Avances en Informática y Automática

Decimoquinto Workshop

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Avances en Informática y Automática

Decimoquinto Workshop

Avances en Informática y Automática

Decimoquinto Workshop

Editores

Héctor Sánchez San Blas
Pablo García Ullán

Publicado en España por:

Departamento de Informática y Automática
Facultad de Ciencias
Universidad de Salamanca
Plaza de los Caídos s/n
37008, Salamanca, España
Tel.: + 34 923294653
Fax: + 34 923294514
Web: `http://mastersi.usal.es`
Email: `mastersi@usal.es`

ISBN **978-84-09-38185-2**

Editores:

Héctor Sánchez San Blas
Pablo García Ullán

Prólogo

El Máster Oficial en Sistemas Inteligentes de la Universidad de Salamanca tiene como principal objetivo promover la iniciación de los estudiantes en el ámbito de la investigación. El congreso organizado por el Departamento de Informática y Automática que se celebra dentro del Máster en Sistemas Inteligentes de la Universidad de Salamanca proporciona la oportunidad ideal para que sus estudiantes presenten los principales resultados de sus Trabajos de Fin de Máster y obtengan una realimentación del interés de los mismos.

La decimoquinta edición del *workshop* "Avances en Informática y Automática", correspondiente al curso 2020 - 2021, ha sido un encuentro interdisciplinar donde se han presentado trabajos pertenecientes a un amplio abanico de líneas de investigación. Todos los trabajos han sido supervisados por investigadores de reconocido prestigio pertenecientes a la Universidad de Salamanca, proporcionando el marco idóneo para sentar las bases de una futura tesis doctoral. Entre los principales objetivos del congreso se encuentran:

- Ofrecer a los estudiantes un marco donde exponer sus primeros trabajos de investigación.
- Proporcionar a los participantes un foro donde discutir ideas y encontrar nuevas sugerencias de compañeros, investigadores y otros asistentes a la reunión.
- Permitir a cada estudiante una realimentación de los participantes sobre su trabajo y una orientación sobre las futuras direcciones de investigación.
- Contribuir al desarrollo del espíritu de colaboración en la investigación.

Organización

El *workshop* "Avances en Informática y Automática" está organizado por el Departamento de Informática y Automática de la Universidad de Salamanca.

Comité Organizador

Angélica González Arrieta
Ángel Zazo Rodríguez
Carlos García Figuerola Paniagua
Emilio Santiago Corchado Rodríguez
Francisco José García Peñalvo
José Luis Alonso Berrocal
José Rafael García-Bermejo Giner
Luis Antonio Miguel Quintales
Maria Gracia Manzano Arjona
Roberto Therón sánchez

Índice general

Sistema de recomendación de música para grupos dependiente del contexto

Adrián Valera Roman, María N. Moreno García, Álvaro Lozano Murciego

Universidad de Salamanca, Salamanca, Plaza de los Caídos s/n 37008, Spain
{adrianvalrom.usal,mmg,loza}@usal.es

Resumen El auge de los servicios de *streaming* de contenidos multimedia ha sido en parte proporcionado por los sistemas de recomendación, que permiten conectar a los usuarios con ítems del catálogo que son de su preferencia. hoy en dia, la música es consumida en grupo tanto en entornos físicos como virtuales, como en salas de chat. Para poder recomendar nuevos ítems a grupos de usuarios es necesarios tener en cuenta todas sus preferencias en conjunto, surgiendo los sistemas de recomendación para grupos. En este trabajo se realiza una revisión del estado del arte sobre los sistemas de recomendación para grupos en el dominio de la música, se plantea un sistema de recomendación en este dominio, sensible al contexto de los usuarios, se propone una aplicación que haga uso de este, y se plantea un experimento offline con un *dataset* de referencia de la literatura. Seguidamente se presentan los resultados obtenidos y se detallan las principales conclusiones extraídas tras analizar los resultados.

Keywords: sistemas de recomendación para grupos, recomendación de música, sistemas de recomendación sensibles al contexto

1. Introducción

La actual era de la información se caracteriza por la disponibilidad generalizada de grandes volúmenes de datos y la democratización del acceso fácil y de bajo coste para todo tipo de usuarios. A pesar de las ventajas e innovaciones, los individuos y las organizaciones se enfrentan a nuevos problemas relacionados con el exceso de información y como hacer llegar aquella relevante a sus clientes o usuarios. Con el fin de superar estos inconvenientes surgen los sistemas de recomendación. Los sistemas de recomendación (SR) utilizan un algoritmo o conjunto de algoritmos para conectar los ítems ofrecidos por un servicio o negocio con sus consumidores a través de técnica de personalización. Si el sistema realiza estas predicciones para grupos de usuarios, agregando las predicciones realizadas para individuos o creando perfiles de grupo con unas determinadas características, estos sistemas se denominan sistemas de recomendación para grupos (SRG).

Con la masificación en el uso de dispositivos portátiles y servicios de *streaming*, los sistemas de recomendación se están convirtiendo esenciales, por ejemplo, en sectores como hostelería, comercio electrónico o cine, entre otros. La

recomendación de contenidos audiovisuales en general, y la recomendación de música en particular [1], es paradigmática ya que son los dominios de ocio aquellos en los que el usuario valora más la novedad y la serendipia, buscando siempre unos límites de novedad dentro de los que se siente cómodo.

Entre las particularidades de la recomendación de música es posible destacar que es un servicio en el que los usuarios consumen artículos de forma rápida y, en consecuencia, es difícil conocer las valoraciones de forma explícita, ya que los usuarios no se detienen a expresarla. Por este motivo, las valoraciones generalmente se deducen a partir del comportamiento del usuario. Además, el problema de la dispersión cobra más importancia que en otros dominios debido a la escasez de valoraciones y a la gran disponibilidad de ítems en los catálogos musicales.

En este dominio, los sistemas de recomendación para grupos adquieren una gran relevancia puesto que un artículo de ocio que suele consumirse en compañía ya sea mediante la reunión de amigos para tal fin o de forma pasiva al realizar otra actividad como una sesión de entrenamiento en un gimnasio o una sesión de trabajo en una biblioteca.

Asimismo, en el dominio de la música, el contexto del usuario también tiene una gran importancia; ya que el momento, el lugar, la tarea que se encuentra realizando, u otros factores como el estado de ánimo, influyen en la valoración que le daría un usuario a una canción. Los sistemas de recomendación basados en contexto utilizan cualquier información que puede ser utilizada para caracterizar la situación del usuario y afinar las predicciones, de tal forma que se pueden realizar predicciones distintas para un mismo ítem y un mismo usuario en función de su estado

Este trabajo tiene como objetivo estudiar el estado del arte actual de los sistemas de recomendación para grupos dependientes del contexto en el dominio de la música; analizar los algoritmos en este dominio buscando aquel que ofrezca mejor desempeño; y finalmente implementar una aplicación que permita al usuario hacer uso del sistema en una situación real.

Este artículo se estructura de la siguiente forma: la sección 2 revisa brevemente el estado del arte de los sistemas de recomendación para grupos, en la sección 3 se presenta el sistema de recomendación para grupos sensible al contexto orientado al dominio de la música propuesto, la sección 4 presenta un caso de estudio que consiste en una serie de experimentos offline que estudian distintas configuraciones del sistema propuesto, para ver como varían las métricas utilizando distintos algoritmos y funciones de agregación y, por último, la sección 5 presenta las conclusiones obtenidas, las limitaciones del estudio y las líneas de trabajo futuras.

2. Estado del arte

Tradicionalmente los sistemas de recomendación se han centrado en los individuos, sin embargo, en los últimos tiempos, con la aparición de aplicaciones móviles y la masificación de su uso, han proliferado las redes sociales, aplicaciones de uso colaborativo y los sistemas de recomendación enfocados en grupos. Un sistema de recomendación para grupos (SRG) [2–5] se centra en escenarios en los cuales las soluciones de

recomendación se proveen a grupos de usuarios que tienen preferencias individuales. Aunque en la mayor parte de características coinciden con los SR [1], ya que suelen ser derivados de ellos a través de métodos como la agregación de SR individuales, surgen nuevos problemas y consideraciones.

En función de cómo se consideran los grupos, los SRG pueden consistir en una agregación de sistemas de recomendación para individuos (predicciones agregadas) o considerar de forma simultánea las preferencias de todos los usuarios agregándolas a un perfil de grupo, para el que se realizan las recomendaciones (modelos agregados). El proceso de generación de las recomendaciones puede verse en la **Fig. 1**.

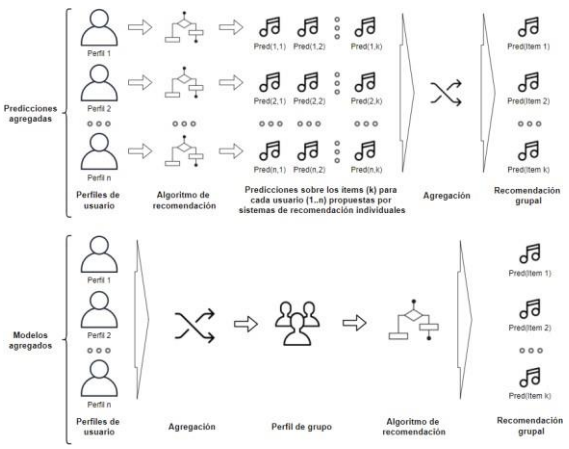


Fig. 1. Formas de agregación en los SRG

Uno de los elementos de discusión más importantes a la hora de desarrollar un SRG es la elección de la estrategia de agregación. Los autores en Felfering et al. [6] discuten distintas estrategias de agregación y los distintos enfoques existentes en la recomendación para grupos. La evaluación del algoritmo de recomendación grupal es primordial para identificar la utilidad de este. Por ello, se analizan los distintos enfoques para evaluar los modelos de recomendación. Los autores también discuten varias métricas utilizadas para evaluar los modelos de recomendación grupal y su relevancia basada en el dominio de un ítem.

Las predicciones se obtienen a partir de las preferencias de los individuos, que pueden ser obtenidas de dos formas:

1. **Obtención explícita:** Cuando los usuarios indican sus preferencias de forma explícita a través de elementos como cuestionarios o la elección de categorías de ítems.
2. **Obtención implícita:** Cuando se inducen las preferencias de los usuarios a través de la interacción que tiene con los ítems de un sistema. Supone un método menos intrusivo, pero mas subjetivo.

2.1 Tipos de grupos

Los tipos de grupos en un SRG pueden clasificarse de distintas formas [4, 7], por ejemplo: en función de los tipos de preferencias de los miembros, en función del método de agregación de los distintos miembros que lo componen o en función del método con el que fueron creados.

Considerando a los integrantes del grupo se establecen estas dos categorías:

1. **Grupos homogéneos:** Grupos cuyos miembros tienen intereses similares.
2. **Grupos heterogéneos:** Grupos cuyos miembros tienen intereses diversos, que pueden llegar a ser muy dispares.

En función de cómo se crea el grupo, pueden clasificarse en tres categorías:

1. **Grupos establecidos:** Formados por individuos que eligieron explícitamente pertener al grupo al tener ciertos intereses en particular.
2. **Grupos ocasionales:** Formados por personas que realizan de forma ocasional alguna actividad juntos. Los miembros pueden tener algunos gustos en particular respecto a un tema o actividad en específico.
3. **Grupos aleatorios:** Grupo de personas que comparten entorno momentáneamente durante un determinado tiempo, y cuyos miembros posiblemente no tengan mucha relación entre ellos.
4. **Grupos automáticamente identificados:** Grupo formado de forma automática juntando a individuos con preferencias similares, como por ejemplo grupos de revisores de artículos que dominan ciertos temas.

2.2 Algoritmos utilizados

Tanto los SR como los SRG pueden ser estar basados en los siguientes enfoques:

- **Sistemas de recomendación basados en contenido:** El filtrado basado en contenido aplicado a grupos se basa en la idea de recomendar nuevos ítems en función de categorías similares a las que contienen las preferencias del usuario. Por ejemplo, la recomendación de libros para un cliente de una librería en función de características como los géneros que ha leído o sus los autores favoritos.
- **Sistemas de recomendación basados en filtrado colaborativo (FC):** Son sistemas de recomendación que utilizan matrices de similitud para estimar valoraciones basándose en las preferencias de los usuarios y ítems vecinos. Siguiendo con el ejemplo anterior, en lugar de centrarse en las propiedades de los libros anteriormente leídos, el algoritmo CF tomaría en consideración las valoraciones de los libros de otros usuarios similares a él/ella a la hora de generar la preferencia para un individuo.

  El FC pueden ser subcategorizado entre los basados en memoria (*memory-based*) y los basados en modelo (*model-based*). Los primeros hacen uso de la matriz de similitud completa a la hora de generar una recomendación, trabajando directamente con el conjunto de datos; mientras que los basados en modelo, utilizando métodos de *machine learning* y minería de datos, crean modelos predictivos.

El FC basado en memoria puede dividirse nuevamente entre los basados en usuario (*user-based*) y los basados en ítems (*item-based*) en función de los elementos considerados para encontrar las similitudes.

- **Sistemas de recomendación híbridos:** son sistemas que combinan varios enfoques buscando una mejora en el desempeño.

Además, para los SRG en particular existen los siguientes enfoques:

- **Recomendación basada en restricciones:** Este tipo de recomendación utiliza restricciones propuestas por los integrantes de los grupos que se deben cumplir en las recomendaciones propuestas.
- **Recomendación basada en la crítica:** Se muestran al usuario ítems de referencia de tal forma que puedan proveer retroalimentación a través de un proceso iterativo que mejore la recomendación.

2.3 Métodos de agregación

Ya sea a través de predicciones agregadas o modelos agregados, un SRG utiliza métodos de agregación para componer las recomendaciones para el grupo.

En [7] se proponen las estrategias de agregación listadas en la **Tabla 1**. Estan agrupadas en función de 3 categorias: *Majority-based* (M), estrategias que se fundamentan en la idea de que la mejor recomendación se encuentra al fomentar aquellos ítems que son más populares; *Consensus-based* (C), estrategias de agregación que tartan de tener en consideración las preferencias de todos los miembros del grupo, buscando un enfoque mas democrático; y *borderline* (B), estrategias de agregación que toman en consideración un subconjunto de todos los usuarios disponibles.

Estrategia de agregación	Descripción	Función
Additive Utilitarian (ADD) [C]	Suma de las valoraciones predichas para un ítem	$argmax_{(t \in I)}(\sum_{u \in G} eval(u,t))$
Multiplicative (MUL) [C]	Multiplicación de las valoraciones predichas para un ítem	$argmax_{(t \in I)}(\prod_{u \in G} eval(u,t))$
Average (AVG) [C]	Media de las valoraciones predichas sobre cada ítem ítems	$argmax_{(t \in I)}\left(\frac{\sum_{u \in G} eval(u,t)}{\lvert G \rvert}\right)$
Average without Misery (AVM) [C]	Media de las predicciones predichas que superan cierto umbral de relevancia	$argmax_{(t \in I))}\left(\frac{\sum_{u \in G} rating(u,t)}{\lvert G \rvert}\right)$ $(t \in I: \nexists u \in G \mid eval(u,t) \leq threshold)$
Approval Voting (APP) [M]	Número de valoraciones predichas que	$argmax_{(t \in I)}\ (\lvert\{u \in G: eval(u,t) \geq threshold\}\rvert)$

	superan cierto umbral de relevancia	
Most Pleasure (MPL) [B]	Máxima valoración de un ítem	$argmax_{(t \in I)}(maxeval(t))$
Least Misery (LMS) [B]	Mínima predicción de un ítem	$argmax_{(t \in I)}(mineval(t))$
Majority Voting (MAJ) [B]	Máxima predicción de un ítem	$argmax_{(t \in I)}(majorityeval(t))$

Tabla 1. Funciones de agregación

2.4 Evaluación de las predicciones

Existen dos protocolos de evaluación de los resultados de un SRG:

- **Evaluación offline:** Basada en la idea de estimar la calidad de la predicción de un algoritmo usando conjuntos de datos que incluyen las valoraciones que dan los usuarios a cada ítem, donde se sigue el esquema clásico de división del *dataset* en conjunto de entrenamiento y de prueba.
- **Evaluación online:** Protocolo de evaluación tradicional basada en el estudio de campo analizando el comportamiento de usuarios, por ejemplo, a través de encuestas o pruebas A/B.

Las métricas utilizadas para evaluar los sistemas de recomendación para grupos suelen ser similares a las utilizadas en los SR. Se pueden clasificar en según el modo en que funciona el SRG:

1. **Predicción de ratings:** El sistema realiza predicciones sobre los ítems que no ha visto un usuario o un grupo, otorgándole una puntuación. Para evaluar estos sistemas sea analizan diferentes tipos de error en la predicción.
2. **Recomendaciones Top-N:** El Sistema muestra al grupo una lista ordenada de los ítems relevantes recomendados. Para evaluar estas recomendaciones se usan medidas de precision o se evalúa la correcta ordenación de la lista.

Las métricas enfocadas en las recomendaciones Top-N suele ser más razonables, ya que un sistema podría acertar con la predicción de los ítems no deseados, pero en cambio ser incapaz de recomendar los ítems relevantes que el usuario finalmente consumiría. Se considera que un ítem es relevante para un grupo cuando supera un umbral de relevancia.

Las métricas más utilizadas para evaluar este tipo de recomendaciones son las de clasificación y el *Normalized Discounted Cumulative Gain* (NDCG). Entre las métricas de clasificación encontramos la precisión y el Recall.

La precisión es la fracción del número de ítems relevantes recomendados en relación con el número total de ítems recomendados (Ecuación (1)),

$$\text{precision@k(g)} = \frac{|\text{predicted}_k(g) \cap \text{relevant}_k(g)|}{k} \quad (1)$$

Donde $\text{predicted}_k(g)$ es la lista de k ítems recomendados al grupo g, y relevant(g) representa todos los ítems relevantes para el grupo g y k el número de ítems recomendados.

El Recall, o sensibilidad (Ecuación (2)), es la fracción del número de ítems recomendados en relación con el número total de ítems relevantes:

$$\text{recall@k(g)} = \frac{|\text{predicted}_k(g) \cap \text{relevant(g)}|}{|\text{relevant(g)}|} \tag{2}$$

La ganancia acumulada descontada, o *discounted cumulative gain* (DCG), se basa en la idea de que aquellos ítems con una menor puntuación para la recomendación deberían ser penalizados decrementando su relevancia de forma logarítmica.

$$DCG@k(g) = \sum_{i=1}^{k} \frac{2^{relevance(t,g)} - 1}{\log_2(i+1)} \tag{3}$$

Sin embargo, este valor puede requerir de normalización si varía el número de ítems recomendados en las listas, esta se realiza en comparación con la iDCG (ideal) (Ecuación (4(4)) obteniendo el nDCG (Ecuación (5)).

$$iDCG@k = \sum_{i=1}^{k} \frac{1}{\log_2(i+1)} \tag{4}$$

$$nDCG@k(g) = \frac{DCG@k(g)}{\text{iDCG@k}} \tag{5}$$

Otra forma de obtener estas métricas es, en lugar de realizar primero una agregación de las valoraciones dadas por los integrantes de cada grupo y evaluar la lista, calcular la métrica para un usuario respecto a la recomendación en primer lugar para todos los usuarios del grupo y luego calcular la media entre ellos (Ecuación (6)).

$$\text{metrica@k(g)} = \frac{\sum_{u \in g} metrica(u)}{|g|} \tag{6}$$

2.5 Sistemas de recomendación sensibles al contexto

Los factores contextuales adquieren una gran importancia en los sistemas de recomendación desde el momento en el que un ítem puede ser valorado muy positivamente por los usuarios en cierto contexto, pero ser inoportuno en otro. Principalmente, existent tres métodos para tratar la información contextual:

- **Métodos 2D:** Son los métodos más comúnmente utilizados, los más sencillos y que menos requisitos necesitan ya que trabajan con un espacio bidimensional:

$$f_{rating}: User \times Item \rightarrow Rating$$

Estos métodos realizan un filtrado de los ítems en función de los requisitos contextuales antes (prefiltrado) o después (postfiltrado) de realizar la recomendación.

- **Modelado contextual:** El modelado contextual consiste en la consideración de toda la información contextual en el momento en el que se realiza la recomendación, incorporandose en el propio algoritmo.

$$f_{rating}: User \times Item \times Context \rightarrow Rating$$

- **Sistemas híbridos:** combinan dos o mas de los filtrados citados anterioremnte

3. Sistema propuesto

El sistema propuesto consta de dos partes: un sistema de recomendación para grupos dependiente de contexto y una Progresive Web App (PWA), llamada *The Group Taste*, que puede ser utilizada por grupos de usuarios, para los que se realizan recomendaciones en función de determinados contextos establecidos por la sala a la que acceden.

En la **Fig. 2 (A)** se muestra un diagrama del sistema de recomendación propuesto, desarrollado en Python y que se detallará en la sección 3.1. En la **Fig. 2 (B)** se muestra la estructura que sigue la aplicación, que se detallará en la sección 3.2. Está desarrollada en Vue.js e integrada con Google Firebase, Spotify y el sistema de recomendación anteriormente citado.

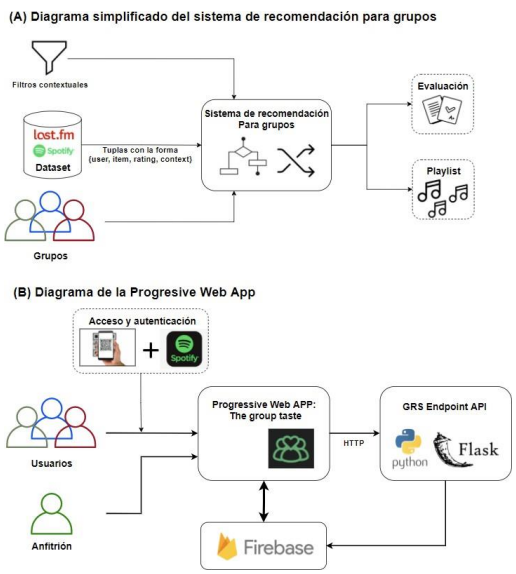


Fig. 2. Diagramas de las dos partes del proyecto

3.1 Sistema de recomendación para grupos

El sistema de recomendación para grupos dependiente del contexto es un programa desarrollado en Python, al que se le introduce un dataset de registros con al menos los campos:

{usuario, canción, valoración, contexto}

Junto a una configuración, que incluye aquellos filtros por los que se desea modelar el contexto. El sistema devuelve listas de recomendación Top-N y/o evaluaciones. Se trata de un SRG que utiliza algoritmos de filtrado colaborativo basados en memoria para realizar predicciones agregadas, donde la información contextual se explota previamente mediante el uso de un método 2D, concretamente el prefiltrado contextual.

El objetivo de la realización de este Sistema es, además de obtener un software que sea capaz de realizar listas de recomendación para grupos y evaluar la validez de éstas, permitir la realización de experimentos de forma automatizada, que se detallarán en la sección 4. Se ha diseñado en Python buscando un enfoque extensible, lo que permite la rápida implementación y evaluación de distintos algoritmos, métodos de agregación, filtros contextuales, grupos y métricas de evaluación. Es decir, se ha desarrollado un *benchmark* que permite comparar distintos sistemas de recomendación para grupos a través de la evaluación offline.

3.1.1. Arquitectura del SRG

Las fases requeridas para la recomendación se encuentran bien delimitadas, por lo que para desarrollar la arquitectura del SRG se ha optado por realizar un sistema donde cada uno de los componentes tenga responsabilidades específicas.

Las fases que componen la recomendación de grupo son: (1) carga del dataset y prefiltrado contextual, (2) creación de los grupos, (3) realización de las recomendaciones individuales, (4) agregación de las recomendaciones en una lista Top-N y (5) evaluación de la recomendación.

Para cada una de las fases se ha creado las siguientes clases, cuya relación se detalla en la **Fig. 3**: (1) *Dataset*, (2) *Group*, (3) *Algorithm* (4) *Aggregator* y (5) *Evaluator*.

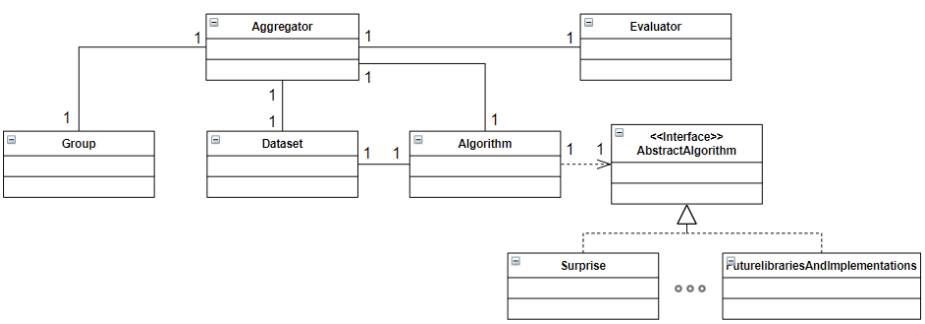


Fig. 3. Diagrama de clases del SRG

3.1.2. Algoritmos y métodos de agregación implementados

Se han implementado un total de 4 algoritmos diferentes: KNN, CoClustering, NMF y Baseline proporcionados por la librería Surprise [8], que tiene buenos resultados de rendimiento a la hora de generar las recomendaciones individuales, al estar escrito parcialmente en C.

Se han implementado un total de 8 métodos de agregación: avg, add, mul, avm, app, lms, maj y mpl, anteriormente citados en el estado del arte (sección 2.3).

3.1.3. API de recomendación para grupos

Una vez concluido el SRG se ha desarrollado un servicio web REST para Python haciendo uso de la librería Flask, de tal modo que una aplicación externa, como la PWA planteada en la siguiente sección, pueda acceder al SRG a través de peticiones HTTP. Tiene como objetivo satisfacer dos necesidades: (1) generar una lista de reproducción Top-N para un grupo que está utilizando la aplicación (2) permitir evaluar a los usuarios las canciones que se han recomendado en el grupo.

3.2 Aplicación desarrollada: The Group Taste

Se ha planteado una Progresive Web App (PWA) para resolver la siguiente situación: El dueño de un establecimiento, como un bar o un gimnasio, desea que los clientes estén satisfechos con la música que allí suena y que además esta esté relacionada con la temática contextual del lugar, para ello coloca un código QR en la entrada y mesas del establecimiento que los clientes pueden escanear para acceder a una sala virtual. En el acceso a dicha aplicación, compartirán sus gustos musicales a través de su cuenta de Spotify, tras lo cual un SRG puede realizar una recomendación musical que satisfaga a todos; finalmente, los usuarios pueden ver la música que actualmente se encuentra reproduciendo en el establecimiento y transmitir retroalimentación a los encargados a través de la aplicación.

Se ha desarrollado utilizando el *framework* Vue.js y se ha integrado con servicios de *Google Firebase*, *Spotify* y el sistema de recomendación desarrollado. A través de *Firebase* se accede a funcionalidades como la base de datos (*Firestore*) y el *hosting* (*Firebase Hosting*) para la aplicación web. Por otra parte, a través de *Spotify* se gestiona la autenticación de los usuarios, se obtienen las preferencias de los usuarios y se reproduce la música.

En la **Fig. 4** se muestra la estructura del funcionamiento de la aplicación web de forma resumida, que se fundamenta principalmente de tres partes: la aplicación, la base de datos de *Firebase* y el *servicio web* con el sistema de recomendación.

El funcionamiento planteado para el sistema propuesto es el siguiente: Una serie de usuarios entran en un establecimiento físico (e.g. un bar o un gimnasio) y acceden a una sala virtual a través de un código QR autenticándose a través de Spotify (1), cuando el anfitrión del establecimiento lo considera oportuno (2) puede solicitar la creación y gestión de una *playlist*, que se genera en el sistema de recomendación para grupos (3) y que se almacena en a la base de datos *Firestore*.

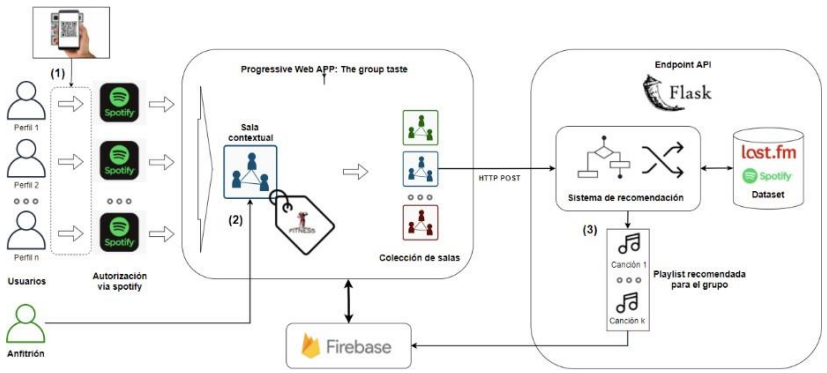


Fig. 4. Estructura de funcionamiento de la aplicación web

3.2.1. Gestión de usuarios y sus preferencias

Los datos sobre los usuarios se obtienen de la información proporcionada por la API de Spotify. Siendo estos los relacionados con el perfil de usuario y las canciones recientemente escuchadas. Una vez la aplicación recopile el perfil con los gustos de los usuarios, se almacenan en una base de datos no relacional (*Firestore*) proporcionada por la plataforma *Google Firebase*.

Spotify ofrece la posibilidad a los desarrolladores de implementar sus propias aplicaciones de música gracias a una API pública [9] accesible por cualquiera que disponga de una cuenta en la plataforma. Para acceder a la API es necesario previamente identificarse con el usuario y la contraseña para obtener un token de acceso único con el que realizar peticiones (1), una vez un usuario disponga del token, podrá realizar cuantas peticiones precise (2) para obtener distinta información sobre su cuenta: perfil de usuario, canciones recientemente escuchadas, playlists disponibles en su perfil…

Todas las canciones de Spotify, que tienen asignadas un identificador único, poseen una serie de características que identifican distintas propiedades de la música, pudiendo estar relacionadas con distintos factores contextuales que se definirán en la sección 4. Estas propiedades contextuales serán utilizadas para definir el contexto con el que un anfitrión decide caracterizar una sala.

3.2.2. Diseño de la PWA

La aplicación se ha desarrollado procurando que sea lo más simple e intuitiva posible ya que las dos principales funciones son las de recoger información sobre los gustos de los usuarios y hacerles sentir partícipes de un grupo mostrándole información sobre la música que se reproduce en un establecimiento.

En la **Fig. 5** se muestran las vistas con las que interactuaría el usuario. El acceso a la aplicación se realiza a través de la lectura de un código QR (**Fig. 5 (A)**), gracias al cual el usuario puede acceder de una forma intuitiva, y el proceso de identificación se lleva a través de Spotify (**Fig. 5 (B)**). Una vez el usuario se identifica, se le muestra la vista

de la *playlist* que actualmente se está reproduciendo (**Fig. 5 (C)**), momento en el cual el sistema de recomendación tendría a disposición sus preferencias musicales, permitiéndole al anfitrión solicitar generar nuevas predicciones.

Además, al usuario se le facilita la posibilidad de valorar positiva o negativamente las canciones propuestas por el GRS.

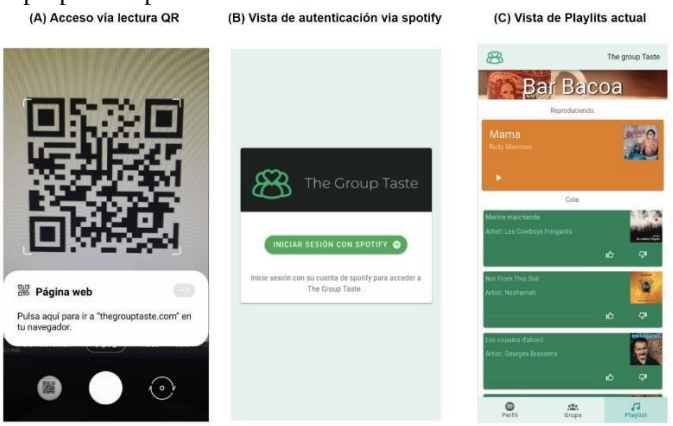


Fig. 5. Vistas del proceso de identificación en *The Group Taste*

El anfitrión tendría acceso a una vista extra, oculta para los clientes, en la que puede gestionar las salas existentes. Cuando lo considere oportuno, será él el que solicite la generación de una recomendación para el grupo que se encuentre en la sala, momento en el cual se hará una petición HTTP al *endpoint* con el SRG que generará la recomendación para el grupo a través de un servicio Rest.

4. Caso de estudio y resultados del experimento

En esta sección se detalla el caso de estudio offline planteado para evaluar el sistema de recomendación de música propuesto en la sección 3.1, que consiste en la automatización de experimentos para comprobar la eficiencia de distintos algoritmos y métodos de agregación para distintos grupos y contextos. Estos experimentos se han realizado sobre un conjunto de datos que se ha obtenido a partir de la fusión del dataset LastFM-1k [10, 11] con la información sobre las canciones proporcionada por la API de Spotify.

4.1 Conjunto de datos utilizado

Para realizar los experimentos se ha creado un *dataset* propio, uniendo el conjunto de datos de LastFM1k, que contiene un total de 19.150.869 tuplas, en las que se incluyen 992 usuarios y 107.528 artistas, con la información sobre canciones provista por Spotify en un único fichero que tiene registros con la tupla:

{user id, song id, rating, {propiedades contextuales}}

Aunque en el conjunto de datos original no existen valoraciones explícitas, la existencia de eventos de escucha basados en marcas temporales permite la inferencia de estas a través del número de reproducciones de una canción. Para ello se ha utilizado el algoritmo propuesto por Maciej Pacula en [12]. Con este método se consiguen aproximar valoraciones explícitas a partir de implícitas.

En la Ecuación (7) se describe la frecuencia para un usuario i y una canción j siento $count(i,\ j)$ el número reproducciones de esa canción para ese usuario y totalCount el número total de reproducciones del usuario para todo el catálogo. En la Ecuación (8) se describe como se obtiene el rating explicito $r_{i,j}$ de un usuario i para una canción j en un ranking de tamaño k ordenado por frecuencias de reproducción. Siendo k'la canción con más reproducciones. De esta forma se obtiene una puntuación en el intervalo (0,4].

$$freq_{i,j} = \frac{count(i,\ j)}{totalCount} \tag{7}$$

$$r_{i,j} = 4 \cdot (1 - \sum_{k^F=1}^{k-1} freq_{k^F}(i)) \tag{8}$$

Para obtener las propiedades contextuales se ha hecho uso de las propiedades proporcionadas por Spotify. Spotify ofrece una API [9] a través de la cual es posible acceder a distintos datos, ya sean relativos a usuarios o a artistas y canciones. En particular asigna a las canciones 12 características: *Danceability, accousticness, duration, energy, instrumentalness, liveness, loudness, mode, explicit, speechiness, tempo y valence*.

Para obtener el *dataset* final, se trataron de realizar peticiones para cada una de las canciones de las tuplas del *dataset* LastFM-1k.Se recuperó información de un total de 688.534 canciones siguiendo este método, aproximadamente un 50% del catálogo de canciones del *dataset*.

Estas características serán utilizadas para la inferencia del contexto. Se han establecido por tanto 4 tipos de contexto siendo estos: (1) *no context*, (2) *party* (3) *fitness* y (4) *chill*. Para la determinación de estos se han tomado umbrales para las siguientes propiedades:

1. **No context:** No considera ninguna propiedad. Este contexto incluye a todas las canciones sin hacer ninguna distinción por propiedad.
2. **Party:** Se considera que una canción es apropiada para un entorno festivo cuando el valor de la Danceability supera el 90%.
3. **Fitness:** Se considera que una canción es apropiada para un entorno deportivo cuando el valor de Energy supera el 90%.
4. **Chill:** Se considera que una canción es apropiada para la relajación cuando el valor de Energy es inferior al 10%.

Una vez obtenido el *dataset* definitivo se realiza un análisis exploratorio de datos (*Exploratory Data Analysis* - EDA), para una primera exploración de los datos obtenidos, donde se encuentran 3410869 registros.

Como se detectó un número medio de 4.95 valoraciones por canción, para evitar problemas en la recomendación en los casos en los que haya demasiadas canciones que tengan pocas reproducciones (han sido escuchadas por pocos usuarios) y por tanto no sean aptas para la realización de un filtrado colaborativo, se ha realizado un filtrado adicional sobre el dataset, estableciendo un umbral mínimo de valoraciones que debe tener una canción para poder utilizarse en el filtrado colaborativo. Se realizaron distintas pruebas y finalmente se consideraron aquellas canciones que tenían al menos 50 valoraciones compartidas por distintos usuarios.

Una vez se realiza este filtrado, el *dataset* se reduce de 3410869 registros a 796937 (23.3%) y las características del *dataset* pasan a ser las de la **¡Error! No se encuentra el origen de la referencia.**:

# Usuarios	#Canciones	#Caract. Contextuales	# medio de valoraciones por canción	# medio de valoraciones por usuario	Valoración media	STD	Valoración Mínima	Valoración Máxima
987	9092	12	87,64	806,6	1,2	1,02	0	4

Tabla 2 Descripción del *dataset*

Y como suele ser común en este tipo de problemas, y como puede observarse en la **Fig. 6**, las valoraciones siguen una distribución de ley de potencia, también conocida como *long tail*, habiendo muchas más canciones con valoraciones bajas que altas. Esto podría suponer un problema, no obstante, en este caso el *long tail* no es tan acentuado al tener una media de 1,082, una desviación de 0,96 y un número suficientemente representativo de valoraciones positivas (aquellas superiores a un umbral de relevancia, como podría ser 0.6*4=2.4)

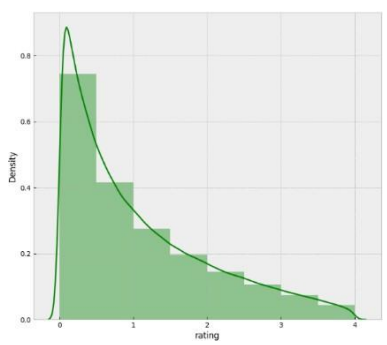


Fig. 6. Distribución de valores de las valoraciones

4.2 Experimento planteado

Una vez se recopilado el conjunto de datos se plantea el experimento para evaluar el sistema. Se busca conocer cuáles son los mejores algoritmos, los mejores métodos de agregación y como varía la consideración del contexto; por ello se plantea el siguiente experimento: se ejecutan todas las posibles combinaciones de algoritmos, métodos de agregación y contextos para obtener listas Top-N para distintos grupos, y posteriormente estas se evalúan con las métricas de precisión y NDCG.

Para la realización de los experimentos se han considerado 10 grupos de 5 miembros por cada uno de los siguientes posibles tipos de grupos:

1. **Grupos aleatorios:** Se consideran aleatoriamente personas de los 987 usuarios totales en el *dataset*.
2. **Grupos contextuales:** Se han considerado personas aleatorias tras haber filtrado el *dataset* en función de los contextos, de los resultados obtenidos se han tomado aquellos individuos que tuviesen un cierto número de canciones relevantes en ese contexto, 10 en este caso, de manera que se puede considerar que esos usuarios tienen un criterio formado en ese contexto.

Se han utilizado exclusivamente las métricas de precisión y evaluación de listas, que como indicaban los trabajos [2, 13] están ganando popularidad frente al resto, pues las métricas de error no proporcionan información demasiado útil a la hora de analizar las listas ordenadas, ya que en una recomendación de lista, y sobre todo en una recomendación agregada para varios usuarios de un grupo, no interesa tanto acertar en la predicción si no elegir aquellos elementos relevantes para la mayoría y poder presentárselos a los usuarios lo antes posible.

Una vez establecidos los grupos y las métricas a utilizar, para la realización del experimento se hacen recomendaciones para las distintas combinaciones de los siguientes elementos:

1. **4 contextos:** No context, party, fitness y chill.
2. **40 grupos de 5 integrantes:** resultado de generar 10 grupos por cada contexto
3. **4 algoritmos:** KNN, NMF, CoClustering y baseline.
4. **8 métodos de agregación:** AVG, ADD, MUL, AVM, APP, LMS, MAJ y MPL.
5. **3 métricas:** Precision, Recall y NDCG
6. **20 distancias distintas para las listas Top-N (valores @k):** Comprendidas entre 5 y 100, a las que se referirá en lo sucesivo como longitud @k de la recomendación.

4.3 Resultados

La ejecución del experimento propuesto generó satisfactoriamente un archivo con las métricas para un total de 102400 recomendaciones. Para analizarlas se han planteado una serie de preguntas, para las cuales se ha filtrado el archivo de resultados con el objetivo de generar las gráficas que pudiesen responderlas:

1. ¿Existe variación en los resultados para grupos creados con el mismo criterio?

2. ¿Cuáles son las mejores agregaciones en función del algoritmo? ¿Cómo afecta la aplicación de un contexto a la evolución de la métrica? ¿Y para un @k fijo?
3. ¿Para qué algoritmos/métodos las métricas varían menos al empezar a aplicar un contexto?
4. Considerando el mejor método de agregación, ¿Qué algoritmos dan mejores resultados tras aplicar un filtrado contextual y cómo varía esto respecto a la longitud de la lista recomendada?
5. ¿Cómo varían las métricas, en función del método de agregación utilizado con cada algoritmo, según incrementa la longitud de la lista Top-N?
6. ¿Qué consideraciones hay que tener para la valoración de los valores de las métricas obtenidas y proceso de selección de un método de agregación o algoritmo?

En la **Fig. 7** y **Fig. 8** se muestran algunas de las gráficas generadas para poder analizar las cuestiones planteadas.

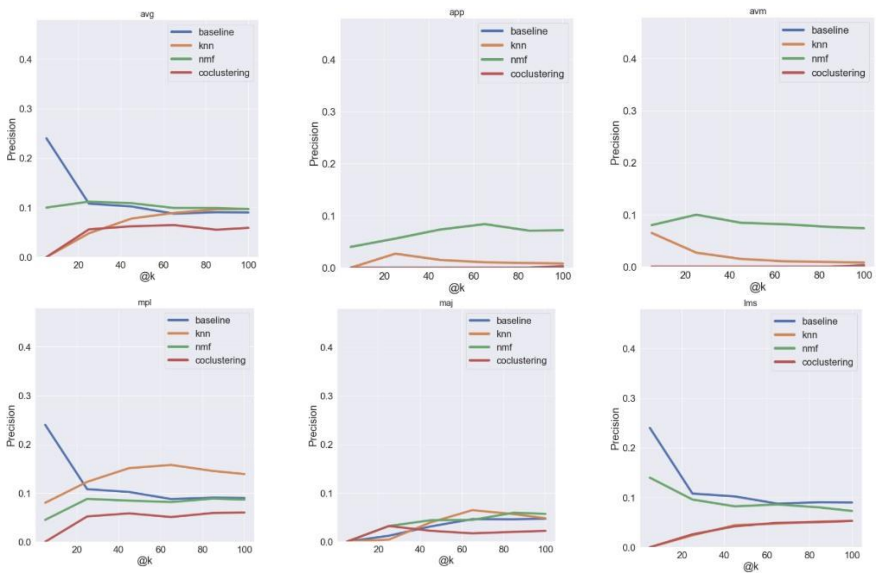


Fig. 7. Evolución de la precisión en función del algoritmo de recomendación utilizado para algunos de los métodos de agregación en el contexto Chill

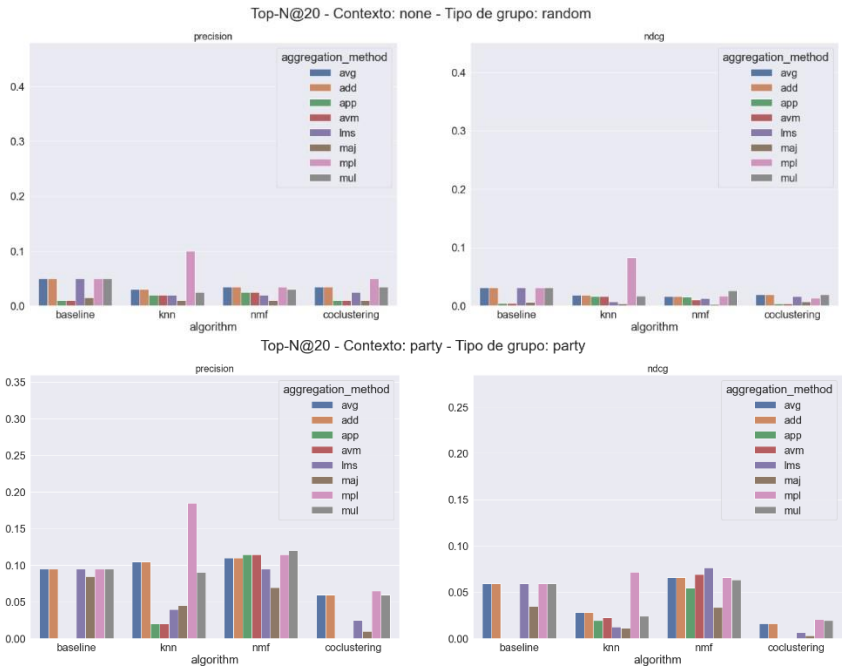


Fig. 8. Comparación entre recomendaciones generadas para grupos aleatorios sin contexto frente las generadas para el grupo de contexto chill, en función del método de agregación y el algoritmo en las recomendaciones Top-N@20

Una vez respondidas las preguntas anteriores y mostradas distintas formas de comparar las métricas de los resultados del experimento, (en función de contexto, de los algoritmos, métodos de agregación y longitudes de las listas Top-N evaluadas) se llega a la conclusión de que dependiendo del caso particular a implementar puede ser interesante considerar una métrica u otra, pues se comportan de diferente forma en función de una circunstancia en particular.

Se ha detectado, por ejemplo, que cuando no hay contexto y los grupos se han formado de forma aleatoria, el único caso que ofrece resultados aceptables es el obtenido al usar el algoritmo KNN y el método de agregación *Most pleasure* (MPL), o que KNN funciona también bien para un determinado contexto, pero es superado por NMF cuando la lista a recomendar tiene que tener pocas canciones.

También se ha encontrado que el algoritmo NMF da resultados aceptables tanto de *Precision* como de NDCG con un mayor número de funciones de agregación que KNN

Si consideramos un ejemplo de caso práctico de selección de parámetros, interesa buscar aquellas implementaciones que ofrezcan un NDCG alto si lo que se desea es la generación de *playlists*, generalmente cortas, en las cuales las canciones deben tener una ordenación correcta y los elementos relevantes al deberían encontrarse al principio. En cambio, si se desea generar recomendaciones de música para un establecimiento, como un bar, es mejor centrarse en la *Precision* y en un Recall alto, que se traduzcan en que haya gran cantidad de recomendaciones relevantes en listas Top-N largas, ya

que la música estará sonando durante más tiempo y los ítems no se consumen de una forma tan inmediata.

5. Conclusiones, limitaciones y líneas de trabajo futuras

Una vez finalizado el TFM, es posible concluir que se han logrado llevar a cabo los objetivos inicialmente propuestos.

En primer lugar, se ha revisado el estado del arte de los sistemas de recomendación, tanto los individuales como los enfocados en grupos, y se han analizado varios sistemas de recomendación para grupos en el dominio de la música.

Por otra parte, se ha desarrollado un SRG con un planteamiento modular y parametrizable que le da la capacidad de ser fácilmente extensible, permitiendo que se integre sin dificultad con otros sistemas. Se han implementado funciones de agregación descritas en la literatura y además el sistema posibilita la integración de distintos algoritmos y métodos de agregación con facilidad por lo que puede utilizarse de una forma cómoda para generalizar y automatizar la realización de nuevos experimentos

Adicionalmente, se ha realizado el estudio experimental propuesto para un caso concreto (Sección 4) y se han evaluado los resultados obtenidos a través de métricas de precisión y ranking, lo que ha permitido valorar los distintos algoritmos de recomendación y métodos de agregación. Los resultados de las recomendaciones obtenidas con varios de los algoritmos y métodos de agregación implementados son buenos, pues se obtienen métricas razonables como precisiones comprendidas entre el 10-25% y *recalls* superiores al 20%. Se concluye que en función del problema a resolver pueden considerarse distintas combinaciones de algoritmos y métodos de agregación, por lo que es interesante la utilización de un *benchmark* como este cuando se desea estudiar un caso práctico.

Finalmente, se ha desarrollado una herramienta web que permite utilizar el sistema de recomendación propuesto en el dominio de la recomendación de música. Gracias a esta aplicación se pueden solventar algunos de los inconvenientes encontrados en el sector como la obtención de valoraciones explícitas y la posibilidad de realizar la evaluación online de una forma sencilla.

5.1 Limitaciones del estudio

Se han encontrado ciertos inconvenientes a la hora de realizar el estudio:

- **Naturaleza de los datos:** El *dataset* utilizado ha sido elaborado a partir de un conjunto de datos antiguo, y ha tenido que tratarse artificialmente para que el SRG realizase predicciones óptimas. Para la realización de experimentos de los que extraer conclusiones más relevantes deberían utilizarse conjuntos de datos obtenidos expresamente para este experimento.
- **Problema de selección de argumentos:** La gran cantidad de combinaciones de argumentos hace inviable la comprobación de todos ellos de una forma sencilla.
- **Problema del arranque en frio y cantidad de** usuarios en The Group Taste: Como se hace uso del filtrado colaborativo, para generar recomendaciones de

calidad en la aplicación es necesario que haya suficientes preferencias pasadas, es un problema común en estas situaciones y de difícil solución.

5.2 Líneas de trabajo futuras

A partir de las conclusiones extraídas del trabajo logrado se plantean las siguientes líneas de investigación futuras:

Inducción del contexto a partir de las propiedades de Spotify: Esta parte del sistema de recomendación puede ser mejorada extrayendo el contexto a partir de las características de las canciones disponibles en el *datasets*, pues pueden definir los contextos establecidos para este trabajo de una forma más precisa. Se plantea como trabajo futuro la implementación de un algoritmo de clasificación que permita inferir el contexto en base a las características que proporciona la API de Spotify. En esta línea de investigación trabajos como el de [14] han comenzado a explorar como realizar el *auto-tagging* contextual de canciones.

Implementación de la recomendación basada en contenido y otras funciones de agregación: El sistema se puede extender con nuevos tipos de algoritmos y más funciones de agregación, lo que permitiría comparar en profundidad que estrategias de agregado son las mejores en cada caso: Majority-based, consensus-based y borderline.

Realización de nuevos experimentos e incorporación de nuevos parámetros y métricas al *benchmark*: Uno de los problemas a los que se enfrenta la recomendación para grupos es la selección de los parámetros correctos dentro de la gran cantidad de opciones disponibles. Para obtener los algoritmos más precisos en distintas situaciones es necesario realizar nuevos experimentos y variar parámetros que se han mantenido estáticos en el caso de estudio, como es el caso del tamaño de grupo. Del mismo modo, hay otros factores interesantes a estudiar que se han obviado en este trabajo, como la forma en la que afecta el método de obtención de las valoraciones, en el caso de estudio se obtiene una valoración sobre las canciones sin considerar el tiempo de escucha, pero trabajos como [15] se proponen métodos de procesamiento de ratings implícitos que podrían ser más precisos al considerar también factores temporales.

Experimento de campo y evaluación online: *The Group Taste* permite la evaluación online a través de una interfaz sencilla, se puede plantear un estudio de campo en el que se muestre a distintos usuarios y grupos recomendaciones para que las evalúen, de esta forma se podrían calcular las distintas métricas como la *Precision* y el NDCG en entornos reales.

6. Referencias

1. Schedl M, Knees P, McFee B, et al Music Recommender Systems. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook. Springer US, Boston, MA, pp 453–492 (2015)
2. Delic A, Masthoff J Group recommender systems. Springer International Publishing, Cham (2018)
3. Boratto L, Carta S State-of-the-art in group recommendation and new

approaches for automatic identification of groups. In: Kacprzyk J, Soro A, Vargiu E, et al (eds) Studies in Computational Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 1–20 (2010)
4. Dara S, Chowdary CR, Kumar C A survey on group recommender systems. J Intell Inf Syst 54:271–295 (2020)
5. Kompan M, Bielikova M Group recommendations: Survey and perspectives. Comput Informatics 33:446–476 (2014)
6. Delic A, Masthoff J Group recommender systems. Springer International Publishing, Cham (2018)
7. Group Recommender Systems - An Introduction | Alexander Felfernig | Springer. https://www.springer.com/gp/book/9783319750668. Accessed 15 Feb 2021
8. Surprise Official Documentation. https://surprise.readthedocs.io/en/stable/. Accessed 30 Jun 2021
9. Spotify API Documentation. https://developer.spotify.com/documentation/web-api/. Accessed 30 Jun 2021
10. Celma Ó Last.FM Dataset - 1K users. http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html
11. Celma O Music Recommendation and Discovery in the Long Tail. Springer (2010)
12. Pacula M A Matrix Factorization Algorithm for Music Recommendation using Implicit User Feedback (2010)
13. McNee SM, Riedl J, Konstan JA Being accurate is not enough: How accuracy metrics have hurt recommender systems. In: Conference on Human Factors in Computing Systems - Proceedings. pp 1097–1101 (2006)
14. Ibrahim KM, Royo-Letelier J, Epure E V., et al Audio-Based Auto-Tagging with Contextual Tags for Music. ICASSP, IEEE Int Conf Acoust Speech Signal Process - Proc 2020-May:16–20 (2020)
15. Sánchez-Moreno D, Zheng Y, Moreno-García MN Time-Aware Music Recommender Systems: Modeling the Evolution of Implicit User Preferences and User Listening Habits in A Collaborative Filtering Approach. Appl Sci 2020, Vol 10, Page 5324 10:5324 (2020)

Sistema para el análisis de sentimiento en dominios especializados

Alberto Montero Fernández, Juan Manuel Corchado Rodríguez, Guillermo Hernández González

Departamento de Informática y Automática, Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
El Grupo de investigación en Bioinformática, Sistemas Informáticos Inteligentes y Tecnología Educativa (BISITE)
{albertomonfdez@usal.es;jm@corchado.net;guillehg@usal.es}
https://www.usal.es/
https://bisite.usal.es/es

Resumen Se propone el diseño de un sistema de etiquetado de textos que funciona de forma distribuida mediante REST, permitiendo definir el sentimiento para cada uno de los registros. Se ha creado un corpus empleando el sistema definido, cuya naturaleza está basada en las opiniones de los estudiantes universitarios, permitiendo emplear estas para posteriormente realizar un análisis de sentimiento mediante la utilización de seis algoritmos supervisados, Naive Bayes Multinomial, Naives Bayes Bernouilli, Vecinos más Cercanos, Support Vector Machines, Long-Short Term Memory y FastText.
El estudio se realiza empleando una validación cruzada con un total de cinco particiones, debido al tamaño del corpus, y empleando las técnicas de tratamiento de texto Tf-idf y *word embbeding*, de las cuales se obtendrá un modelo con los mejores parámetros posibles para cada algoritmo empleado. Una vez realizado lo anterior, se lleva a cabo un estudio de los resultados empleando las métricas, precisión y *recall*, así como la realización de las matrices de confusión y curvas ROC del sentimiento negativo para cada uno de los modelos, ya que, es el sentimiento que mejores resultados ofrece en el corpus definido, debido al desequilibrio existente entre las clases. *abstract* environment.

Keywords: Análisis de sentimiento, procesamiento del lenguaje natural,sistema de etiquetado de textos, aprendizaje automático supervisado

1. Introducción

Las técnicas del procesamiento del lenguaje natural, ligadas a los avances del campo del *machine learning*, permiten la extracción automática de información de bloques de texto, como puede ser noticias, tweets, comentarios... Una de las aplicaciones más populares es el llamado "análisis de sentimiento", que trata de atribuir al texto una valoración de la intención comunicativa del autor en un

rango de “positivo” a “negativo”, bien de forma dicotómica, multiclase o en un rango numérico continuado.

La cantidad de corpus disponibles en Internet ha experimentado un importante aumento en los últimos años [45], gracias a redes sociales y aplicaciones de terceros que ofrecen una funcionalidad a costa de los datos proporcionados. Sin embargo, un problema habitual es que, a pesar de la existencia de modelos generalistas preentrenados, el desempeño de estos puede no ser satisfactorio en aquellos dominios alejados del usado para componer el modelo, es decir, existen necesidades específicas para las que es necesario elaborar modelos a medida, por lo que es pertinente diseñar una metodología científica en la que se enmarque el proceso. Otra importante carencia es la falta de modelos en idiomas distintos al inglés, que es la lengua imperante en la investigación.

Este trabajo se dividirá en cincos parte bien definidas: en §2 se realiza una revisión de la literatura relacionada con el análisis de sentimiento. En §3 se muestra todo el proceso para la elaboración del sistema propuesto y los distintos modelos empleados para su evaluación. En §4 se muestra los resultados obtenidos en el trabajo, seguido por §5 donde se muestran las conclusiones derivadas del proyecto y trabajos futuros posibles.

2. Estado del arte

El propósito de este apartado es realizar una revisión de la literatura relacionada con el procesamiento del lenguaje natural (PLN) y concretamente, la aplicación de este para el análisis de sentimientos. En esta revisión se busca conocer las investigaciones relacionadas con esta aplicación del PLN, particularmente los distintos modelos y dominios en los que se han aplicado.

El análisis de sentimientos es una de las áreas de investigación de mayor crecimiento en los últimos años, teniendo el artículo más citado del campo mayor repercusión que los mejores artículos en el área de investigación, también popular, como es la ingeniería de software [24, 53]. Un caso que llama la atención, puesto a que esta segunda área de investigación en 2015 tenía un total de 70.000 artículos, frente a los casi poco más de 5000 artículos que existían relacionados con el análisis de sentimientos. Así nos cuenta el artículo [53] que realiza un repaso de la evolución en el crecimiento de esta área de investigación, y como su uso es cada vez más común debido a la enorme utilidad de esta, reafirmando una vez más la popularidad del análisis de sentimientos.

Centrándonos en la utilidad del análisis de sentimientos, existen multitud de artículos relacionados con la aplicación de este en las redes sociales. En concreto, la red social Twitter, es muy empleada debido a la facilidad en la extracción de los tuits si se tiene el debido consentimiento de los usuarios. Debido a esto, se realizan númerosas investigaciones, en donde se llevan a cabo la generación de modelos genéricos preentrenados con datos de Twitter, en donde el idioma inglés es el lenguaje predominante [40, 46, 54, 60], o generando corpus para distintos dominios específicos, como es caso de las investigaciones que están relacionadas con la política [69], el cáncer [26], el clima [59] y la energía [38].

Una de las ventajas de realizar este análisis, es que permite conocer el sentimiento expresado por las personas relacionadas a estos campos de manera automatizada, y en consecuencia, realizar acciones teniendo en cuenta los resultados de este análisis, atendiendo a la necesidades reales, y por lo tanto, las necesidades de las personas implicadas en ella.

El avance debido al paso de los años , y por consiguiente, el aumento de artículos relacionados con el análisis de sentimientos, ha conllevado a la mejora de los modelos en esta técnica del PLN. Partiendo de la base de algoritmos más simples y sencillos como puede ser KNN Vecino más Cercano [15], a otros con mayor efectividad, donde encontramos el algoritmo de Naive Bayes, que es uno de los más utilizados [62] y en concreto una de sus variantes, el denominado Naive Bayes Multinomial, que es común su empleo en investigaciones relacionadas con el análisis de sentimientos, debido a sus características, que hacen de este, un algoritmo con gran rendimiento en tareas PLN [12, 31]. A menudo, se compara con otro algoritmo muy similar, el denominado Naive Bayes Bernoulli [65] ya que es interesante la comparativa de cara a mostrar los resultados.

Actualmente, aunque los anteriores algoritmos se siguen empleando en las tareas PLN, es muy común el uso de *deep learning* o aprendizaje profundo [49] para llevar a cabo esta tarea, debido a la inmensa variedad de algoritmos existentes, que facilitan el análisis empleando redes neuronales profundas, que son especialmente adecuadas para el trabajo intensivo que se realiza en estos análisis, ya que, estas redes cuentan con un número significativo de capas de unidades de procesamiento conectadas, activados por los cálculos ponderados de las neuronas precedentes, que les permite realizar la tarea con gran precisión [63]. Tanto es así, que se han realizado diversos estudios para probar la eficacia de estos, en el análisis de sentimientos [14, 56, 71] demostrando que sus resultados son muy precisos en redes como las redes neuronales recurrentes (RNN) y en concreto dentro de estas *Long-short term memory* (LSTM) [35] .

Es interesante conocer como mejorar estos algoritmos, a través del tratamiento de los textos empleados para entrenar los modelos. Tanto el mecanismo de pesados de términos *term frequency-inverse term* frequency (Tf-idf), como el de *word embedding* o incrustación de palabras, son algunas de las técnicas utilizadas para el tratamiento de los textos, que proporcionan distintos resultados en la precisión de los modelos, tal y como nos indica el artículo [29], que realiza un estudio en profundidad de los distintos algoritmos mencionados anteriormente aplicando estas técnicas, llegando a la conclusión de que la incrustación de palabras ofrece mejores resultados en el análisis de sentimientos.

Como se puede observar, es un área de investigación cada vez más llamativa y por lo tanto investigada, debido al enorme uso que se puede realizar de esta aplicación del PLN.

Sin embargo, pese al gran abanico de investigaciones que se realizan en torno a este campo, el idioma español pasa muy desapercibido a la hora de generar corpus, ya que el inglés es el lenguaje imperante. Añadido a esto, existe el problema de que los corpus predominantes están preentrenados con un número masivo de datos genéricos o de ámbito muy particular, como es el caso de aquellos modelos producidos a partir de reseñas de productos en Amazon debido a la disponibi-

lidad de valoraciones numéricas asociadas a las mismas que se pueden emplear como etiquetas [17] o tuits de la red social Twitter [13]. Esto produce que el desempeño de estos puede no ser satisfactorio en aquellos dominios alejados del usado para componer el modelo.

Por consiguiente, para mitigar la dificultad que tiene reunir corpus en otros idiomas, en este trabajo se ha realizado un sistema para facilitar el etiquetado de textos que pueda funcionar de manera distribuida mediante REST, para permitir el etiquetado del sentimiento a través de distintos algoritmos vistos anteriormente, como son Naives Bayes Multinomial o LSTM, en un conjunto de textos recogidos. La realización de este sistema solventa la problemática de la falta de corpus en dominios específicos o idiomas.

3. Metodología

En este apartado se describe la metodología definida en el estudio, distinguiendo entre la parte relativa a la elaboración del corpus y de su etiquetado en el apartado3.1 y la algoritmia para la construcción de modelos en el apartado3.2

3.1. Corpus

El eje temático que se ha seguido para generar el corpus en este caso, dado a que el sistema proporciona lo necesario para la creación de cualquier corpus, se ha basado en la idea de un estudio relacionado con la educación. En concreto, en las opiniones de estudiantes universitarios respecto a su universidad y específicamente qué opinan de cómo se imparte su grado universitario.

Para la realización del corpus, se parte de la recolección inicial de las opiniones mediante una plataforma web (desarrollada en Node.js, Vue.js y Python), el registro en esta, se realiza a través de la plataforma Twitter, como se observa en la Fig. 1 dado a que se ha buscado partir de la integración con esta red social con vistas a que la metodología pueda extenderse para recopilar corpus en el ámbito de la misma en líneas de investigación futuras.

Una vez registrados en la plataforma, los usuarios tienen habilitado un espacio para redactar la valoración con un total de 500 caracteres, como se aprecia en la Fig. 2.

En total para esta investigación han participado tres etiquetadores, que están encargados de indicar a qué sentimiento corresponde cada uno de los textos que se han ido registrando en la plataforma, como se aprecia en la Fig. 3. Esto es debido a que para generar el corpus es interesante que más de una persona etiquete los textos, para que el sentimiento definido como etiqueta tenga la mayor validez posible, ya que, al contar con más de un etiquetador y por lo tanto, otro puntos de vista, permite que los corpus generados, sean menos subjetivos y por consiguiente, también los resultados del análisis de sentimientos. Para generar el corpus de entrenamiento con una etiqueta única a partir de múltiples proporcionadas por los etiquetadores se propone un sistema basado en reglas:

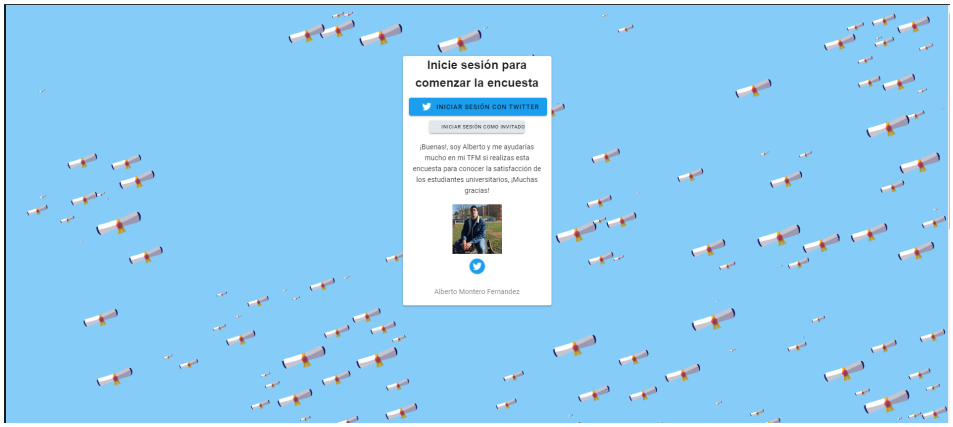


Figura 1: Registro en la plataforma Web

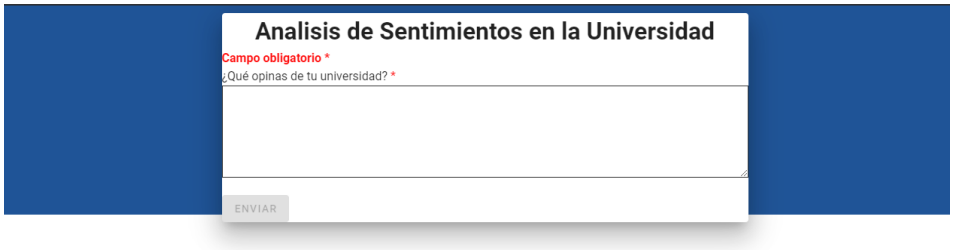


Figura 2: Casilla para rellenar la opinión mediante la web proporcionada.

1. Si el texto es clasificado por dos etiquetadores con un mismo sentimiento de tipo positivo o negativo y el tercero indica el polo opuesto, entonces se vuelve a etiquetar el texto y en caso de ocurrir nuevamente se descarta, en otro caso se etiqueta con la opción mayoritaria.
   Ejemplo: dos etiquetadores clasifican el texto como positivo y el tercero como negativo, se revisa el texto de nuevo, si es la segunda vez que ocurre se descarta. En otro caso, dos etiquetadores etiquetan el texto como positivo y el tercero como neutro, el texto se etiqueta como positivo.
2. Si el texto no es clasificado por dos o más etiquetadores con un mismo sentimiento, entonces se descarta el texto.
   Ejemplo: los tres etiquetadores expresan cada uno un sentimiento diferente para el texto, entonces el texto es descartado, ya que, para los etiquetadores el texto expresa un sentimiento ambiguo.

Finalmente, tras realizar este proceso, se define un conjunto de datos del tipo (texto, etiqueta) en formato .csv que será empleado para generar los modelos de análisis de sentimiento en los distintos algoritmos, en total el conjunto de datos empleado cuenta con un total de 102 registros y las distribución de la polaridades de las opiniones se aprecia en la Fig. 4.

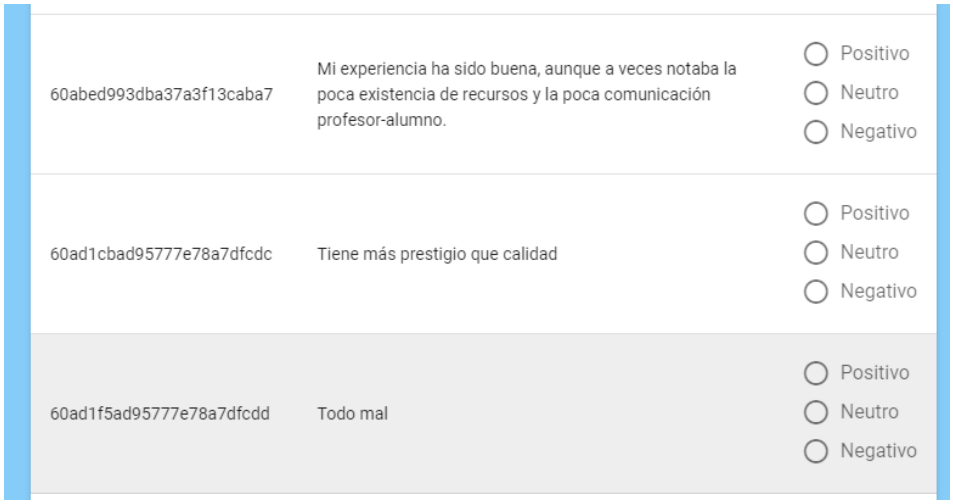


Figura 3: Pantalla que les permite a los etiquetadores clasificar los textos.

3.2. Modelos

En este apartado se trata todo lo relacionado con los modelos empleados a la hora de realizar el análisis de sentimientos del corpus previamente desarrollado en el apartado 3.1.

Se emplean en total seis algoritmos, tres de ellos se usan como enfoque base, debido a su naturaleza más sencilla, estos son Naive Bayes Multinomial, Naive Bayes Bernoulli y Vecinos más Cercanos. El estudio de estos es interesante para el corpus empleado, ya que son ampliamente utilizados en el análisis de sentimientos. Por otro lado, se emplean Support Vector Machine, Long-Short Term Memory y FastText al contrario que en los otros algoritmos, estos, previamente a la creación del modelo realizan una búsqueda de los mejor hiperparámetros de cara a obtener el mejor modelo para las técnicas Tf-idf y *word embedding*. A continuación se detalla la configuración de cada uno de ellos.

Procesamiento de los textos Para el tratamiento de los textos a la hora de emplearlos en los modelos, es necesario eliminar las palabras vacías *stop words*, que fueron introducidas por primera vez en 1958 por Hans Peter Luhn [51], un informático y experto en información que preparó el camino para la indexación automática y la recuperación de información. Fue en la Conferencia Internacional Información Científica de Washington DC donde se empleó el término *stopwords* por primera vez, debido a la técnica de indexación *Keyword-in-Context* (KWIC), que discrimina entre palabras clave y palabras que no lo son, es decir esta ultimas son las *stopwords*. Esta innovación se adoptó debido al rendimiento que ofrecia y en 1979, Van Rijsbergen publicó la segunda edición de su libro [61] en el que muestra una lista de 250 *stopwords* en inglés. Desde entonces, su uso es común para emplearlas en tareas relacionadas con el tratamiento de los textos.

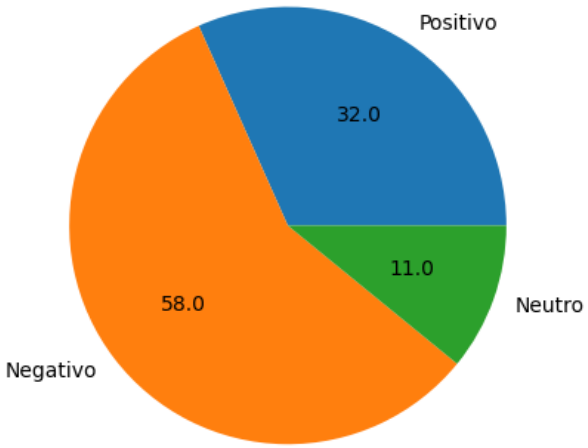


Figura 4: Distribución de las opiniones según su polaridad.

Una vez se han tratado las *stopwords* empleando las listas proporcionadas por la librería NLTK [7], se procede a eliminar los signos de puntuación, los acentos y otros signos diacríticos que pueden dificultar el proceso para hallar el sentimiento. Una vez realizado, se comprueban las palabras restantes mediante el *word cloud*, técnica de visualización sencilla pero potente para el tratamiento de textos, que muestra la palabra más frecuente con letras más grandes y más gruesas, y con diferentes colores. Cuanto menor sea el tamaño de la palabra, menor será su frecuencia de aparición en el corpus. Como se puede observar en la Fig. 5 "universidad", "profesorado", "carrera", "asignatura" o "alumno" son algunas de las palabras con mayor ocurrencia, lo cual es normal, debido a que se trata de un dominio relacionado con la educación y en concreto, la etapa universitaria.

Posteriormente, para que los algoritmos puedan manejar los textos es necesario aplicar técnicas para el tratamiento de estos. Estas técnicas permiten transformas los textos en vectores numéricos para que se puedan generar los modelos necesarios para el análisis de sentimientos, entre estas técnicas se emplean Tf-idf, es decir, la frecuencia de ocurrencia del término en la colección de documentos [58], técnica muy empleada en modelos de sentimientos debido a sus resultados [57], y *word embedding*, que se ha demostrado que captura con precisión la semántica y el contexto de las palabras, mientras que su uso en un

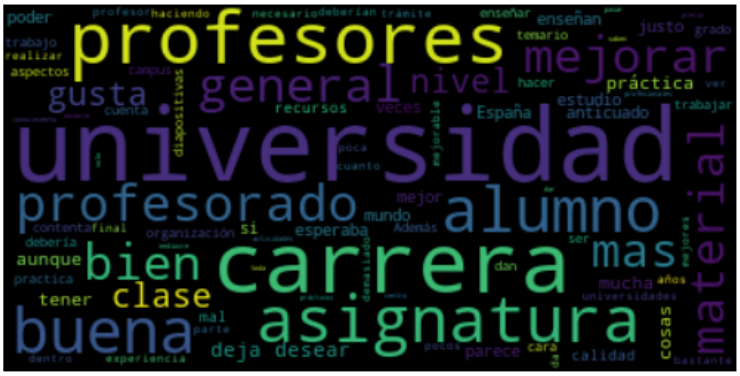


Figura 5: Word Cloud del corpus generado.

entorno de clasificación supervisado puede mejorar los modelos de sentimientos entrenados [48]. En este caso, se emplean y comparan ambas de cara a obtener el mejor clasificador en el análisis.

Tf-idf será implementado mediante la librería Scikit-learn [9] de Python, mediante su clase `TFIDFVectorizer` que permite tanto convertir una colección de documentos de texto en una matriz de recuentos de tokens y transformar una matriz de conteo en una representación tf o Tf-idf normalizada, se usa el parámetro `max_df`, cuyo valor en el rango [0.7-1.0) permite eliminar las *stopwords* contenidas en el texto y predeterminadamente se transforman todas las palabras a minúsculas, como se aprecia en siguiente código.

```
self.vectorizer = TF-IDFVectorizer(max_df=1,
                                   sublinear_tf=True,
                                   use_idf=True)
```

Por otro lado, *word embedding* será implementado mediante word2vec, empleando para ello Gensim [2] y la librería Keras de Tensorflow [6] de Python, para tokenizar las palabras. Tras eliminar las *stopwords* y signos de puntuación mediante NLTK [7], se emplea la clase `Tokenizer` encargada de vectorizar un corpus de texto, convirtiendo cada texto del conjunto de entrenamiento en una secuencia

de números enteros (cada número entero es el índice de un token en un diccionario), para después aplicar `fit_on_text` actualizando el vocabulario interno basado en una lista de textos extraídos del conjunto de datos, después se aplica `texts_to_sequences` para transformar cada texto en textos en una secuencia de números enteros. Tras esto, se realiza la llamada al método `pad_sequences`, esta función transforma una lista (de longitud `num_samples` de secuencias, en una matriz para emplearla posteriormente con el modelo construido de *word embedding*, donde se define la representación vectorial de la palabra para emplearla en los modelos, esto se realiza tanto para los datos de entrenamiento y validación.

Naive Bayes multinomial Para su implementación se emplea la funcionalidad que ofrece Scikit-learn [9] para Python. Mediante esta librería empleamos el modelo `MultinomialNB` empleando los valores por defecto del modelo, contando con el parámetro de suavizado `alpha activado`, y sin prioridad de clases para los ajustes.

Naive Bayes Bernoulli Al igual que en el caso de la sitribucion Multinomial, Bernoulli hace uso de la librería que ofrece Scikit-learn [9] para Python. En este caso se emplea el modelo `BernoulliNB` cuyos parámetros serán los valores por defecto del modelo, entre ellos están, el parámetro de suavizado `Alpha` activado, y el parámetro `class_prior` para indicar que no se dé prioridad de clases para los ajustes.

Vecinos más Cercanos Este modelo así como los dos anteriores, también emplea la librería que ofrece Scikit-learn [9] para Python. Para su implementación se hace uso del modelo `KNeighborsClassifier` empleando los valores por defecto del modelo, contando con un total de 5 vecinos para este.

Es un método sencillo que permite definir un enfoque base para esta aplicación del PLN. Ofrece una visión de una clasificación rápida y sencilla de implementar, que es interesante en el estudio de la comparativa con el resto de modelos empleados en el análisis.

SVM Para la implementación de los distintos modelos de la máquina SVM se hace uso de la librería Sklearn, se realiza una búsqueda del mejor modelo dada su precisión, teniendo en cuenta los siguientes hiperparámetros:

1. **C**: parámetro de regularización, la fuerza de la regularización es inversamente proporcional al parámetro C, este debe ser estrictamente positivo. Se emplean 1, 10, 100, 1000 C como los parámetros en la búsqueda, dado a que es interesante ver como el parámetro de regularización en la precisión de los modelos.
2. **Kernel**: tipo de kernel empleado para seleccionar el hiper plano que se usa para separar los datos en las distinta clases objetivo. Se emplean rbf,linear como kernels.

3. **Gamma**: este parámetro es empleado para los hiperplanos no lineales, es decir, el kernel rbf. Este parámetro define hasta dónde llega la influencia de un solo ejemplo de entrenamiento, con valores bajos que significan "lejos" y valores altos que significan "cerca". Los parámetros **gamma** pueden verse como la inversa del radio de influencia de las muestras seleccionadas por el modelo de soporte vectorial. Se emplean 0.001, 0.0001 como parámetros en la búsqueda del mejor modelo.

LSTM Para comprobar como rinde este modelo frente al conjunto de datos de dominio especifico generado, se hace uso de la librería Tensorflow [6], en donde se especifica el modelo LSTM y se indica la función de activación **softmax**. Para comprobar el mejor modelo dada la precisión, se hace uso del método `GridSearchCV` en donde se indican los hiperparametros que se desean modificar de cara a la generación de modelos, este proceso permite conseguir el mejor modelo definido por la precisión. Los hiperparametros empleados para generar los distintos modelos son los siguientes:

1. **Batch_size**: número entero para indicar el número de muestras por lote de cálculo. Se emplean 256, 521, 1024, 2048, 4096 muestras, ya que, es interesante conocer como se comporta con un rango de valores entre un número bajo y elevado de muestras por lote de cálculo.
2. **Epochs**: número entero para indicar el número de etapas de entrenamiento, es decir el número de iteraciones de entrenamiento sobre el conjunto de datos proporcionado. Se emplean 4, 6, 8, 10 etapas, dado a que número mayor de etapas únicamente empeoraba los resultados al emplearse un número pequeño de muestras.
3. **Optimizer**: algoritmo empleado para manejar las tasas de errores en el entrenamiento del analizadores de sentimientos, empleándose para resolver problemas de optimización minimizando la función. Se emplean Adadelta [70] y Adam [44] como optimizadores, dado a que para los modelos de PLN estos algoritmos son rápidos y ofrecen buenos resultados en los modelos de *deep learning*.

FastText Para su implementación se emplea la librería que ofrece FastText para Python. Para la optimización en los hiperparámetros del modelo se hace uso del Script *hyperparameter_ optimisation.sh* [8] procedente del libro [18], realizándose algunas modificaciones para también, contemplar las etapas como uno de los parámetros a modificar, ya que el Script solo contemplaba el *learning rate* y la dimensión. FastText emplea a nivel interno word2vec para tratar internamente los textos empleados como vectores de palabras. Por lo tanto, solo se contempla *word embedding* en el tratamiento de los textos debido a la naturaleza del propio modelo. Los hiperparametros empleados para generar los distintos modelos son los siguientes:

1. **Lr**: la tasa de aprendizaje durante las iteraciones de entrenamiento del algoritmo. Se utilizan los siguientes valores para definir el mejor modelo 0.001, 0.01, 0.1, 0.3.

2. **Epoch**: número de etapas, es decir cada una de las iteraciones que se realizan sobre el conjunto de datos. Se utilizan los siguientes valores para definir el mejor modelo 5, 10, 20.
3. **WordNgrams**: máxima longitud del n-gramas de palabras "wordNgrams": se emplea el valor por defecto.
4. **Dim**: tamaño del vector de palabras "dim". se utilizan los siguientes valores para definir el mejor modelo 10, 20 .

4. Resultados

En este apartado se muestran los resultados de la búsqueda de los mejores hiperparametros en los modelos LSTM, SVM y FastText, además, de los resultados del análisis de sentimiento en los modelos generados por ambas técnicas de tratamiento de texto, empleando las métricas, precisión y *recall*, así como el estudio de las matrices de confusión de los modelos y sus curvas ROC generadas para el sentimiento negativo.

4.1. Configuración de los modelos de SVM, LSTM y FastText

SVM Para la técnica Tf-idf se ha obtenido que entre los modelos generados, los cuales se pueden observar en la Fig. 6, que todos los modelos contienen una precisión del 57.6 % y se escoge el que contiene los siguientes hiperparametros:

1. **C**: 1.
2. **gamma**: 0.001.
3. **kernel**: rbf.

Para la técnica *word embedding* se ha obtenido que entre los modelos generados, los cuales se pueden observar en la Fig. 6, que el mejor modelo con una precisión del 65.9 % es el que contiene los siguientes hiperparametros:

1. **C**: 10.
2. **gamma**: 0.001.
3. **kernel**: rbf.

Ambas técnicas coinciden en parámetros, exceptuando el parámetro de regularización, que es menos en el modelo Tf-idf, sin embargo, como se aprecia en la Fig. 6, todos los modelos de Tf-idf tienen la misma precisión, caso que ocurre en los modelos de *word embedding* pero solo en dos de ellos, respecto a los más precisos.

LSTM Para la técnica Tf-idf se ha obtenido que entre los modelos generados, los cuales se pueden observar en la Fig. 8, que el mejor modelo con una precisión del 57.6 % es el que contiene los siguientes hiperparametros:

1. **Batch_size**: 256.
2. **Epochs**: 4.

```
Best parameters set found on development set:

{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.576 (+/-0.041) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.576 (+/-0.041) for {'C': 1, 'kernel': 'linear'}
0.576 (+/-0.041) for {'C': 10, 'kernel': 'linear'}
0.576 (+/-0.041) for {'C': 100, 'kernel': 'linear'}
0.576 (+/-0.041) for {'C': 1000, 'kernel': 'linear'}
```

Figura 6: Ranking de modelos SVM empleando Tf-idf.

3. **Optimizer**: Adadelta.

Para la técnica *word embedding* se ha obtenido que entre los modelos generados, los cuales se pueden observar en la Fig. 9, que el mejor modelo con una precisión del 50 % es el que contiene los siguientes hiperparametros:

1. **Batch_size**: 2048.
2. **Epochs**: 8.
3. **Optimizer**: Adam.

Ambas técnicas difieren en todos sus parámetros, lo cual indica que en LSTM a diferencia de SVM, si que afecta en gran medida el uso de una u otra técnica a la hora de seleccionar los mejores parámetros para generar el modelo. Al igual que en SVM, en los modelos de la técnica Tf-idf existen varios con la mejor precisión y para *word embedding* en este caso, solo un modelo que destaca por encima del resto.

FastText Una vez realizada la búsqueda del mejor modelo el resultado final y por lo tanto, el modelo con el que se trabajará en los resultados es el siguiente:

1. **Lr**: 0.3.
2. **Epoch**: 20.
3. **Dim**: 20.

Análisis de la precisión con validación cruzada Se realiza un estudio de la precisión de los modelos generados mediante la validación cruzada, técnica que permite realizar varias particiones sobre el conjunto de datos calculando la

Figura 7: Ranking de modelos SVM empleando *word embedding*.

precisión en cada una de ellas. En concreto, se emplean 5 particiones, y finalmente se realiza la media de las precisiones obtenidas en cada etapa, para hallar la precisión final considerando ambas técnicas. Con este proceso se busca realizar la comparación entre el valor de las precisiones de los distintos modelos. A continuación, en la Fig. 10 se muestran las precisiones obtenidas para ambas técnicas.

```
{'batch_size': 256, 'epochs': 4, 'optimizer': 'Adadelta'}

Grid scores on development set:

0.261 (+/-0.261) for {'batch_size': 256, 'epochs': 4, 'optimizer': 'adam'}
0.576 (+/-0.109) for {'batch_size': 256, 'epochs': 4, 'optimizer': 'Adadelta'}
0.380 (+/-0.283) for {'batch_size': 256, 'epochs': 6, 'optimizer': 'adam'}
0.359 (+/-0.543) for {'batch_size': 256, 'epochs': 6, 'optimizer': 'Adadelta'}
0.576 (+/-0.109) for {'batch_size': 256, 'epochs': 8, 'optimizer': 'adam'}
0.109 (+/-0.043) for {'batch_size': 256, 'epochs': 8, 'optimizer': 'Adadelta'}
0.304 (+/-0.348) for {'batch_size': 256, 'epochs': 10, 'optimizer': 'adam'}
0.576 (+/-0.109) for {'batch_size': 256, 'epochs': 10, 'optimizer': 'Adadelta'}
0.109 (+/-0.043) for {'batch_size': 512, 'epochs': 4, 'optimizer': 'adam'}
0.380 (+/-0.283) for {'batch_size': 512, 'epochs': 4, 'optimizer': 'Adadelta'}
0.304 (+/-0.348) for {'batch_size': 512, 'epochs': 6, 'optimizer': 'adam'}
0.261 (+/-0.261) for {'batch_size': 512, 'epochs': 6, 'optimizer': 'Adadelta'}
0.174 (+/-0.130) for {'batch_size': 512, 'epochs': 8, 'optimizer': 'adam'}
0.174 (+/-0.130) for {'batch_size': 512, 'epochs': 8, 'optimizer': 'Adadelta'}
0.359 (+/-0.543) for {'batch_size': 512, 'epochs': 10, 'optimizer': 'adam'}
0.120 (+/-0.022) for {'batch_size': 512, 'epochs': 10, 'optimizer': 'Adadelta'}
0.576 (+/-0.109) for {'batch_size': 1024, 'epochs': 4, 'optimizer': 'adam'}
0.120 (+/-0.022) for {'batch_size': 1024, 'epochs': 4, 'optimizer': 'Adadelta'}
0.380 (+/-0.283) for {'batch_size': 1024, 'epochs': 6, 'optimizer': 'adam'}
0.326 (+/-0.174) for {'batch_size': 1024, 'epochs': 6, 'optimizer': 'Adadelta'}
0.370 (+/-0.522) for {'batch_size': 1024, 'epochs': 8, 'optimizer': 'adam'}
0.261 (+/-0.261) for {'batch_size': 1024, 'epochs': 8, 'optimizer': 'Adadelta'}
0.304 (+/-0.130) for {'batch_size': 1024, 'epochs': 10, 'optimizer': 'adam'}
0.326 (+/-0.391) for {'batch_size': 1024, 'epochs': 10, 'optimizer': 'Adadelta'}
0.511 (+/-0.239) for {'batch_size': 2048, 'epochs': 4, 'optimizer': 'adam'}
0.109 (+/-0.043) for {'batch_size': 2048, 'epochs': 4, 'optimizer': 'Adadelta'}
0.163 (+/-0.152) for {'batch_size': 2048, 'epochs': 6, 'optimizer': 'adam'}
0.511 (+/-0.239) for {'batch_size': 2048, 'epochs': 6, 'optimizer': 'Adadelta'}
0.359 (+/-0.543) for {'batch_size': 2048, 'epochs': 8, 'optimizer': 'adam'}
0.250 (+/-0.239) for {'batch_size': 2048, 'epochs': 8, 'optimizer': 'Adadelta'}
0.120 (+/-0.022) for {'batch_size': 2048, 'epochs': 10, 'optimizer': 'adam'}
0.326 (+/-0.391) for {'batch_size': 2048, 'epochs': 10, 'optimizer': 'Adadelta'}
0.261 (+/-0.261) for {'batch_size': 4096, 'epochs': 4, 'optimizer': 'adam'}
0.304 (+/-0.130) for {'batch_size': 4096, 'epochs': 4, 'optimizer': 'Adadelta'}
0.315 (+/-0.152) for {'batch_size': 4096, 'epochs': 6, 'optimizer': 'adam'}
0.163 (+/-0.152) for {'batch_size': 4096, 'epochs': 6, 'optimizer': 'Adadelta'}
0.315 (+/-0.370) for {'batch_size': 4096, 'epochs': 8, 'optimizer': 'adam'}
0.576 (+/-0.109) for {'batch_size': 4096, 'epochs': 8, 'optimizer': 'Adadelta'}
0.185 (+/-0.109) for {'batch_size': 4096, 'epochs': 10, 'optimizer': 'adam'}
0.174 (+/-0.130) for {'batch_size': 4096, 'epochs': 10, 'optimizer': 'Adadelta'}
```

Figura 8: Ranking de modelos LSTM empleando Tf-idf.

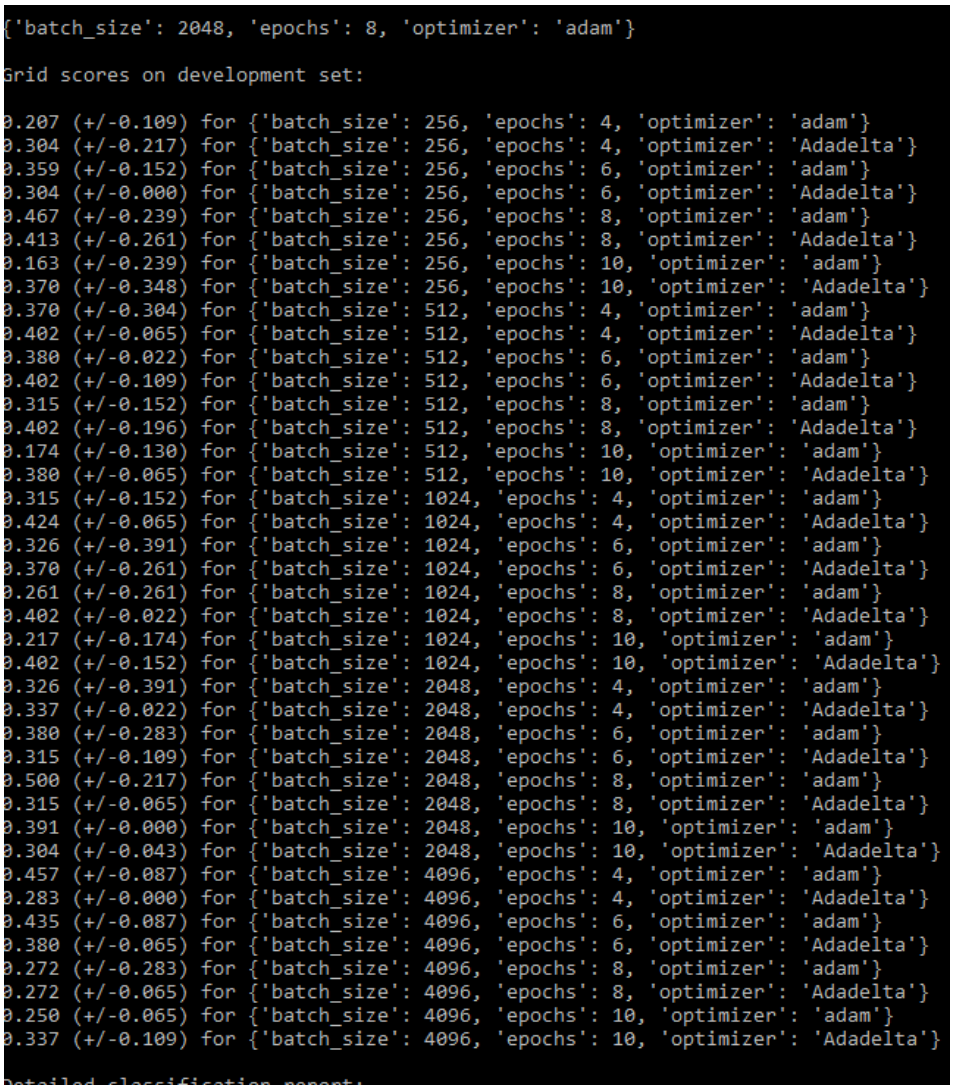


Figura 9: Ranking de modelos LSTM empleando *word embedding*.

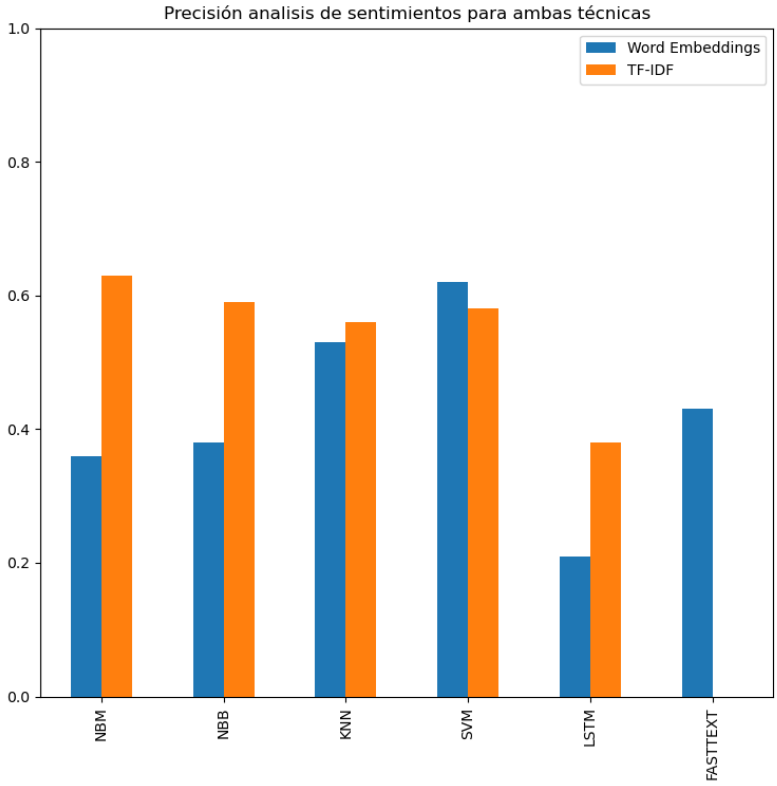


Figura 10: Precisiones en la clasificación de ambas técnicas para los modelos generados.

Se pueden observar distintos resultados interesantes en la Fig. 10. En primer lugar, los algoritmos que comentábamos anteriormente que eran más sencillo de implementar debido a su naturaleza, es decir, NBM, NBB y KNN, obtienen precisiones a la par o incluso mejores, que el resto de algoritmos, entre esto el único en conseguir superar el umbral de 50 % de precisión es SVM, que lo consigue aplicando la técnica Tf-idf que al igual que en los anterior algoritmos es la técnica que consigue mayor precisión. Por otro lado, al emplear dos distribuciones de Naive Bayes es interesante observar la comparativa entre ambas, la distribución Multinomial supera los resultados de la distribución Bernoulli en Tf-idf, mientras en *word embedding* es algo inferior, como se aprecia en la Fig. 10, indicando que, el recuento de ocurrencias que realiza el primero, funciona mejor que la comparación que realiza el segundo de la presencia de términos de forma booleana. Este caso es común en las clasificación de textos, es decir, no es especifico de este corpus definido en el estudio, otros estudios [66] contrastan estos mismos resultados.

En tanto, los algoritmos LSTM y FastText que se comentaba que estaban a la par en las tareas PLN han obtenido precisiones similares, en donde se observa como sus resultados no alcanzan el umbral del 50 % y el comportamiento en el caso de LSTM es idéntico a lo que ocurría anteriormente con la mejor precisión para la técnica Tf-idf. El pobre resultado de estos dos algoritmos indica que se confirma lo que se sospechaba y es que debido al reducido vocabulario del corpus, tanto LSTM por ser de tipo *deep learning*, como FastText por su naturaleza, obtienen malos resultados.

Matrices de confusión A continuación, para observar más detalladamente el corpus, se observan las matrices de confusión de los modelos generados, donde las columnas pertenecen a las predicciones y las filas a los valores reales, permitiendo observar exhaustivamente que ocurre con cada clase, ya que es interesante de cara a observar el comportamiento de los modelos frente al corpus, que recordamos tenia un número de registro por cada clase desequilibrado.

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	0	6	26
Clase real negativo	0	58	0
Clase real neutro	10	0	1

Tabla 1: Matriz de confusión perteneciente al algoritmo **NBM** para la técnica **Tf-idf**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	0	6	26
Clase real negativo	0	19	39
Clase real neutro	0	0	11

Tabla 2: Matriz de confusión perteneciente al algoritmo **NBM** para la técnica **word embedding**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	6	10	16
Clase real negativo	0	53	5
Clase real neutro	0	11	0

Tabla 3: Matriz de confusión perteneciente al algoritmo **NBB** para la técnica **Tf-idf**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	0	5	27
Clase real negativo	4	42	12
Clase real neutro	5	6	0

Tabla 4: Matriz de confusión perteneciente al algoritmo **NBB** para la técnica **word embedding**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	22	0	10
Clase real negativo	0	42	16
Clase real neutro	2	9	0

Tabla 5: Matriz de confusión perteneciente al algoritmo **KNN** para la técnica **Tf-idf**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	15	11	6
Clase real negativo	5	48	5
Clase real neutro	5	3	3

Tabla 6: Matriz de confusión perteneciente al algoritmo **KNN** para la técnica **word embedding**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	0	0	32
Clase real negativo	0	58	0
Clase real neutro	10	1	0

Tabla 7: Matriz de confusión perteneciente al algoritmo **SVM** para la técnica **Tf-idf**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	5	0	27
Clase real negativo	0	58	0
Clase real neutro	11	0	0

Tabla 8: Matriz de confusión perteneciente al algoritmo **SVM** para la técnica **word embedding**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	3	3	26
Clase real negativo	0	35	20
Clase real neutro	7	4	0

Tabla 9: Matriz de confusión perteneciente al algoritmo **LSTM** para la técnica **Tf-idf**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	0	0	32
Clase real negativo	0	5	53
Clase real neutro	0	0	11

Tabla 10: Matriz de confusión perteneciente al algoritmo **LSTM** para la técnica **word embedding**

	Clase predicha positivo	Clase predicha negativo	Clase predicha neutro
Clase real positivo	18	0	14
Clase real negativo	10	10	38
Clase real neutro	0	0	11

Tabla 11: Matriz de confusión perteneciente al algoritmo **FastText** para la técnica **word embedding**

Como se puede observar en las tablas anteriores, las cuales pertenecen a las matrices de confusión de los modelos generados para cada algoritmo, se obtiene, en primer lugar, que la mayoría de los modelos confunden las clases de sentimiento neutro, con el sentimiento positivo, esto puede deberse en parte, al número de registros pertenecientes a cada clase y como los textos neutros y positivos son similares en sus palabras. En algunos casos, ocurre lo mismo con el sentimiento neutro y negativo, esto se debe a que algunas de las opiniones neutras emplean tanto palabras positivas, como negativas y por lo tanto, puede llevar al modelo a crear falsos positivos.

La clase de sentimiento negativo es el caso de estudio más interesante, debido a la precisión que se obtiene en ella, siendo esto causado por lo mencionado anteriormente, nos referimos al número de casos que se obtienen de esa clase al realizar el corpus, ya que predomina con 58 registros de 102 como se observaba en §3, y por lo tanto, esto provoca que sea la clase con mayor precisión, de igual manera, para mostrar esto se hace uso de la métrica *recall*, que permite observar los verdaderos positivos (vp), frente a los falsos negativos (fn) mediante $tp/(vp + fn)vpfn$, en la Fig. 11 se puede observar la métrica mencionada para la clase de sentimiento negativo.

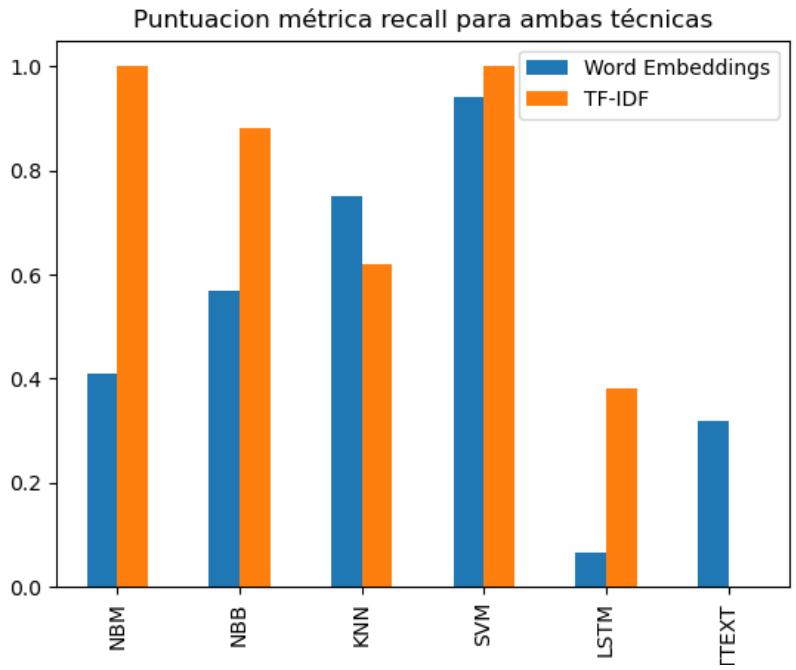


Figura 11: Puntuación de la métrica *recall* para el sentimiento negativo de ambas técnicas para los modelos generados.

Se observa como la precisión del sentimiento negativo es muy elevada, en concreto se aprecia el caso de los modelos SVM para ambas técnicas que indica que la mayor parte de verdaderos positivos pertenecen a la clase de sentimiento negativo, al igual que ocurre con el modelo NBM y NBB de la técnica Tf-idf, debido a esto, la precisión obtenida anteriormente mostrada en el gráfico es elevada para ellos, ya que el rendimiento con esta clase es muy alto.

Por otro lado, los modelos de LSTM y FastText, al igual que NBM para la técnica *word embedding* tienen una precisión en la clasificación de sentimiento negativo muy baja y por lo tanto, debido a este baja precisión en el sentimiento que predomina en el número de registros, se puede explicar en el caso de LSTM su baja precisión. Por otra parte, llama la atención como FastText sí que realiza una clasificación con una precisión del 43 %, que se debe al número de aciertos en las dos clases restantes, algo impresionante, teniendo en cuenta el número reducido de registros pertenecientes a estas clases. Una posible explicación, es que a la hora de observar su matriz de confusión en la Tab. 11, se aprecia una buena clasificación del sentimiento positivo respecto al resto de modelos. El modelo KNN en la técnica *word embedding* parece ser el que mejor precisión ofrece en cuanto equilibrio de clases, ya que su precisión con la clase de sentimiento negativo no es tan alta, en comparación a SVM. Como se observa en la Fig. 10, SVM es la que mayor precisión obtiene para ambas técnicas a la hora de predecir el sentimiento negativo.

Por lo tanto, se ha definido un análisis de sentimiento que permite reconocer con bastante precisión aquellos textos que denotan una actitud negativa. Para realizar una observación más precisa de la clase que expresa el sentimiento negativo, se dispone de la curva ROC de cada modelo para observar su comportamiento.

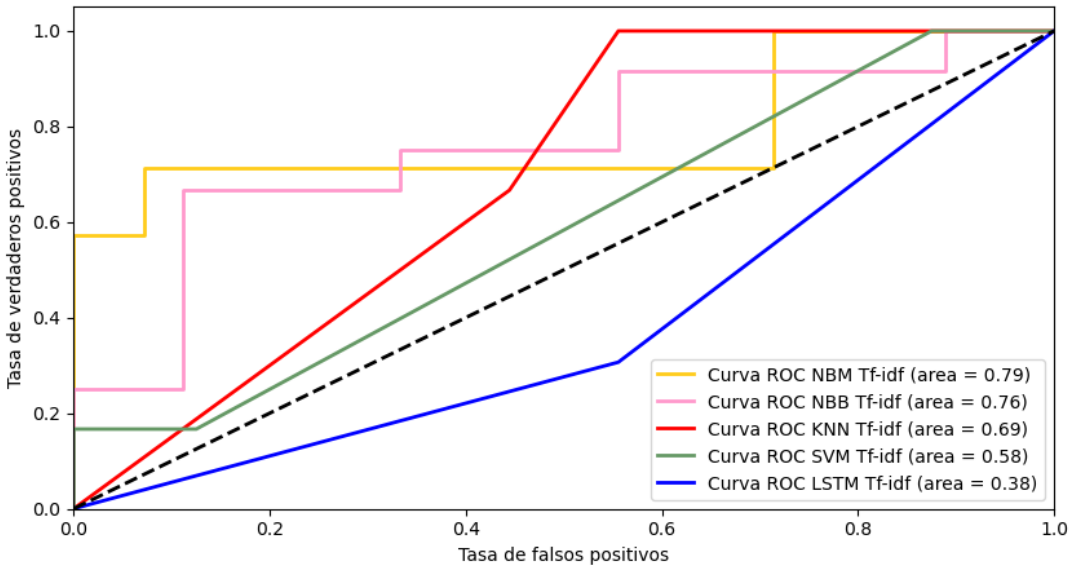


Figura 12: Curva ROC del sentimiento negativo empleando Tf-idf.

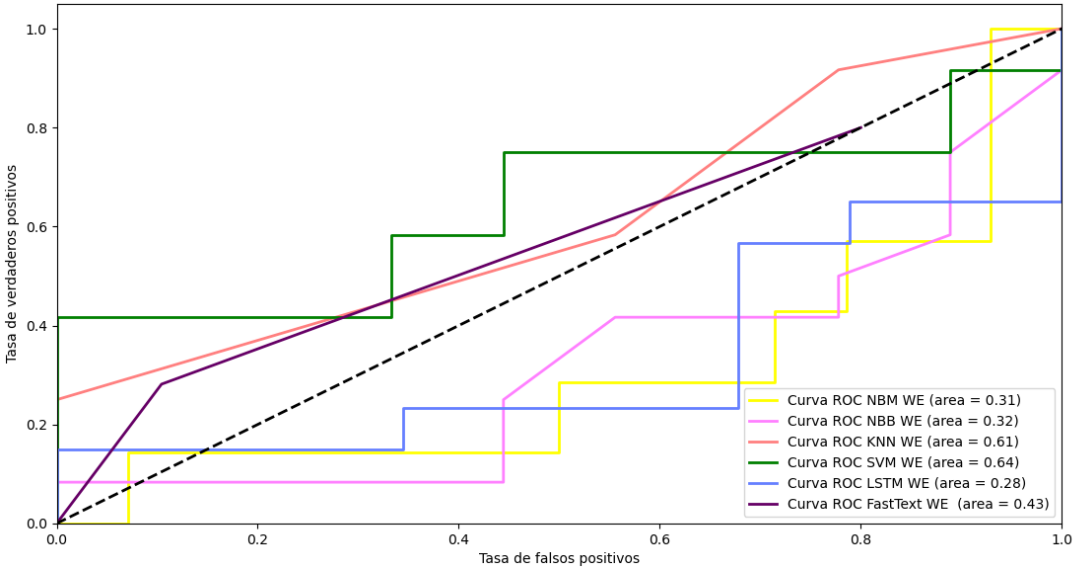


Figura 13: Curva ROC del sentimiento negativo empleando *word embedding*.

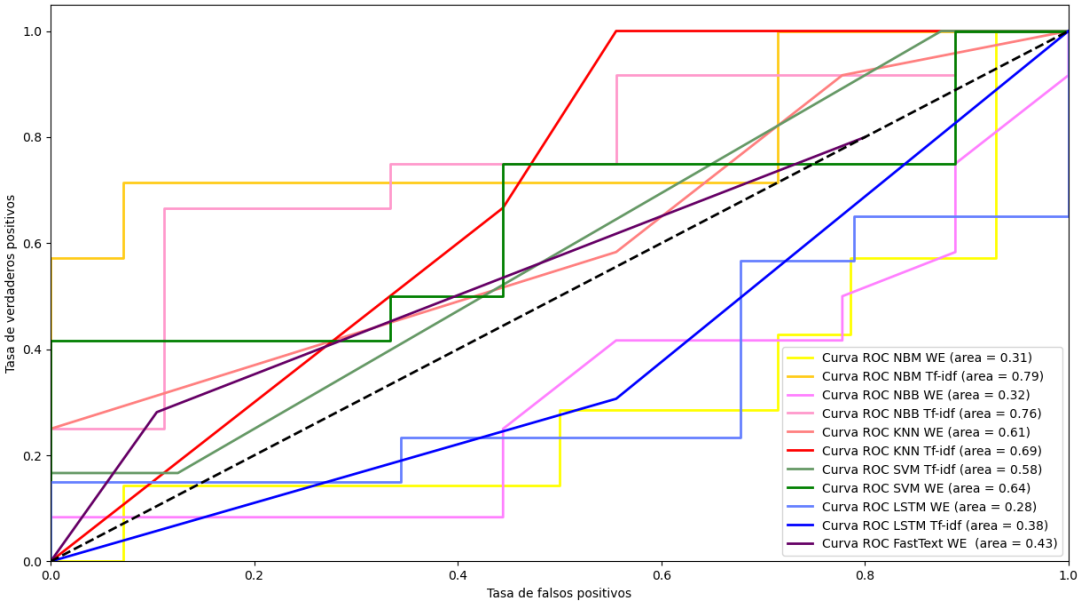


Figura 14: Curva ROC del sentimiento negativo empleando ambas técnicas.

Se confirma en las curvas ROC generadas, las cuales permiten visualizar gráficamente el comportamiento para la clase de sentimiento negativo y enfrentar los verdaderos positivos contra los falsos positivos, visualizando así, lo ya mencionado anteriormente con la métrica *recall*, que la mayor parte de los algoritmos, exceptuando los ya comentados en el apartado anterior, son capaces de reconocer con exactitud los textos que expresan un sentimiento negativo, como se observa en la FIG. 14, en donde destacan aquellos modelos que aplican la técnica Tf-idf, entre los más destacados se encuentran, NBM, NBB y KNN, superando así a la otra técnica de tratamiento de texto empleada.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

En esta investigación se buscaba definir una metodología y herramientas para solventar la problemática de corpus genéricos predefinidos que se alejaban del dominio específico deseado, además de la necesidad de corpus en otros idiomas distintos al inglés, ya que es el predominante en este tipo de investigaciones.

Por lo tanto, se ha propuesto un método para definir un corpus mediante el etiquetado de textos a través de un sistema definido, empleando una aplicación web y un conjunto de etiquetadores humanos (en este caso tres etiquetadores), encargados de clasificar cada uno de los textos registrados en el sistema con el sentimiento positivo, negativo o neutro, para posteriormente, tratarlos mediante un conjunto definido de reglas y obtener como resultado un fichero en formato CSV que pueda ser utilizados en modelos de análisis de sentimientos.

En este estudio se ha definido un corpus de dominio educativo en donde se han obtenido un total de 102 registros con las opiniones de estudiantes universitarios que han aceptado colaborar en la investigación. Una vez definido el corpus, se ha realizado un estudio del análisis de sentimientos de este, empleando un total de seis algoritmos (Naive Bayes Multinomial, Naive Bayes Bernoulli, KNN Vecinos más Cercanos, SVM, LSTM y FastText) y distintas técnicas de tratamiento de textos (Tf-idf, *word embedding*) para mostrar la precisión conseguida por cada uno de los modelos generados, de donde se han obtenido las siguientes conclusiones:

1. La distribución Multinomial en la técnica Tf-idf es el mejor modelo de ambas distribuciones bayesianas en el análisis realizado para el corpus definido.
2. FastText funciona con una precisión pobre si el vocabulario definido es muy reducido, debido a que es necesario un corpus mucho mayor para que realice resultados precisos.
3. Se confirma que, al igual que ocurría con FastText, el algoritmo de tipo aprendizaje profundo LSTM debido a su naturaleza, ofrece resultados muy poco precisos debido a tamaño reducido de la muestra.
4. SVM obtiene resultados similares respecto a la precisión de sus resultados para ambas técnicas, ofreciendo dos modelos que funcionan bien con el corpus.

5. Algoritmos sencillos y rápidos son más eficaces que los complejos, debido a las características del corpus, siendo el algoritmo NBM el modelo óptimo para el corpus definido.
6. La técnica de tratamientos de textos Tf-idf ofrece mejores modelos en el análisis de sentimientos para este caso en particular.
7. El sentimiento Neutro es el que más problemas deriva en función de la precisión obtenida respecto a los modelos generados, esto se debe al número reducido de ejemplos que se tiene de él, siendo el que menor número de registros tiene de las tres clases definidas.

La investigación, ha cumplido con proporcionar un metodología y herramientas para poder realizar análisis de sentimientos sobre el corpus definido, en el idioma español, caso que no ocurría en otras investigaciones al emplear corpus genéricos relacionados con otros dominios alejados de su propuesta.

5.2. Trabajo futuro

A continuación, se muestra posibles lineas de investigación futuras para realizar un trabajo mas completo partiendo de lo realizado hasta ahora:

1. Realizar un corpus que comprenda un mayor número de registros, creando un conjunto de datos que ofrezca mejores resultados respecto a la precisión de los modelos para crear un equilibrio en todas las clases.
2. Incluir un mayor número de etiquetadores para el etiquetado de los textos estudiando con herramientas estadísticas la concordancia entre sus opiniones.
3. Añadir más modelos al estudio del análisis y realizar una tercera técnica de tratamiento de textos, que consista en un híbrido de las dos propuestas.

Referencias

1. Definición curva roc y auc. `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419`. [Online; accessed 01-July-2021].
2. Gensim library. `https://radimrehurek.com/gensim/`. [Online; accessed 01-July-2021].
3. Imagen curva roc. `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419`. [Online; accessed 06-July-2021].
4. Imagen knn. `https://pythondiario.com/2018/01/introduccion-al-machine-learning-9-k.html`. [Online; accessed 06-July-2021].
5. Imagen svm. `https://ichi.pro/es/explicacion-de-la-maquina-de-vectores-de-soporte-svm-9774310469091` [Online; accessed 06-July-2021].
6. Keras library. `https://keras.io/`. [Online; accessed 01-July-2021].
7. Nltk library. `https://www.nltk.org/`. [Online; accessed 01-July-2021].
8. Script fasttext. `https://github.com/PacktPublishing/fastText-Quick-Start-Guide/blob/master/chapter2/hyperparameter_optimisation.sh`. [Online; accessed 01-July-2021].

9. Sk-learn library. `https://scikit-learn.org/`. [Online; accessed 01-July-2021].
10. Wikipedia curva roc. `https://es.wikipedia.org/wiki/Curva_ROC`. [Online; accessed 01-July-2021].
11. word2vec. `https://code.google.com/archive/p/word2vec/`. [Online; accessed 01-July-2021].
12. Muhammad Abbas, K Ali Memon, A Aleem Jamali, Saleemullah Memon, and Anees Ahmed. Multinomial naive bayes classification model for sentiment analysis. *IJCSNS*, 19(3):62, 2019.
13. Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on language in social media (LSM 2011)*, pages 30–38, 2011.
14. Qurat Tul Ain, Mubashir Ali, Amna Riaz, Amna Noureen, Muhammad Kamran, Babar Hayat, and A Rehman. Sentiment analysis using deep learning techniques: a review. *Int J Adv Comput Sci Appl*, 8(6):424, 2017.
15. N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.
16. T.L. Berg and D.A. Forsyth. Animals on the web. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1463–1470, 2006.
17. Aashutosh Bhatt, Ankit Patel, Harsh Chheda, and Kiran Gawande. Amazon review classification and sentiment analysis. *International Journal of Computer Science and Information Technologies*, 6(6):5107–5110, 2015.
18. J. Bhattacharjee. *FastText Quick Start Guide: Get Started with Facebook's Library for Text Representation and Classification*. Packt Publishing, 2018.
19. B. Bhavitha, Anisha Rodrigues, and Niranjan Chiplunkar. Comparative study of machine learning techniques in sentimental analysis. pages 216–221, 03 2017.
20. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2016.
21. Felipe Bravo-Marquez, Marcelo Mendoza, and Barbara Poblete. Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, 69:86–99, 2014.
22. Gerd Brewka. Artificial intelligence—a modern approach by stuart russell and peter norvig, prentice hall. series in artificial intelligence, englewood cliffs, nj. *The Knowledge Engineering Review*, 11(1):78–79, 1996.
23. Xue Chen and Chengjin Xu. Disturbance pattern recognition based on an alstm in a long-distance phi-otdr sensing system. *Microwave and Optical Technology Letters*, 62(1):168–175, 2020.
24. S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.
25. Vapnik V. Cortes C. Support-vector networks, 1995. `https://link.springer.com/content/pdf/10.1007/BF00994018.pdf`.
26. W. Christian Crannell, Eric Clark, Chris Jones, Ted A. James, and Jesse Moore. A pattern-matched twitter analysis of us cancer-patient sentiments. *Journal of Surgical Research*, 206(2):536–542, 2016.
27. Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer, 2008.
28. Novelty Octaviani Faomasi Daeli and A. Adiwijaya. Sentiment analysis on movie reviews using information gain and k-nearest neighbor. 2020.
29. Nhan Cach Dang, María N. Moreno-García, and Fernando De la Prieta. Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3), 2020.

30. Hai Ha Do, PWC Prasad, Angelika Maag, and Abeer Alsadoon. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications*, 118:272–299, 2019.
31. Arif Abdurrahman Farisi, Yuliant Sibaroni, and Said Al Faraby. Sentiment analysis on hotel reviews using multinomial naïve bayes classifier. *Journal of Physics: Conference Series*, 1192:012024, mar 2019.
32. Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. Multilingual language processing from bytes, 2016.
33. David Hand and Keming Yu. Idiot's bayes: Not so stupid after all? *International Statistical Review*, 69:385 – 398, 05 2007.
34. Has¸im Sak Heiga Zen. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43266.pdf`.
35. Schmidhuber J. Hochreiter S. Long short-term memory. neural comput. 1997. `https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory`.
36. David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
37. Mohammad Rezwanul Huq, Ahmad Ali, and Anika Rahman. Sentiment analysis on twitter data using knn and svm. *International Journal of Advanced Computer Science and Applications*, 8(6), 2017.
38. Victoria Ikoro, Maria Sharmina, Khaleel Malik, and Riza Batista-Navarro. Analyzing sentiments expressed on twitter by uk energy company consumers. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 95–98, 2018.
39. Md. Sanzidul Islam, Sadia Sultana Sharmin Mousumi, Sheikh Abujar, and Syed Akhter Hossain. Sequence-to-sequence bangla sentence generation with lstm recurrent neural networks. *Procedia Computer Science*, 152:51–58, 2019. International Conference on Pervasive Computing Advances and Applications- PerCAA 2019.
40. Anurag P. Jain and Vijay D. Katkar. Sentiments analysis of twitter data using data mining. In *2015 International Conference on Information Processing (ICIP)*, pages 807–810, 2015.
41. Hamed Jelodar, Yongli Wang, Rita Orji, and Shucheng Huang. Deep sentiment classification and topic discovery on novel coronavirus or covid-19 online discussions: Nlp using lstm recurrent neural network approach. *IEEE Journal of Biomedical and Health Informatics*, 24(10):2733–2742, 2020.
42. Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.
43. Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification, 2016.
44. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
45. Akshi Kumar and Arunima Jaiswal. Systematic literature review of sentiment analysis on twitter using soft computing techniques. *Concurrency and Computation: Practice and Experience*, 32(1):e5107, 2020. e5107 CPE-18-1167.R1.
46. Monu Kumar and Anju Bala. Analyzing twitter sentiments through big data. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2628–2631, 2016.

47. Francisco Lacueva, J. Vea-Murguía, Rafael del Hoyo-Alonso, and Rosa María Salas. Fasttext as an alternative to using deep learning in small corpus. 09 2017.
48. Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. *31st International Conference on Machine Learning, ICML 2014*, 4, 05 2014.
49. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
50. Yang Li, Ting Liu, Jing Jiang, and Liang Zhang. Hashtag recommendation with topical attention-based lstm. Coling, 2016.
51. Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.
52. Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 10 2013.
53. Kuutila M. Mäntylä M.V., Graziotin D. The evolution of sentiment analysis—a review of research topics, venues, and top cited papers, 2018.
54. Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326, 2010.
55. S. Madeh Piryonesi and Tamer E. El-Diraby. Role of data analytics in infrastructure asset management: Overcoming data size and quality problems. *Journal of Transportation Engineering, Part B: Pavements*, 146(2):04020022, 2020.
56. Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49, 2016. New Avenues in Knowledge Bases for Natural Language Processing.
57. Wahyu Prabowo and Fitriani Azizah. Sentiment analysis for detecting cyberbullying using tf-idf and svm. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 4, 12 2020.
58. Shahzad Qaiser and Ramsha Ali. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181, 07 2018.
59. Jun Qian, Zhendong Niu, and Chongyang Shi. Sentiment analysis model on weather related tweets with deep neural network. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, ICMLC 2018, page 31–35, New York, NY, USA, 2018. Association for Computing Machinery.
60. Adyan Marendra Ramadhani and Hong Soon Goo. Twitter sentiment analysis using deep learning methods. In *2017 7th International Annual Engineering Seminar (InAES)*, pages 1–4, 2017.
61. C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, USA, 2nd edition, 1979.
62. Irina Rish. An empirical study of the naïve bayes classifier. *IJCAI 2001 Work Empir Methods Artif Intell*, 3, 01 2001.
63. Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
64. B.W. Silverman and M. C. Jones. E. fix and j.l. hodges(1951): an important contribution to nonparametric discriminant analysis and density estimation. *International Statistical Review*, 57(3):233–247, 1989.
65. Gurinder Singh, Bhawna Kumar, Loveleen Gaur, and Akriti Tyagi. Comparison between multinomial and bernoulli naïve bayes for text classification. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pages 593–596, 2019.

66. Gurinder Singh, Bhawna Kumar, Loveleen Gaur, and Akriti Tyagi. Comparison between multinomial and bernoulli naïve bayes for text classification. pages 593–596, 04 2019.
67. K et al SRIPATH ROY. Student career prediction using advanced machine learning techniques, 2021.
68. Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.
69. A. Tumasjan, T.O. Sprenger, P.G. Sandner, and I.M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment, 2010. cited By 51.
70. Matthew Zeiler. Adadelta: An adaptive learning rate method. 1212, 12 2012.
71. Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

Systems based on Artificial Intelligence applied to urban mobility problems

Benjamin Begic, María Belén Pérez Lancho, Sara Rodríguez González, Dr. Roberto Casado Vara

Departamento de Informática y Automática, Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
{bbegic,lancho,srg,rober}@usal.es
https://www.usal.es/

Resumen The aim of this study is to understand the diffusion of systems based on artificial intelligence within mobility, especially urban areas in 2021. The research has led to understand how pathfinding is dominated by the Graph-Based Approach for importing maps and by two main algorithms for the calculation of optimal paths, the A* and Dijkstra algorithm.
Since the early 2000s, these algorithms have been the subject of study and evolution in some variants, often compared with other strategies for researching optimal paths, but they still manage to provide very satisfactory performances. For this reason, combined with the growing need to streamline mobility through the reduction of private vehicles and new communication technologies, the research focused on 4 major areas: interaction between road users for a global pathfinding strategy that reduces traffic, solving problems related to car sharing and dispatchimg by mixing pathfinding algorithms with time windows, creating new infrastructures or new methods of using existing ones, and obviously researching new pathfinding algorithms.

Keywords: *delivery, dispatching, pathfinding algorithms, route planning, urban areas, maps*

1. INTRODUCTION

The first studies on pathfinding algorithms date back to the early 50s but a strong push was born with robotics in the early 90s. In this period the implementation of the Graph-Based Approach has allowed to model maps and implement the pathfinding algorithms even with the limited capacity of resources of that time.

The big turning point took place in the mid-2000s thanks to the spread of the internet and smartphones, and in particular with the need for users to use navigation systems that are more intuitive than paper maps. This availability of new technologies has pushed the scientific world to deepen studies in this area, with simulations and tests, implementation of new algorithms and development

of existing ones and, as a result, the solution of the pathfinding problem is dominated by two main algorithms:

- A* Algorithm
- Dijkstra Algorithm.

These algorithms have the great advantage of being reliable, of exploring space and always finding the shortest path if it exists, they are easily implemented with a Graph-Based Approach and several variants have been already developed. However A* performs slightly better and I found its implementation more interesting and for this reason, I decided to learn more about it.

Nowadays we are facing a further evolution in the resolution of pathfinding problems; thanks to the new technologies under development and the superior capacity in managing data through communication systems, the attention of researchers has shifted from solving pathfinding problems to solving macro problems, such as global management of mobility or improvement of car-sharing and dispatching services thanks to pathfinding. After a thorough reading of the research carried out in the last 5 years, 4 macro-areas of development have been identified in systems based on A.I.:

- reaserching for a global pathfinding strategy to reduces traffic
- multi-pathfinding problems with a time-window constraint
- research and implementation of new pathfinding algorithms
- creating new infrastructures or new methods of using the existing ones.

The aim of this study is the understanding of pathfinding algorithms, an application of one of them and the analysis of the evolution of pathfinding applications in mobility problems in urban areas. The structure of this work will follow with the description of the Graph-Based Approach as it is the basis on which all the algorithms are developed, the description of the A* algorithm, an implementation of the same, the in-depth analysis of research in the field of mobility in urban areas and, finally, the conclusions.

2. Graph-Based Approach

A **Graph G** = (V, Γ), as it is possible to see in Figure 1, is a set of of nodes (vector [V]) connected by arches (vector [Γ]) that have numeric weights, that represent for example a movement costs, attached to them. The biggest advatage is that a a grid can be viewed as a special case of a graph; this is fundamental to allow the implementation of the two biggest algorithm used for path-finding, the A* and Dijkstra algorithms. Scientists keep using grids in pathfinding problems because they're easy to use, and works perfectly with the path searching algorithms.

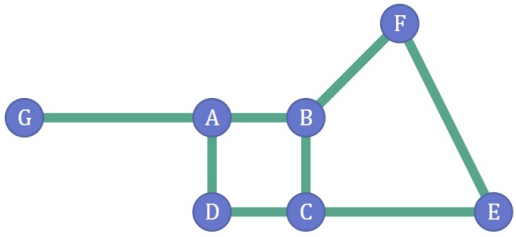


Figura 1: Example of a graph **G**

A graph-based pathfinding algorithm needs to know what the nodes are and also which locations are connected to which other ones. Obviously, building a graph, the scientist knows much more information about it, like the coordinates of the nodes or the exact shape of the arches, but the A* and Dijkstra doesn't really required this deep knowledge. This is a huge limits that slow down the implementation of the algorithm and for this reason an evolution of A* and other algorithms have been proposed to speed-up the searching process.

2.1. Grids

When I'm working with a Graph-Based Approach, it is fundamental to convert the graph into its visualization as a grid, this because my vector of nodes are distributed in the space and they are connected between them with the arches. How this connection is done is the most important part because having a good and regular shape of the grid helps to run the algorithm; the ideal scenario is that every node could have 4 connections (North, East, South, West) and the arches are straight lines as it possible to see in Figure 2.

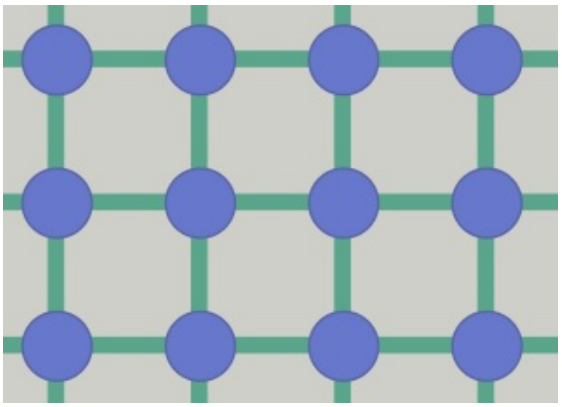


Figura 2: Example of a grid of a graph

This configuration is very advantageous because it is easy to know the **neighbors** nodes, which are the other nodes connected to node considered by an edge. Another important feature is that the algorithms will apply, for each arc, a weight or cost function depending from the length of the arc; this configuration allows to speed-up the calculation because the possible direction are just four and the lengths are multiples of a basic chosen unit. For example if I allows also connections to the diagonals, this means I would have 8 possible connection for every node and I should assign different cost functions to the cross direction and the diagonal direction because moving across the diagonal would generate a larger path.

2.2. Graph arches variants

In graph theory, there are several ways to handle arches:

- An **undirected** graph has edges that go in both directions [A↔B].
- A **directed** graph has arches that can go one direction but not the other[A→B]
- A **multigraph** can have multiple edges between the same nodes, the node A is connected with node B with different arches, not just one.

An interesting quality is that it is possible to add a weight to an arch; for example the arc's weight could be proportional to his length or I can choose to prefer using arches corresponding to suburbans road than urban ones assigning a higher weight value to the second ones. The propriety of before allows me to implement obstacles inside a grid, simply setting the weight of the arc equal to infinite.

2.3. RoadMaps

This work aim to solve path searching problems in an urban environment; this means that a map of a city should be loaded and preprocessed in the following way:

- road intersections becomes the nodes of the grid
- roads becomes the arches connecting nodes.

A pathfinding algorithm will search for the shortest road between two nodes, which means two intersections; if the starting point or the destination point are not intersections, a simple way to solve the problem is to add these points as nodes inside the grid. It is important to remove these nodes after using the algorithm, because in realty those intersections does not exist.

In [9] there is a very smart technique for importing a map; instead of considering every single road as it is in real life, they made an approximation of it, in particular the simplify the diagonal sides of the map with a combination of straight in order to work with the simplest grid possible, the one that consider only 4 possible directions, as it is possible to observe in Figure 3

Figura 3: Grid of city created from a real map [9]

3. A* Algorithm

It is one of the most implemented pathfinding algorithm because it is relatively easy to understand and implement [19][20]; from the starting node, the algorithm assign a cost for moving in each direction, as it is possible to see in the Figure 4 (left), where moving vertically or horizontally has a cost of 10, and in diagonal has a cost of 14.

This algorithm is based on reducing the total cost to reach the point B from the point A. For each node, it calculates three costs, that are reported in Figure 4 (right):

- **The G cost:** it is the distance from the starting node. It represents the cost to reach the considered node from the starting point; in this case it is 2 diagonals far away from A, this means that the cost is 2*14 = 28.
- **The H cost:** it is the distance from the end node (Heuristic cost). It represents the cost to reach the end point starting from the considered node; in this case it is 1 diagonals far away from B, this means that the cost is 1*14 = 14.
- **The F cost** $=Gcost + Fcost$. It represents the total cost of the considered node if we follow that route to reach the end point B from the starting point A.

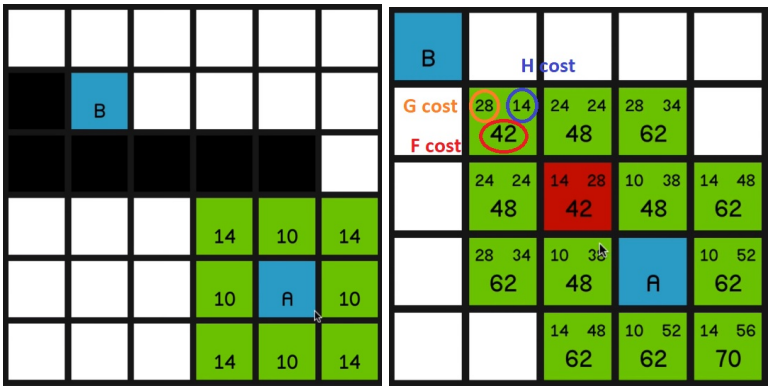


Figura 4: G cost, H cost and F cost associated to a node of a graph in A* algorithm

The algorithm would find the shortest route considering a path with nodes that has the smaller F_{cost}.

3.1. Dijkstra Vs A* Algorithm

Dijkstra's Algorithm [17],[18],[21] works almost as the same way of A* algorithm but with an important difference, Dijkstra explorate the space by visiting vertices in the graph beginning with the object's starting point. Later it repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined. It expands outwards from the starting point until it reaches the goal, the destination point. This is very time consuming algorithm compared to A* that explore the space searching a path in the direction of the destination point. This is very clear after analyzing the behaviour of the two algorithm in the presence of a concave obstacle (Figure 5).

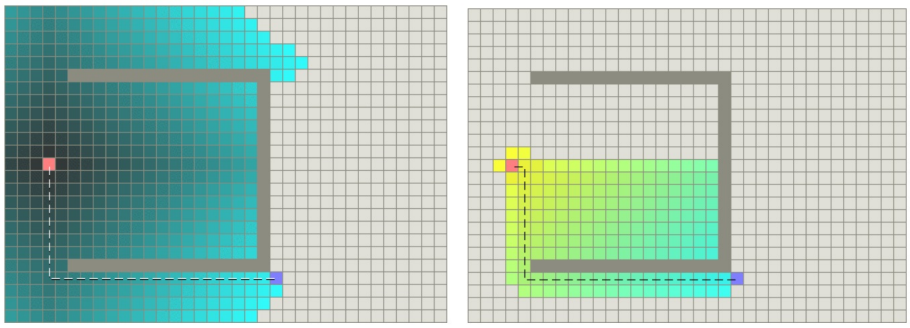


Figura 5: Dijkstra vs. A* Algorithm on the same obstacle

As it is possible to see from Figure 5, the Dijkstra Algorithm (left) tries to expand widely in all the vertices considering also the space far from the destination point that finding a shortest path in that direction would be statistically quite impossible. Meanwhile A* Algorithm would expand the exploration towards the goal, having at the same time a good knowledge of the space and the obstacles in it and later calculating the shortest path.

3.2. Heuristics for Grid Maps

On a grid, there are well-known heuristic distance functions to implement in order to calculate the G_{cost} and the h_{cost}, in particular they depends from the movements allowed by the grid:

- On a square grid that allows 4 directions of movement, it is recommended **Manhattan distance** (L_1).
- On a square grid that allows 8 directions of movement, it is recommended **Diagonal distance** (L_∞).
- On a square grid that allows any direction of movement, it might or might not be used the **Euclidean distance** (L_2). If A* is finding paths on the grid but they are allowed movement not on the grid, it is better to consider other representations of the map.
- On a hexagon grid that allows 6 directions of movement, it is recommended **Manhattan distance** adapted to hexagonal grids.

Working with maps it is always recommended to fix a unit of measurement like distance or time; for example fixing a unity of distance equal to 500 meters, if a segment road is 3 kms, the heuristic cost associated to cover this arc, the distance between the two nodes defining this arc, is $3000/500$ m $= 6$.

4. Implementation of pathfinding algorithms in new technologies

In the last five years the researchers has focused their attention in solving new problems using path finding algorithms because the A* and Dijkstra Algorithms have proven their efficiency over the years. For this reason, and thanks to the development of new technologies, the scientific world focus their attention on implementing those algorithm in four major areas. The applications are several and they could be founded in the bibliography, here I will talk about the most significant ones

4.1. Reaserching for a Global Pathfinding Strategy to reduces traffic

The main disadvantage of the applications like Google Maps and Waze is that they are not able to deal the pathfinding problem as a global problem but only for the single user. This means that their intelligence is able to suggest a better route when in the calculated one there is an obstacle but they are not able to see the whole picture; the consequence is that they will suggest the new route to all the users and in few minutes create a congestion in the route that was supposed to be faster. Another problem is that they are not real-time algorithms, this means that they are able to detect a congestion when it is already too late.

Thanks to the new technologies of the recent years that allows the transport of a huge amount of data, the researchers are working to find new solution for the mobility that involves a huge number of vehicles that should move in harmony between them [1][2][3][4][14].

Following this idea, the **VANET** architecture has becoming everyday more important in urban enviroment. A VANET [1][2][4] is an emerging vehicular ad hoc wireless networks that can be used to deliver on Intelligent Transport System with improved communications capabilities real-time traffic information more efficiently and in a cost-effective manner.

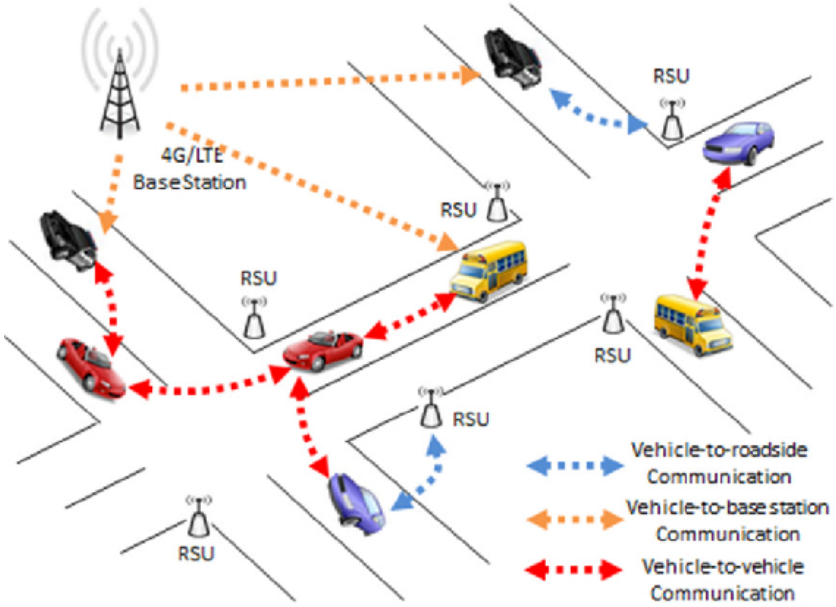


Figura 6: VANET structure

A VANET is characterized by (Figure 6):

- Vehicle-to-Roadside unit (V2R): a system that allows communication between vehicles and fix units on roadsides.
- Vehicle-to-Vehicle communications, so that the information could be shared between vehicles, and consequently through V2R.
- Road-Side Units (RSUs): elements that are able to communicate with a central system and with the vehicles
- Base Station: it has the computational power to analyze all the data received, implement a global strategy, and communicate it to the vehicles and RSUs.

All this real-time information, which will be collected, can be used for route planning in individual vehicles, freeway-traffic-flow management, and vehicle localization. Another important advantage is that knowing real-time traffic information can lead to a completely new pathfinding algorithms that can be designed with the aim of discovering the most efficient routes that individual vehicles can take considering all vehicles surrounding it and the real-time environment.

In [1] it has implemented the ORPA ALgorithm that calculates the shortest path avoiding the traffic and moving in coordination with the other vehicles increasing the overall efficiency. it has been conducted a comparative study on the same condition (starting/destination point and traffic condition) between ORPA and the most used algorithms like A*, VBA* and Dijkstra proving that ORPA calculates fastes routes in every condition.

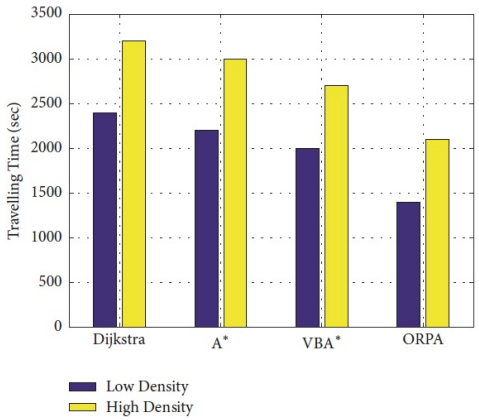


Figura 7: Comparison of travelling time in low-and high-density traffic condition

4.2. Multi-Pathfinding problems with a Time-Window constraint

A very important problem to solve is the optimization of all services that implies different delivery/pick-up points merge together with a time-window constrain. This kind of problems could be easily applied to a huge amount of situation and cases, indeed the literature dealing with these urban problems is large. All delivery services, all ride sharing services but also heavy transportation and supplies services could use pathfinding to optimize their service, reducing the time of travelling and allowing to respect the time-window constrain [5][7][10][11][12][15].

The [12] proposes a very smart futuristic solution for urban delivery companies; the idea is to use self-driving delivery robots to provide flexibility for on-time deliveries and help to protect both driver and customers by minimizing contact. In this configuration it is possible to have dispatching robots to serve nearby customers while a driver is also serving a customer (Figure 8).

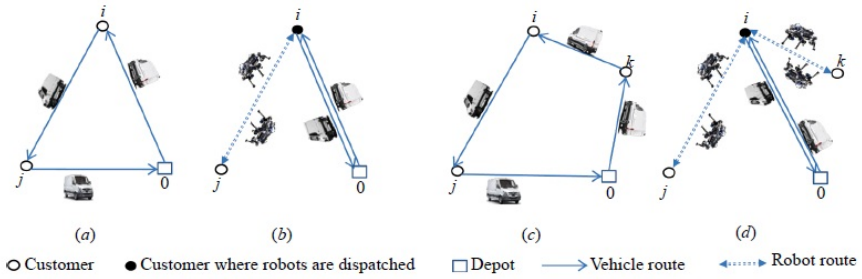


Figura 8: Four representative routes of vehicle-visit versus robot-visit

This structure presents different positives points like the limited time of the vehicle on roads and the reduction of traffic, the optimization of working time because with one driver's delivery it is possible to cover a whole area, reduce the stress of the driver and improve the delivery capacity of the company.

The researchers called the algorithm used to implement this structure Vehicle Routing Problem with Time Windows and Delivery Robots (**VRPTWDR**); indeed they started from a graph-based approach, they implemented a mathematical model for the VRPTWDR and investigate the challenges and benefits of using delivery robots as assistants for city logistics.

Even if the real application of this structure looks futuristic, the simulation brought to astonished results, for this reason it won't be a surprise to find this solution implemented in the next decade.

4.3. Research and Implementation of new Pathfinding Algorithms

In this area of studies I found four article [1][2][9][16] that tries to solve the pathfinding problem implementing new algorithms. The first two article [1][2] try to develop new algorithms knowing the real-time situation of the traffic but the last two [9][16] are my favourite two in the all research.

The aim of [9] is to propose a new algorithm, **RRNbDTs** that would solve the main problem of the Dijkstra, A*, D* algorithms, the high computational complexities. But the [16] is as genius as it is simply. The researchers from the Department of Electrical Engineering of the Universitas Indonesia, create an APP for motorcycles suitable specifically for the city of Jakarta. Their city doesn't have any smart-infrastructure and the budget of the department was very low but they succeeded to find a way of solving the major deficiency of APPs like Waze or Google-maps. They imported the map from OpenStreetMap into PostgreSQL databases and use the pgRouting library for route planning. From the last library they implemented the **entropy-balanced k shortest paths (EBkSP)** routing algorithm; the idea under the algorithm is to create all the possible shortest routes and give them a weight based on the popularity. At the beginning all the routes have popularity zero but when a vehicle chose to follow a particular route, that path assume a weight (increase its popularity) and the following user query would be direct to a path with less popularity. Implementing this strategy, their idea is to control globally the traffic through only the smartphones of the users.

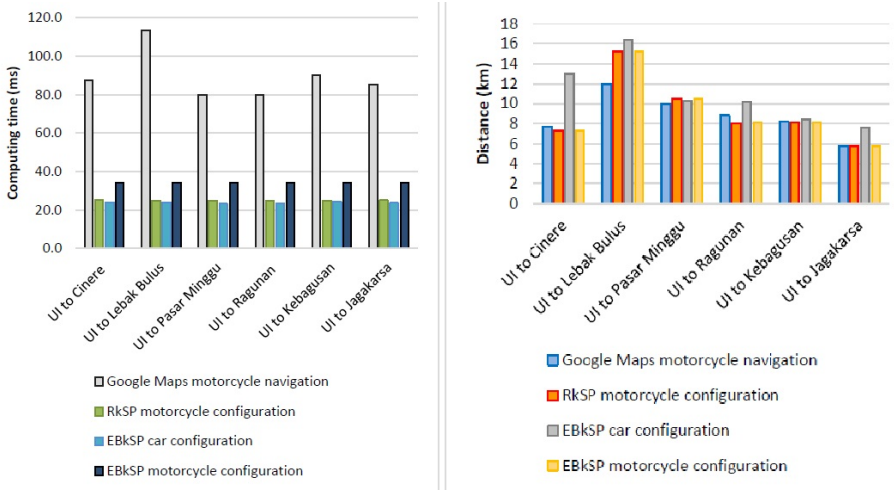


Figura 9: EBkSP vs Google Maps

In Figure 9 it is possible to see how, without having all the technology and financial resources of Google, they accomplished to obtain the same results (Right) but also with a lower computational time (Left).

4.4. Creating new infrastructures or new methods of using the existing ones

The problem that researchers are trying to solve is to create new infrastructure [8], modify the existing ones [13] or using the existing one in a different way [6] in order to reduce the traffic in the cities, which will allow to loose less time in daily driving, improve the environment and the air quality, and improve the efficiency of the city.

The [6] is the one I found more interesting in this area because the researchers try to solve the problem of urban transportation using a line metro. A metro line has trains that are using the binaries only for a limited window of time, which means that it is possible to have the metro line free to be used by AGV robots or to share the metro train between passengers and goods, with the idea of creating a framework (Figure 10) to integrate metro for urban logistics delivery in order to avert the conflicts with ground transportation, reduce delivery cost, transport distance and delivery time of vehicles.

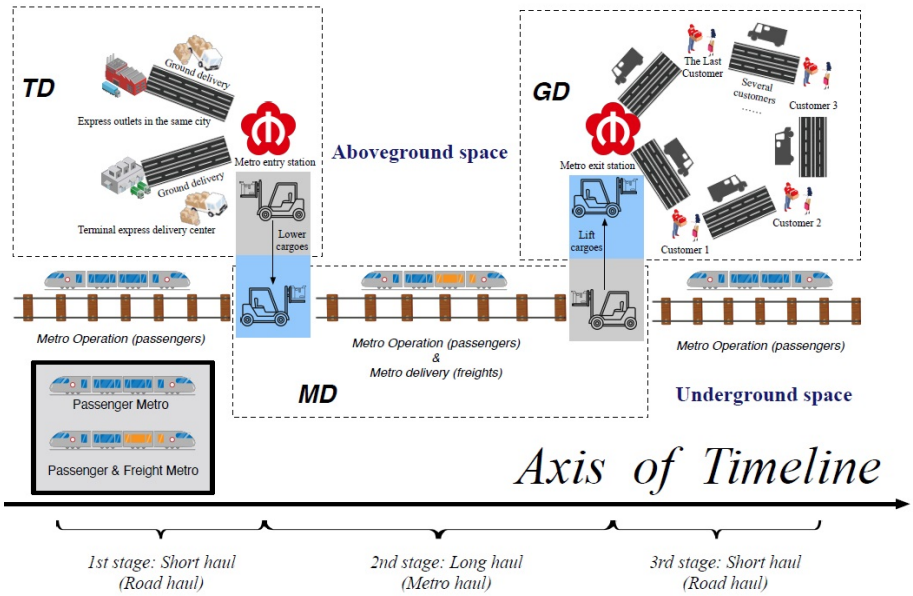


Figura 10: Urban logistics delivery process based on a single Metro Line

By means of the saving algorithm (CW algorithm) and the improved Tabu Search algorithm (CWTS), the established model is solved to optimize the delivery routes under metro-based Underground Logistic System (ULS) + Ground Delivery (GD). The performance of this framework has been compared to Ant Colony Optimization applied on surface transport, obtaining better results in terms of delivery distance, delivery cost and delivery time.

5. CONCLUSIONS

The Graph-Based Approach is the best way to import a map and make it suitable for a computer and the other programming languages. Also pathplanning is the best example of how is hard to improve an efficient algorithm like the A* or Dijkstra Algorithm.

The theory behind the A* algorithm is not difficult to understand, as also the Graph-Based Approach but the implementation over complex real maps is quite challenging.

The most interesting part of this research is the deep connection between technologies, business and scientific world.

Nowadays the logistic area represents one of the most complicated and less efficient inside every company and, for this reason, they are willing to invest in new technologies and in Research and Development in order to improve the overall efficiency of the company.

Also social problems like environment and the quality of city life has given a huge acceleration to innovative applications of the pathfinding algorithms as it is clear from this paper. Ride sharing, new infrastructure, new ways to accomplish jobs like the couriers have become a huge point of focus but the ***big vision*** is to create a central A.I., integrated with smart-cities, that would manage the road urban transportation, for work or leisure, and that would eliminate traffic and accidents.

The future of pathfinding is brilliant because with the increase of the investment towards a green economy and smart-cities, would keep giving motivation and resources to the scientific community to develop new and better solution or technologies that would definitely improve our quality of life.

Referencias

1. Al-Mayouf, Yusor Rafid Bahar and Mandi, Omar Adil and Taha, Namar A. andAbdullah, Nor Fadzilah and Khan, Suleman and Alam, Muhammad. *Accident Management System Based on Vehicular Network for an Intelligent Transportation System in Urban Environments*. Journal of Advanced Transportation 2018 (2018).
2. Wan Xiangpeng and Ghazzai Hakim and Massoud Yehia. *Mobile Crowdsourcing for Intelligent Transportation Systems: Real-Time Navigation in Urban Areas*. IEEE Access 7 (2019): 136995-137009.
3. Junhua Wang, Kai Liu, Ke Xiao, Xiumin Wang, Qingwen Han and Victor Chung Sing Lee. *Delay-Constrained Routing via Heterogeneous Vehicular Communications in Software Defined BusNet*. IEEE Transactions on Vehicular Technology 68.6 (2019): 5957-5970.
4. Luis R. Gallego-Tercero, Rolando Menchaca-Mendez, Mario E. Rivero-Angeles and Ricardo Menchaca-Mendez. *Efficient Time-Stable Geocast Routing in Delay-Tolerant Vehicular Ad-Hoc Networks*. IEEE Access 8 (2020): 171034-171048.
5. Yanshuo Sun, Zhi-Long Chen and Lei Zhang. *Nonprofit peer-to-peer ridesharing optimization*. Transportation Research Part E: Logistics and Transportation Review 142 (2020): 102053.
6. Changjiang Zheng, Yuhang Gu, Jinxing Shen and Muqing Du. *Urban Logistics Delivery Route Planning Based on A Single Metro Line*. IEEE Access 9 (2021): 50819-50830.
7. Kaushik Manchella, Abhishek K. Umrawal, and Vaneet Aggarwal. *FlexPool: A Distributed Model-Free Deep Reinforcement Learning Algorithm for Joint Passengers and Goods Transportation*. IEEE Transactions on Intelligent Transportation Systems 22.4 (2021): 2035-2047.
8. Sachin Nataraj, Daniele Ferone, Carlos Quintero-Araujo, Angel A. Juan and Paola Festa. *Consolidation centers in city logistics: A cooperative approach based on the location routing Problem*. International Journal of Industrial Engineering Computations 10.3 (2019): 393-404.
9. Jiangtao Kong, Jian Huang, Hongkai Yu, Hanqiang Dengand, Jianxing Gong and Hao Chen. *RNN-based default logic for route planning in urban environments*. Neurocomputing, 338, 307-320.
10. Kittipong Thawongklang, Ladda Tanwanichkul. *Application of production scheduling techniques for dispatching ready-mixed concrete*. International Journal of Technology 7.7 (2016): 1163-1170.

11. Yuan Shia, Meng Chena, Ting Qub, Wei Liua, and Yiji Caica. *Digital connectivity in an innovative joint distribution system with real-time demand update.* Computers in Industry 123 (2020): 103275.
12. Chen, C. and Demir, E. and Huang, Y. and Qiu, R. *The adoption of self-driving delivery robots in last mile logistics.* Transportation research part E: logistics and transportation review 146 (2021): 102214.
13. Huang, Y. and Li, Z. *Optimal multimodal travelway design for an urban street network.* IEEE Access 8 (2020): 187700-187712.
14. Cun Cao, Xuefeng Guan, Na Zhang, Xinglei Wang and Huayi Wu, *A Hybrid Deep Learning-Based Traffic Forecasting Approach Integrating Adjacency Filtering and Frequency Decomposition.* IEEE Access 8 (2020): 81735-81746.
15. Rafael Grosso, Jesús Muñuzuri, Alejandro Escudero-Santana and Elena Barbadilla-Martín *Mathematical Formulation and Comparison of Solution Approaches for the Vehicle Routing Problem with Access Time Windows.* Complexity 2018 (2018).
16. Asvial, M. and Pandoyo, M.F.G. and Arifin, A.S. *Entropy-based k shortest-path routing for motorcycles: A simulated case study in Jakarta.* International Journal of Advanced Computer Science and Applications 11.7 (2020): 442-449.
17. Zhang, J.-D. and Feng, Y.-J. and Shi, F.-F. and Wang, G. and Ma, B. and Li, R.-S. and Jia, X.-Y. *Vehicle routing in urban areas based on the Oil Consumption Weight-Dijkstra algorithm.* IET Intelligent Transport Systems 10.7 (2016): 495-502.
18. H. Wang, Y. Yu, Q. Yuan. *Application of Dijkstra Algorithm in robot pathplanning.* Proceedings of International Conference on Mechanic Automation and Control Engineering, 2011, pp. 1067–1069 .
19. Moschovakis, Yiannis N. *What is an algorithm?.* Mathematics unlimited—2001 and beyond. Springer, Berlin, Heidelberg, 2001. 919-936.
20. Ostrowski, Dustin, et al. *Comparative analysis of the algorithms for pathfinding in gps systems.* ICN 2015 (2015): 114.
21. Goyal, Abhishek, et al. *Path finding: A* or dijkstra's*? International Journal in IT and Engineering 2.1 (2014): 1-15.

Moneyball: Análisis y predicción de resultados en el deporte

Luis Carlos Arroyo Vicente, María Angélica González Arrieta

Departamento de Informática y Automática
Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
carlosarroyo@usal.es;angelica@usal.es
https://www.usal.es/

Resumen El objetivo principal del trabajo es utilizar diferentes métodos supervisados de inteligencia artificial para la predicción de resultados deportivos. Se utilizarán cinco conjuntos de datos que contiene información de todos los partidos de varias temporadas de cinco ligas de fútbol. Se realizará una revisión teórica de una serie de algoritmos seleccionados de inteligencia artificial. Posteriormente, se implementan dichos algoritmos para analizar los resultados de aplicación de inteligencia artificial en el campo objeto de estudio modificando determinadas variables.
Estas variables serán obtenidas a través del procesamiento de los datos. Se analizará la importancia de cada una de ellas y su importancia conjunta para la predicción precisa de resultados.
Se concluirá la memoria determinando cuál de los modelos tiene un mayor rendimiento en cada una de las ligas objeto de estudio en función de las características utilizadas para dicha predicción.

Keywords: Inteligencia artificial, predicción, fútbol, machine learning, modelos supervisados

1. Introducción

El análisis y la predicción de resultados deportivos siempre ha sido un tema relevante en cualquier tipo de deporte. Desde hace tiempo se ha tratado de predecir resultados deportivos buscando patrones que pudiesen asegurar una buena predicción.
El aumento de importancia de este tipo de predicciones ha sido determinante para la elección del tema. Este aumento de importancia es debido a varios factores, entre ellos, el mercado está en auge, debido a las casas de apuestas y a la búsqueda de estadísticas relacionadas con los deportes, la mejora de tomas de medidas para formar estos conjuntos de datos y por lo tanto, la mejora de los propios conjuntos de datos en sí, ha provocado también que aumente el afán por el estudio de ellos y por supuesto, el crecimiento exponencial de la potencia las

CPU, ha provocado que estos algoritmos sean mucho más rápidos, obteniendo resultados eficaces de manera veloz.
En este trabajo se buscará obtener un modelo de predicción de resultados de fútbol a través de set de datos referentes a ligas de diferentes países, a saber, Alemania (Bundesliga), España (Liga BBVA), Francia (Ligue 1), Italia (Serie A) e Inglaterra (Barclays Premier League). En los conjuntos de datos se encuentra información sobre las últimas 20 temporadas (el número exacto de temporadas varía en cada liga). Para ello, se realizará un preprocesamiento de los datos antes de aplicarles los algoritmos de inteligencia artificial y obtener las predicciones finales.

La predicción de resultados deportivos atañe a cualquier deporte. Existen modelos que estudian variables internas del propio deporte en sí, como puede ser, la importancia de los jugadores más inteligentes en cada equipo utilizando el modelo de red neuronal ABPNN (Adaptive Back Propagation Neural Network) en el artículo [10] , o predicción de las fuerzas y momentos de reacción en el suelo de los atletas a través de redes neuronales convolucionales (CNN), se puede encontrar en [9]. Incluso en baloncesto se han realizado estudios mediante técnicas de aprendizaje automático para predecir el crecimiento de estrellas en potencia como se describe en [13].

Con los datos y la ayuda de inteligencia artificial se pretende encontrar modelos que pronostiquen los resultados de manera fiable. El crecimiento de juegos y de casas de apuestas ha propiciado que cualquier deporte sea estudiado mediante los datos. En la literatura se pueden encontrar artículos sobre modelos para Volleyball, como en el artículo [11] donde utilizan aprendizaje automático, en concreto, el algoritmo 'a priori' para pronosticar las estrategias de un partido de Volleyball, además realizan un análisis y optimizan este algoritmo para obtener mejores resultados. También existen modelos para Bádminton en el documento [17] donde se utilizan varios clasificadores, una red bayesiana y un modelo CHIRP (Composite Hypercubes Random Projections), para la predicción de resultados en los torneos Open de Australia, Malasia, Alemania y Singapur de 2019 o el baloncesto, [18] donde presentan un algoritmo para predicción de resultados en tiempo real basado en datos del partido y datos históricos de los equipos.

Centrándonos en el fútbol podemos encontrar algoritmos de predicción no solamente para los resultados, si no al igual que en otros deportes, estudios que pretenden comprender y evitar las lesiones más comunes en el fútbol como [1] o artículos que clasifican y exponen los mejores jugadores y su influencia como se puede comprobar en el artículo [12]. Incluso hay quienes han intentado descifrar las variables ocultas en el fútbol, por qué no es un suceso extraño que un equipo pequeño derrote a uno grande, o un partido cambie completamente en cuestión de minutos, [14].

Aunque este trabajo se centrará sobretodo en los modelos de predicción de resultados. Respecto a este tema, se pueden encontrar diversos artículos que

utilizan diferentes técnicas de inteligencia artificial. Desde las ya reconocidas redes neuronales en [15] donde se enfocan en las redes LSTM, u otros métodos supervisados, como el caso de [7] donde presentan un modelo de red bayesiana llamado pi-football para la predicción de resultados de la Premier League en la temporada 2010-2011.También existen artículos como [2] donde se comparan diferentes algoritmos que se utilizan para la predicción de resultados y a través del error que comete cada uno se determina el mejor.

En la predicción de resultados de fútbol encontramos artículos que tratan la Premier League, como es el caso de [19].

Como hemos observado en los artículos mencionados anteriormente, hay una gran cantidad de métodos para la predicción de resultados.
En este trabajo se utilizarán varios de los modelos supervisados y se compararán entre ellos, intentando determinar cual ofrece mejores predicciones para cada una de las ligas.

La disposición de este trabajo sigue la disposición general de un informe técnico.
En el capítulo 2 se realizará un desarrollo teórico de los algoritmos a tratar, en el capítulo 3 se encontrará el análisis y el procesamiento de los conjuntos de datos, en la sección 4 se recogerán los resultados y se realizará un análisis de los mismos, por último, se encontrarán las conclusiones.

2. Desarrollo teórico

Este trabajo utilizarán diferentes algoritmos de aprendizaje automático, por ello se realizará primero un estudio de cada uno de ellos. Primero se tratará el algoritmo SVM (Support Vector Machines), seguido de algoritmos basados en árboles y bosques aleatorios. A parte de estos algoritmos se utilizará también una regresión logística.

La regresión logística, ya conocida por todos, es un instrumento estadístico de análisis multivariado de uso predictivo. Resulta útil e interesante su empleo cuando se tiene una variable dependiente y un conjunto de variables independientes que se utilizarán como predictoras. El propósito del análisis consiste en predecir la variable dependiente determinando mediante los pesos, qué variables son más importantes y ayudan mejor a predecir correctamente los resultados. De esta misma manera se expondrán los demás algoritmos a utilizar.
Las máquinas de vector de soporte son un conjunto de algoritmos de aprendizaje estadístico supervisado desarrollado por Vladimir Vapnik (laboratorio AT&T). Pertenecen a la familia de clasificadores lineales.
Una SVM construye un hiperplano en un espacio de dimensionalidad cualquiera (normalmente alta, incluso infinita). Este hiperplano busca separar de forma

óptima los puntos pertenecientes a una clase o a otra. No solo trata de encontrar el hiperplano que separe los puntos de las diferentes clases, si no que además debe existir la máxima distancia o margen, entre los puntos más cercanos de diferentes clases. También se les conoce, por tanto, como clasificadores de margen máximo (WSVM).
Definición. Un hiperplano separador se considerará óptimo si y solo si equidista del ejemplo más cercano de cada clase.

La demostración se puede realizar por reducción al absurdo, se puede comprobar en 'Introducción a las máquinas de vector soporte (SVM) en aprendizaje supervisado' [5].
La distancia entre un hiperplano de separación $D(x)$ y un ejemplo x' viene dada por:

$$\frac{|D(x')|}{||w||_2} \tag{1}$$

La formulación matemática implica que, [5]:

$$y_i D(x_i) \ \geq \ 1 \ i = 1, ..., n \tag{2}$$

Un ejemplo se dirá separable si satisface la ecuación (2), y llamaremos vectores de soporte, figura 1, a aquellos ejemplos que satisfacen esta ecuación con igualdad. Los vectores de soporte aplicados a la función de decisión cumplen que $|D(x_i)| = 1$, por lo que la distancia de cualquier punto al hiperplano siempre será $\frac{1}{||w||}$.

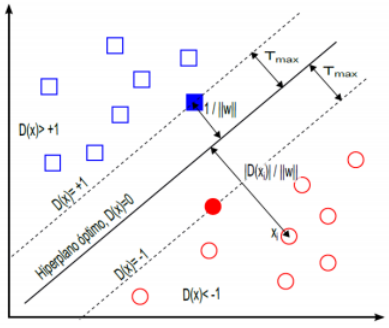


Figura 1: La distancia de cualquier ejemplo x_i, al hiperplano de separación óptimo es $\frac{1}{||w||}$, figura procedente de [6].

Los métodos basados en árboles, son modelos de predicción construidos mediante la sucesión de particiones de un conjunto de datos y del ajuste de un modelo simple a cada una de ellas. En este trabajo se utilizará tanto un árbol de decisión como un bosque aleatorio para predecir los resultados.

Los árboles de clasificación incorporan un método de clasificación supervisado, el nombre es debido a la similitud con los árboles que se componen de raíz, nodos, ramas y hojas. De manera similar, estos modelos se componen de nodos que son relacionados entre sí debido a las ramas (segmentos) que los unen, pero, de manera contraria estos modelos se inician desde la raíz y se extienden hacia abajo. El nodo inicial se llama 'nodo raíz' mientras que los nodos finales se les conoce como 'nodos hoja'.
El método se basa en una partición inicial que divide el espacio total en dos regiones devolviendo como respuesta el valor medio de Y para cada región. A continuación, se selecciona la variable y su nuevo valor de partición para poder realizar la siguiente partición.
A partir de las particiones es fácil componer el árbol, y obtener el valor de la predicción correspondiente significa simplemente seguir las ramas hasta llegar a una de las 5 regiones definidas, figura 2.

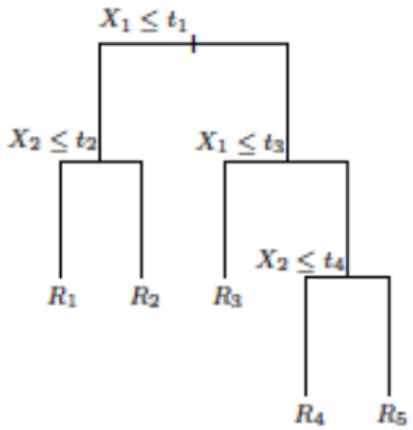


Figura 2: Árbol creado a partir de las particiones binarias [8].

El modelo de Random Forest, es una combinación de árboles predictivos, es decir, una modificación del Bagging [16]. Utiliza una colección de árboles (clasificadores débiles) no correlacionados y los promedia [8]. Cada árbol depende de los valores de un vector aleatorio de la muestra de manera independiente y con la misma distribución de todos los árboles en el bosque. Si el número de árboles es

grande, el error converge a un límite. Este error depende individualmente de la potencia de cada árbol y además de la correlación que tengan entre todos. El uso de una selección aleatoria de las características provoca que sean más robustos con respecto al ruido (clasificador fuerte). El error, la potencia y la correlación se miden a través de cálculos internos.
Éste método busca la disminución de la variabilidad y por lo tanto consigue un aumento en la fiabilidad y robusted, lo que genera ventajas a la hora de trabajar con grandes cantidades de datos.

3. Análisis y preprocesamiento de los conjuntos de datos

A nivel teórico en este trabajo se buscará predecir el ganador, vencedor o si fuera empate de un partido. Para ello se necesitará una base de datos con resultados de partidos, pero con esto no será suficiente, se realizará un proceso de procesamiento donde se tratará de encontrar características intrínsecas en los datos pero que no se ven a simple vista.
Los conjuntos de datos han sido obtenidos a través de la plataforma Kaggle, se utilizarán cinco conjuntos de datos de las cinco ligas (Liga BBVA, Barclays Premier League, Bundesliga, Serie A, Ligue 1). De ahora en adelante se utilizará el conjunto de datos de la bundesliga, siguiéndose el mismo procedimiento para los demás.

El conjunto de datos de la Bundesliga es un único fichero con información de las temporadas desde 1993-2018 con un total de 7649 partidos.
En la figura 3, se puede observar las diferentes características que se pueden encontrar en el conjunto de datos. A parte de las características comunes a los otros conjuntos de datos, en este encontramos también la división. No dispone de la temporada pero si de la fecha por lo que se creará esta característica lo primero.
Todas las temporadas cuentan con 18 equipos, por lo que son 34 jornadas distintas por temporada.

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG
0	D1	7/8/1993	Bayern Munich	Freiburg	3	1	H	NaN
1	D1	7/8/1993	Dortmund	Karlsruhe	2	1	H	NaN
2	D1	7/8/1993	Duisburg	Leverkusen	2	2	D	NaN
3	D1	7/8/1993	FC Koln	Kaiserslautern	0	2	A	NaN
4	D1	7/8/1993	Hamburg	Nurnberg	5	2	H	NaN
...	..	...	...	...	...	...	..	...
7645	D1	12/5/2018	Hoffenheim	Dortmund	3	1	H	1.0
7646	D1	12/5/2018	Leverkusen	Hannover	3	2	H	2.0
7647	D1	12/5/2018	Mainz	Werder Bremen	1	2	A	1.0
7648	D1	12/5/2018	Schalke 04	Ein Frankfurt	1	0	H	1.0
7649	D1	12/5/2018	Wolfsburg	FC Koln	4	1	H	1.0

Figura 3: Conjunto de datos de la Bundesliga.

Para procesarlo, se creará un código que lea el fichero y posteriormente se introducirá en las diferentes rutinas. Además crearemos la característica para diferenciar temporada a temporada.
En la gráfica 4 se puede observar la distribución de los datos que vamos a utilizar en el conjunto de datos de la Bundesliga. Es notable la diferencia que existe entre los datos, habiendo muchos más partidos en los que gana el equipo local que en los que pierde o empata.

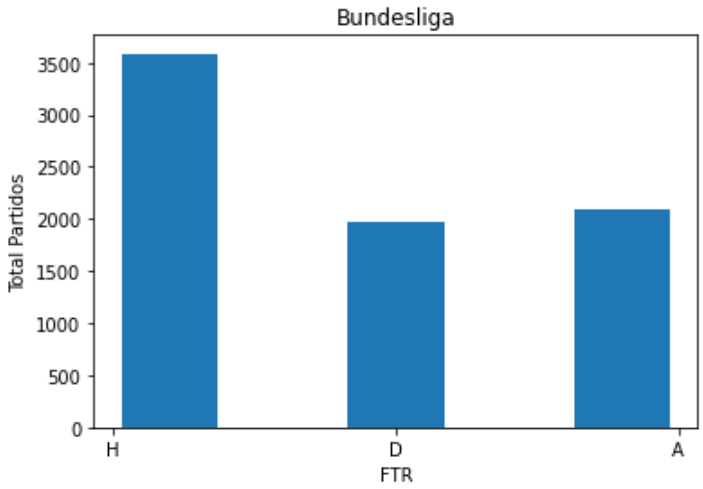


Figura 4: Distribución de los datos del conjunto de la Bundesliga.

Se puede comprobar, por lo tanto, que se dispone de una distribución desigual de los datos, siendo los datos que reportan un empate los que se encuentran en menor número.
Se encontrará una distribución similar para el resto de las ligas.

3.1. Procesamiento de los datos

A través del procesamiento de los datos se tratará de obtener nuevas características que ayuden a realizar un análisis y una predicción más completa. Se utilizarán rachas de victorias, la diferencia de puntos entre los dos equipos que se enfrentan la cantidad de goles recibidos y producidos por cada equipo.
Para crear estas nuevas características definiremos una serie de funciones que realizarán estos procesos a través de Python.

Función de obtención de goles Esta función trata el conjunto de datos y obtiene los goles referentes a cada equipo, por un lado el equipo que juega en casa y por otro lado el equipo que juega fuera de casa. Esta rutina se divide en tres partes.

- Obtención de los goles marcados por el equipo.
- Obtención de los goles recibidos por el equipo.
- Creación de las características diferenciando en si los goles de cada equipo fueron como local o como visitante.

Se obtienen los goles teniendo en cuenta en qué momento de la temporada sucede el encuentro por lo que no se sumarán los goles que sucedan después del momento del partido, básicamente, solo se tendrán en cuenta los goles hasta la fecha del partido.

Función de obtención de los puntos Dado que en el propio dataset se tiene información sobre todos los partidos y el resultado de cada uno de ellos, al tratarse de una temporada de una liga, una buena idea es calcular los puntos que tienen cada uno de los equipos. Para ellos primero habrá que definir como se reparten los puntos, esto se hará de manera tradicional. El ganador recibirá tres puntos y el perdedor cero puntos. En caso de empate, los dos equipos recibirán 1 punto.

Función de obtención de la forma Esta función se encargará de obtener la forma de un equipo. Se centrará principalmente en rachas de victorias y de derrotas.Se tendrán en cuenta los últimos cinco partidos y se crearán rachas de los últimos tres y cinco partidos.
Se asigna el valor '1' si se cumple la racha, tanto de victorias como de derrotas. Se asigna el valor '0' si la racha no se cumple. Finalmente se tendrán en cuenta las rachas de victorias y derrotas de los últimos 3 y 5 partidos.

Función diferencia de puntos y goles Se crea una última función para obtener la diferencia de goles encajados y goles anotados por cada equipo. Por último, se creará una función que devuelva la diferencia de puntos que tienen los dos equipos y además, la diferencia de puntos existentes respecto a las últimas jornadas.

Con esta serie de variables nuevas, se comenzará a realizar el análisis y la predicción de los resultados de los partidos.

4. Resultados y análisis

En este apartado se tendrán en cuenta todos los modelos obtenidos tratados en esta memoria. Se presentarán en una tabla en la que el parámetro de estudio será la precisión. La precisión se puede calcular matemáticamente:

$$Precision = \frac{TP + TN}{TP + TN + FP + FN} = \frac{NClasificadosCorrectamente}{NTotal} \quad (3)$$

Donde TP es el número de positivos clasificados correctamente, TN es el número de negativos clasificados correctamente, FP es el número de falsos positivos y FN es el número de falsos negativos.

Análisis de los resultados de la bundesliga A continuación se expondrán en la tabla 1 los resultados obtenidos en los aparatados anteriores.

Modelo	**Puntos**	**Goles**	**Rachas**	**Goles, puntos**	**Rachas, puntos**	**Rachas, goles**	**Todos**	**Media**
Random Forest	40,89	42,28	41,65	44,23	41,33	41,95	43,71	42,29
SVM	46,31	45,78	44,47	45,92	45,75	46,22	46,1	45,79
Decision Tree	37,42	38,2	40,35	37,8	38,25	38,3	37,95	38,32
Logistic R.	47,17	46,95	46,57	47,23	44,17	46,74	47,38	46,59
Media	43	43,30	43,26	43,79	42,37	43,30	43,72	43,25

Tabla 1: Resultados para los algoritmos de la Bundesliga.

Desde esta tabla, 1, se pueden determinar varias conclusiones. Al calcular tanto la media de cada algoritmo, como la media de cada columna, es fácil determinar tanto el mejor algoritmo, como la mejor manera de predecir los resultados a través de los datos.

El mejor clasificador por media es el que utiliza el modelo de regresión logística con 46,59 %, seguido del modelo SVM con 45,79 %. El peor modelo es el que utiliza un árbol de decisión con 38,32 % de precisión.

La mejor manera de predecir resultados es a través de los goles y puntos o a través de todas las variables. A la vista de estos resultados, tan similares no se puede concluir que las predicciones a través de las rachas sean peores pero si parecen tener menor importancia.

Además, el que mayor precisión ha obtenido a la vista de los resultados es el modelo de regresión logística con todas las variables, seguido muy de cerca por

el mismo modelo pero que utiliza goles y puntos, es más, se puede observar también que estos dos últimos por separado han obtenido buena puntuación.

En general la precisión media de predicción para el conjunto de la bundesliga es del 42,25 %.
Para los demás algoritmos existe también un comportamiento similar aun que con ciertas excepciones dado que en general, para cada modelo, los valores de la precisión son bastante similares independientemente de las características que utilicemos. No hay ninguna que destaque por su mala precisión.

Modelo	**Puntos**	**Goles**	**Rachas**	**Goles, puntos**	**Rachas, puntos**	**Rachas, goles**	**Todos**	**Media**
Random Forest	46,92	44,7	45,47	49,51	46,61	45,21	48,78	46,74
SVM	49,87	49,77	47,54	50,7	49,87	49,92	50,96	49,80
Decision Tree	40,04	40,25	44,02	40,25	41,85	39,16	40,71	40,89
Logistic R.	50,28	49,56	48,16	51,84	50,54	49,66	52,35	50,34
Media	46,77	46,07	46,29	48,07	47,21	45,98	48,2	46,94

Tabla 2: Resultados para los algoritmos de la BBVA.

Análisis de los resultados de la BBVA En la tabla 2 se pueden observar las diferentes precisiones que han tenido los diferentes modelos en cada uno de los casos.

A simple vista también se puede comprobar que cuando se utilizan todas las variables se suelen obtener los mejores resultados. También destaca, al igual que con el conjunto de datos de la Bundesliga, el caso en el que utiliza los goles y puntos.

El mejor clasificador vuelve a ser el modelo de regresión logística seguido por el modelo SVM. El peor modelo, al igual que antes, es el modelo del árbol de decisión.
El valor medio de la precisión para este conjunto de datos es de 46,94 %, mejorando la puntuación anterior. Puede influir que en este conjunto de datos hay temporadas que contienen 20 equipos, por lo que, hay un mayor número de datos.

Modelo	Puntos	Goles	Rachas	Goles, puntos	Rachas, puntos	Rachas, goles	Todos	Media
Random Forest	45,05	42,62	41,73	44,51	45,80	43,63	48,10	44,49
SVM	50,75	51,29	49,59	51,08	50,54	51,29	51,36	50,84
Decision Tree	38,35	38,14	40,65	39,16	38,41	38,28	36,99	38,57
Logistic R.	51,08	50,81	49,80	51,36	50,47	50,88	51,56	50,85
Media	46,31	45,72	45,44	46,53	46,31	46,02	47,00	46,19

Tabla 3: Resultados para los algoritmos de la Ligue 1.

Análisis de los resultados de la Ligue 1 En la tabla 3 se encuentran los resultados de los estudios de los modelos de predicción para la Ligue 1.
Los dos mejores modelos siguen siendo SVM y la regresión logística. El peor, el árbol de decisión.
La mejor manera de predecir aquí también es respecto a todas las variables, mientras que la peor, a través de las rachas.
La media general de predicción de resultados en la Ligue 1 es de 46,19 %

Modelo	Puntos	Goles	Rachas	Goles, puntos	Rachas, puntos	Rachas, goles	Todos	Media
Random Forest	47,30	49,12	46,27	50,44	47,81	48,98	50,22	48,59
SVM	46,56	50,51	46,35	50,29	46,49	50,22	50,37	48,68
Decision Tree	40,28	39,91	43,06	40,28	39,33	38,98	40,72	40,37
Logistic R.	51,46	52,57	46,71	52,63	51,46	52,34	52,49	51,38
Media	46,40	48,03	45,60	48,41	46,27	47,63	48,45	47,26

Tabla 4: Resultados para los algoritmos de la Barclays.

Análisis de los resultados de la Barclays El mejor algoritmo es la regresión logística, los modelos SVM y Random Forest tienen una puntuación parecida, mientras que el modelo de árbol de decisión sigue siendo el peor clasificador, como se puede observar en la tabla 4.
La mejor manera de predicción es a través de todas las variables, y la peor, como en el resto de los casos, es a través de las rachas.
La media general es de 47,26 %.

Modelo	Puntos	Goles	Rachas	Goles, puntos	Rachas, puntos	Rachas, goles	Todos	Media
Random Forest	46,09	45,67	44,34	48,07	46,03	45,79	47,23	46,17
SVM	50,36	46,87	46,87	50,54	50,48	50,84	50,42	49,48
Decision Tree	41,88	39,23	42,36	38,63	41,03	38,15	39,35	40,09
Logistic R	41,58	51,08	47,11	50,42	50,30	50,78	50,45	48,82
Media	44,98	45,71	45,17	46,92	46,96	46,39	46,86	46,14

Tabla 5: Resultados para los algoritmos de la Serie A.

Análisis de los resultados de la Serie A El modelo SVM es el que mejores resultados ha obtenido en estas simulaciones, después la regresión logística, como se puede observar en 5.

Las mejores predicciones son utilizando todas las características o simplemente, los goles y los puntos.
El valor medio general de las predicciones en la Serie A es de 46,15 %.

5. Conclusiones

En este trabajo de fin de máster hemos realizado predicciones de resultados deportivos, concretamente de fútbol, a través de cinco conjuntos de datos de cada una de las cinco ligas europeas más importantes. Para ello, se han utilizado cuatro modelos supervisados distintos con diferentes características para comprobar la importancia de éstas en la precisión a la hora de predecir los resultados.
Se ha realizado un desarrollo teórico de los métodos de inteligencia artificial que se han utilizado, y se han analizado los conjuntos de datos para posteriormente, realizar un procesamiento de los mismos que nos permita encontrar nuevas características que ayuden a la predicción.
Realizando las distintas simulaciones estamos capacitados para responder las preguntas que antes nos hacíamos. En cuanto a los métodos de aprendizaje automático más óptimos, tenemos:

- Los mejores clasificadores han sido las máquinas de vector de soporte (SVM) con una media general de precisión de 48,92 % y la regresión logística con 49,60 %. El modelo de Random Forest ha obtenido resultados peores pero bastante similares con una media de 45,66 %.

- El modelo de árbol de decisión ha obtenido los peores resultados con una precisión de solo 39,65 %.

Se ha observado la dificultad que tienen estos clasificadores (sobretodo SVM y la regresión logística) para predecir el resultado del partido como empate. Esto puede deberse a la distribución desigual de los datos ya que tenemos muchos más datos de victoria del equipo local o la victoria del equipo visitante, habiendo aun así muchos más datos para la victoria del equipo que juega en su estadio. Para un trabajo futuro sería conveniente encontrar características que ayuden en la predicción de este tipo de resultados.
A la vista de los resultados de la precisión de los algoritmos utilizando las diferentes características, se puede concluir que, son mucho más relevantes los goles de cada equipo y la diferencia de puntos entre ellos que las rachas de victorias y derrotas. Esto se puede deber a la escasez de datos, dado que existe poca representación de ellos con rachas positivas de victorias (o negativas de derrotas).

En cuanto a la influencia de las diferentes ligas, se ha comprobado que la ligas con mejores resultados son la Barclays y la BBVA, también puede deberse a que en estas ligas, al ser la mayoría (o todas) de las temporadas de 20 equipos, disponemos de una mayor cantidad de datos y por lo tanto de partidos. En el caso de la BBVA además, es la liga con más partidos registrados.
Como trabajo futuro sería interesante realizar predicciones sobre un conjunto de datos más extenso. Además el uso de otras características también podría mejorar la predicción, una característica que recoja la historia de cada equipo, generando así un ranking de ellos y como ya se comentó antes, una variable que ayude a la predicción de empates.

Referencias

1. F. Ayala, A. López-Valenciano, J.A. Gámez Martín, M. De Ste Croix, F.J. Vera-Garcia, M.D.P. García-Vaquero, I. Ruiz-Pérez, and G.D. Myer. A preventive model for hamstring injuries in professional soccer: Learning algorithms. *International Journal of Sports Medicine*, 40(5):344–353, 2019. cited By 6.
2. Rahul Baboota and Harleen Kaur. Predictive analysis and modelling football results using machine learning approach for english premier league. *International Journal of Forecasting*, 35(2):741–755, 2019.
3. R. Beal, T.J. Norman, and S.D. Ramchurn. Artificial intelligence for team sports: A survey. *Knowledge Engineering Review*, 2019. cited By 6.
4. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
5. Elena Campo León and José Tomás Alcalá Nalvaiz. Introducción a las máquinas de vector soporte (SVM) en aprendizaje supervisado. 2016. Aporta en secretaría material físico.
6. Enrique Carmona. tutorial sobre maquinas de vectores soporte (svm). *UNED, Consultada en http://www. ia. uned. es/˜ ejcarmona/publicaciones/[2013-Carmona] % 20SVM. pdf (fecha de consulta 01-07-2017)*, 2014.
7. Anthony C. Constantinou, Norman E. Fenton, and Martin Neil. pi-football: A bayesian network model for forecasting association football match outcomes. *Knowledge-Based Systems*, 36:322–339, 2012.
8. Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
9. W.R. Johnson, J. Alderson, D. Lloyd, and A. Mian. Predicting athlete ground reaction forces and moments from spatio-temporal driven cnn models. *IEEE Transactions on Biomedical Engineering*, 66(3):689–694, 2019. cited By 15.
10. M.A. Khan, M. Habib, S. Saqib, T. Alyas, K.M. Khan, M.A. Al Ghamdi, and S.H. Almotiri. Analysis of the smart player's impact on the success of a team empowered with machine learning. *Computers, Materials and Continua*, 66(1):691–706, 2021. cited By 0.
11. Q. Liu and Q. Liu. Prediction of volleyball competition using machine learning and edge intelligence. *Mobile Information Systems*, 2021, 2021. cited By 0.
12. R. Maanijou and S.A. Mirroshandel. Introducing an expert system for prediction of soccer player ranking using ensemble learning. *Neural Computing and Applications*, 31(12):9157–9174, 2019. cited By 4.
13. Zafar Mahmood, Ali Daud, and Rabeeh Ayaz Abbasi. Using machine learning techniques for rising star prediction in basketball. *Knowledge-Based Systems*, 211:106506, 2021.
14. Massimo Marchiori and Marco de Vecchi. Secrets of soccer: Neural network flows and game performance. *Computers & Electrical Engineering*, 81:106505, 2020.
15. Daniel Pettersson and Robert Nyquist. Football match prediction using deep learning. *Psychol. Sport Exerc.*, 15(5):538–547, 2017.
16. J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *Aaai/iaai, Vol. 1*, pages 725–730, 1996.
17. M. Sharma, Monika, N. Kumar, and P. Kumar. Badminton match outcome prediction model using naïve bayes and feature weighting technique. *Journal of Ambient Intelligence and Humanized Computing*, 2020. cited By 1.
18. Kai Song, Yiran Gao, and Jian Shi. Making real-time predictions for nba basketball games by combining the historical data and bookmaker's betting line. *Physica A: Statistical Mechanics and its Applications*, 547:124411, 2020.

19. Ben Ulmer, Matthew Fernandez, and Michael Peterson. *Predicting soccer match results in the english premier league*. PhD thesis, Doctoral dissertation, Ph. D. dissertation, Stanford, 2013.

Sistemas de detección y seguimiento de objetos a través de visión artificial

Denis Pato Martínez, Vivian Félix López Batista y Juan Francisco de Paz Santana

Departamento de Informática y Automática, Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
{denispato679,vivian,fcofds}@usal.es
https://www.usal.es/

Resumen Este trabajo está enfocado en el campo de la detección y recuento de objetos a través de la visión artificial, centrándose en un tema muy popular en la actualidad, las ciudades inteligentes. Por ello se presenta una propuesta, tras evaluar diferentes sistemas utilizados en la literatura, basada en técnicas de aprendizaje profundo junto al uso de videocámaras para detectar y analizar la densidad del tráfico. La técnica desarrollada se corresponde con un algoritmo You Only Look Once (YOLO), que permite usar pesos previamente entrenados o entrenar los pesos con un conjunto de datos personalizado, para la detección del tráfico y una serie de condiciones para el recuento del mismo. Para la evaluación de los resultados se han implementado diferentes técnicas de agrupamiento, *clustering* en inglés, con los datos obtenidos mediante la detección para obtener agrupaciones con los diferentes niveles de densidad de tráfico.

Keywords: Detección de tráfico, algoritmos de visión artificial, YOLO, CNN, algoritmos de agrupamiento

1. Introducción

En la actualidad, cada vez es más común escuchar hablar del término "Smart city" o ciudad inteligente, definida como una ciudad que hace uso de la tecnología, junto a los recursos más primitivos con la finalidad de mejorar su desarrollo de forma sostenible y en consecuencia mejorar la calidad de vida de sus habitantes. Uno de los aspectos a mejorar en las ciudades actuales se corresponde con el tráfico urbano, y es por ello, que desde hace años han aparecido diferentes propuestas que hacen uso de diferentes tecnologías y que tienen como finalidad mejorar el tráfico calculando la densidad de este [21,87,100]. Mediante estas densidades se busca obtener conclusiones en base a las diferentes franjas horarias o épocas del año con la finalidad de implementar un sistema de semaforización inteligente que minimice los tiempos de espera y mejore la fluctuación del tráfico.

Empresas muy conocidas y con un enorme poder como Samsung o Hauwei, ya han hecho propuestas de cara al futuro para desarrollar e implantar sistemas

inteligentes que mejoren las condiciones del tráfico en ciudades importantes de nuestro planeta. Otras empresas más enfocadas al desarrollo de dispositivos y hardware como Siemens han desarrollado diferentes sensores y cámaras que se encargan de detectar el tráfico como el caso de la cámara Sivicam[1], que se corresponde con una cámara CMOS combinada con un sistema de procesamiento de imágenes para la detección de vehículos. También se han hecho más conocidas empresas como Teledyne FLIR, compañía que se especializa en el diseño y producción de cámaras y sensores para un gran número de ámbitos y finalidades como la detección del tráfico.

Gracias a los avances en la tecnología y en la capacidad de computación durante estos últimos años, estas propuestas actuales suelen estar basadas en técnicas de aprendizaje profundo haciendo uso de una amplia variedad de algoritmos de visión artificial [6, 13, 60].

A la hora de plantear estas propuestas de ciudades inteligentes, diversos investigadores y medios de comunicación se hicieron eco para proponer los problemas que esto podría suponer en la sociedad en la que vivimos actualmente. Como bien indican Losavio et al. [64], el Internet de las cosas (IoT, en inglés) y el propio desarrollo de una ciudad inteligente tienen la necesidad de realizar una recogida de datos de forma masiva sobre las personas, por ejemplo, en forma de vídeo o imágenes obtenidas mediante la implantación de cámaras en la vía pública o mediante sensores. Todos estos datos son necesarios para el correcto funcionamiento de todos los elementos de una ciudad inteligente, y así, poder mejorar los servicios públicos y privados, sobre todo, centrándonos en la seguridad pública que es uno de los puntos primordiales que se buscan optimizar. El problema aparece en este punto, puesto que la propia seguridad pública, entre otras cosas, protege la libertad de estar solo y de tener una esfera privada de autonomía personal, hecho que se contradice con la necesidad de implantar sensores y cámaras que recojan datos de los usuarios en la vía pública sin haber un consentimiento previo. En este punto se empieza a hablar de investigación digital legal y de ciencia forense digital. La primera se encarga de aducir principios de seguridad de la información, regulación legal de la vida pública y privada y seguridad pública, y con relación a la segunda, se responsabiliza del análisis de pruebas encontradas en diferentes dispositivos electrónicos con la finalidad de tomar decisiones en diferentes ámbitos como el empleo, o decisiones judiciales. Por ello, se requiere de análisis que establezcan la procedencia, integridad y fiabilidad de los datos.

Estos motivos son más que suficientes para que exista la posibilidad de que estas mejoras en las ciudades puedan generar amenazas contra la personalidad y la privacidad, por lo que es necesario sopesar el bien generado frente a las amenazas que se puedan causar. Por este motivo los sistemas informáticos implementados serán los responsables tanto de lo bueno como de lo malo, producido con su uso y estas amenazas generadas no serán provenientes únicamente de la intrusión en

[1] https://assets.new.siemens.com/siemens/assets/api/uuid:2f786376-7405-4c40-a518-04bdee6b5cb7/sivicam-en.pdf

áreas privadas, sino que también se producirán por deducciones erróneas debido a análisis que no sean del todo fiables.

Este último tema tratado, hace que sea necesario mencionar otro problema que aparece en nuestra sociedad, debido a la informatización en los diferentes ámbitos de nuestras vidas. Este problema se corresponde con el abuso racial, étnico, por géneros, o por grupos de edad que se encuentra presente en la aplicación de técnicas de análisis de los datos. Esta discriminación se ha hecho cada vez más notoria con la implantación de sistemas informáticos que se encargan de tareas como la concesión de créditos, la contratación de personal en empresas o incluso la resolución de casos judiciales. Todos ellos temas muy importantes que pueden atentar contra los derechos de las personas si su funcionamiento no es el correcto o cuenta con un factor de discriminación racial. En la mayoría de los países desarrollados actuales existen leyes que se encargan de prohibir la discriminación en base a características como la raza, color, religión, nacionalidad, sexo o edad de las personas a la hora de la realización de la mayoría de las tareas mencionadas. Ahora bien, como exponen Pedreshi et al. [75] en las últimas décadas se han estudiado los motivos de esta discriminación en diferentes ámbitos de las vidas de las personas y se han obtenido pruebas de un trato injusto debido a la existencia de perfiles raciales. El problema llega en el momento en el que estos perfiles se proyectan en los modelos de clasificación usados para realizar estas tareas de contratación de personal y demás. Esta discriminación, presente en los modelos se debe a que el aprendizaje en base a datos históricos hace que el modelo acabe presentando los mismos prejuicios que se encuentran visibles en la sociedad. Estos autores estudian los riesgos de discriminación presentes en la minería de datos dando a entender que es un aspecto a tener en cuenta a la hora de implementar cualquier sistema que haga uso de estas técnicas.

Lo que resta de este trabajo se centrará en conseguir cumplir los objetivos del estudio que se corresponden con la revisión de la literatura relacionada con este tema, la descripción de los métodos utilizados en el trabajo para la detección y análisis del tráfico y la exposición de un caso de estudio junto a sus resultados, finalizando con las conclusiones y líneas de trabajo futuras.

En cuanto a la estructura del trabajo, este se encuentra estructurado de la siguiente manera: en la sección 2 se presentará una revisión del estado del arte de los temas relacionados con la detección y el análisis del tráfico, en la sección 3 se describirán los métodos desarrollados para realizar la detección de elementos de tráfico y los métodos utilizados para el análisis del tráfico, en la sección 4 se presentará el caso de estudio llevado a cabo para comprobar el funcionamiento del sistema, en la sección 5 se expondrán los resultados obtenidos, y por último, en la sección 6 se expondrán las conclusiones y las líneas de trabajo futuro de este trabajo.

2. Estado del arte

Debido a los avances y las mejoras de los algoritmos de visión artificial en los últimos años, se ha producido un aumento del interés en el estudio del campo del

análisis de tráfico, y se han realizado un gran número de estudios [6, 21], de los cuales una gran mayoría hacen uso de redes neuronales a través del aprendizaje profundo (Deep learning). Dichas redes están compuestas por capas convolucionales para implementar la detección y el recuento de los vehículos, mientras que otros trabajos más antiguos hacían uso de técnicas de procesamiento de imágenes [2], que se basan en las características de los objetos presentes en una imagen como el color y la forma, detectando su contorno, con la finalidad de decidir si están presentes los objetos deseados. Los trabajos que basan la detección en redes neuronales convolucionales, hacen uso de una gran variedad de algoritmos de visión artificial dependiendo del conjunto de datos utilizado para el entrenamiento y la disposición de los datos de estudio.

En cuanto al uso de técnicas de aprendizaje profundo, los primeros trabajos realizados se enfocaban en el uso de sensores de tráfico, aunque también hay estudios actuales como el realizado por Owais et al. [71], en él hacen un estudio del uso de diferentes tipos de sensores para la recolección de los datos y la aplicación de redes neuronales y lógica difusa para la detección de accidentes de tráfico. Otro estudio realizado en 2017 por Yu et al. [99] hace uso de sensores de detección y una arquitectura de red neuronal recurrente Long short-term memory (LSTM) para predecir las horas punta de tráfico y obtener características relevantes. Relacionado con este último punto, existen un gran número de estudios que se encuentran enfocados en el análisis del tráfico, que, en gran parte, se basan en la detección de semáforos. En el trabajo realizado por E. Lee y D. Kim [60], los autores proponen un sistema de detección de semáforos pequeños haciendo uso de una red neuronal convolucional que usa la pérdida de regresión focal para aumentar la precisión, con la finalidad de mejorar el flujo del tráfico. El algoritmo que utiliza para generar mapas de características, se corresponde con un algoritmo ResNet-101 [38] y gracias a la pérdida de regresión focal implementada durante el entrenamiento del detector mejoran su precisión. Hay más trabajos que usan regresores a partir de los datos para cumplir con este cometido como el realizado por Tomar et al. [87] en el que el objetivo de los investigadores se centra en regular el tráfico de forma eficiente, teniendo en cuenta características como el día de la semana, la hora o la distancia. Para ello combinan la regresión logística y la lógica difusa aplicada a los datos obtenidos y así consiguen que el sistema de control se adapte correctamente. En otro trabajo bastante reciente [52], los autores proponen un modelo de regresión lineal múltiple con el objetivo de predecir los tiempos óptimos de los semáforos. En su investigación se centran en dos aplicaciones, una haciendo uso de una cámara para detectar cuantos coches se encuentran en el carril y así calcular los tiempos, y otra para cuando no se sabe cuantos coches se encuentran en el carril, calculando los tiempos mediante un regresor a partir de datos antiguos.

Ahora bien, gracias al gran avance en las tecnologías y la resolución de las cámaras de vídeo actuales, recientemente se ha observado una tendencia a aportar soluciones que usan técnicas de aprendizaje profundo basadas en el uso de videocámaras. Estas soluciones tratan de obtener la densidad del tráfico y las horas punta mediante la detección de vehículos y peatones aplicando modelos de aprendizaje profundo a los vídeos de tráfico. Aparecen una gran variedad de

propuestas de diferentes autores como los estudios de Bedruz et al. [4] y Li et al. [61], en los que los investigadores implementan la detección y el conteo de los vehículos usando métodos basados en la extracción de borrones o manchas (*blob analysis*, en inglés). Estos métodos ofrecen mejores resultados que los obtenidos mediante el uso de sensores, pero su precisión aún podría ser mejorable por lo que aparecen métodos que se basan en la detección de vehículos con modelos entrenados con imágenes. Debido a los avances en la tecnología de visión artificial y el incremento de la potencia de computación, estos últimos métodos son los más utilizados y los que mayor precisión consiguen en sus resultados.

En el trabajo realizado por Chakraborty et al. [13], los autores implementan un sistema de detección de tráfico haciendo uso de videocámaras e implementando dos algoritmos diferentes de visión artificial para comparar sus resultados. Las dos técnicas de aprendizaje profundo que utilizaron para la detección de la congestión de tráfico fueron You Only Look Once (YOLO) y una red neuronal convolucional profunda, Deep Convolutional Neural Network (DCNN) y, por último, implementaron un algoritmo Support Vector Machine (SVM) para realizar las comparaciones de los algoritmos.

En otro trabajo realizado por Iyer et al. [48] sus autores realizan la detección de los vehículos implementando un algoritmo de detección de objetos Single Shot Detector (SSD), en concreto, el algoritmo de Google Single Shot MultiBox Detector (SSMD). Este tipo de algoritmos, entre los que también se enmarca YOLO, mencionado anteriormente, consiste en detectar varios objetos presentes en una imagen obtenida de un único *frame* mediante el uso de cajas múltiples. La arquitectura utilizada por estos algoritmos ofrece una mayor velocidad de procesamiento y una gran precisión comparándola con otras arquitecturas de aprendizaje profundo anteriores.

También es importante mencionar los algoritmos R-CNN [33], Fast R-CNN [32] y Faster R-CNN [78], que junto a YOLO han sido los más utilizados en estas tareas de detección de objetos puesto que son los que ofrecen una mayor rapidez. Existen numerosos estudios [25, 32, 44] en los que se han usado estos algoritmos de visión artificial, sobre todo, en las últimas décadas. Cabe destacar el trabajo de Chen et al. [15], en el que los autores se centran en el problema que genera la detección de objetos muy pequeños. Para resolver este problema plantean un algoritmo R-CNN, que se corresponde con una red neuronal convolucional profunda, pero ampliándolo mediante una red de propuesta de región (Region Proposal Network, RPN) con el objetivo de proponer múltiples objetos que son identificables en una imagen en particular. Cabe mencionar que este método fue propuesto por Ren et al. [78] en su trabajo publicado en 2016 llegando a ser muy popular y ampliamente utilizado.

Por otro lado, los avances en el análisis y minería de los datos, con la intención y finalidad de obtener información a partir de sus características, han hecho posible la aparición de muchos algoritmos que llevan a cabo estas tareas. Dentro de este ámbito se habla de algoritmos de *clustering* o segmentación que tienen como finalidad realizar una agrupación de la información o los datos en diferentes categorías o grupos [49]. Estas técnicas se han hecho cada vez más relevantes por su uso en diferentes ámbitos como el *marketing* digital, para centrar una campaña

en grupos de personas concretas, o para el análisis del tráfico, obteniendo grupos de menor a mayor congestión del tráfico en base a los datos de estudio. Se hace uso de una gran variedad de algoritmos de clustering para realizar todo tipo de agrupaciones siendo más o menos eficientes dependiendo del caso de estudio y de los datos a agrupar. En el trabajo realizado por Rokach et al. [79], los autores realizan un estudio de los diferentes métodos de clusterización clasificándoles según la metodología que usan para realizar la agrupación.

Uno de los métodos más simple y comúnmente usado se corresponde con el algoritmo K-means, que emplea un criterio que busca el mínimo error medio entre las "distancias" de todos los elementos a clasificar. En la literatura, numerosos autores han usado este algoritmo implementándolo de formas diferentes. En el trabajo realizado por Kanungo et al. [51], los autores implementan este método mediante una de sus heurísticas populares más conocidas, que recibe el nombre de algoritmo de agrupación K-means de Lloyd, en honor a su creador. Existen numerosos artículos en los que se realiza una agrupación de la congestión de tráfico mediante la implementación de este algoritmo, como el trabajo realizado por Pattanaik et al. [74], en el que realizan una estimación de la congestión del tráfico en diferentes calles con la finalidad de obtener la ruta óptima, o el estudio de Hongsakham et al [42], en el que estiman el nivel de congestión de una calle en base a los datos de tráfico obtenidos.

Otro de los métodos de agrupación ampliamente usado se corresponde con el algoritmo de maximización de la esperanza, también conocido como Expectation Maximization (EM), el cual se enmarca en los métodos de agrupación basados en densidad. Este método se caracteriza por ser un algoritmo de máxima verosimilitud de propósito general, usado en problemas de datos perdidos, y se puede aplicar en un gran número de ámbitos como podemos ver, por ejemplo, en la segmentación de imágenes realizada por Tatiraju et al. [86] en un trabajo en el que hacen uso de este algoritmo y del algoritmo K-means para llevar a cabo esta tarea. En otro trabajo publicado por Liu et al. [63], los autores implementan una clasificación en tiempo real del estado de la congestión del tráfico mediante un modelo de mezcla gaussiano, también denominado, Guassian Mixture Model (GMM), que hace uso de un algoritmo EM para llevar a cabo esta tarea.

Otro algoritmo de agrupación utilizado, y que a diferencia de los anteriores se corresponde con un método no supervisado, es el algoritmo Mean Shift, el cual representa un procedimiento general no paramétrico de agrupación [20]. Este algoritmo se caracteriza por no hacer uso de suposiciones integradas sobre la forma de distribución ni el número de grupos en los que se agruparán los datos. Existen numerosos trabajos en los que este método es utilizado para diferentes aplicaciones. En concreto, también hay estudios en los que se hace uso de este método para agrupar datos numéricos en grupos como en el estudio de Chang-Chien et al. [14].

También existen métodos de agrupamiento que hacen uso de una red neuronal [23]. Un claro ejemplo de ello es el algoritmo neuronal Self-Organizing Map (SOM) o mapa autoorganizado. En este tipo de algoritmos cada grupo se representa como una neurona, al igual que cada uno de los datos de entrada y se forma una red neuronal de una sola capa con el conjunto de los datos. Existen nume-

rosos trabajos en la literatura en los que los autores han utilizado este algoritmo para implementar la agrupación de datos en sus estudios. En el trabajo realizado por Gu et al. [36], los autores usan el algoritmo SOM para agrupar y reconocer patrones de tráfico entre diferentes segmentos de paradas de bus. Otro ejemplo de este tipo de algoritmos, y que está inspirado en el algoritmo neuronal SOM, es el algoritmo Neural Gas, que se caracteriza por buscar las representaciones de datos óptimas basándose en vectores de características. Existen un gran número de artículos en los que los autores han utilizado este método para llevar a cabo la agrupación de datos como los publicados por Ghesmoune et al. [30] y Daszykowski et al. [18].

Por último, mencionar las redes Growing Cell Structures (GCS) [26], que pueden considerarse como otra modificación del algoritmo neuronal SOM integrando funciones de reclutamiento y depuración de los nodos para llevar a cabo tareas de agrupación. Desde la presentación de este modelo de red neuronal por parte de Friszke, muchos han sido los investigadores que han optado por su implementación para llevar a cabo tareas de agrupación como ocurre en el trabajo realizado por Su et al. [84], en el que los autores aplican el algoritmo GCS para llevar a cabo la agrupación de datos y posteriormente aplican un algoritmo de cuantificación de vectores a los resultados obtenidos para refinarlos y mejorar el rendimiento final.

Por lo tanto, y después de haber realizado un estudio de los diferentes métodos y técnicas utilizadas para lograr estas tareas, en este trabajo se va a analizar el uso del algoritmo YOLO en su versión 4 para la detección de los diferentes elementos de tráfico, y se van a utilizar los algoritmos de agrupamiento K-means, EM, Mean Shift y SOM para el estudio de los resultados.

3. Metodología

En términos generales, en esta sección se detallan en profundidad los métodos utilizados para llevar a cabo las tareas de detección de los diferentes elementos del tráfico, el modo en el que se realiza el recuento de estos elementos y los métodos utilizados para obtener el agrupamiento de los datos que permitirán obtener conclusiones respecto a la congestión del tráfico.

Para ello, a continuación, se desarrollará el sistema implementado haciendo hincapié en dos aspectos: cómo se lleva a cabo la detección de los elementos del tráfico y, por último, qué métodos se han implementado para realizar el agrupamiento de los datos para el estudio de la congestión.

3.1. Detección del tráfico mediante YOLOv4 y Tensorflow

Desde un principio, la intención de este trabajo consistía en la detección de los diferentes elementos de tráfico en vídeos o en tiempo real a partir de la información obtenida a través de videocámaras. Para llevar a cabo estas tareas de detección de objetos, los sistemas actuales hacen uso de redes neuronales convolucionales como se ha visto en la sección 2. YOLO es un ejemplo, pero

la diferencia que presenta YOLO en todas sus versiones, respecto a propuestas anteriores, es que para realizar el proceso de detección sólo mira y evalúa una imagen una única vez, mientras que otros modelos de clasificación de redes neuronales evalúan una misma imagen cientos o miles de veces modificando las ubicaciones o las escalas de esta. YOLO hace este proceso con una sola red neuronal convolucional, que predice simultáneamente en cada imagen, múltiples cuadros delimitadores y las probabilidades de la clase con la que se corresponde cada uno de ellos. Por todos estos motivos, YOLO es extremadamente rápido y fácil de generalizar a diferentes situaciones.

Entrando en mayor detalle acerca de la arquitectura usada por YOLOv4 en este trabajo, esta se corresponde con una red neuronal convolucional simple. En la Tabla 1 se muestra la composición de esta red que se caracteriza por usar solamente dos tipos de capas estándar: capas de convolución con un núcleo de 3x3 y capas de agrupación máxima (*max pooling*, en inglés) con un núcleo de 2x2. Por último, la red cuenta con una capa convolucional con un núcleo 1x1, que tiene la función de reducir los datos a un formato 13x13xN (siendo N, (número de clases + 5) * 5, el número 5 representa el número de cuadros delimitadores que detecta YOLO para cada una de las cuadrículas), que es el tamaño de la cuadrícula en la que se divide cada imagen, como se puede observar en la Figura 1 [9]. En este punto es importante mencionar que para este trabajo se han realizado pruebas con dos conjuntos de datos que se explicarán en la sección 4, Caso de estudio, siendo el primero, el compuesto por los pesos de YOLO preentrenados que contiene datos de 80 clases de objetos, por lo que N es igual a 425, y un conjunto de datos con 2 clases de objetos, siendo N 35. Volviendo al funcionamiento de la arquitectura, cada una de las celdas de la cuadrícula tienen asociados 35 canales, que se representan como 35 números que simbolizan los datos de los cuadros delimitadores y las predicciones de la clase. Los 35 números se corresponde con los 3 elementos de datos de los cinco cuadros delimitadores que se predicen para cada una de las celdas, siendo estos siete elementos los siguientes:

- Las coordenadas x e y, la anchura y la altura del rectángulo del cuadro delimitador, que definen la posición.
- El valor de la confianza de la clase o clases predichas.
- Por último, la distribución de la probabilidad sobre las dos clases (congestionada y no congestionada).

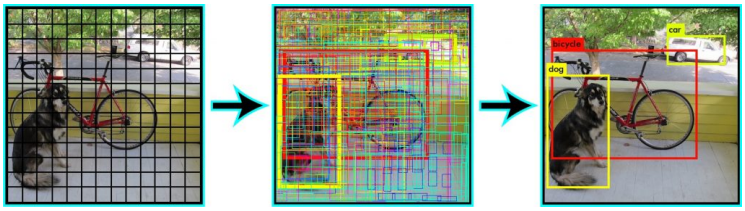


Figura 1: División de la imagen en celdas realizada por YOLO y obtención de los cuadros delimitadores

Tabla 1: Arquitectura del modelo YOLO usado

Capa	Núcleo	Avance	Formato de salida
Entrada			[416, 416, 3]
Convolucional	3x3	1	[416, 416, 16]
Max Pooling	2x2	2	[208, 208, 16]
Convolucional	3x3	1	[208, 208, 32]
Max Pooling	2x2	2	[104, 104, 32]
Convolucional	3x3	1	[104, 104, 64]
Max Pooling	2x2	2	[52, 52, 64]
Convolucional	3x3	1	[52, 52, 128]
Max Pooling	2x2	2	[26, 26, 128]
Convolucional	3x3	1	[26, 26, 256]
Max Pooling	2x2	2	[13, 13, 256]
Convolucional	3x3	1	[13, 13, 512]
Max Pooling	2x2	2	[13, 13, 512]
Convolucional	3x3	1	[13, 13, 1024]
Convolucional	3x3	1	[13, 13, 1024]
Convolucional	1x1	1	[13, 13, N]

Los pasos que sigue YOLO para la detección son los siguientes:

1. Primero, se redimensiona la imagen de entrada a un formato 416x416.
2. Se pasa la imagen redimensionada a la red neuronal convolucional simple.
3. Esta red se encarga de obtener el tensor con formato 13x13xN (ya explicado) que contiene las cajas delimitadoras de las celdas de la cuadrícula.
4. Por último, se calculan los valores de confianza de las clases predichas de todos los cuadros delimitadores y se desestiman los que se encuentren por debajo de un valor umbral predefinido.

Por último, y con la intención de mejorar el rendimiento del sistema e implementar funciones personalizadas programadas en Python, se ha decidido convertir los pesos previamente entrenados con la red neuronal convolucional YOLOv4 en un modelo de TensorFlow con el que se llevará a cabo la detección. La particularidad de TensorFlow es que permite una implementación de la detección usando el lenguaje Python de una forma muy sencilla puesto que los propios

nodos y tensores de TensorFlow son objetos de Python. Por estos motivos se ha utilizado esta herramienta en conjunto con el algoritmo YOLO. En la sección 4 Caso de estudio, se detallará en mayor profundidad la utilización de este sistema.

3.2. Métodos de agrupamiento para el estudio de los datos

El otro gran desafío planteado en este trabajo se corresponde con la aplicación de algoritmos de agrupación a los datos de flujo, obtenidos tras aplicar el sistema de detección a los elementos de tráfico durante el caso de estudio. Estos algoritmos se corresponden con el algoritmo K-means, el algoritmo EM, el algoritmo Mean Shift y el algoritmo neuronal SOM, todos ellos mencionados en la sección 2 y que han sido usados ampliamente en la literatura. A continuación, se describirá el funcionamiento de cada uno de estos métodos.

Algoritmo K-means Se corresponde con un algoritmo de tipo no supervisado que agrupa los elementos en k grupos definidos, basándose en sus características. Para llevar a cabo el agrupamiento se busca minimizar la suma de distancias entre cada objeto y el centroide del grupo, y esta distancia suele ser la distancia cuadrática. Se suele organizar este algoritmo en tres acciones:

1. **Inicialización**: después de definir el número de grupos k de la representación, el algoritmo establece los k centroides en el espacio de datos de forma aleatoria por lo general.
2. **Asignación de cada elemento a un grupo**: cada uno de los elementos se va asignando a uno de los grupos.
3. **Actualización de los centroides**: tras cada asignación se comprueba la posición de los centroides de cada grupo eligiendo nuevas posiciones como promedio de los elementos de cada grupo.

Posteriormente, se repiten las últimas dos acciones hasta que no se actualizan los centroides consiguiendo así resolver un problema de optimización y minimizar la suma de las distancias cuadráticas de todos los elementos con su centroide. En cuanto a la representación de los elementos, éstos se representan con vectores reales de n dimensiones $(x_1, ..., x_n)$ y el algoritmo define los k grupos que minimizan la suma de las distancias entre los objetos, dentro de cada grupo S = $S_1, ..., S_k$, a su centroide. La ecuación (1) representa este problema de asignación

$$\min_{\mathbf{S}} E\left(\boldsymbol{\mu_i}\right) = \min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\|\mathbf{x}_j - \boldsymbol{\mu}_i\right\|^2 \quad (1)$$

donde S es el conjunto de datos cuyos elementos son los objetos que queremos agrupar x_j representados por vectores, donde cada uno de sus elementos representa una característica o atributo. Tendremos k grupos con su correspondiente centroide μ_i. En cuanto a la función que se encarga de la actualización de los centroides se toma cada centroide como el promedio de los elementos de cada grupo mediante la función (2):

$$\frac{\partial E}{\partial \boldsymbol{\mu}_i} = 0 \implies \boldsymbol{\mu}_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j \quad (2)$$

Algoritmo EM En lo que al algoritmo EM o *expectancy maximization* respecta, se corresponde con un algoritmo de agrupación no supervisada al igual que K-means. El funcionamiento del algoritmo se basa en calcular los grupos alternando dos acciones de forma iterativa:

1. **Expectación**: se encarga de asignar cada elemento a un grupo de forma probabilística. En resumen, se calcula la probabilidad de que el elemento pertenezca a cada uno de los grupos respectivamente.
2. **Maximización**: se actualizan los parámetros de cada grupo (ubicación media ponderada de todos los elementos y la matriz de varianza-covarianza) en función de los elementos que pertenecen al grupo, que se habían asignado en el primer paso calculando su probabilidad de pertenencia.

En cuanto a las diferencias que presenta respecto al algoritmo K-means, el algoritmo EM es generativo, lo que quiere decir, que este método busca entender cómo se han generado los datos con la finalidad de generar una mayor cantidad de datos (falsos) para mejorar la precisión de la agrupación.

Ahora se presentará un ejemplo, donde se tienen dos grupos y los elementos de datos representados mediante un vector $(x_1, ..., x_n)$, para explicar el funcionamiento del algoritmo. Los grupos siguen una distribución normal N (μ, σ^2) y la probabilidad de que un elemento forme parte de un grupo es π. En definitiva, para los grupos se tendrán los parámetros: π o probabilidad de mezcla conjunta a todos los grupos, μ o media de todos los elementos de un grupo y σ o desviación estándar de un grupo, y todos ellos se denotarán colectivamente como θ.

A continuación, se llevaría a cabo la primera acción del algoritmo y se calcularía la función de densidad de probabilidad ϕ de que un elemento x pertenezca a uno de los grupos como muestra la ecuación (3)

$$p(x;\theta) = \pi\phi(x;\mu_1,\sigma_1) + (1-\pi)\theta(x;\mu_2,\sigma_2) \quad (3)$$

Y la probabilidad asociada al conjunto de datos compuesto por n elementos se suele representar como una suma logarítmica (4):

$$l(\theta;data) = \sum_{i=1}^{n} log[\pi\phi(x;\mu_1,\sigma_1) + (1-\pi)\theta(x;\mu_2,\sigma_2)] \quad (4)$$

Ahora tras realizar esta función de expectación, lo que se busca es maximizar (paso 2 del algoritmo) el resultado buscando que los elementos estén asociados a un grupo con la mayor probabilidad posible (5). Para realizar esta maximización se hace uso de una variable latente denotada como Δ. Esta variable se corresponde con una variable binaria y determina si un elemento se encuentra en un grupo o en el otro, de forma que si conocemos este valor para cada uno de los elementos conseguiremos maximizar las probabilidades.

$$l(\theta;data,\Delta) = \sum_{data}[(1-\Delta_i)log\phi(x_i;\mu_1,\sigma_1) + \Delta_i log\phi(x_i;\mu_2,\sigma_2)] + \sum_{data}[(1-\Delta_i)log(1-\pi) + \Delta_i log\pi] \quad (5)$$

Hay que tener en cuenta que las sumas se encuentran fuera del algoritmo y que se ha añadido otra suma para poder tener en cuenta la probabilidad de cada Δ. Además, hay que seleccionar los valores de los estimadores de máxima verosimilitud. Para μ, se obtiene la media de cada uno de los grupos; para σ, se calcula la desviación estándar de cada grupo y para π, se toma la proporción muestral de los elementos del segundo grupo. Por último, para llevar a cabo el algoritmo primero se seleccionan arbitrariamente los parámetros, luego se realiza el paso de la expectación calculando los valores para Δ y seguidamente la función de la maximización calculando los valores de los estimadores de máxima verosimilitud para volver realizar el paso 1, y así sucesivamente se repiten estos dos pasos hasta que se obtiene el resultado deseado.

Algoritmo Mean Shift El algoritmo Mean Shift o desplazamiento de la media se enmarca también dentro de los algoritmos de tipo no supervisado con la particularidad de que asigna los elementos de datos a los grupos de forma iterativa desplazando los elementos hacia el modo, entendido como la mayor densidad de elementos dentro del grupo. De esta forma, se denomina a este método como un algoritmo de búsqueda de modo. A diferencia de los dos algoritmos anteriores, en este método no se necesita especificar el número de grupos en los que se enmarcarán los elementos y el algoritmo es el que se encarga de determinar el número de grupos óptimo respecto a los elementos.

En cuanto a su funcionamiento, el algoritmo de desplazamiento de la media se basa en el concepto de estimación de la densidad del núcleo (EDN), donde los elementos se muestrean a partir de una distribución de probabilidad. Por lo tanto, la EDN es un método que estima la distribución subyacente y en definitiva se corresponde con una función de densidad de probabilidad aplicada al conjunto de datos de entrada. Este método comienza colocando un núcleo (entendido como una función de ponderación que se utiliza en la convolución) a cada uno de los elementos del conjunto de datos. En cuanto al núcleo utilizado, existen diferentes tipos, siendo el núcleo gaussiano el más utilizado y la suma de todos los núcleos da lugar a una función de densidad de ejemplo de superficie de probabilidad. Esta suma dependerá también del parámetro de ancho de banda del núcleo utilizado.

Para comprender mejor este funcionamiento, supongamos que tenemos un conjunto de datos $(x_1, ..., x_n)$ de elementos en un espacio d-dimensional que han sido obtenidos de una población mayor, y que se ha elegido un núcleo K con un ancho de banda h. La función del núcleo (6) junto a estos datos devuelve el siguiente estimador de densidad del núcleo para llevar a cabo la función de densidad de la población completa.

$$f_k(u) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{u-u_i}{h}) \quad (6)$$

Esta función tiene como requisitos satisfacer las dos condiciones que se muestran a continuación. La primera (7) es necesaria para asegurar que la estimación está normalizada y la segunda (8) está relacionada con la simetría del espacio definido.

$$\int K(u)du = 1 \quad (7)$$

$$K(u) = K(|u|) \; para \; todos \; los \; valores \; de \; u \quad (8)$$

Las dos funciones de núcleo más utilizadas que cumplen estas condiciones son la uniforme (9), y la ya mencionada anteriormente, la función gaussiana (10).

$$K(u) = \frac{1}{2} \begin{cases} 1 & -1 \leq |u| \leq 1 \\ 0 & else \end{cases} \quad (9)$$

$$K(u) = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{1}{2}|u|^2} \quad (10)$$

Algoritmo SOM El último de los algoritmos utilizado para la agrupación se corresponde con el algoritmo neuronal SOM o mapa autoorganizado. Es un tipo de red neuronal que se entrena mediante aprendizaje no supervisado con la finalidad de generar una representación discretizada de baja dimensión a partir del conjunto de datos de entrada, denominada mapa. Véase la Figura 2. Por este motivo, se considera a este algoritmo como un método de reducción de la dimensionalidad. En comparación a otras redes neuronales, los mapas autoorganizados aplican un aprendizaje competitivo y hacen uso de una función de vecindad para conservar las propiedades topológicas del espacio de entrada.

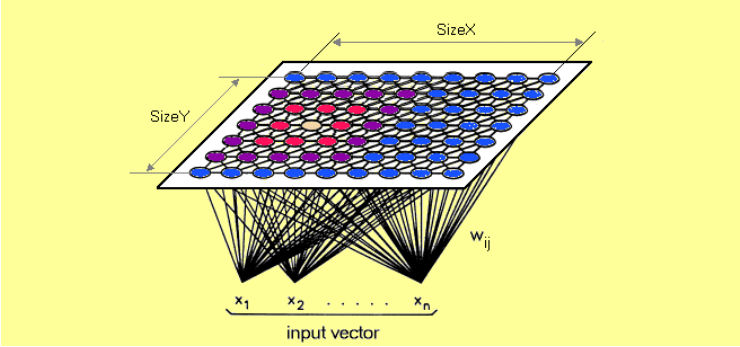


Figura 2: Arquitectura del algoritmo neuronal SOM

En cuanto al funcionamiento de este algoritmo, se basa en que cada elemento del conjunto de datos se reconoce a sí mismo compitiendo por ser representante y se comienza la creación del mapa inicializando los vectores de pesos de estos elementos. A continuación, se selecciona un vector de muestra de forma aleatoria y se busca en el mapa el peso que mejor representa a ese vector de muestra. El peso que se elige se recompensa por haber sido elegido, al parecerse más al vector de muestra aleatorio y los pesos vecinos de ese peso también son recompensados por parecerse. Gracias a este funcionamiento el mapa va creciendo y, por lo general, se generan formas cuadradas o rectangulares en el espacio de características de dos dimensiones.

Los pasos seguidos por estos algoritmos se organizan de la siguiente manera:

1. Se inicializan los pesos de cada uno de los nodos.
2. Se selecciona un vector de muestra al azar del conjunto de datos inicial.
3. Cada uno de los nodos del mapa se examina para obtener cuáles son los pesos más parecidos al vector de muestra, y al nodo más parecido se le conoce como la Unidad de Mejor Correspondencia (UMC).
4. A continuación, se calcula la vecindad de esta UMC.
5. Ahora se recompensa al peso ganador por parecerse más al vector muestra y cuanto más cerca esté un nodo de este peso, más se modificarán sus pesos y cuanto más lejos, menos aprenderá.
6. Por último, se repiten los pasos del 2 al 5 N veces, definidas.

4. Caso de estudio

En esta sección se describirá el caso de estudio llevado a cabo para comprobar la correcta funcionalidad del sistema de detección, así como explicar el recuento de los elementos de tráfico y el análisis de los resultados de congestión obtenidos. Además, se mostrará el origen de los datos de vídeo de entrada y los datos utilizados para el entrenamiento del modelo de detección YOLOv4.

Lo primero que hay que mencionar en esta sección es el sistema desplegado para llevar a cabo este trabajo. Se desarrolló una aplicación en lenguaje de programación Python, haciendo uso de las bibliotecas OpenCV y TensorFlow, desplegada con el kit de desarrollo NVIDIA Jetson AGX Xavier[2], que se corresponde con un sistema en módulo embebido de la familia NVIDIA AGX Systems. Este sistema incluye una GPU Volta integrada con núcleos Tensor, dos aceleradores de aprendizaje automático profundo, una CPU NVIDIA Carmel ARMv8.2 de ocho núcleos, 32 GB de LPDDR4x de 256 bits con 137 GB/s de ancho de banda y 650 GB/s de E/S de alta velocidad, incluyendo PCIe de cuarta generación y 16 conexiones de cámara de MIPI CSI-2.

Como dato, este sistema estaba pensado para estar integrado junto a un despliegue de cámaras compatibles y así realizar una detección de congestión del tráfico en tiempo real. Lamentablemente, por falta de tiempo y por motivos legales no fue posible realizar este despliegue y para el caso de estudio se han realizado detecciones en vídeos obtenidos de Youtube[3], de una cámara implantada en una de las calles de Tokio. En la Figura 3, se puede observar un ejemplo de los vídeos mencionados y utilizados para este caso.

Respecto a esto, hay que tener en cuenta el hecho de que en Japón se circula por la izquierda, es decir, en dirección contraria a como se circula en España, por lo que habrá que tenerlo en cuenta porque cualquier cambio podría modificar los resultados de congestión obtenidos.

El siguiente punto a tratar en el caso de estudio está relacionado con los datos que se han utilizado para generar el modelo de pesos de YOLO4. En este trabajo se han realizado pruebas con el modelo de pesos previamente entrenado

[2] Nvidia Jetson AGX Xavier: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[3] Fuente de los vídeos: https://www.youtube.com/watch?v=RQA5RcIZlAM

Figura 3: Ejemplo de los vídeos del caso de estudio

de YOLO4 y pruebas con un modelo de pesos generado a partir de imágenes obtenidas de la plataforma de Google Open Images Dataset V6[4], que contiene un gran número de imágenes de varias clases de objetos y para diferentes usos. Como ya había mencionado anteriormente, el primer modelo de pesos contiene el entrenamiento de 80 clases de objetos y se puede obtener a través de Internet del repositorio del proyecto de YOLO y para realizar la detección con este modelo simplemente hay que convertirlo en un modelo de TensorFlow y ejecutar la aplicación de detección.

Por otro lado, para la generación del modelo de pesos personalizado se han obtenido dos conjuntos de datos, uno de entrenamiento compuesto por 3000 imágenes (1500 de la clase persona y 1500 de la clase vehículo) y un conjunto de prueba compuesto por 600 imágenes (300 para la clase persona y 300 para vehículo), con la particularidad de que cada una de las imágenes tiene que ir acompañada de un fichero de anotaciones que contiene todos los parámetros correspondientes al cuadro o cuadros delimitadores de los objetos presentes en la imagen. En la Figura 4, se muestra como se representan los cuadros delimitadores de los objetos en YOLO y, por lo tanto, el formato que tendrá que seguir cada uno de los ficheros de anotación.

Para llevar a cabo este entrenamiento se ha desarrollado un cuaderno de Google Colab[5], servicio *cloud* basado en los cuadernos de Jupyter y que permite usar GPUs de Google gratuitas, para poder realizar las tareas de entrenamiento simultáneamente con las tareas de detección. Tras la ejecución de estas tareas se obtuvieron datos pertinentes a la precisión de detección de este modelo personalizado, entrenado con imágenes, que se pueden observar en contraposición con la precisión del modelo pre-entrenado de YOLO comparando la precisión media mAP50 [39], en la Tabla 2, y en las Figuras 5 y 6 se observa la diferencia de la detección de un mismo *frame* obtenido de los vídeos utilizados, en el cuaderno.

[4] Fuente de las imágenes: https://storage.googleapis.com/openimages/web/index.html
[5] Google Colab: https://colab.research.google.com/notebooks/welcome.ipynb?hl=es

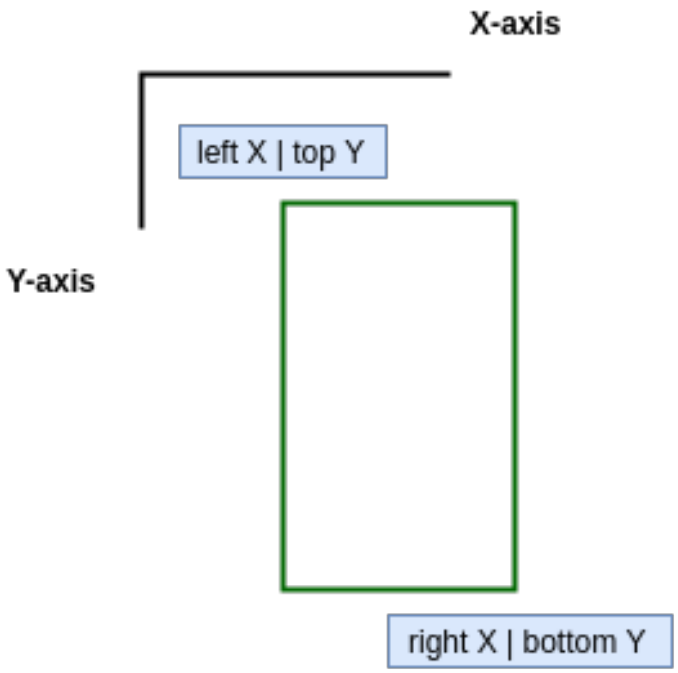


Figura 4: Definición de los cuadros delimitadores para el entrenamiento

Como se puede ver a simple vista, la precisión es mayor en el caso de la utilización del modelo pre-entrenado de YOLO con el COCO Dataset 2017[6], por lo que se utilizará este modelo en las tareas de detección posteriores, para obtener la congestión de tráfico del cruce.

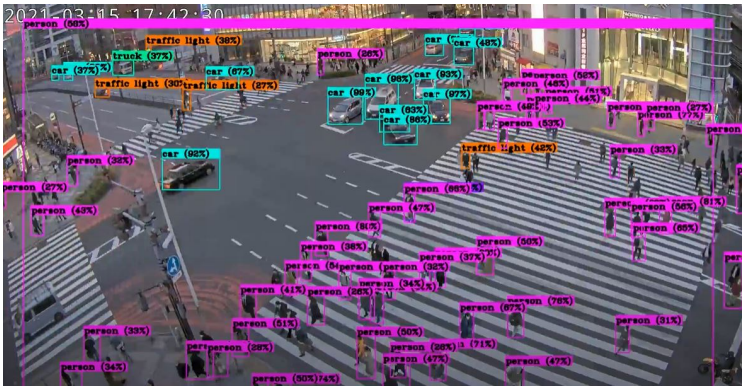


Figura 5: Ejemplo de detección usando el modelo pre-entrenado de YOLO

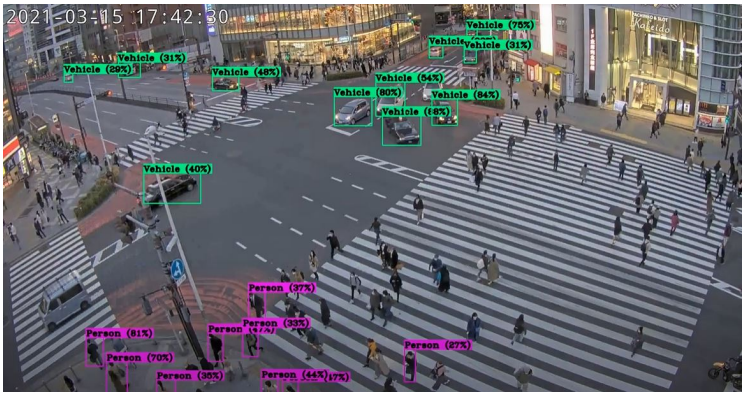


Figura 6: Ejemplo de detección usando un modelo YOLO personalizado entrenado

[6] COCO Dataset: https://cocodataset.org/#home

Tabla 2: Comparativa de precisión de los modelos

Algoritmo	Datos	Precisión mAP50
YOLOv3	COCO 2017 Dataset	52,32 %
YOLOv4	COCO 2017 Dataset	57,33 %
YOLOv4	Dataset personalizado	34,36 %

Después de saber qué modelo de pesos será el utilizado para la detección, se pasa a detallar cómo funciona esta tarea y cómo se consiguen los datos de congestión. Primero es importante mencionar que, gracias a las funciones que aporta el módulo OpenCV, se han definido dos sectores en el cruce, siendo el sector 1 el de la derecha (que representa el paso de peatones más cercano a la cámara) y el sector 2 el de abajo a la izquierda (que representa un paso de peatones más lejano y con peor precisión) que se observan en los vídeos, como podemos ver en la Figura 7. También, hay que destacar que a de cada objeto que detecta el algoritmo se obtienen dos valores, su clase (por ejemplo, *car*, que significa que es un coche) y la confianza de esa detección, que es el valor numérico que aparece al lado de la clase y su rango de valores va de 0 a 1.

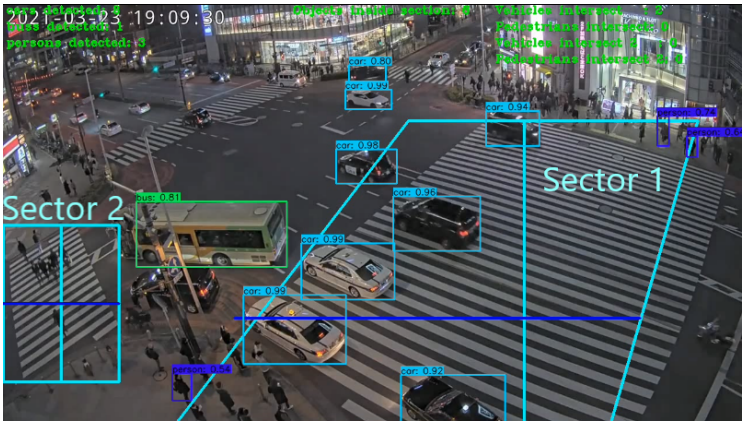


Figura 7: Ejemplo de segmentación en sectores y recuenteo

Además, por cada sector se han definido dos líneas que marcan las coordenadas referentes al recuento de los elementos del tráfico. De esta forma cuando un vehículo o un peatón pasa por la línea que se corresponde a su clase se cuenta como uno y se aumenta el contador de esa clase. Hay que mencionar en este punto, que en un principio se había planteado un algoritmo de seguimiento [43] para mejorar la precisión del recuento, pero por problemas surgidos a la hora de la implementación se cambió esta idea por la de usar una serie de condiciones matemáticas relacionadas con las coordenadas. El principal problema de este

planteamiento se debe a que se puede dar el caso, en el que justo cuando un vehículo o un peatón va a cruzar la línea imaginaria de recuento, el algoritmo de detección no lo detecte y en consecuencia no se sume al contador y la precisión del sistema de recuento baje.

Respecto a las detecciones de los elementos, se pueden observar diferencias relacionadas con el sector y con la hora del día. En cuanto al primer caso, el sector 2 (el de la izquierda) se encuentra en una posición más alejada de la cámara, por lo que esto tiene como consecuencias que la detección tenga una precisión más baja sobre todo en el caso de los peatones, ya que estos se corresponden con elementos bastante pequeños. En la Figura 7 se puede observar la diferencia de detección entre las dos secciones en la misma hora de un día.

Por último, también se pueden observar cambios en los resultados del algoritmo de detección en relación con la hora del día en la que se está realizando la detección. De esta forma, podemos observar en las Figuras 8 y 9 que cuando es de noche, debido a que hay menos luz, la cámara cuenta con peor calidad y eso repercute en que el algoritmo de detección pierda precisión y, en resumen, que se cuente peor.

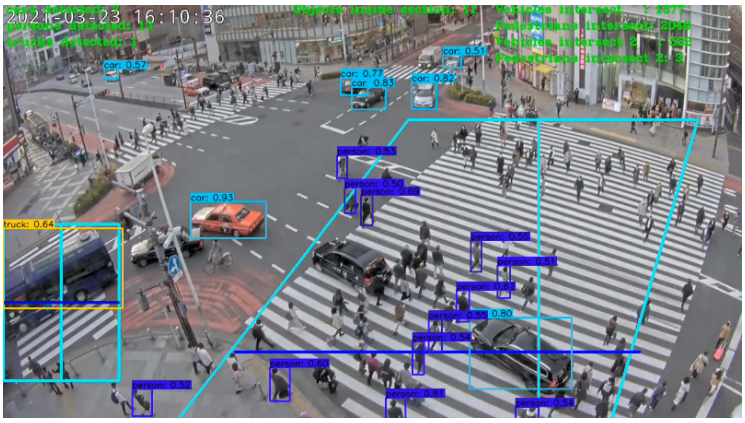


Figura 8: Ejemplo de detección del modelo durante el día

En la siguiente sección se van a presentar y detallar los resultados de congestión de tráfico obtenidos a partir de la detección de las 24 horas de un día mediante la aplicación de los algoritmos de agrupamiento previamente mencionados.

5. Resultados

Los resultados de este trabajo se corresponden con los obtenidos después de aplicar el algoritmo de detección durante 24 horas de un día. Para agrupar estos datos se han dividido las 24 horas de vídeo en franjas de 15 minutos, y para

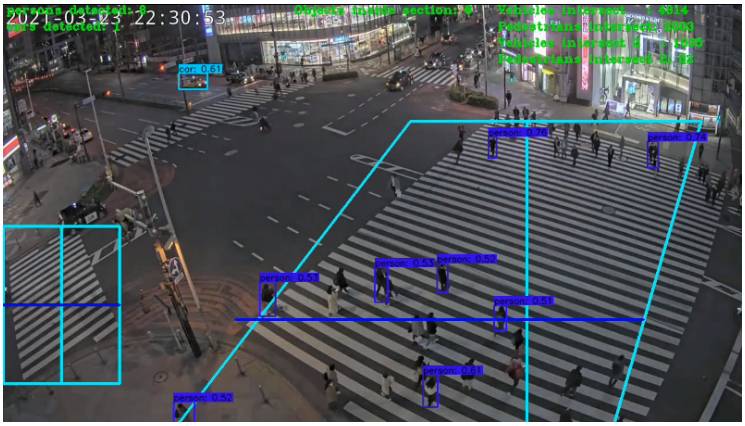


Figura 9: Ejemplo de detección del modelo durante la noche

cada franja se obtiene el número de peatones y de vehículos que han pasado por las secciones, dando lugar así a un conjunto de datos con coordenadas x e y, y a representaciones bidimensionales. Quiero mencionar que solo se tendrán en cuenta los resultados obtenidos para el sector 1, puesto que en el caso del sector 2, debido a los problemas explicados en la sección anterior, el número de peatones detectados era muy inferior a la realidad. Tras aplicar los cuatro algoritmos de agrupación explicados en la sección 2 a los datos de congestión racabados se han obtenido resultados sobre la densidad de tráfico. En la figura 10 se observa la agrupación de los datos realizada por el algoritmo K-means, en la figura 11 se puede ver la agrupación de los datos realizada por el algoritmo EM, en la figura 12 se observa la agrupación de los datos realizada por el algoritmo Mean Shift, y por último en la figura 13 se muestra la agrupación de los datos realizada por el algoritmo SOM.

En relación con los resultados obtenidos, podemos ver como en todos los algoritmos se diferencian tres grupos o *clusters* principales, un grupo representado con el color verde, como el grupo que contiene las horas con menor concentración de tráfico; un grupo representado con el color amarillo, como el grupo con una concentración media; y, por último, un grupo representado con el color rojo y que se corresponde con las horas en las que se produce una mayor concentración de tráfico. Se han obtenido fácilmente relaciones entre las franjas horarias, observando que las horas con menor concentración son las que se corresponden con la madrugada y las horas con mayor concentración se corresponden con las horas a las que la gente suele ir o salir del trabajo, mientras que el grupo amarillo representa las franjas horarias en las que la gente no suele estar en la calle durante el día.

Por último, cabe mencionar que los tres primeros algoritmos arrojan resultados similares al agrupar las densidades de tráfico dependiendo proporcionalmente del número de peatones y vehículos, ya que estos tres métodos de agrupamiento basan su funcionamiento en fórmulas matemáticas. Por otro lado, el algoritmo

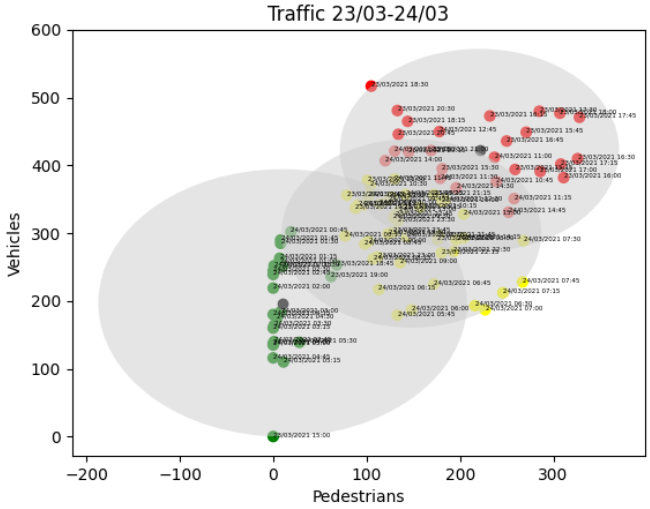


Figura 10: Agrupación de la congestión de tráfico con K-means

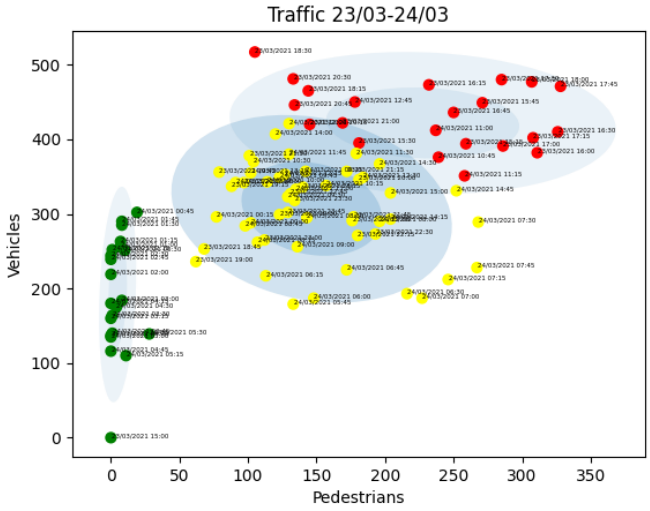


Figura 11: Agrupación de la congestión de tráfico con EM

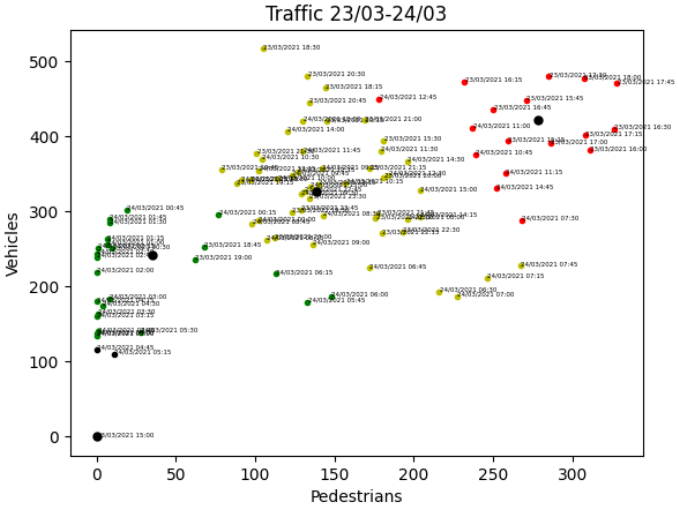


Figura 12: Agrupación de la congestión de tráfico con Mean Shift

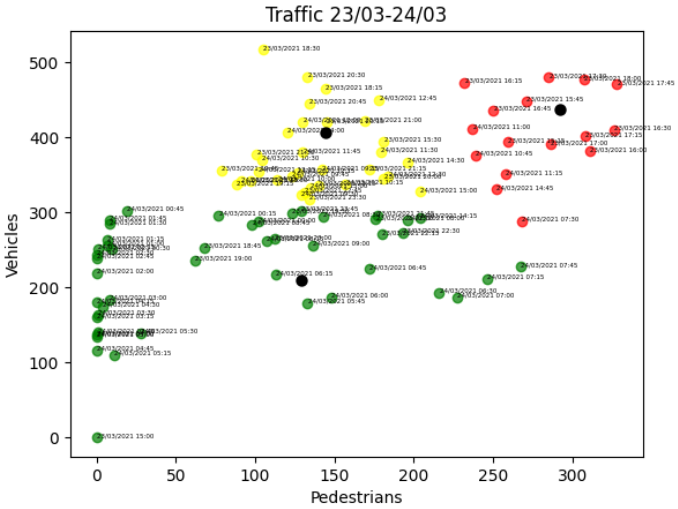


Figura 13: Agrupación de la congestión de tráfico con SOM

SOM, que hace uso de redes neuronales, devuelve un resultado algo diferente a los demás. En su caso, se puede observar como hay franjas horarias que con aproximadamente 200 vehículos y 300 peatones se han añadido al grupo de menos congestión, cuando en los demás algoritmos se habían añadido al grupo de congestión media. Por último, mencionar que, al tratarse de métodos de aprendizaje no supervisado y no tener los datos etiquetados previamente, no se ha realizado una comparación de métricas como la precisión o *accuracy*. En el caso del algoritmo SOM, se ha obtenido un error de cuantización del 85 %, valor bastante alto que representa la diferencia entre los datos de entrada y los resultados obtenidos por este algoritmo tras el entrenamiento.

6. Conclusiones y trabajo futuro

El campo de las ciudades inteligentes y el análisis del tráfico es un ámbito que viene siendo muy estudiado desde hace años y al que se le está dando gran importancia en la actualidad, sobre todo, en forma de proyectos planificados para un futuro en ciudades muy importantes. Por este motivo, existen una gran cantidad de algoritmos y modelos que han sido utilizados o planteados para llevar a cabo este cometido. En este trabajo se ha realizado una revisión de estos modelos y técnicas para poder compararlas y así obtener más conocimineto acerca de este tema.

La revisión de los diferentes métodos de detección y agrupamiento utilizados en la literatura, ha hecho posible la elección de los métodos utilizados para la realización de este trabajo, centrándose en características como la precisión o la rapidez.

La decisión de utilizar el algoritmo YOLOv4 para la detección del tráfico, debido a su rapidez, ha permitido conseguir datos de congestión de tráfico que han hecho posible la obtención de resultados tras el uso de los métodos de agrupamiento implementados. De todas formas, existen numerosos algoritmos que podrían arrojar buenos resultados e incluso mejores en los que se incluye la versión 5 de este algoritmo, YOLOv5. El uso de esta versión se plantea como un desafío futuro que permita mejorar la precisión puesto que en el momento de la implementación del sistema, esta versión no era soportada por la arquitectura del dispositivo Jetson AGX Xavier, el cual era necesario para poder realizar el análisis de los datos, ya que permite distribuir el cómputo de toda la información obtenida a través de las videocámaras, y así acelerar el proceso.

Por otro lado, la precisión del modelo utilizado podría ser mejorable, planteando de cara al futuro un modelo mejor entrenado con una mayor cantidad de imágenes y también con imágenes de mayor calidad y que aporten más información. Dentro de este punto, y también como trabajo futuro se podría incluir la posibilidad de que el entrenamiento del modelo tuviese en cuenta el abuso racial que suele ir de la mano con la aplicación de las técnicas de aprendizaje automático, e incluir imágenes de peatones de diferentes partes del mundo y culturas.

Otro aspecto que mejorar sería el recuento de los elementos del tráfico. Para la realización de este trabajo, la utilización de condiciones y la definición de

secciones ha sido suficiente como para obtener unos datos de congestión factibles, pero de cara al futuro y con la intención de mejorar el sistema sería conveniente la implementación de un algoritmo de seguimiento o algún mecanismo que permita que el recuento de los elementos sea más preciso de lo que es actualmente.

Por último, he de mencionar que este trabajo está pensado para obtener resultados de congestión y patrones del tráfico en una ciudad en tiempo real con la implantación de una red de cámaras y dispositivos que permitan esta tarea. Por ello, se presenta como un modelo de prueba de lo que conllevaría realizar un despliegue a gran escala. Además, este estudio forma parte de un sistema mayor y los resultados obtenidos serán utilizados para la implementación de un sistema de control de tráfico automático haciendo uso de semáforos inteligentes y con la finalidad de mejorar los tiempos de espera en el tráfico de una ciudad.

Referencias

1. Amineh Amini, Teh Ying Wah, Mahmoud Reza Saybani, and Saeed Reza Aghabozorgi Sahaf Yazdi. A study of density-grid based clustering algorithms on data streams. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 3, pages 1652–1656, 2011.
2. Luciano Oliveira Andrews Sobral, Leizer Schnitman, and Felippe De Souza. Highway traffic congestion classification using holistic properties. In *10th IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, 2013.
3. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
4. Rhen Anjerome Bedruz, Edwin Sybingco, Argel Bandala, Ana Riza Quiros, Aaron Christian Uy, and Elmer Dadios. Real-time vehicle detection and tracking using a mean-shift based blob analysis and tracking approach. In *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pages 1–5, 2017.
5. Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.
6. Massimo Bertozzi, Alberto Broggi, Massimo Cellario, Alessandra Fascioli, Paolo Lombardi, and Marco Porta. Artificial vision in road vehicles. *Proceedings of the IEEE*, 90(7):1258–1271, 2002.
7. Margrit Betke, Esin Haritaoglu, and Larry S Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine vision and applications*, 12(2):69–83, 2000.
8. Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2000.
9. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
10. Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

11. Raymond B Cattell. The description of personality: Basic traits resolved into clusters. *The journal of abnormal and social psychology*, 38(4):476, 1943.
12. Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.
13. Pranamesh Chakraborty, Yaw Okyere Adu-Gyamfi, Subhadipto Poddar, Vesal Ahsani, Anuj Sharma, and Soumik Sarkar. Traffic congestion detection from camera images using deep convolution neural networks. *Transportation Research Record*, 2672(45):222–231, 2018.
14. Shou-Jen Chang-Chien, Wen-Liang Hung, and Miin-Shen Yang. On mean shift-based clustering for circular data. *Soft Computing*, 16(6):1043–1060, 2012.
15. Chenyi Chen, Ming-Yu Liu, Oncel Tuzel, and Jianxiong Xiao. R-cnn for small object detection. In *Asian conference on computer vision*, pages 214–230. Springer, 2016.
16. Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.
17. Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.
18. Michael Daszykowski, Beata Walczak, and Desire L Massart. On the optimal partitioning of data with k-means, growing k-means, neural gas, and growing neural gas. *Journal of chemical information and computer sciences*, 42(6):1378–1389, 2002.
19. Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE, 2013.
20. Konstantinos G Derpanis. Mean shift clustering. *Lecture Notes*, page 32, 2005.
21. Sorin Draghici. A neural network based artificial vision system for licence plate recognition. *International Journal of Neural Systems*, 8(01):113–126, 1997.
22. HE Driver and AL Kroeber. Quantitative expression of cultural relationships. university of california publications in american archaeology and ethnology 31: 211–256. ester, m.; kriegel, h.-p.; sander, j.; and xu, x. 1996. a density-based algorithm for discovering clusters in large spatial databases with noise. *Driver21131Quantitative Expression of Cultural Relationships1932*, 1932.
23. K-L Du. Clustering: A neural network approach. *Neural networks*, 23(1):89–107, 2010.
24. Adel S Elmaghraby and Michael M Losavio. Cyber security challenges in smart cities: Safety, security and privacy. *Journal of advanced research*, 5(4):491–497, 2014.
25. Quanfu Fan, Lisa Brown, and John Smith. A closer look at faster r-cnn for vehicle detection. In *2016 IEEE intelligent vehicles symposium (IV)*, pages 124–129. IEEE, 2016.
26. Bernd Fritzke. Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, 7(9):1441–1460, 1994.
27. Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.
28. Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

29. Robert Gentleman and Vincent J Carey. Unsupervised machine learning. In *Bioconductor case studies*, pages 137–157. Springer, 2008.
30. Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. A new growing neural gas for clustering data streams. *Neural Networks*, 78:36–50, 2016.
31. Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. Understanding deep learning techniques for image segmentation. *ACM Computing Surveys (CSUR)*, 52(4):1–35, 2019.
32. Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
33. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.
34. Shreyank N Gowda and Chun Yuan. Colornet: Investigating the importance of color spaces for image classification. In *Asian Conference on Computer Vision*, pages 581–596. Springer, 2018.
35. Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.
36. Yanyan Gu, Yandong Wang, and Shihai Dong. Public traffic congestion estimation using an artificial neural network. *ISPRS International Journal of Geo-Information*, 9(3):152, 2020.
37. John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.
38. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
39. Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*, pages 198–213. Springer, 2016.
40. Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He, and Paul Kennedy. Deep learning techniques for medical image segmentation: achievements and challenges. *Journal of digital imaging*, 32(4):582–596, 2019.
41. Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.
42. W Hongsakham, W Pattara-Atikom, and R Peachavanish. Estimating road traffic congestion from cellular handoff information using cell-based neural networks and k-means clustering. In *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 1, pages 13–16. IEEE, 2008.
43. Xinyu Hou, Yi Wang, and Lap-Pui Chau. Vehicle tracking using deep sort with low confidence track filtering. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2019.
44. Shih-Chung Hsu, Chung-Lin Huang, and Cheng-Hung Chuang. Vehicle detection using simplified fast r-cnn. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1–3. IEEE, 2018.
45. R. Hussin, M. Rizon Juhari, Ng Wei Kang, R.C. Ismail, and A. Kamarudin. Digital image processing techniques for object detection from complex background image. *Procedia Engineering*, 41:340–344, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

46. Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys (CSUR)*, 50(2):1–38, 2017.
47. A.G Ivakhnenko. *Cybernetics and forecasting techniques.*
48. Parinith R Iyer, Shrutheesh Raman Iyer, Raghavendran Ramesh, MR Anala, and KN Subramanya. Adaptive real time traffic prediction using deep neural networks. *IAES International Journal of Artificial Intelligence*, 8(2):107, 2019.
49. Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.
50. IT Jolliffe, OB Allen, and BR Christie. Comparison of variety means using cluster analysis and dendrograms. *Experimental Agriculture*, 25(2):259–269, 1989.
51. Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.
52. Ajinkya Khadilkar, Kunal Sunil Kasodekar, Priyanshu Sharma, and J. Priyadarshini. Intelligent traffic light scheduling using linear regression. In Hasmat Malik, Smriti Srivastava, Yog Raj Sood, and Aamir Ahmad, editors, *Applications of Artificial Intelligence Techniques in Engineering*, pages 329–335, Singapore, 2019. Springer Singapore.
53. Zaheer Khan, Zeeshan Pervez, and Abdul Ghafoor. Towards cloud based smart cities data security and privacy management. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 806–811. IEEE, 2014.
54. Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson. Mini-batch gradient descent: Faster convergence under data sparsity. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2880–2887. IEEE, 2017.
55. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
56. Jason Kurniawan, Sensa GS Syahra, Chandra K Dewa, et al. Traffic congestion detection: learning from cctv monitoring images using convolutional neural network. *Procedia computer science*, 144:291–297, 2018.
57. Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. *Advances in neural information processing systems*, 20:833–840, 2007.
58. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
59. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
60. Eunseop Lee and Daijin Kim. Accurate traffic light detection using deep neural network with focal regression loss. *Image and Vision Computing*, 87:24–36, 2019.
61. Da Li, Bodong Liang, and Weigang Zhang. Real-time moving vehicle detection, tracking, and counting system implemented with opencv. In *2014 4th IEEE International Conference on Information Science and Technology*, pages 631–634, 2014.
62. Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

63. Xiong Liu, Li Pan, and Xiaoliang Sun. Real-time traffic status classification based on gaussian mixture model. In *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, pages 573–578. IEEE, 2016.
64. Michael M Losavio, KP Chow, Andras Koltay, and Joshua James. The internet of things and the smart city: Legal challenges with digital forensics, privacy, and security. *Security and Privacy*, 1(3):e23, 2018.
65. Pooja Mahto, Priyamm Garg, Pranav Seth, and J Panda. Refining yolov4 for vehicle detection. *International Journal of Advanced Research in Engineering and Technology (IJARET)*, 11(5), 2020.
66. Minsky Marvin and A Papert Seymour. Perceptrons, 1969.
67. John McCarthy and Edward A Feigenbaum. In memoriam: Arthur samuel: Pioneer in machine learning. *AI Magazine*, 11(3):10–10, 1990.
68. Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
69. Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York, 1988.
70. Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.
71. Mahmoud Owais, Ghada S Moussa, and Khaled F Hussain. Robust deep learning architecture for traffic flow estimation from a subset of link sensors. *Journal of Transportation Engineering, Part A: Systems*, 146(1):04019055, 2020.
72. Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.
73. Benjamin N. Passow, David Elizondo, Francisco Chiclana, Simon Witheridge, and Eric Goodyer. Adapting traffic simulation for traffic management: A neural network approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1402–1407, 2013.
74. Vishwajeet Pattanaik, Mayank Singh, PK Gupta, and SK Singh. Smart real-time traffic congestion estimation and clustering technique for urban vehicular roads. In *2016 IEEE region 10 conference (TENCON)*, pages 3420–3423. IEEE, 2016.
75. Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568, 2008.
76. Lakshmish Ramaswamy, Bugra Gedik, and Ling Liu. Connectivity based node clustering in decentralized peer-to-peer networks. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, pages 66–73. IEEE, 2003.
77. Šarūnas Raudys. Evolution and generalization of a single neurone: I. single-layer perceptron as seven statistical classifiers. *Neural Networks*, 11(2):283–296, 1998.
78. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.
79. Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.
80. Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
81. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
82. Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

83. Jude W Shavlik, Thomas Dietterich, and Thomas Glen Dietterich. *Readings in machine learning*. Morgan Kaufmann, 1990.
84. Zhong Su, Li Zhang, and Yue Pan. Document clustering based on vector quantization and growing-cell structure. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 326–336. Springer, 2003.
85. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
86. Suman Tatiraju and Avi Mehta. Image segmentation using k-means clustering, em and normalized cuts. *Department of EECS*, 1:1–7, 2008.
87. Anurag Singh Tomar, Mridula Singh, Girish Sharma, and KV Arya. Traffic management using logistic regression with fuzzy logic. *Procedia computer science*, 132:451–460, 2018.
88. Liesbet Van Zoonen. Privacy concerns in smart cities. *Government Information Quarterly*, 33(3):472–480, 2016.
89. Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.
90. Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.
91. David L Wallace. Clustering. *International encyclopedia of the social sciences*, 2:519–524, 1968.
92. Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
93. Zhiqiang Wei, Xiaopeng Ji, and Peng Wang. Real-time moving object detection for video monitoring systems. *Journal of Systems Engineering and Electronics*, 17(4):731–736, 2006.
94. Paul Werbos. Beyond regression:"new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.
95. Qing Wu, Yungang Liu, Qiang Li, Shaoli Jin, and Fengzhong Li. The application of deep learning in computer vision. In *2017 Chinese Automation Congress (CAC)*, pages 6522–6527. IEEE, 2017.
96. Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, and Xiaolin Hu. Traffic sign detection based on convolutional neural networks. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2013.
97. Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. A robust em clustering algorithm for gaussian mixture models. *Pattern Recognition*, 45(11):3950–3961, 2012.
98. Jian Yao and Jean-Marc Odobez. Multi-layer background subtraction based on color and texture. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.
99. Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM international Conference on Data Mining*, pages 777–785. SIAM, 2017.
100. Shiqi Yu, Sen Jia, and Chunyan Xu. Convolutional neural networks for hyperspectral image classification. *Neurocomputing*, 219:88–98, 2017.
101. Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116, 2004.

102. Zhenhua Zhang, Qing He, Jing Gao, and Ming Ni. A deep learning approach for detecting traffic accidents from social media data. *Transportation research part C: emerging technologies*, 86:580–596, 2018.
103. Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

Análisis visual interactivo para la segmentación y sincronización de películas y guiones cinematográficos

Héctor Sánchez San Blas, Roberto Therón Sánchez

Departamento de Informática y Automática, Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
{hectorsanchezsanblas,theron}@usal.es
https://www.usal.es/

Resumen Dentro de los géneros literarios, la narrativa es aquel en el que intervienen varios personajes en un espacio-tiempo narrativo donde puede ser muy complejo llegar a comprender todas las interacciones que tienen lugar. Las herramientas visuales que son capaces de computar grandes volúmenes de información y que pueden ser grandes aliados para convertirlos en formas de representación interactivas que faciliten esta comprensión. Concretamente, el objetivo es el desarrollo de una plataforma que permita llevar a cabo un análisis visual de una película, es decir, identificar de forma visual los distintos elementos que actúan dentro de una película así como la interacción entre ellos. Para ello, se desarrollan una serie de procesos y técnicas capaces de extraer secuencias y escenas de una película y sincronizarlos con las descripciones y diálogos presentes en un guión cinematográfico, permitiendo así llevar a cabo un análisis visual de la película.

Keywords: Análisis visual, Cine, Películas, Representación

1. Introducción

El análisis cinematográfico consiste en el estudio de los componentes del lenguaje cinematográfico, es decir, aquellos elementos que permiten distinguir una película de otra. La finalidad del análisis cinematográfico de una película es encontrar la forma en que los signos y estructuras de comunicación de la película interactúan entre sí para generar un efecto en el espectador. Teniendo en cuenta que el resultado final de una película es un conjunto de técnicas y elementos de distintas ramas, existen diversos tipos de análisis que pueden llevarse a cabo, además, se superponen y complementan según el objetivo que busquen alcanzar:

- **Textual**: se centra en el análisis del texto cinematográfico, lo que incluye, no solo el guión de la película, sino todos los elementos que aparecen en pantalla, sin tener en cuenta interpretaciones o conjeturas que puedan realizarse, lo que se traduce en un análisis totalmente objetivo de una película.

- **Contextual**: estudia el contexto en que la película es estrenada y cómo es recibida. Esta parte incluye un análisis teniendo en cuenta la época, sociedad y cultura.
- **Subtextual**: centrado en el análisis de los conceptos no reconocibles de forma visual y que surgen de la interpretación de la parte textual. A diferencia del análisis textual, este reúne una mayor subjetividad en el análisis.
- **Intertextual**: determina la relación existente entre las películas y otras disciplinas, por ejemplo, el teatro o la literatura.

El presente trabajo, pretende centrarse en análisis textual y subtextual de un guión cinematográfico, analizando los distintos elementos formales presentes en texto. Por tanto, se tiene como objetivo el diseño de una herramienta de análisis visual interactivo que sea capaz de recoger la información del guión de una película y los subtítulos de la misma para ser analizados de forma visual, de forma que pueda llevarse a cabo un análisis textual de la misma, reconociendo los distintos elementos que distinguen a dicha película. Para ello, se adopta un enfoque que combinará la información existente del guión original con los subtítulos que aparecen en la película final.

El artículo se encuentra organizado de la siguiente forma: la Sección 2 se centra en una revisión del estado de la literatura acerca de los temas de similitud de texto y análisis visual. La Sección 3, en la que se describirá el funcionamiento del sistema propuesto. La Sección 4, que mostrará el caso de estudio llevado a cabo con los resultados obtenidos. Finalmente, las conclusiones obtenidas se presentarán en la Sección 5.

2. Estado del arte

En esta sección se realizará un análisis de las técnicas y métodos utilizados, no solo para hallar la similitud entre dos textos, sino aquellas que permiten realizar análisis de forma visual de narraciones. Para ello, un análisis individual de cada uno de los temas será realizado con el objetivo de abarcar una mayor variedad de posibilidades que permitan determinar aquellos métodos y técnicas que se adecúan a las necesidades del trabajo.

2.1. Similitud de textos

El campo de la estilometría así como el de similitud de textos pertenece a un ámbito muy estudiado para su utilización en el desarrollo de distintas aplicaciones, por ejemplo la detección de plagios [10]. El problema que subyace a la operación de búsqueda es que se debe medir la similitud o disimilitud de dos cadenas, lo que ha sido sido un tema de investigación durante más de cinco décadas, desde las primeras operaciones hasta el moderno aprendizaje automático y el análisis de datos. Además, cada método utiliza diferentes aspectos y características de los datos [57]. Las soluciones propuestas son varias y van desde fórmulas o modelos matemáticos hasta la utilización de análisis visual. Las técnicas empleada son variadas y pero las más importantes son:

- **Técnicas Tradicionales**: Se tratan de técnicas que buscan hallar la similitud entre dos conjuntos de palabras ya sea hallando el número de letras que cambian, el conjunto de palabras que tengan en común o realizando una transformación del texto a valores numéricos para aplicar técnicas matemáticas. Dentro de este campo entran la Divergencia Kullback-Leibler [42], la Divergencia Jansen-Shannon [9] y la Distancia de Levenshtein [5]. Dentro de este apartado destacar la Similitud del Coseno que se trata de una técnica que mide la similitud entre dos vectores de un espacio, más concretamente, el coseno del ángulo entre dos vectores y determina si ambos apuntan aproximadamente a la misma dirección [62].
- **Técnicas basadas en estructura, semántica y distribución del texto**: Se tratan de técnicas que buscan hallar la similitud entre dos conjuntos de palabras mediante un análisis de las relaciones entre dichas palabras, la estructura del conjunto de palabras o la clasificación de las mismas (sustantivos, verbos, adjetivos y adverbios). En este entorno nos encontramos técnicas como PoS Similarity (Part of Speech Similairty) [7] y Latent Semantic Analysis (LSI) [55]. Dentro de este campo existe una que destaca sobre el resto aunque basada en PoS Similairty, estas son las medidas basadas en el conocimiento cuantifican la relación semántica de las palabras mediante una red semántica. Muchas medidas han demostrado funcionar bien para la base de datos léxica WordNet para el inglés [70]. Los sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos (synsets), cada uno de los cuales expresa un concepto distinto. La estructura de WordNet la convierte en una herramienta útil para la lingüística computacional y el procesamiento del lenguaje natural.
- **Técnicas basadas en *Word Embeddings* y *Deep Learning***: Una de las técnicas más usadas para la similitud de texto es la denominada *Word Embedding*. Se trata de una representación aprendida a partir de un texto dado donde las palabras que tienen el mismo significado poseen un representación similar. Se trata de una serie de técnicas en las que palabras individuales se representan como vectores de valor real en un espacio vectorial predefinido. Cada palabra se asigna a un vector y los valores del vector son aprendidos de tal manera que se asemeja a una red neuronal y, por tanto, se suele agrupar en el campo del *Deep learning*. La representación distribuida se aprende a partir del uso de las palabras. Esto permite que las palabras que se utilizan de forma similar tengan representaciones similares, capturando de forma natural su significado. En estos campos destacan en utilización y resultados las técnicas Word2Vec [45], GloVe [48], LSTM-Siamese [54] y BERT [53].

2.2. Análisis visual de películas

Actualmente, el crecimiento exponencial de los datos que se pueden obtener hace que sea necesaria la búsqueda de métodos que permitan analizarlos y extraer información útil de los mismos. Uno de los métodos que permiten un desarrollo rápido de aplicaciones encargadas de extraer esta información son las técnicas de visualización de información. Las técnicas de visualización de basan en el uso de

distintas representaciones, como la utilización de gráficas, para expresar de forma visual información oculta dentro de los datos, así como mostrar tendencias de forma más intuitiva. Existen distintos estudios que hacen uso de estas técnicas para llevar a cabo estos análisis, por ejemplo, análisis visual de jerarquías de datos junto a su evolución temporal [66].

La visualización de datos centra nuestra atención en el contenido importante. Cuando vemos un gráfico, percibimos rápidamente las tendencias y los valores atípicos. Si podemos ver algo, lo interiorizamos rápidamente. Si nos centramos en las películas, muchas son las aplicaciones que se han realizado para su clasificación [61], recomendación [13], análisis de imágenes [28], etc. Dentro de estas posibilidades nos encontramos la del análisis visual de ciertos elementos de dichas películas. Un ejemplo de esto es el trabajo [23] que llevan a cabo un análisis de la serie de libros de "Canción de Hielo y Fuego" de George R. R. Martin donde hacen uso de un grafo de interacciones de los personajes a medida que se desarrolla la historia donde se encuentra que las propiedades estructurales permanecen aproximadamente estables y comparables a las redes sociales del mundo real. Además, los grados de los personajes más conectados reflejan un límite cognitivo de conexiones sociales concurrentes que los humanos tienden a mantener. Similar al trabajo anterior nos encontramos [33] donde proponen un enfoque no supervisado para construir un grafo dinámico de personajes a partir de la agrupación de rostros.

Por otra parte, en el trabajo [25] se propone un método de análisis intuitivo y semántico mediante el diseño de una visualización de red de sentimientos multinivel basada en palabras de emoción. El análisis de redes sociales desempeña un papel importante en la comprensión y la búsqueda de soluciones para los problemas funcionales de la sociedad mediante el examen de la estructura y las relaciones originales de la red. Por lo tanto, la visualización de redes se aplica en una amplia variedad de campos, incluyendo el análisis de redes basado en la similitud de los datos, el análisis de redes sobre situaciones socio-científicas, el algoritmo de dibujo de grafos dirigido por la fuerza es un algoritmo de diseño estándar para diseñar un grafo.

Finalmente, nos encontramos con el trabajo [12] que visualiza datos de películas desde múltiples dimensiones a través de D3.js con el fin de llevar a cabo una observación y análisis más profundos. Son numerosos los tipos de información que obtienen, encontrándonos con las películas más vistas, las provincias de China con más cines, las palabras más utilizadas en las críticas de las películas y las relaciones entre directores, actores y películas.

3. Sistema Propuesto

Tras la realización del estudio en cuanto al estado del arte se refiere acerca de las técnicas de similitud de textos y de soluciones relativas al desarrollo de herramientas para el análisis y clasificación de películas, podemos comenzar con la propuesta del sistema a realizar. Cabe destacar que el sistema propuesto tiene como principal funcionalidad poder llevar a cabo un análisis visual e interactivo

de películas del cine a partir de sus guiones y subtítulos, permitiendo distinguir ciertas características que las distinguen del resto. Para ello, se propone un sistema que lleve a cabo los procesos necesarios de forma secuencial tal y como se puede observar en la Figura 1.

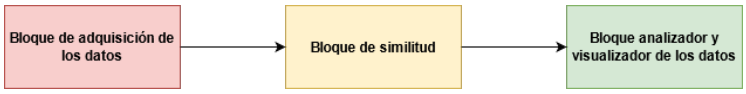


Figura 1: Sistema secuencial propuesto.

Cada uno de estos bloques se encarga de llevar a cabo distintas tareas necesarias en este proceso:

- **Bloque de adquisición de los datos**: este bloque será el encargado de extraer los datos que se pueden encontrar en un guión cinematográfico para poder ser analizados posteriormente.
- **Bloque de similitud**: esta parte es la encargada de cruzar los diálogos del guión original con los subtítulos presentes en la película final, permitiendo obtener los instantes de tiempo de la película final que se corresponden con las escenas del guión original.
- **Bloque analizador y visualizador de los datos**: este bloque llevará a cabo el análisis de los datos obtenidos en los bloques anteriores, extrayendo información de estos visualizándolos para un mayor entendimiento.

Teniendo este enfoque como base, a lo largo de este apartado se irán explicando de forma más detallada las propuestas para cada uno de los bloques para lograr la consecución del sistema final.

3.1. Bloque de adquisición de los datos

Como se ha mencionado antes, este bloque es el encargado de extraer los datos contenidos en el guión de una película. El guión de una película es un documento que tiene habitualmente entre 70 y 180 páginas y que relata los sucesos de una película, con cierto parecido a una novela. Se desarrolla de forma completa un argumento teniendo en cuenta que todo hay que grabarlo, filmarlo y montarlo.

Conociendo la estructura básica de un guión, es es posible llevar a cabo un algoritmo capaz de extraer la información de este tipo de documentos, lo que permitiría obtener datos con los que poder analizar la película. El algoritmo que se propone realizará esta extracción de información siguiendo las pautas marcadas en el diagrama de secuencia de la Figura 2.

Como se puede observar, el algoritmo recorre cada una de las líneas del fichero identificándolas y clasificándolas para su posterior análisis. Para ello, se recoge la información del fichero y se va leyendo línea por línea por línea, entendiendo

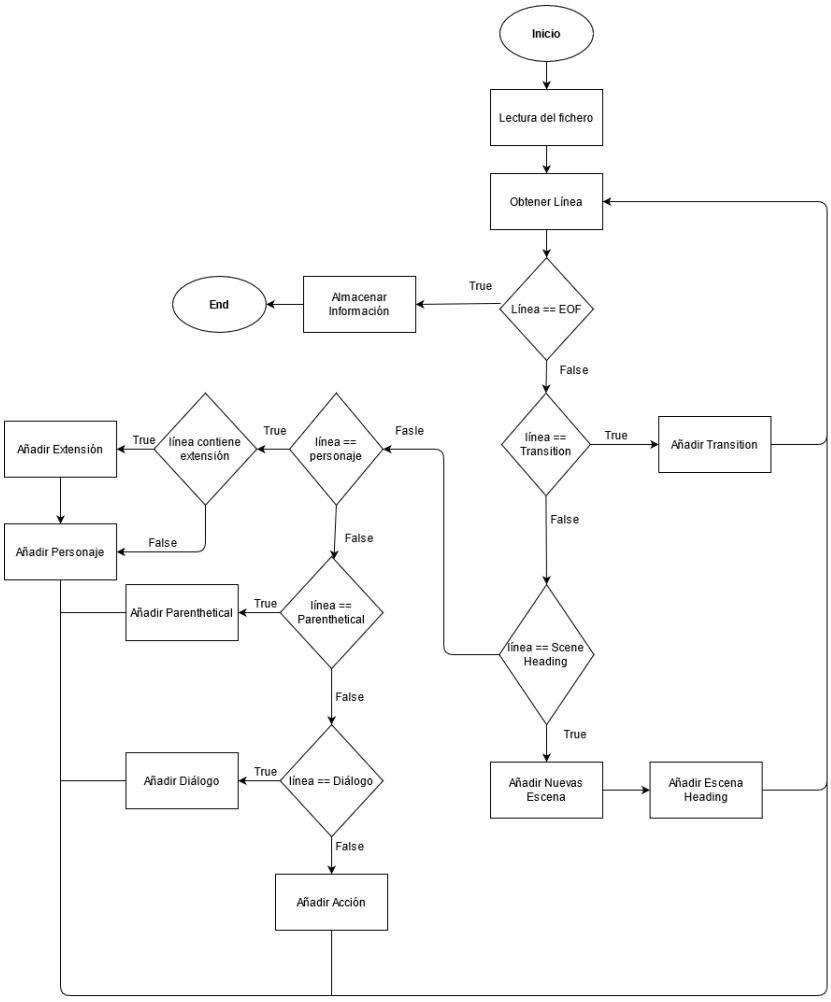


Figura 2: Diagrama extracción datos guión.

dicha línea como un *string* cuyo final sea un retorno de carro. En caso de que la línea no sea el final del fichero, se analiza su contenido:

- En caso de que el *string* contenga palabras de transacciones en la cabecera 'INT.' o 'EXT.', se identifica como *Scene Heading*, lo que permite determinar el inicio de una nueva escena.
- En caso de que el *string* contenga palabras de transacciones predefinidas como 'CUT TO' o 'ANOTHER ANGLE', se identifica como transición y se añade al conjunto de datos.
- Llegados a este punto, se lleva a cabo un análisis mediante expresiones regulares que permitan identificar las siguientes partes:
  - Personaje: que a su vez puede ir acompañado de *Extension* o no. En cualquier caso, se añaden los elementos identificados como tal.
  - *Parenthetical*.
  - Diálogo.
  - Acción.

Al final, se obtiene un conjunto de datos identificados por los tipos ya descritos y separados para cada una de las escenas.

3.2. Bloque de similitud

Este bloque tiene como objetivo hallar una correspondencia entre los diálogos del guión con los subtítulos de la película final. Para ello, ha de emplearse una de las técnicas de similitud de textos que se ha investigado en el la Sección 2. En este caso, se utiliza WordNet junto a las herramientas que provee para realizar esta tarea ya que se trata de una herramienta con buenos resultados en este campo y con necesidades computacionales bajas.

Teniendo esto en cuenta, realiza una lectura del archivo con el contenido de subtítulos de la película real junto con los puntos, en tiempo, donde se producen. Este se cruza con los diálogos resultantes para cada una de las escenas que se han extraído del apartado anterior. Cabe destacar que, se deben contemplar los siguientes aspectos:

- Un diálogo tan solo puede asignarse a un único subtítulo de película.
- Un diálogo no puede asignarse a una escena donde el código de tiempo no coincida con el código de tiempo de la escena. Se entiende como código de tiempo el tiempo de inicio y fin de un subtítulo. En el caso de la escena, el código de tiempo viene determinar por una secuencia de subtítulos cuya aparición en el tiempo es secuencial y cuya diferencia de tiempo entre dos subtítulos que aparecen de forma consecutiva no supera los 30 segundos. De esta forma, el código de tiempo de una escena se corresponderá con el tiempo de inicio del primer subtítulo vinculado al primer diálogo asignado y el tiempo final del último subtítulo vinculado a último diálogo asignado.

Teniendo esto en cuenta, se asignan aquellos subtítulos con los diálogos correspondientes, siempre que se cumplan las condiciones anteriores y el resultado

de la similitud con WordNet sea una puntuación mayor o igual a 0.5. El resultado de este proceso es el documento final con todos los datos accesibles e identificados para su análisis.

3.3. Bloque de análisis visual

Este bloque se encarga de recoger los datos extraídos por los dos bloques anteriores y llevar a cabo un análisis de esta información. Son tres los posibles análisis que se pretenden realizar:

- **Análisis de la película con todas sus escenas**: este análisis permite tener una idea general de las características de la película.
- **Análisis de los personajes**: en este caso, se analizarán determinados aspectos de los personajes, como las relaciones entre ellos que se producen a lo largo de la película.
- **Análisis de las escenas**: no solo mostrará aspectos de las propias escenas, sino que, si se encuentra incluida se mostrará el momento correspondiente de la película.

4. Caso de estudio

La propuesta presentada ha sido aplicada al caso de estudio de la película Rocky de 1976 escrita y protagonizada por Sylvester Stallone y dirigida por John G. Avildsen. De esta forma, se pretende comprobar cuáles son las capacidades del sistema para extraer datos a partir del guión original de la película y los subtítulos de la película final. Además, el sistema debe analizar dichos datos para poder extraer características del sistema que permitan comprender las películas de inicio a fin, distinguiéndolas del resto de películas. Para poder conocer toda la información relativa a la película, el sistema realiza un análisis en cuanto a aspectos generales de la película, personajes y cada una de las escenas de la mismas. Por tanto, a lo largo de este punto, se irán comprobando cuáles son los resultados obtenidos para estos casos de estudio explicando los aspectos más importantes de cada uno de los apartados en los cuáles se divide la aplicación.

4.1. Análisis general de la película

Esta sección ofrece información relativa a peculiaridades generales de toda la película. Se corresponde con la sección *Película* de la herramienta desarrollada. El diagrama que muestra una visión general de la película se corresponde a la primera imagen que nos encontramos al acceder a esta sección, que es similar a la que se puede observar en la Figura 3.

Esta gráfica representa lo que podría denominarse la "huella de la película", es decir, se trata de una gráfica que permite identificar visualmente a la película. Más concretamente y explicando con detenimiento, representa la duración de

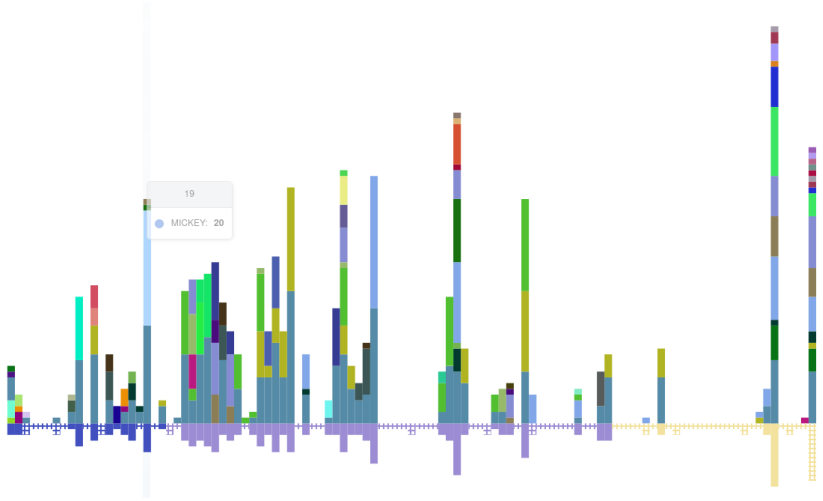


Figura 3: Huella de la película.

cada escena, si dicha escena se ha identificado o no mediante la similitud realizada, el acto al que pertenece cada escena, los personajes que participan en cada escena y el número de veces que aparecen dichos personajes.

En lo referente a los personajes, representados por las barras positivas, nos encontramos que representan el número de intervenciones que realiza cada uno de los personajes que sale durante esa escena. Esto se representa a partir de distintos colores para cada uno de los personajes, además, la altura del color representa el número de intervenciones que realiza cada uno en dicha escena. Por otra parte, permite comprobar varias cosas, la primera, cuáles son los personajes que más aparecen e intervienen a lo largo de la película, en este caso, tenemos a Rocky representado en azul grisáceo, que aparece en casi todas las escenas que contienen diálogos y que, por tanto, se muestra como el protagonista de la película. Además, permite conocer que interacciones existen entre los personajes para cada una de las escenas, así como conocer que escenas no poseen diálogos.

En lo que se refiere a las escenas, representadas por las barras con valor negativo, cabe destacar que un guión suele dividirse en tres actos: el primer acto suele representar el 20 % del total de escenas de un guión, el segundo acto un 55 % y el tercer acto un 25 %. En este caso, cada uno de los actos se representan en verde claro , naranja claro y púrpura respectivamente, para cada uno de los actos. Cada una de las escenas que pertenecen a cada acto se representan con el color que los identifica, obteniendo como resultado, en este caso, que el primer acto dura hasta la escena 21, el segundo de la 22 a la a la 80 y el tercero de la 81 hasta la 107, que es la última. Por otra parte, las escenas que aparecen en recuadros representan aquellas escenas que no se han identificado mediante el proceso de similitud ya que no se ha encontrado referencia a nivel de subtítulos sobre ello así como aquellas que no poseen diálogos, cuya comprobación no se

puede realizar. Claramente destacan aquellas escenas que no tienen diálogos y se tratan de simples descripciones de las mismas mediante acciones y cambios de lugar. Finalmente, la altura de las escenas representa la duración de las misma según el número de páginas que ocupen, destacando la duración de la escena 102 ya que se trata de la que más duración representa y donde se encuentran el antecedente al final de la trama.

4.2. Análisis de los personajes

Esta sección ofrece información relativa a los personajes de la película. Se corresponde con la sección *Personajes* de la herramienta desarrollada. La primera imagen que nos encontramos al acceder a esta sección es similar a la que se puede observar en la Figura 4.

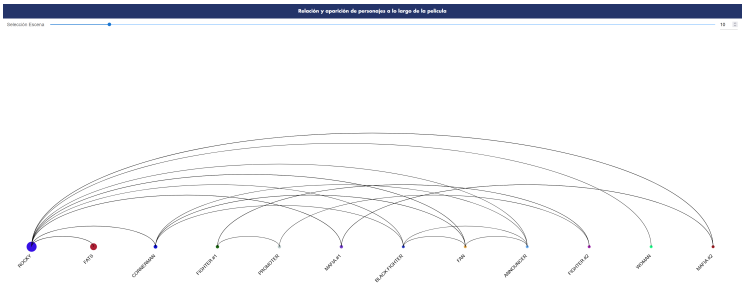


Figura 4: Interacciones entre personajes y el número de veces que intervienen hasta la escena 10.

Este diagrama representa las interacciones que se van produciendo entre los personajes a lo largo de la película, así como el número de veces que aparece un personaje en la escena. El número de veces que interviene un personaje viene representado por el tamaño del punto que se visualiza, además, las interacciones entre usuarios vienen presentadas por la unión con arcos entre sí. Por ejemplo, en la Figura 4 se ha podido observar que, en la escena 10 que pertenece al primer acto de la película, se han presentado algunos personajes, por el tamaño, podrían ser secundarios, aunque el punto de Rocky ya es algo mayor.

Si accedemos a la representación para la escena 43 (segundo acto) observamos que la figura de Rocky como personaje empieza a destacar, al aparecer un mayor número de veces y tener un gran número de interacciones, esto último pudiendo comprobarse en la Figura 5, lo que nos da a entender que se trata del protagonista de la película, además, se empiezan a discernir otras figuras importantes como Adrian, Paulie o Apollo que podrían ser personajes secundarios destacándose de los terciarios que tan solo aparecen en ciertos momentos..

Además, en esta escena, se puede comprobar como han aparecido ciertas subtramas en la película, ya que existen nodos con arcos entre sí pero pocos o

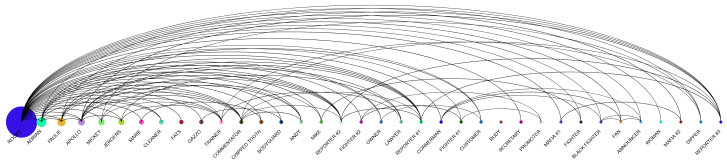


Figura 5: Interacciones entre personajes y el número de veces que intervienen hasta la escena 43.

ninguno con el resto de personajes. Destacar que no existe interacción alguna con el personaje Rudy, lo que puede tratarse de una escena en forma de monólogo del mismo.

Por otra parte, si observamos el resultado obtenido para la escena 83 y que puede observarse en la Figura 6, podemos asegurar que el protagonista es Rocky por su número de intervenciones junto a las interacciones que realiza con gran parte de los personajes de la película, además, se puede terminar de confirmar los roles de personajes secundarios Mickey, Adrian, Paulie y Apollo que poseen también un gran número de interacciones con otros personajes.

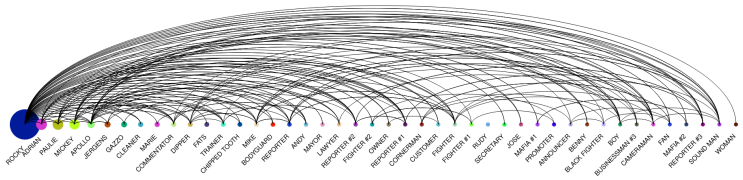


Figura 6: Interacciones entre personajes y el número de veces que intervienen hasta la escena 83.

Para estos diagramas de arcos es posible la ordenación alfabética y por intervenciones. Respecto a todo el análisis anterior, según las estructuras y tramas que se logran identificar así como el progreso en cuanto a las intervenciones e interacciones de los personajes, se puede observar una estructura común en el cine y que se denomina 'El Viaje del Héroe'. El Viaje del Héroe, o el monomito, es una estructura narrativa común a todas las culturas del mundo, en la que un personaje se aventuran en territorio desconocido para recuperar algo que necesita. Enfrentándose a conflictos y adversidades, el héroe acaba triunfando antes de volver a casa transformado.

4.3. Análisis individual de las escenas

Esta sección ofrece información relativa a peculiaridades generales de toda la película. Se corresponde con la sección *Escenas* de la herramienta desarrollada.

El ejemplo aqui mostrado pertenece a la primera escena de la película. La primera imagen que nos encontramos al acceder a esta sección es similar a la que se puede observar en la Figura 7.

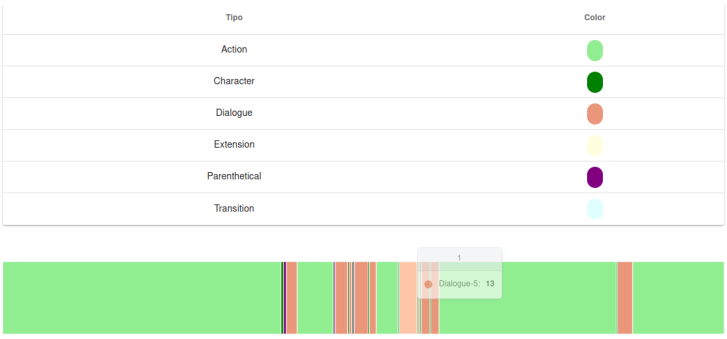


Figura 7: Huella de la escena 1 de la película.

Esta sección permite realizar un análisis individual de las características de cada una de las escenas. Lo que se obtiene es algo similar a la "huella de la película", sin embargo, en esta ocasión, se trata de la "huella de la escena". Esta representación nos permite discernir cuál es la secuencia de aparición de Acciones, Personajes, *Extension*, *Parenthetical*, Diálogos y Transiciones. Para cada una de estas posibilidades se asigna un color, tal como se observar en la Figura 7. El tamaño de las barras que lo componen viene determinado por el número de palabras que forma parte de cada una de esas clases. Este valor puede comprobarse colocándose encima del dato concreto que se quiere conocer, tal y como se puede observar en la Figura 7. Esta representación permite conocer en todo momento la estructura de cada escena, conociendo la extensión de cada uno de estos apartados dentro de la escena y, por tanto, la importancia de los mismo. Posteriormente, nos encontramos con el contenido que puede observarse en la Figura 8.

Esta sección permite acceder al instante de película correspondiente a un diálogo. En la tabla que se puede observar en la Figura 9, se observa la lista de diálogos con los cuales se ha encontrado correspondencia acerca de un subtítulo. Para poder acceder al momento de la película que se quiere consultar, a partir del diálogo o subtítulo requerido, se puede realizar de dos forma:

- Seleccionando uno de los diálogos que se encuentra que se encuentra en la tabla de la derecha, tal y como se puede observar en la Figura 9.
- Seleccionando uno de los diálogos que aparecen en la "huella de la escena" ya mencionados anteriormente.

Destacar que esta correspondencia solo se hará para aquellos diálogos en los cuáles se haya encontrado cierta similitud entre ellos. Por otra parte, mencionar

Momento correspondiente

Figura 8: Representación de la escena y su correspondencia en diálogo

Diálogo original y *subtítulo* correspondiente

Número Diálogo	Diálogo	Subtítulo	Inicio	Fin	Porcentaje
0	... Ya waltzin' -- Give the suckers some action.	You're waltzing. Give this sucker some action.	00:01:51,909	00:01:54,400	0.86
1	... Ya want some good advice?	You want some advice?	00:01:57,014	00:01:58,345	0.80
2	... Just gimme the water.	Water.	00:01:58,416	00:01:59,576	0.70
3	Should I bet the fight don't go the distance -- Ya feel strong?	Rocky, should I bet the fight don't go three rounds?	00:02:02,787	00:02:05,085	0.67
4	... Ya want some good advice?	You want some good advice?	00:02:08,993	00:02:10,654	0.95

Rows per page: 5 1-5 of 11

Figura 9: Ejemplo de comprobación de diálogo en la película

que hasta que no se haya enviado toda la película, no será posible hacer estos saltos en las últimas escenas de las películas. Mediante esta correlación entre la película final y el guión original, puede comprobarse si se han seguido las directrices del guión, que es lo que ha cambiado, la ambientación, etc. Esta herramienta permite comprobar cómo ha sido realizada la asignación, además, se permite filtrar por un *treshold* para poder comprobar con qué valor se obtiene mejores resultados.

5. Conclusiones

A lo largo de este trabajo, se ha llevado a cabo la investigación en técnicas de similitud de textos y análisis visual que han permitido el desarrollo una herramienta de análisis interactivo y visual para el análisis de guiones cinematográficos que permite conocer de forma más educativa las características que poseen ciertas películas, lo que permite conocer el desarrollo de sus tramas, interacciones entre personajes y duración de las escenas entre otras cosas. El sistema desarrollado posee tres partes bien diferenciadas. La primera de ellas, el bloque adquisición de datos ha permitido extraer datos de los documentos de texto correspondientes a los guiones cinematográficos que posean una estructura similar a la estructura básica en la que se ha basado el trabajo. Con estos datos, junto a los subtítulos de la película final, el bloque de similitud ha permitido obtener cierta correspondencia entre los subtítulos de la película final y los diálogos originales. Toda esta información ha sido procesada por le bloque de análisis, dando lugar a la herramienta final, que permite conocer la información contenida en los guiones y subtítulos para lograr interactuar con estos datos de forma que permite al usuario conocer ciertos aspectos que podrían estar ocultos o llegar a conocer ciertas características de la película sin haberla visto. Para comprobar el funcionamiento de la misma, se ha realizado un análisis de la película Rocky 1976 comprobando la información que permitía conocer junto a su posibilidad de poder determinar de forma visual la estructura de la película y su evolución a lo largo del tiempo que dura la misma.

Después del desarrollo de la herramienta y a partir de las limitaciones que se han ido conociendo durante el desarrollo de la misma, se propone como trabajo futuro:

- Mejorar el algoritmo de similitud de textos para adecuarse a las restricciones de texto y tiempo existentes.
- Optimizar el algoritmo de similitud de textos para lograr un tiempo de respuesta bajo y que permita que la herramienta pueda ser utilizada de forma instantánea.
- Mejorar el algoritmo de extracción de datos de los guiones, de forma que se adapte de forma más flexible a las distintas estructuras y posibilidades que existen en la redacción de estos documentos.
- Llevar a cabo una representación temporal que permita comparar la estructura del guión con el de la película final.

- Realizar un análisis de sentimientos a los textos de las películas representando información relativa a las emociones de las escenas, de forma que se pueda añadir información semántica y del contexto de las escenas.

6. Referencias

Referencias

1. *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. IEEE.
2. EnglishWordNet: A new open-source wordnet for English, howpublished = `https://kln.lexicala.com/kln28/mccrae-rudnicka-bond-english-wordnet/`, note = Accessed: 2021-05-12.
3. What is standard screenplay format?, howpublished = `https://www.studiobinder.com/blog/how-to-write-a-screenplay/`, note = Accessed: 2021-05-20.
4. Story and Discourse: Narrative Structure in Fiction and Film by Seymour Chatman. 1980.
5. Safa Abdul-Jabbar and Loay George. A Comparative Study for String Metrics and the Feasibility of Joining them as Combined Text Similarity Measures. *ARO-The Scientific Journal of Koya University*, 5(2):6–18, 2017.
6. A. Abdul-Rahman, G. Roe, M. Olsen, C. Gladstone, R. Whaling, N. Cronk, R. Morrissey, and M. Chen. Constructive Visual Analytics for Text Similarity Detection. *Computer Graphics Forum*, 36(1):237–248, 2017.
7. Mohammad AL-Smadi, Zain Jaradat, Mahmoud AL-Ayyoub, and Yaser Jararweh. Paraphrase identification and semantic text similarity analysis in Arabic news tweets using lexical, syntactic, and semantic features. *Information Processing and Management*, 53(3):640–652, 5 2017.
8. Arifah Che Alhadi, Aziz Deraman, Masita Masila Abdul Jalil, Wan Nural Jawahir Wan Yussof, and Rosmayati Mohemad. A computational analysis of short sentences based on ensemble similarity model. *International Journal of Electrical and Computer Engineering*, 9(6):5386–5394, 2019.
9. Eduardo G. Altmann, Laércio Dias, and Martin Gerlach. Generalized entropies and the similarity of texts. *Journal of Statistical Mechanics: Theory and Experiment*, 2017(1), 1 2017.
10. Salha Alzahrani and Hanan Aljuaid. Identifying cross-lingual plagiarism using rich semantic features and deep neural networks: A study on Arabic-English plagiarism cases, 2020.
11. Karlo Babić, Francesco Guerra, Sanda Martinčić-Ipšić, and Ana Meštrović. A comparison of approaches for measuring the semantic similarity of short texts based on word embeddings. *Journal of Information and Organizational Sciences*, 44(2):231–246, 2020.
12. An Bing and Zhu Li-Gu. Film Big Data Visualization Based on D3.js. In *2020 International Conference on Big Data and Social Sciences (ICBDSS)*, pages 50–53. IEEE, 8 2020.
13. Konstantinos Bougiatiotis and Theodore Giannakopoulos. Enhanced Movie Content Similarity Based on Textual, Auditory and Visual Information. 11 2017.

14. Chris Bryan, Kwan Liu Ma, and Jonathan Woodring. Temporal Summary Images: An Approach to Narrative Visualization via Interactive Annotation Generation and Placement. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):511–520, 1 2017.
15. Enric Castelló. Storytelling in applications for the EU quality schemes for agricultural products and foodstuffs: Place, origin and tradition, 2020.
16. David Chartash. Maths meets myths: Quantitative approaches to ancient narratives. *Folklore*, 130(3):315–317, 2019.
17. Emeric Dynomant, Stéfan J. Darmoni, Émeline Lejeune, Gaëtan Kerdelhué, Jean-Philippe Leroy, Vincent Lequertier, Stéphane Canu, and Julien Grosjean. Doc2Vec on the PubMed corpus: study of a new approach to generate related articles. 11 2019.
18. Vanessa Echeverria, Roberto Martinez-Maldonado, Simon Buckingham Shum, Katherine Chiluiza, Roger Granda, and Cristina Conati. Exploratory versus Explanatory Visual Learning Analytics: Driving Teachers' Attention through Educational Data Storytelling. *Journal of Learning Analytics*, 5(3), 11 2018.
19. Cameron Edmond and Tomasz Bednarz. Three trajectories for narrative visualisation. *Visual Informatics*, 5(2):26–40, 6 2021.
20. Velitchko Filipov, Victor Schetinger, Kathrin Raminger, Nathalie Soursos, Susana Zapke, and Silvia Miksch. Gone full circle: A radial approach to visualize event-based networks in digital humanities. *Visual Informatics*, 5(1):45–60, 3 2021.
21. Therón Sánchez R. Flores González, E. Análisis visual interactivo de espacio-tiempo narrativo. *Avances en Informática y Automática*, 10:97–113, 2017.
22. Gerard Genette. *Narrative discourse : an essay in method*. Cornell University Press, 1980.
23. Thomas Gessey-Jones, Colm Connaughton, Robin Dunbar, Ralph Kenna, P ´ Adraig Maccarron, Cathal O'conchobhair, Joseph Yose, and Kenneth W Wachter. Narrative structure of A Song of Ice and Fire creates a fictional world with realistic measures of social complexity.
24. Negin Ghasemi and Saeedeh Momtazi. Neural text similarity of user reviews for improving collaborative filtering recommender systems. *Electronic Commerce Research and Applications*, 45(October 2019):101019, 2021.
25. Hyoji Ha, Hyunwoo Han, Seongmin Mun, Sungyun Bae, Jihye Lee, and Kyungwon Lee. An improved study of multilevel semantic network visualization for analyzing sentiment word of movie review data. *Applied Sciences (Switzerland)*, 9(12), 6 2019.
26. Mahmoud M. Hammad, Mohammad Al-Smadi, Qanita Bani Baker, Muntaha Al-Asa'd, Nour Al-Khdour, Mutaz Bni Younes, and Enas Khwaileh. Question to question similarity analysis using morphological, syntactic, semantic, and lexical features. *Journal of Universal Computer Science*, 26(6):671–697, 2020.
27. Phan Hieu Ho, Trung Hung Vo, Ngoc Anh Thi Nguyen, and Ha Huy Cuong Nguyen. A narrative method for evaluating documents similarity based on unique strings. *International Journal of Recent Technology and Engineering*, 8(2 Special Issue 11):473–479, 2019.
28. Md Naimul Hoque, Nazmus Saquib, Syed Masum Billah, and Klaus Mueller. Toward Interactively Balancing the Screen Time of Actors Based on Observable Phenotypic Traits in Live Telecast. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2), 10 2020.
29. Xiaoli Huang, Jimmy Lin, and Dina Demner-Fushman. Evaluation of PICO as a knowledge representation for clinical questions. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, (November):359–363, 2006.

30. Rezarta Islamaj, W. John Wilbur, Natalie Xie, Noreen R. Gonzales, Narmada Thanki, Roxanne Yamashita, Chanjuan Zheng, Aron Marchler-Bauer, and Zhiyong Lu. PubMed Text Similarity Model and its application to curation efforts in the Conserved Domain Database. *Database*, 2019(1):1–13, 2019.
31. Arkadiusz Janz, Pawel Kedzia, and Maciej Piasecki. Graph-based complex representation in inter-sentence relation recognition in Polish texts. *Cybernetics and Information Technologies*, 18(1):152–170, 2018.
32. Suhyeon Kim, Haecheong Park, and Junghye Lee. Word2vec-based latent semantic analysis (W2V-LSA) for topic modeling: A study on blockchain technology trend analysis. *Expert Systems with Applications*, 152, 8 2020.
33. Prakhar Kulshreshtha and Tanaya Guha. Dynamic character graph via online face clustering for movie analysis. *Multimedia Tools and Applications*, 79(43-44):33103–33118, 11 2020.
34. O. Joun Lee, Jason J. Jung, and Jin Taek Kim. Learning hierarchical representations of stories by using multi-layered structures in narrative multimedia. *Sensors (Switzerland)*, 20(7), 4 2020.
35. Gregor Leusch, Nicola Ueffing, and Hermann Ney. A Novel String-to-String Distance Measure With Applications to Machine Translation Evaluation. *Proceedings of MT Summit*, (August 2014):33–40, 2003.
36. Chengcheng Li, Fengming Liu, and Pu Li. Text similarity computation model for identifying rumor based on bayesian network in microblog. *International Arab Journal of Information Technology*, 17(5):731–741, 2020.
37. Xiaoman Li, Cui Wang, Xuefu Zhang, and Wei Sun. Generic SAO Similarity Measure via Extended Sorensen-Dice Index. *IEEE Access*, 8:66538–66552, 2020.
38. Xu Li, Chunlong Yao, Fenglong Fan, and Xiaoqiang Yu. A text similarity measurement method based on singular value decomposition and semantic relevance, 2017.
39. Ying Li, Shih Hung Lee, Chia Hung Yeh, and C. C.Jay Kuo. Techniques for Movie Content Analysis and Skimming Tutorial and overview on video abstraction techniques. *IEEE Signal Processing Magazine*, 23(2):79–89, 2006.
40. Ying Li, Shrikanth Narayanan, and C. C.Jay Kuo. Content-based movie analysis and indexing based on audio visual cues. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(8):1073–1085, 8 2004.
41. Xiangyu Liao, Xingyu Liao, Wufei Zhu, Lu Fang, and Xing Chen. An efficient classification algorithm for NGS data based on text similarity. *Genetics Research*, (September 2018), 2018.
42. Yao Lifang, Qin Sijun, and Zhu Huan. Feature selection algorithm for hierarchical text classification using Kullback-Leibler divergence. In *2017 2nd IEEE International Conference on Cloud Computing and Big Data Analysis, ICCCBDA 2017*, pages 421–424. Institute of Electrical and Electronics Engineers Inc., 6 2017.
43. Weiming Lu, Pengkun Ma, Jiale Yu, Yangfan Zhou, and Baogang Wei. Metro maps for efficient knowledge learning by summarizing massive electronic textbooks. *International Journal on Document Analysis and Recognition*, 2019.
44. Christian Metz. *Film Language: A Semiotics of the Cinema*. University of Chicago Press, 1974.
45. Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting Similarities among Languages for Machine Translation. 9 2013.
46. Hamid Mohammadi and Seyed Hossein Khasteh. A fast text similarity measure for large document collections using multireference cosine and genetic algorithm. *Turkish Journal of Electrical Engineering and Computer Sciences*, 28(2):999–1013, 2020.

47. Dheyaa Abdulameer Mohammed and Nasreen J. Kadhim. Extractive multi-document summarization model based on different integrations of double similarity measures. *Iraqi Journal of Science*, 61(6):1498–1511, 2020.
48. Shapol M. Mohammed, Karwan Jacksi, and Subhi R.M. Zeebaree. A state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, 22(1):552–562, 4 2021.
49. David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G. Altman, Doug Altman, Gerd Antes, David Atkins, Virginia Barbour, Nick Barrowman, Jesse A. Berlin, Jocalyn Clark, Mike Clarke, Deborah Cook, Roberto D'Amico, Jonathan J. Deeks, P. J. Devereaux, Kay Dickersin, Matthias Egger, Edzard Ernst, Peter C. Gøtzsche, Jeremy Grimshaw, Gordon Guyatt, Julian Higgins, John P.A. Ioannidis, Jos Kleijnen, Tom Lang, Nicola Magrini, David McNamee, Lorenzo Moja, Cynthia Mulrow, Maryann Napoli, Andy Oxman, Bá Pham, Drummond Rennie, Margaret Sampson, Kenneth F. Schulz, Paul G. Shekelle, David Tovey, and Peter Tugwell. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *PLoS Medicine*, 6(7), 2009.
50. A. Palmer. *Storyworlds and groups*, pages 176–192. 01 2010.
51. Yaohua Pan, Zhibin Niu, Jing Wu, and Jiawan Zhang. InSocialNet: Interactive visual analytics for role—event videos. *Computational Visual Media*, 5(4):375–390, 12 2019.
52. Seung Bo Park, Kyeong Jin Oh, and Geun Sik Jo. Social network analysis in a movie using character-net. In *Multimedia Tools and Applications*, volume 59, pages 601–627, 7 2012.
53. Nicole Peinelt, Dong Nguyen, and Maria Liakata. tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection. Technical report.
54. Elvys Linhares Pontes, Stéphane Huet, Andréa Carneiro Linhares, and Juan-Manuel Torres-Moreno. Predicting the Semantic Textual Similarity with Siamese CNN and LSTM. 10 2018.
55. Robbi Rahim, Nuning Kurniasih, Muhammad Irawan, Yustria Siregar, Abdurrozzaq Hasibuan, Deffi Sari, Tiarma Simanihuruk, Dian Sutiksno, Erland Mouw, Idris Sudin, and Achmad GS. Latent semantic indexing for indonesian text similarity. *International Journal of Engineering & Technology*, 7:73, 03 2018.
56. Anil Ramakrishna, Victor R. Martínez, Nikolaos Malandrakis, Karan Singla, and Shrikanth Narayanan. Linguistic analysis of differences in portrayal of movie characters. In *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 1, pages 1669–1678. Association for Computational Linguistics (ACL), 2017.
57. Konrad Rieck and Christian Wressnegger. Harry: A tool for measuring string similarity. *Journal of Machine Learning Research*, 17(October), 2016.
58. Safa Sami, Loay E George, and Safa S Abdul-Jabbar. Article in ARO-The Scientific Journal of Koya University. 2017.
59. Rodrigo Santamaría and Roberto Therón. Treevolution: Visual analysis of phylogenetic trees. *Bioinformatics*, 25(15):1970–1971, 8 2009.
60. Patrina Sexton Topper and José A. Bauermeister. Relationship Timelines, Dyadic Interviews, and Visual Representations: Implementation of an Adapted Visual Qualitative Technique. *International Journal of Qualitative Methods*, 20:160940692110167, 1 2021.
61. Prashant Giridhar Shambharkar, M. N. Doja, Dhruv Chandel, Kartik Bansal, and Kunal Taneja. Multimodal KDK classifier for automatic classification of movie

trailers. *International Journal of Recent Technology and Engineering*, 8(3):8481–8490, 9 2019.

62. Sahar Sohangir and Dingding Wang. Improved sqrt-cosine similarity measurement. *Journal of Big Data*, 4(1), 2017.
63. Sahar Sohangir and Dingding Wang. Improved sqrt-cosine similarity measurement. *Journal of Big Data*, 4(1), 12 2017.
64. Krishna Somandepalli, Naveen Kumar, Tanaya Guha, and Shrikanth S. Narayanan. Unsupervised Discovery of Character Dictionaries in Animation Movies. *IEEE Transactions on Multimedia*, 20(3):539–551, 3 2018.
65. Peter Steiner. *Russian Formalism*, volume 8 of *The Cambridge History of Literary Criticism*, page 11–30. Cambridge University Press, 1995.
66. Roberto Therón. Hierarchical-temporal data visualization using a tree-ring metaphor. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4073 LNCS, pages 70–81. Springer Verlag, 2006.
67. Roberto Theron and Laura Fontanillo. Diachronic-information visualization in historical dictionaries. *Information Visualization*, 14(2):111–136, 4 2015.
68. Roberto Therón, Carlos Seguín, Laura De La Cruz, and María Vaquero. Highly interactive and natural user interfaces: Enabling visual analysis in historical lexicography. In *ACM International Conference Proceeding Series*, pages 153–158. Association for Computing Machinery, 2014.
69. Therón Sánchez R. Toimil Martín, D. Estudio de técnicas de análisis visual de guiones cinematográficos. *Avances en Informática y Automática*, 7:119–132, 2013.
70. Sonakshi Vij, Devendra Tayal, and Amita Jain. A Machine Learning Approach for Automated Evaluation of Short Answers Using Text Similarity Based on WordNet Graphs. *Wireless Personal Communications*, 111(2):1271–1282, 3 2020.
71. Qianwen Wang, Zhen Li, Siwei Fu, Weiwei Cui, and Huamin Qu. Narvis: Authoring Narrative Slideshows for Introducing Data Visualization Designs. Technical report.
72. Wu Wen, Xiaobo Xue, Ya Li, Peng Gu, and Jianfeng Xu. Code Similarity Detection using AST and Textual Information. *International Journal of Performability Engineering*, 15(10):2683–2691, 2019.
73. Chung Yi Weng, Wei Ta Chu, and Ja Ling Wu. RoleNet: Movie analysis from the perspective of social networks. *IEEE Transactions on Multimedia*, 11(2):256–271, 2 2009.
74. Polly W Wiessner. Embers of society: Firelight talk among the Ju/'hoansi Bushmen. *PNAS*, 111(39):14027–14035, 2014.
75. Bing Xu, Institute of Electrical and Electronics Engineers. Beijing Section, and Institute of Electrical and Electronics Engineers. *Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC 2017) : March 25-26, 2017, Chongqing, China.*
76. Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, Xiaolong Wang, Zhuoran Wang, and Chao Qi. Neural Response Generation via GAN with an Approximate Embedding Layer *. Technical report.
77. Lin Yao, Zhengyu Pan, and Huansheng Ning. Unlabeled Short Text Similarity with LSTM Encoder. *IEEE Access*, 7:3430–3437, 2019.
78. Tao Zheng, Yimei Gao, Fei Wang, Chenhao Fan, Xingzhi Fu, Mei Li, Ya Zhang, Shaodian Zhang, and Handong Ma. Detection of medical text semantic similarity based on convolutional neural network. *BMC Medical Informatics and Decision Making*, 19(1):1–12, 2019.

79. Shenghan Zhou, Xingxing Xu, Yinglai Liu, Runfeng Chang, and Yiyong Xiao. Text Similarity Measurement of Semantic Cognition Based on Word Vector Distance Decentralization with Clustering Analysis, 2019.

Redes LSTM para la predicción de series temporales

Pablo García Ullán [1], Roberto López González [2], María Angélica González Arrieta [3]

Departamento de Informática y Automática, Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
pablete197@usal.es;robertolopez@artelnics.com;angelica@usal.es
https://www.usal.es/
https://www.artelnics.com/

Resumen La predicción de series temporales mediante redes neuronales es uno de los problemas más demandados en la actualidad, desde predicción de parámetros meteorológicos hasta predicción de valores de bolsa [16]. Los modelos de redes neuronales más simples, como el perceptrón, no son capaces de resolverlos correctamente [12]. Hay que recurrir a redes neuronales más complejas, como las capas densas, redes recurrentes o redes LSTM (Long Short Term Memory) para resolverlos [17].
En este trabajo se desarrollará el marco teórico necesario para implementar redes neuronales LSTM. Se comenzará explicando como preprocesar un conjunto de datos de series temporales, se continuará describiendo el funcionamiento de las redes LSTM, se proseguirá comentando la importancia y la utilidad de la función de costo, se explicarán los algoritmos de entrenamiento y se finalizará exponiendo un ejemplo de validación y un problema real de predicción de series temporales.

Keywords: LSTM (Long Short Term Memory), redes neuronales, series temporales.

1. Introducción

Los problemas de predicción son aquellos en los que la red neuronal tiene que predecir el estado futuro de un sistema en función de observaciones pasadas del mismo. Algunos autores consideran que este tipo de problemas deben considerarse como problemas de aproximación, ya que se podrían resolver aproximando los valores de entrada a una función. En este tipo de problemas las variables de entrada son series temporales con información del pasado y las variables de salida son las correspondientes series temporales de eventos futuros. De este modo, el objetivo de estos problemas es crear un modelo que pueda realizar predicciones de futuro en función de eventos pasados.

Este trabajo se centrará en la resolución de problemas de predicción mediante redes neuronales LSTM. Este tipo de redes se caracterizan por tener "memoria",

y por lo tanto, están especializadas en resolver este tipo de problemas.

La redes LSTM se dieron a conocer en [14]. En este artículo se muestra el funcionamiento de las redes LSTM mediante la explicación de las funciones que la definen. Pero en este trabajo no se explica, entre otras cosas, como habría que minimizar el error de estas redes.

En [13], se explica como debería de llevarse a cabo el proceso que minimiza el error de la red, el cual se denomina retropropagación del error. Pero en este artículo se realiza un planteamiento general, y no se muestran resultados específicos para las redes LSTM.

En [26], una review del año 2020, se realiza un análisis exhaustivo de las redes LSTM. Se detalla su funcionamiento, el proceso mediante el cual la red genera predicciones, el proceso de la retropropagación del error, etc. Pero este trabajo es tan especifico que no explica cómo se deberían conectar las redes LSTM con otro tipo de capas o cómo se debería calcular el error de las redes que contengan LSTM.

Otra fuente de consulta muy frecuente en el ámbito de las redes LSTM es [4]. En esta página web se muestra mucha información relativa a las redes LSTM, pero al tener un fin didáctico, no se muestran los resultados exactos, las figuras están simplificadas, etc.

Como se puede comprobar, no hay una única fuente en la que se englobe todo el marco necesario para poder entender e implementar redes neuronales que contengan capas LSTM. Por esto, en este trabajo se construirá un marco que englobe todos estos conceptos, y sobre el cual, se han desarrollado las redes LSTM de la librería de código abierto OpenNN [3]. Sirva este trabajo para que otros puedan utilizarlo como base para implementar redes neuronales que contengan capas de LSTM.

Para llevar a cabo el desarrollo de dicho marco, el trabajo se ha estructurado de la siguiente forma: En el capítulo 2 se explicará como tienen que ser los conjuntos de datos en los problema de predicción. En el capítulo 3 se mostrará el funcionamiento de las redes LSTM. En el capítulo 4 se explicará la función de costo y el proceso de retropropagación. En el capítulo 5 se explicarán los algoritmos de entrenamiento. En el capítulo 6 se mostrará un ejemplo que valida las LSTM desarrolladas. En el capítulo 7 se realizará un problema real. En el capítulo 8 se expondrán las conclusiones obtenidas. Y finalmente, en el capítulo 9 se expondrán las lineas futuras.

2. Conjunto de datos

Las redes neuronales aprenden a partir del conocimiento almacenado en los conjuntos de datos denominados datasets. Los datasets de series temporales tienen el formato que aparece en la Tabla 1. Como se puede observar, este formato solo dispone de columnas que corresponden con distintas series temporales, es decir, no tiene definidas muestras en sí.

$x_{1,1}$	$x_{1,2}$	$\dots$	$x_{1,v}$
$x_{2,1}$	$x_{2,2}$	$\dots$	$x_{2,v}$
$\dots$	$\dots$	$\dots$	$\dots$
$x_{m,1}$	$x_{m,2}$	$\dots$	$x_{m,v}$

Tabla 1: Estructura de los datos en los problemas de predicción.

Para poder utilizar estos datasets será necesario transfórmalos. Para realizar dicha transformación habrá que establecer un número de *lags* y de *steps ahead*. El número de *steps ahead* definirá el horizonte de tiempo que se quiera predecir, mientras que el número de *lags* definirá cuantos eventos del pasado se utilizarán.

Por ejemplo, si la Tabla 1 se transforma con dos *lags* y dos *steps ahead*, donde la serie v se considera la salida, se generaría el dataset representado en la Tabla 2.

$x_{1,1}$	$x_{2,1}$	$x_{1,2}$	$x_{2,2}$	$\dots$	$x_{1,v-1}$	$x_{2,v-1}$	$x_{1,v}$	$x_{2,v}$
$x_{2,1}$	$x_{3,1}$	$x_{2,2}$	$x_{3,2}$	$\dots$	$x_{2,v-1}$	$x_{3,v-1}$	$x_{2,v}$	$x_{3,v}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$x_{m-1,1}$	$x_{m,1}$	$x_{m-1,2}$	$x_{m,2}$	$\dots$	$x_{m-1,v-1}$	$x_{m,v-1}$	$x_{m-1,v}$	$x_{m,v}$

Tabla 2: Formato de dataset para problema de predicción.

De este modo, cada muestra está formada por el número de variables de entrada, multiplicadas por el número de *lags*, más las variables de salida, multiplicadas por el número de *steps ahead*. El dataset original tenía unas dimensiones de $(m) \times (v)$ y el dataset transformado de $(lags \cdot steps - 1) \times (Entradas \cdot lags + Salidas \cdot steps)$.

Un factor a tener en cuenta a la hora de elegir el número de *lags* y de *steps ahead* son las dimensiones del dataset. Se puede comprobar que el número de filas del dataset transformado es inferior al número de filas del original. Por lo tanto, si la suma de *lags* más *steps ahead* menos uno, es mayor que el número de muestras no se podrá realizar la transformación.

3. Capa LSTM

Las neuronas LSTM son un tipo de neuronas especializadas en la predicción de series temporales. En particular son un tipo de neuronas recurrentes modificadas que evitan los problemas de desvanecimiento de gradiente y aumento de gradiente. De este modo, las redes LSTM podrán realizar mejores predicciones en los problemas de dependencias temporales grandes que las redes recurrentes [26].

3.1. Funcionamiento

La idea principal detrás de las redes LSTM es el estado de la celda. Se podría decir que el estado de la celda es la memoria de la LSTM, y en función de esta memoria la red hará distintas predicciones. Esta memoria sólo sufrirá ligeras modificaciones lineales a lo largo de todo el proceso de transmisión de la información dentro de la LSTM [11].

La red LSTM es capaz de añadir o quitar información del estado de la celda mediante unas estructuras denominadas puertas. La red LSTM cuenta con 4 puertas diferentes, la puerta de entrada, la puerta de salida, la puerta de estado y la puerta de olvido.

Puerta de olvido El proceso de transmisión de la información dentro de la neurona LSTM comienza con la puerta de olvido. Esta se encarga de modificar el estado de la celda de memoria, y es la que decide si se mantiene o se borra la memoria de la red. Si la puerta genera un 1, la memoria se mantiene, pero si genera un 0, la información se borra.

La función de salida de la puerta de olvido se representa mediante la siguiente ecuación:

$$f_t = \sigma(W_f x_t + W_f^R h_{t-1} + b_f) \tag{1}$$

Donde W_f hace referencia a los pesos de olvido, W_f^R a los pesos recurrentes de la puerta de olvido, b_f a los bias de la puerta de olvido, h_{t-1} son los estados ocultos del estado anterior y x_t son las entradas de la neurona.

Puerta de entrada Una vez se ha borrado o no la memoria, llega el momento de decidir si se va a almacenar nueva información o se va modificar la existente. Esta decisión la tomara la puerta de entrada. Si la puerta genera un 1, la información se actualiza, pero si genera un 0, la información no se actualiza.

La función de salida de la puerta de entrada se representa mediante la siguiente ecuación:

$$i_t = \sigma(W_i x_t + W_i^R h_{t-1} + b_i) \tag{2}$$

Donde W_i hace referencia a los pesos de entrada, W_i^R a los pesos recurrentes de la puerta de entrada, b_i a los bias de la puerta de entrada , h_{t-1} son los estados ocultos del estado anterior y x_t son las entradas de la neurona.

Puerta de estado Una vez se ha decidido si la información se actualiza, se decide que elementos son candidatos para ser modificados. Este decisión la lleva a cabo la puerta de estado. Esta puerta decide que elementos son candidatos a ser actualizados.

La función de salida de la puerta de estado se representa mediante la siguiente ecuación:

$$\bar{c}_t = \sigma(W_{\bar{c}} x_t + W_{\bar{c}}^R h_{t-1} + b_{\bar{c}}) \tag{3}$$

Celda de memoria Una vez se ha decidido si borrar o no la memoria, si actualizar o no la información, y en el caso que se decida actualizar, que elementos actualizar, se modifica la celda de memoria. Este proceso de actualización de la celda lo lleva a cabo el estado de la celda de memoria.

La celda de memoria se representa mediante el vector:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \bar{c}_t \tag{4}$$

Puerta de salida Una vez se ha actualizado la celda de memoria, se genera una señal de salida. La puerta encargada de este proceso se denomina puerta de salida. Esta se encarga de crear una señal que definirá la salida del sistema.

La función de salida de la puerta de salida se representa mediante la siguiente ecuación:

$$o_t = \sigma(W_o x_t + W_o^R h_{t-1} + b_o) \tag{5}$$

Estados ocultos Finalmente, una vez se tiene la señal de la puerta de salida de salida, esta se combina con el estado actual de la celda de memoria para dar lugar a un nueva salida de la red o estado oculto.

Los estados ocultos o salidas de las neurona se representa mediante el vector:

$$h_t = y'_t = o_t \cdot \sigma(c_t) \tag{6}$$

Este proceso de transmisión de la información dentro de la neurona se denomina propagación hacia adelante o *forward propagation*, y se recoge en el diagrama de actividad que aparece en la Figura 1.

Se puede comprobar que todas las funciones de salida de las puertas están formadas por dos funciones. Una función de activación, que esta representada por σ y una función de combinación que corresponde con $W_x x_t + W_x^R h_{t-1} + b_x$ a la que denotaremos C o *comb*. En la Figura 1, las puertas están representadas por la unión del círculo $c^{(f)t}(n)$ y el cuadrado $a^t(n)$.

Además, la LSTM podrá disponer de un parámetro denominado paso temporal. Este parámetro se encargará de establecer a cero el valor del estado oculto y estado de la celda cada cierto número de iteraciones. Se podría decir que el paso temporal se encarga de borrar la memoria de la LSTM. Este parámetro deberá ser establecido por el usuario.

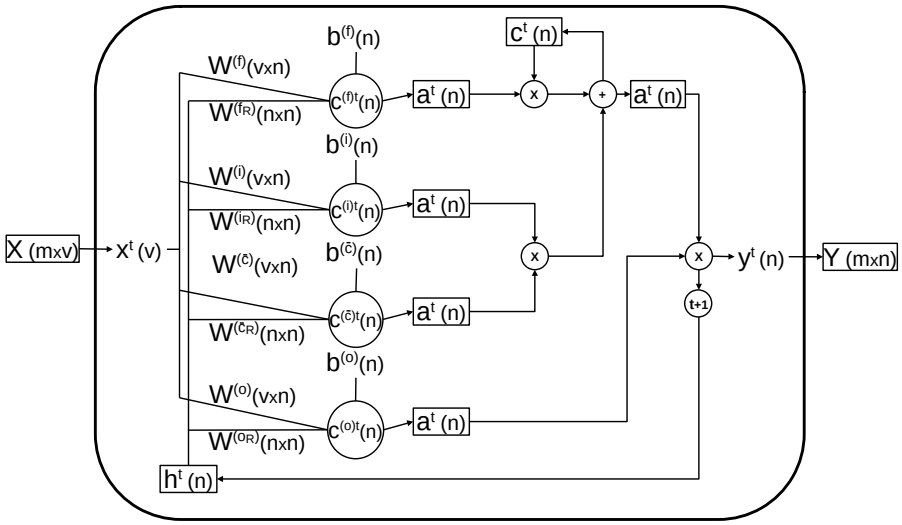


Figura 1: Diagrama de actividad de una neurona LSTM.

Si se quisiera conectar una capa de LSTM con otros capas, no necesariamente de LSTM, simplemente habría que utilizar la salida de la capa como entrada de la siguiente. Matemáticamente este proceso está representado por la composición de funciones. Por lo tanto, la unión de varias capas se expresaría de la siguiente forma:

$$\mathbf{Y} = \mathbf{Y}^l \circ \mathbf{Y}^{l-1} \circ ... \circ \mathbf{Y}^2 \circ \mathbf{Y}^1 \tag{7}$$

Donde $\mathbf{Y}$ hace referencia a la salida de la red, e $\mathbf{Y}^l$ hace referencia a la salida de cada capa.

4. Función de costo

La función de costo tiene un papel muy importante dentro de las redes neuronales. Esta función define la tarea que la red neuronal tiene que lleva a cabo y da una estimación de la calidad de la información que la red tiene que aprender. Se podría decir que el problema del aprendizaje automático se resume como la minimización de esta función [13].

Toda red neuronal tiene una función de costo asociada, la cual contiene un término de error. Este término de error tiene una función de error asociada, definida en función de los parámetros de la red neuronal. El problema del aprendizaje de la red se podría resumir como la búsqueda de los parámetros que minimizan esta función de error, puesto que si la red no comente error alguno, esta realizará predicciones exactas. Para llevar a cabo esta búsqueda es necesario calcular el

gradiente del error, pues este indicará la dirección en la que el error decrece [25].

El gradiente del error de una red neuronal se define como la función de error derivada respecto de todos los parámetros de la red:

$$\nabla e = \begin{pmatrix} \frac{\partial e}{\partial \theta_1} \\ \vdots \\ \frac{\partial e}{\partial \theta_d} \end{pmatrix} \tag{8}$$

Donde d es el numero de parámetros.

El proceso con el que se determina el gradiente del error para poder modificar los parámetros de la red neuronal y así minimizar la función de costo se denomina retropropagación del error.

Para llevarse a cabo el proceso de retropropagación del error es necesario calcular todas las derivada del tipo:

$$\frac{\partial e}{\partial \theta_i^{(l)}} \tag{9}$$

Para poderlas hallar será necesario aplicar la regla de la cadena:

$$\frac{\partial e}{\partial \theta_i^{(l)}} = \sum_{j=1}^{m} \sum_{k=1}^{n} \frac{\partial e}{\partial a_{jk}^{(l)}} \frac{\partial a_{jk}^{(l)}}{\partial \theta_i^{(l)}} \tag{10}$$

El primer término se denomina delta, $\boldsymbol{\Delta}$, y el segundo término es la derivada de la activación respecto de cada parámetro.

Si la red neuronal solo cuenta con una capa, solo habrá una delta, se denominará delta de salida y corresponderá con la derivada del error. En cambio, si hay más de una capa, habrá una delta por capa. En este caso la delta de la última capa se seguirá llamando delta de salida, mientras que el resto deltas ocultas.

La función de la delta de salida es calcular el error con respecto de la función de error escogida, mientras que la función de las deltas ocultas es transmitir el error desde las capas finales a las capas iniciales.

La delta de salida se calcula como:

$$\Delta_{jk}^{(l)} \equiv \frac{\partial e}{\partial a_{jk}^{(l)}} \tag{11}$$

La delta oculta se calcula como:

$$\Delta_{jk}^{(l)} = \sum_{k=1}^{n^{(l+1)}} \Delta_{jk}^{(l+1)} \frac{\partial a_{jk}^{(l+1)}}{\partial a_{jk}^{(l)}} \tag{12}$$

En el caso específico de las redes LSTM, las derivadas de la activación con respecto de cada parámetro son las siguientes:

$$\frac{\partial a_{jk}^{(l)}}{\partial W_f}, \frac{\partial a_{jk}^{(l)}}{\partial W_f^R}, \frac{\partial a_{jk}^{(l)}}{\partial b_f}, \frac{\partial a_{jk}^{(l)}}{\partial W_i}, \frac{\partial a_{jk}^{(l)}}{\partial W_i^R}, \frac{\partial a_{jk}^{(l)}}{\partial b_i}, \frac{\partial a_{jk}^{(l)}}{\partial W_o},$$

$$\frac{\partial a_{jk}^{(l)}}{\partial W_o^R}, \frac{\partial a_{jk}^{(l)}}{\partial b_o}, \frac{\partial a_{jk}^{(l)}}{\partial W_{\bar{c}}}, \frac{\partial a_{jk}^{(l)}}{\partial W_{\bar{c}}^R}, \frac{\partial a_{jk}^{(l)}}{\partial b_c}$$

Si se combina el concepto de delta y el de derivada parcial del error con respecto a cada parámetro, se determina el gradiente del error y se puede llevar a cabo el proceso de retropropagación.

Para hacerse una idea de como son las derivadas del error con respecto a cada parámetro en el caso de las LSTM, a continuación se muestra la derivada parcial del error con respecto de los pesos sinápticos de la puerta de olvido:

$$\frac{\partial e_t}{\partial W_f} = \Delta \cdot \left(\frac{\partial o_t}{\partial W_f} \cdot a(c_t) + o_t \cdot \frac{\partial a(c_t)}{\partial W_f} \right) \tag{13}$$

$$\frac{\partial o_t}{\partial W_f} = \frac{\partial o_t}{\partial comb_t^o} \cdot \frac{\partial comb_t^o}{\partial h_{t-1}} \cdot \frac{\partial h_{t-1}}{\partial W_f} \tag{14}$$

$$\frac{\partial a(c_t)}{\partial W_f} = \frac{\partial a(c_t)}{\partial c_t} \left(\frac{\partial f_t}{\partial W_f} \cdot c_{t-1} + f_t \cdot \frac{\partial c_{t-1}}{\partial W_f} + \frac{\partial i_t}{\partial W_f} \cdot \bar{c}_t + i_t \cdot \frac{\partial \bar{c}_t}{\partial W_f} \right) \tag{15}$$

$$\frac{\partial i_t}{\partial W_f} = \frac{\partial i_t}{\partial comb_t^i} \cdot \frac{\partial comb_t^i}{\partial h_{t-1}} \cdot \frac{\partial h_{t-1}}{\partial W_f} \tag{16}$$

$$\frac{\partial \bar{c}_t}{\partial W_f} = \frac{\partial \bar{c}_t}{\partial comb_t^{\bar{c}}} \cdot \frac{\partial comb_t^{\bar{c}}}{\partial h_{t-1}} \cdot \frac{\partial h_{t-1}}{\partial W_f} \tag{17}$$

$$\frac{\partial f_t}{\partial W_f} = \frac{\partial f_t}{\partial comb_t^f} \left(x_t + W_f^R \frac{\partial h_{t-1}}{\partial W_f} \right) \tag{18}$$

Si la red neuronal está compuesta por una sola capa, la delta que aquí aparece será la delta salida. Mientras que si se está formada por varias, será la oculta. Gracias a este sistema de deltas, denominado regla de las deltas, se puede transmitir el error entre las distintas capas de una red neuronal [22].

Las otras once derivadas parciales se realizan de manera análoga a la aquí expuesta. Y una vez calculadas, se puede proceder a minimizar el error de manera sistemática mediante un algoritmo de entrenamiento.

5. Estrategia de entrenamiento

Como se mencionó en la sección anterior, el problema del aprendizaje automático se podría resumir como la optimización de la función de costo. Este proceso de optimización se podría entender como la búsqueda de los parámetros de la red que minimizan dicha función. El proceso de búsqueda de los parámetros óptimos se lleva a cabo mediante un algoritmo de entrenamiento.

El proceso de entrenamiento se podría resumir de la siguiente manera: Primero, las muestras de entrenamiento se introducen en la capa de entrada. Segundo, la información se transmite a lo largo de toda la red en el proceso de propagación hacía adelante. Tercero, se calcula el gradiente del error mediante un algoritmo de retropropagación. Cuarto, se modifican los parámetros de la red, y quinto, se repite todo el proceso hasta minimizar el error.

Este proceso de entrenamiento es llevado a cabo por los denominados algoritmos de entrenamiento. Hay muchos tipos de algoritmos de entrenamiento, pero los más famosos se podrían considerar el método de Quasi-Newton y el algoritmo de estimación de momento adaptativo. En este trabajo se utilizará el algoritmo estimación de momento adaptativo o ADAM debido a su rápida convergencia [18].

El algoritmo de entrenamiento tendrá que aplicar un proceso de retropropagación del error, puesto que el gradiente que se obtiene le indicará como tiene que modificar los distintos parámetros de la red. De este modo, el algoritmo los modificará de manera que se minimice la función de costo.

Habitualmente las funciones de costo a minimizar son funciones con parámetros no lineales, y como consecuencia, no se puede encontrar un algoritmo de optimización cerrado para hallar los mínimos. Por esto, la búsqueda de los parámetros se realizará mediante una sucesión de pasos como la siguiente:

$$\theta_{i+1} = \theta_i + \Delta\theta_i \tag{19}$$

Donde i denota el número de la iteración, θ_i es el valor del parámetro actual, $\Delta\theta_i$ es la variación que se introduce en la iteración i, y θ_{i+1} es el nuevo parámetro.

Así, cuando se inicia el proceso de entrenamiento de la red neuronal se parte con unos parámetros (habitualmente elegidos de forma aleatoria), y en cada iteración del algoritmo de entrenamiento, se van modificando teniendo en cuenta la dirección del gradiente. Este proceso se repite iterativamente hasta que el error se minimiza.

Para que el proceso de entrenamiento se detenga y no se realicen infinitas iteraciones, se definen criterios de parada. Algunos de los criterios más habituales son los siguientes: el valor de la función de costo es menor que un valor

preestablecido, se ha realizada un número determinado de iteraciones o se ha estado entrenando durante un tiempo determinado.

6. Ejemplo de validación: Péndulos acoplados

Una red neuronal con al menos una única capa oculta de neuronas con función de activación no lineal generará un marco de trabajo en el que cualquier función de un espacio dimensional de dimensiones finitas y con un número de discontinuidades finitas se podrá aproximar siempre y cuando tenga el número suficiente de neuronas. A esto se le denomina teorema de aproximación universal de las redes neuronales. [15]

Esto quiere decir que si se introduce en la red neuronal un dataset que sea totalmente analítico, la red lo tiene que aproximar sin cometer error alguno. El dataset analítico realizado en este trabajo corresponde con el movimiento de 10 péndulos acoplados. Este dataset es totalmente analítico puesto que el movimiento de los péndulos se rige por el siguiente hamiltoniano:

$$\frac{1}{2}I\ddot{\theta}_n - C(\theta_{n+1} + \theta_{n-1} - 2\theta_n) + mgl\sin\theta_n \tag{20}$$

Mediante este hamiltoniano se generan diez series temporales que corresponden con la posición de los péndulos durante su movimiento acoplado. Si la red consigue predecir con un error mínimo el movimiento de dichos péndulos, se validaría el funcionamiento de la misma.

6.1. Series de entrada

En las Figuras 2 y 3 se representan 2 de las 10 columnas generadas en el dataset. Para que se pueda distinguir la forma de la series temporales no se representaran todas las muestras de la serie, sino que sólo se representarán las 500 primeras.

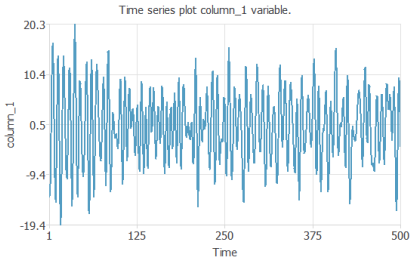


Figura 2: Representación de las 500 primeras muestras de la serie 1

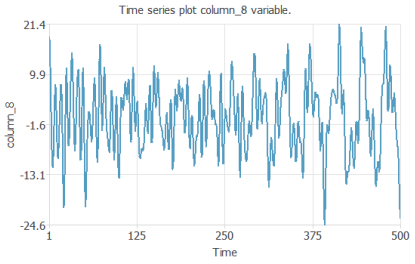


Figura 3: Representación de las 500 primeras muestras de la serie 8

Como ya se comentó en la sección 2, estas series no se pueden introducir tal cual en la red neuronal, hace falta transformarlas. La transformación que mejores resultados ha proporcionado es la que se genera mediante el uso de tres *lags* y un *step ahead*.

6.2. Red neuronal

La red neuronal que proporciona mejores resultados es la que se refleja en la Figura 4.

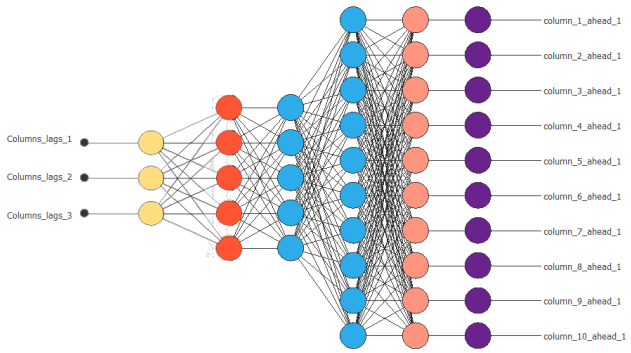


Figura 4: Arquitectura de la red neuronal

En esta imagen sólo se muestran 3 variables de entrada (puntos negros) y 3 neuronas de escalado (círculos amarillos) puesto que las treinta varaibles de entrada se han agrupado dentro de los tres *lags* correspondientes.

Esta red cuenta con una capa capa de escalado de 30 neuronas (círculos amarillos), una para cada entrada, una capa de 5 neuronas LSTM (círculos rojos), dos capas de perceptrones de 5 y 10 neuronas respectivamente (círculos azules), una capas de 10 neuronas de desescalado (círculos color salmón), una para cada salida, y una capa de 10 neuronas de bounding (círculos morados). Las 30 entradas y neuronas de desescalado se han agrupado en 3 grupos correspondiendo con los *lags*.

La capa de bounding al final no se utilizó para el desarrollo del trabajo, pero al estar implementada se mantuvo. Esta capa se utilizaría para acotar el rango de las predicciones de la red.

6.3. Estrategia de entrenamiento

Para llevar a cabo el entrenamiento del modelo se eligió el error cuadrático normalizado puesto que el error no aumenta con el tamaño del dataset y un

valor de error próximo a cero suele coincidir con un error real próximo a cero. Además, se utilizó el algoritmo de entrenamiento ADAM debido a su rápida convergencia [18].

En la Figura 5 se muestra el proceso de entrenamiento por el cual se va disminuyendo el error que la red neuronal comente.

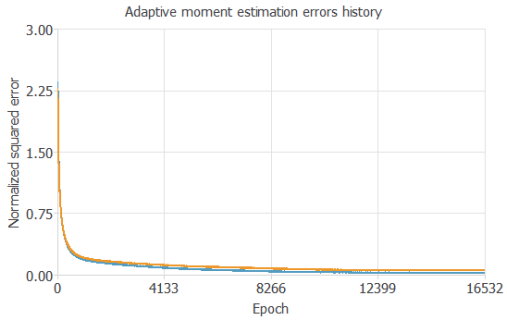


Figura 5: Descenso del error de la red mediante el algoritmo ADAM

En esta gráfica se muestra con una linea naranja el error que la red comente con las muestras de entrenamiento, y en azul el error que comete con las muestras de selección.

Como se puede observar, a partir de los valores aleatorios que se asignan a los parámetros de la red, esta comente un error de entorno a 2.25, pero tras 16.532 épocas de entrenamiento el error de la red disminuye hasta un 0.009. Un error muy pequeño que, a priori, muestra que la red es casi perfecta.

Si se quisiera disminuir este error habría que aumentar el número de muestras del dataset, pero esto aumenta de manera muy considerable el tiempo de entrenamiento. Y un error de 0.009 es suficiente para mostrar que las redes LSTM funcionan correctamente.

6.4. Validación de los resultados

La forma más sencilla de validar los resultados es utilizar muestras que la red no haya visto con anterioridad y comparar las predicciones que genera con los valores esperados. En las Figuras 6 y 7 se muestran las predicciones realizadas por la red y sus correspondientes valores esperados para 200 muestras nuevas.

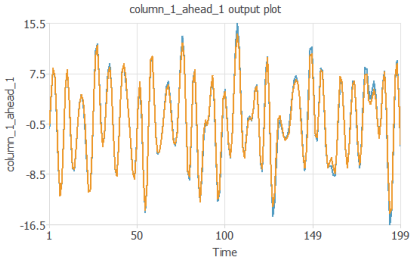


Figura 6: Predicciones y valores esperados para la columna 1

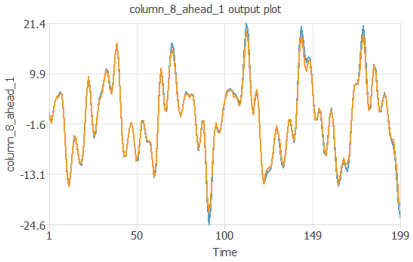


Figura 7: Predicciones y valores esperados para la columna 8

Donde la linea naranja corresponde con los resultados predichos y la linea azul con los esperados. Como se pude observar la red predice sin cometer casi error alguno, lo cual es consistente con el descenso de error obtenido durante el entrenamiento.

En la Figura 8 se muestran los errores mínimo, máximo y medio de la red neuronal, junto con su desviación estándar.

	Minimum	Maximum	Mean	Deviation
Absolute error	0.000607491	1.81079	0.403384	0.397477
Relative error	1.5215e-5	0.0453526	0.010103	0.0099551
Percentage error	0.0015215	4.53526	1.0103	0.99551

Figura 8: Estadísticas de los errores

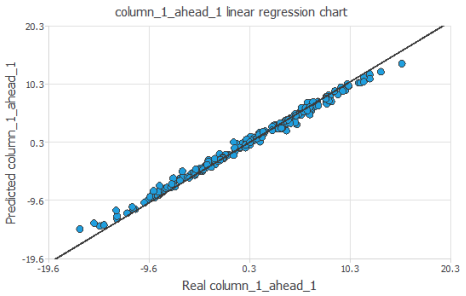
Se puede observar que todos los valores de error obtenidos son muy bajos. El valor medio del error relativo es de 0.01 lo cual corresponde con un error de un uno por ciento en cada predicción realizada. Este porcentaje es muy bajo teniendo en cuenta que se está realizando un problema de predicción.

Otro método para validar los resultados generados es la regresión lineal. Esta técnica estadística mide el rendimiento del modelo comparando las relaciones entre las predicciones realizadas y los valores esperados.

La regresión lineal está definida por tres parámetros a, b y r^2. Los parámetros a y b corresponden con la ordenada en el origen y la pendiente de la mejor regresión lineal posible entre las variables de salida y las predicciones de la red. El tercer parámetro r^2, es el coeficiente de correlación lineal entre las salidas y las predicciones.

En el caso de una predicción perfecta, los puntos se adaptarán a la recta de regresión generada, la pendiente de la recta de regresión sera 1, la ordenada en el origen 0 y la correlación 1.

En la Figura 9 se muestra la regresión y sus parámetros obtenidos usando este modelo.



	Value
Intercept	-0.00638
Slope	1.04
Correlation	0.997

Figura 9: Representación y parámetros de las regresión lineal de la salida de la columna 1

Como se puede observar los resultados obtenidos son muy buenos. Los puntos se adaptan de forma correcta a la recta, el valor de la ordenada en el origen es muy próximo a cero y el valor de la pendiente y de la correlación son muy próximos a uno.

Gracias a los resultados obtenidos se puede confirmar que las redes implementadas en la librería OpenNN funcionan de manera de correcta. Sólo faltaría aplicarlas a un problema real para observar su funcionamiento.

7. Aplicación real: Predicción de la calidad del aire

En esta sección se desarrollará un problema de predicción en el que se verá como se comportan las redes LSTM frente a problemas del mundo real. Al no ser un problema analítico habrá ruido, será imposible incluir todas las variables que afecten a dicho modelo, podrá haber eventos imposibles de contemplar, etc. Por lo tanto, este ejemplo mostrará el verdadero potencial de las redes LSTM.

El problema ha resolver es el de la predicción de la contaminación atmosférica en la ciudad de Madrid a lo largo de un año. Este modelo se centrará en predecir los niveles de dióxido de nitrógeno en la atmósfera. Este gas es generado principalmente por la polución de los coches y de las centrales de generación de electricidad. Su predicción es de importancia en la actualidad puesto que es un

gas que afecta a la capacidad pulmonar de las personas, especialmente la de los niños, y es el principal causante de la lluvia ácida.

El dataset que se ha generado para predecir la calidad del aire en la ciudad de Madrid consta de dos partes. La primera, corresponde con las condiciones climatológicas de la ciudad: temperatura media, precipitaciones, velocidad media del aire, rachas del aire y presión máxima. La segunda, corresponde con dos indicadores de la calidad del aire: niveles de dióxido de nitrógeno y niveles de ozono. Ambos dataset contienen valores desde el 19/08/2005.

La información relativa a las condiciones climatológicas ha sido extraída de la base de datos libres de AEMET, mientras que los niveles de la calidad del aire han sido extraídos de la base de datos del ayuntamiento de Madrid.

7.1. Series de entrada

En las Figuras 10 y 11 se muestran dos de las series temporales que componen el dataset, específicamente se muestran los valores de la temperatura media y los niveles de NO_2 a lo largo de 500 días.

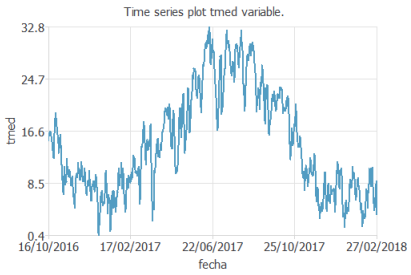


Figura 10: Representación de la temperatura media

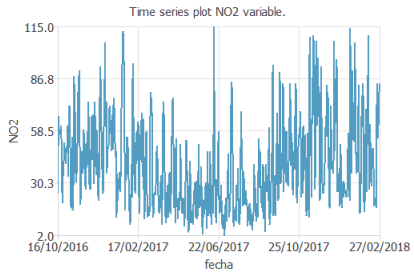


Figura 11: Representación de los niveles de NO_2

Al igual que en el ejemplo de validación, estas series no se pueden introducir tal cual en la red neuronal, hace falta transformarlas. En este caso, la transformación que mejores resultados ha proporcionado es la que se genera mediante el uso de un *lag* y de un *step ahead*.

7.2. Red neuronal

La red neuronal que proporciona mejores resultados es la que se refleja en la Figura 12.

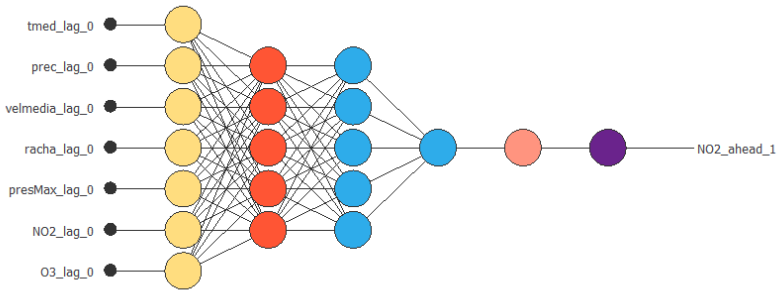


Figura 12: Arquitectura de la red neuronal

En este caso la red cuenta con una capa capa de escalado de 7 neuronas (círculos amarillos), una para cada entrada, una capa de 5 neuronas LSTM (círculos rojos), dos capas de perceptrones de 5 y 1 neuronas respectivamente (círculos azules), una capas de 1 neurona de desescalado (círculos color salmón), una para cada salida, y una capa de 1 neurona de bounding (círculos morados).

7.3. Estrategia de entrenamiento

Al igual que en el caso de validación, el proceso de entrenamiento del modelo se llevó a cabo utilizando el error cuadrático normalizado y el algoritmo de entrenamiento ADAM.

En la Figura 5 se muestra el proceso de entrenamiento por el cual se va disminuyendo el error que la red neuronal comente.

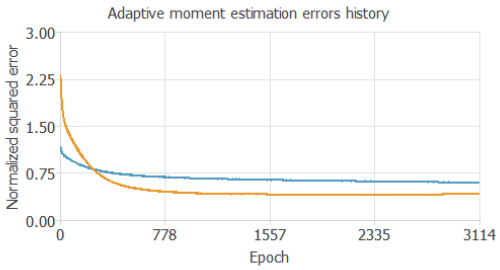


Figura 13: Descenso del error de la red mediante el algoritmo ADAM

En esta gráfica se muestra con una linea naranja el error que la red comente con las muestras de entrenamiento, y en azul el error que comete con las mues-

tras de selección.

Como se puede observar, el error que la red comente en el proceso de selección sólo disminuye hasta un 0.416. A primera vista, este error de entrenamiento puede parecer muy grande comparado con el obtenido en el caso de validación, pero habrá que observar cual es el error real cometido en cada predicción.

Es normal que el error obtenido sea mayor, puesto que ya que no se está trabajando con un dataset analítico, sino que con uno real. Al ser un dataset real habrá ruido, variables que no se estén considerando, variables que puedan no estar correlacionadas y por lo tanto introduzcan fallos en el modelo, etc.

En las siguientes secciones se mostrará que este error no es tan grande como puede parecer a priori.

7.4. Validación de los resultados

Al igual que en el ejemplo de validación, en la Figura 14 se muestran las predicciones de los niveles de NO_2 que la red produce junto con sus valores esperados.

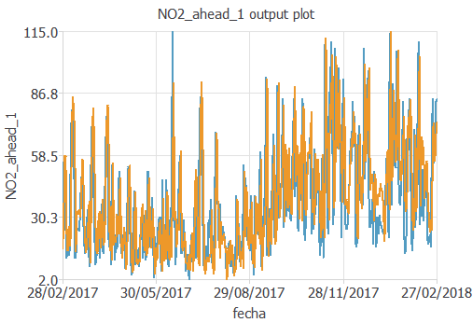


Figura 14: Predicciones y valores esperados para el nivel de NO_2

Donde la linea naranja corresponde con los resultados predichos y la linea azul con los resultados esperados. Como se pude comprobar, la red no realiza mala predicciones, sólo comete fallos en los picos bruscos, lo cual es comprensible puesto que son las franjas más difíciles de predecir, pero el modelo aproxima de forma correcta el resto de franjas.

En la Figura 15 se muestran los errores mínimo, máximo y medio de la red neuronal, junto con su desviación estándar.

	Minimum	Maximum	Mean	Deviation
Absolute error	0.0227737	71.9795	19.548	13.3714
Relative error	0.000106969	0.338091	0.0918176	0.0628059
Percentage error	0.0106969	33.8091	9.18176	6.28059

Figura 15: Estadísticas de los errores

En esta imagen se puede apreciar que los valores de error absoluto son muy grandes, esto se debe a que el error absoluto se acumula, y al haber muchas predicciones, su valor aumenta de forma considerable. Así, para este tipo de problemas es conveniente utilizar el error medio y el porcentaje. Si se observan dichos valores se puede comprobar que la red sólo comente un error de un 9 % en cada predicción.

Teniendo en cuenta los resultados mostrados en la Figura 14 y que el error es solo del 9 % se puede concluir que se ha realizado un buen modelo de predicción.

Gracias a los resultados obtenidos se puede concluir que las redes LSTM también realizan un buen trabajo en los problemas de predicción reales.

8. Conclusiones

A partir del trabajo mostrado en los capítulos anteriores se pueden extraer las siguientes conclusiones:

- En este trabajo se ha explicado el marco de trabajo de las redes LSTM, se han validado correctamente y se han aplicado a un caso real.

- Se ha realizado un ejemplo de validación, problema de los diez péndulos acoplados, que se resuelve llegando a obtener un error muy próximo a cero. Gracias a las propiedades de las redes neuronales y al teorema de aproximación universal, el hecho de que se haya resuelto el problema con error cero demuestra que las redes LSTM funcionan de manera correcta.

- Se ha realizado un ejemplo de aplicación real, problema de la predicción de la calidad del aire en la ciudad de Madrid, obteniendo buenos resultados. Con este ejemplo se han llegado a realizar predicciones con un error del 9 %. Si se tiene en cuenta que estamos trabajando con un problema real, un error del 9 % es un error muy bajo. Esto se refleja en la Figura 14, en la cual se puede observar que las predicciones se ajustan de una manera más que correcta a los valores esperados.

- Los resultados obtenidos mediante LSTM son mejores que los obtenidos mediante otros métodos, como por ejemplo, las capas densas de perceptrones. Este último método comete un error de un 9.8 % en el problema de la calidad

del aire de la ciudad de Madrid. Este hecho no se ha destacado previamente pues una mejora de un 0.8 % no es realmente notoria, pero merece la pena mencionarlo.

9. Trabajo futuro

Como continuación a este trabajo se podrían plantear varias opciones:

- Modificar las redes LSTM ya definidas para crear redes LSTM-FCN (Long Short Term Memory Fully Convolutional Network), que teóricamente proporcionan mejores predicciones.

- Implementar métodos de selección de modelo óptimo. Estos métodos establecerían cuales son los parámetros óptimos que han de introducirse en la red neuronal. Algunos de estos algoritmos podrían ser *growing neurons* y *prunning inputs*.

- Aplicar las redes LSTM a otros tipos de problemas que no sean de predicción numérica. Por ejemplo, se podrían utilizar para procesos de traducción automática, modelado del lenguaje, procesamiento de lenguaje multilingüe, etc.

- Añadir otros tipos de capas, como las convolucionales, recurrentes, etc. Para así poder aplicar las redes a otros tipos de problemas, como por ejemplo, el subtitulado automático de imágenes.

Referencias

1. Deriving back propagation on simple rnn/lstm. `https://towardsdatascience.com/back-to-basics-deriving-back-propagation-on-simple-rnn-lstm`.
2. Neuraldesigner. `https://www.neuraldesigner.com/`.
3. Open neural network. `https://www.opennn.net/`.
4. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.
5. Mark Hudson Beale, Martin T Hagan, and Howard B Demuth. Neural network toolbox. *User's Guide, MathWorks*, 2:77–81, 2010.
6. Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.
7. George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
8. Thierry Dauxois and Michel Peyrard. *Physics of solitons*. Cambridge University Press, 2006.
9. Serge Vasilevich Fomin et al. *Elements of the theory of functions and functional analysis*, volume 1. Courier Corporation, 1999.
10. Izrail Moiseevitch Gelfand, Richard A Silverman, et al. *Calculus of variations*. Courier Corporation, 2000.
11. F.A. Gers, N.N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 3(1):115–143, 2003. cited By 787.

12. K. Greff, R.K. Srivastava, J. Koutnik, B.R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017. cited By 1566.
13. S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, 6(2):107–116, 1998. cited By 705.
14. S. Hochreiter and J. Schmidhuber. Lstm can solve hard long time lag problems. pages 473–479, 1997. cited By 324.
15. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
16. Caihong Hu, Qiang Wu, Hui Li, Shengqi Jian, Nan Li, and Zhengzheng Lou. Deep learning with a long short-term memory networks approach for rainfall-runoff simulation. *Water*, 10:1543, 10 2018.
17. F. Karim, S. Majumdar, H. Darabi, and S. Chen. Lstm fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669, 2017. cited By 283.
18. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
19. Roberto Lopez and Eugenio Oñate. A variational formulation for the multilayer perceptron. In Stefanos D. Kollias, Andreas Stafylopatis, Włodzisław Duch, and Erkki Oja, editors, *Artificial Neural Networks – ICANN 2006*, pages 159–168, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
20. David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.
21. F.J. Ordóñez and D. Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors (Switzerland)*, 16(1), 2016. cited By 895.
22. DL Prados and SC Kak. Neural network capacity using delta rule. *Electronics Letters*, 25(3):197–199, 1989.
23. William H. Press, William T. Vetterling, Saul A. Teukolsky, and Brian P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, USA, 2nd edition, 2001.
24. Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.
25. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
26. A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404, 2020. cited By 130.

Algoritmos de Detección de Anomalías en Aprendizaje Automático

Manuel Luque Cuesta [1], Roberto López González [2], María Angélica González Arrieta [3]

Departamento de Informática y Automática, Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
idu23153@usal.es;robertolopez@artelnics.com;angelica@usal.es
https://www.usal.es/
https://www.artelnics.com/

Resumen El aprendizaje automático se utiliza cada vez más y más, desde la mejora de la vida cotidiana hasta en procesos industriales avanzados. Uno de sus mayores problemas es la ocurrencia de anomalías en los datos, las cuales desestabilizan los modelos de aprendizaje. Estas anomalías son datos que se salen de las distribuciones "normales" de los datos y que merece la pena encontrar. Este trabajo surge como respuesta a esta problemática, cuyo objetivo es crear algoritmos de detección de anomalías en la librería *OpenNN*, de forma que estos sean capaces de enfrentarse a grandes conjuntos de datos en tiempos no limitantes, es decir, mejorando, en cuanto a eficiencia, las librerías de aprendizaje automático que sean estado del arte y que posean algoritmos de este tipo.

Keywords: Detección de anomalías, C++, algoritmos de detección de anomalías, inteligencia artificial

1. Introducción

En el mundo del aprendizaje automático se necesitan datos, datos en grandes cantidades, extraídos generalmente a través de mecanismos (*e.g.: procesos industriales, factores ambientales*) o aplicaciones (*e.g.: compras de usuarios, tipo de uso*). Estos datos son utilizados para entrenar los modelos, en su mayoría matemáticos, que más tarde permitirán realizar predicciones o estimaciones.

Dado que los modelos dependen intrínsecamente de los datos, para obtener un modelo "perfecto" para un determinado problema estos deberían ser del mismo modo. Empero, los datos contienen, en muchas ocasiones, "errores"; errores entendidos como ruido o anormalidades (más adelante se detallará esto). Cuando estos se salen de las distribuciones espaciales "normales" de los datos se consideran anomalías (*outliers*). Estos datos fuera de lugar se llevan investigando desde los albores del aprendizaje automático y el análisis de datos [16], ya que entorpecen el aprendizaje del modelo y la validación del mismo, y se pretende, por tanto, analizar su origen o eliminarlos.

A la par del desarrollo de algoritmos de aprendizaje automático han ido surgiendo librerías de código abierto de estos, librerías de las que, hoy más que nunca, se promueve su uso. Entre estas librerías se encuentra *OpenNN* (*Open Neural Network*) [24], la cual tiene como características principales el manejo de conjuntos de datos (*datasets*) y la creación, entrenamiento y evaluación de modelos de redes neuronales; permitiendo resolver problemas de clasificación, regresión, predicción y asociación. Además, esta librería se usa como base para otros programas, por lo que se espera que sea óptima computacionalmente hablando, es decir, lo más rápida posible en cuanto a velocidad de ejecución y económica en cuanto a espacio. Cabe remarcar también que esta librería aporta muchas ventajas desde el punto de vista de eficiencia computacional, como la mayor capacidad de patrones que otras librerías similares o la mayor velocidad de computo utilizando también la tarjeta gráfica. Además, existe una herramienta gráfica llamada *Neural Designer* para construir redes neuronales, la cual está basada en la librería y permite su utilización para usuarios con escasos conocimientos.

Se espera que esta librería sea capaz de detectar y eliminar las anomalías antes del proceso de aprendizaje, de forma que este sea el mejor posible. Además, sirve de muy poco crear algoritmos que ya estén previamente implementados, si no mejoran en nada, de tal forma que se espera que mejoren en cuanto a eficiencia l estado del arte que en este caso son librerías como *PyOD* o *Scikit-Learn*, las cuales son dos librerías enfocadas al aprendizaje automático; la primera más enfocada a la detección de anomalías y la segunda con propósito general.

Por ello, el objetivo de este trabajo consiste en desarrollar algoritmos de detección de anomalías en la librería *OpenNN*, consiguiendo mejores resultados que *Scikit-Learn*. Esto se logrará mediante un conocimiento experto sobre anomalías y los algoritmos de detección, así como del uso de las estructuras más eficientes y del paralelismo.

Este documento, se dividirá en las siguientes secciones: en la sección 2 Bases teóricas se expone la teoría necesaria para comprender las anomalías y librerías relacionadas; en la sección 3 Marco experimental, se exponen las pautas bajo las que se experimenta; en la sección 4 *Local Outlier Factor* se explican las bases de dicho algoritmo, las mejoras realizadas y sus experimentos relacionados en los cuales se demostrarán sus cualidades, asimismo ocurre con la sección 5 *Isolation Forest* en la que se exponen las bases de este algoritmo y sus experimentos. La última sección será 6 Conclusiones y líneas futuras.

2. Bases teóricas

2.1. Detección de anomalías

Las anomalías son: "instancias de datos que se desvían, de manera significante de la mayoría de las instancias"; tal y como están definidas en [20]. En este otro artículo las definen como: "patrones en los datos que no se ajustan a su comportamiento esperado" [31]. En general, se podría decir que las anomalías

son instancias o patrones que se alejan de las distribuciones normales de los datos, o que perteneciendo a una distribución su etiqueta difiere de esta.

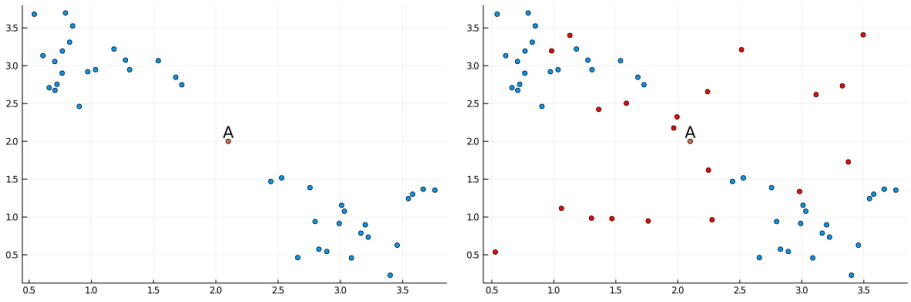
El origen de estas anomalías, es variado, de forma que se podrían considerar varios tipos; en este documento se van a considerar dos: anormalidades y ruido.

Las anormalidades serán entendidas como patrones que, por causas mal intencionadas o a causa de una nueva tendencia en un entorno como puede ser una población, forman reducidos grupos homogéneos en el espacio (en ocasiones, de un único patrón), alejados de sus valores típicos. Por ejemplo: nuevas tendencias climatológicas en una zona, debido al cambio climático.

Mientras que el ruido será entendido como aquellos patrones que, por causas no mal intencionadas o errores en las mediciones, no están localizados espacialmente donde deberían. Estos patrones suelen ser heterogéneos debido a su que su origen proviene de errores. El ruido también puede provocar que patrones se camuflen en distribuciones de otras clases, o que incluso se mantengan en su propia clase, pero en otro lugar espacial; sin embargo en muchos de estos casos no son considerados anomalías. Por ejemplo: una persona contesta de manera perezosa una encuesta o la entiende de manera incorrecta.

Ambos, el ruido y las anormalidades, convienen ser detectados y eliminados antes de realizar cualquier enfoque de aprendizaje automático, debido a que entorpecen el aprendizaje del modelo, o mejor dicho, de las distribuciones normales de los datos.

Para comprenderlo mejor, en la Figura 1a aparece un ejemplo, en dos dimensiones, de anormalidad o anomalía fuerte (el punto "A"), entre dos distribuciones de datos, pudiéndose tratar, por ejemplo, de una enfermedad; en tanto que en la Figura 1b, se encuentra el mismo punto "A" y las distribuciones, aunque se observan muchos puntos más, estos puntos podrían considerarse ruido junto con el punto "A" ya que se observa que no siguen ninguna distribución y deberían eliminarse para poder aprender correctamente las distribuciones.



(a) Ejemplo de anomalía. (b) Ejemplo de ruido.

Figura 1: Diferencia entre anomalía y ruido

La detección de anomalías intenta encontrar aquellos eventos minoritarios, inesperados y raros, teniendo únicas problemáticas que son comunes a todos los métodos:

- **Desconocimiento**: muchas incógnitas sobre de donde provienen las anomalías.
- **Clases anómalas y heterogéneas**: las anomalías pertenecen todas a una clase, pero son muy diferentes entre sí.
- **Tipos diversos de anomalías**: (clasificación según el espacio) anomalías puntuales, son aquellas que se encuentran aisladas en el espacio; anomalías condicionales, son aquellas que según el contexto son anomalías o no; y las anomalías grupales, son aquellas que forman una nueva tendencia o un grupo infracciones, pero que son un grupo minoritario.

Existen varias clasificaciones mediante las que encasillar algoritmos para detección de anomalías, entre ellas, se expondrán dos: según la salida del algoritmo y según la metodología a la hora de encontrar anomalías.

Según el resultado que produce el algoritmo de detección de anomalías, se pueden encontrar dos tipos:

- **Puntuación**: se valora cómo de anómalos son cada uno de los patrones que conforman el conjunto de datos, pero no se puede estimar con seguridad cuantos *outliers* hay.
- **Binario**: la salida determina si un determinado punto o patrón es anomalía o no. Si bien es posible transformar un algoritmo de puntuación en uno binario utilizando un umbral, perdiéndose información.

Por otro lado, los algoritmos de detección de anomalías se pueden clasificar según su enfoque a la hora de encontrarlas:

- **Análisis de valores extremos**: hay anomalías en las que se asume que los valores de sus variables son muy altos o muy bajos.
- **Estadísticos y probabilísticos**: la clave en este tipo de modelos es determinar la distribución de los datos. Lo más común es utilizar una mixtura de gausianas, cuyos parámetros son extraídos mediante el algoritmo de esperanza-maximización.
- **Lineales**: estos algoritmos se basan en reducir el espacio dimensional, como por ejemplo con la técnica *PCA*, para más tarde emplear una regresión y crear un plano o recta que intente cometer el mínimo error al aproximar los puntos.
- **Basados en proximidad**: la idea de la que se parte es la detectar patrones que están aislados, existiendo tres enfoques: análisis de *cluster* (grupo), basados en densidad y en vecinos más cercanos.

2.2. Librerías con detección de anomalías

El principal objetivo es desarrollar algoritmos de detección de anomalías que sean más eficientes que el estado del arte, para ello se han de analizar librerías de detección anomalías, las cuales sean usadas asiduamente.

La librería más utilizada para el aprendizaje automático en el lenguaje de programación *Python* es *Scikit-Learn*. Esta posee una pequeña sección [7] sobre detección de anomalías, en la que expone las bases sobre esta rama y unos cuantos algoritmos de aprendizaje. En su pequeño tutorial, dividen a las anomalías en dos tipos: *novelties* y *outliers*. Al primero lo consideran como las anomalías que aparecen como datos nuevos; mientras que al segundo lo llaman así cuando las anomalías ya se encontraban presentes en los datos. Los algoritmos más representativos de esta librería para la detección de anomalías son *Local Outlier Factor* y *Isolation Forest*.

Otra librería desarrollada en *Python* es *PyOD* (*Python Outlier Detection*) [32], aunque esta si que se centra completamente en la detección de anomalías. Fue creada en 2017 y cuenta con hasta 32 algoritmos de detección de anomalías, algunos muy similares entre sí. Lo que es más, ha sido usada en multitud de investigaciones científicas. De los 32 algoritmos, tanto *Local Outlier Factor* como *IsolationForest* son utilizado desde la librería *Scikit-Learn*.

3. Marco experimental

El objeto de esta sección es definir los experimentos comunes a los dos algoritmos implementados, con el supuesto de que pudieran realizarse más pruebas o experimentos para un determinado algoritmo dada su naturaleza.

3.1. Hardware

Son los componentes del ordenador utilizados para el desarrollo, pruebas y experimentación. Aparecen en la Tabla 1.

Dispositivo	**Especificaciones**
CPU	Intel(R) Core(TM) i5-9600K CPU @ 3.70 GHz, 6 núcleos
Memoria RAM	16 *GB* DDR4
Almacenamiento	500 *GB* (SSD)
GPU	*NVidia* GeForce RTX 2060 Super

Tabla 1: Entorno hardware.

3.2. Conjuntos de datos reales

Los conjuntos de datos para detección de anomalías se han obtenido de ODDs (*Outlier Detectetion DataSet*) [8], un dominio web en el que existen muchos conjuntos en los que se sabe a ciencia cierta qué puntos son anomalías y cuales no. De entre estos, se han seleccionado variados en cuanto a número de patrones, número de características y porcentaje de anomalías; los seleccionados pueden verse en la Tabla 2.

Nombre	Nº de patrones	Variables	Contaminación %
Arrhythmia	452	274	14.6
Cardio	1831	21	9.61
ForestCover	286048	10	0.9
Ionosphere	351	33	36
Lympho	148	18	4.1
Mnist	7603	100	9.2
Satellite	6435	36	32
Shuttle	49097	9	7
Vowels	1456	12	3.4

Tabla 2: Conjuntos de datos utilizados en la experimentación.

3.3. Métricas

A la hora de realizar la experimentación, la variable que determina la anomalía, será extraída para, a posteriori, obtener unas métricas que determinen cómo han desempeñado los algoritmos. Las métricas seleccionadas son *Precision* y *Recall*; la primera indicará cuantas anomalías han sido detectadas del total de anomalías, mientras que la segunda determinará cuantas anomalías "reales" hay de entre las que han sido detectadas. Cabe remarcar que si se conoce el porcentaje de anomalías, ambas métricas producirán el mismo resultado, por lo que en los casos en los que se utilice dicho porcentaje, solo se reflejará una métrica. Para calcular las métricas se realizarán 10 ejecuciones independientes, salvo si el algoritmo es determinista, en cuyo caso solo se calculará el tiempo en 10 ejecuciones distintas.

Es destacable el tiempo ya que será la métrica que determine si realmente se ha mejorado con respecto a otras implementaciones.

4. *Local Outlier Factor*

Local Outlier Factor (LOF) [27] es un algoritmo de detección de anomalías basado en proximidad, o para ser más precisos, en densidad. Este algoritmo surge por la necesidad de detectar anomalías que sean locales, es decir, que sean anomalías debido a su entorno más próximo, ya que, en la mayoría de las situaciones, en el conjunto de datos se dan diferentes grupos (o *clusters*), con diferente densidad; por ello, lo anómalo que sea un determinado patrón (según la densidad), será debido a la densidad de los patrones más cercanos.

Este algoritmo ha sido utilizado en multitud de entornos diferentes, como puede ser a la hora de detectar atascos puntuales [25] o para detectar fallos en procesos químicos complejos [29]. Además, para la comprensión de este algoritmo se han revisado los capítulos 1 y 4 de [11].

4.1. Definición

Local Outlier Factor necesita conocer el vecindario de cada uno de los patrones, basándose, en parte, en el algoritmo *K-Nearest Neighbors* (k vecinos más

próximos) [6]. De aquí es obtiene la distancia K de cada uno de los objetos o patrones. Sea $\mathbf{x}$ un patrón contenido en un conjunto de datos, D, de forma que $\mathbf{x}$ es un vector de características o variables.

Definition 1 (Distancia-k de un patrón). *Para cualquier entero positivo k, la distancia-k de un patrón, denotada como $D^k(\mathbf{x})$, esta definida como la distancia $dist(\mathbf{x}, \mathbf{y})$ entre $\mathbf{x}$ y un patrón $\mathbf{y} \in D$, de modo que:*

1. *para al menos k patrones $\mathbf{y}' \in D \setminus \{\mathbf{x}\}$ se mantiene que $dist(\mathbf{x}, \mathbf{y}') \leq dist(\mathbf{x}, \mathbf{y})$ y*
2. *para como mucho $k-1$ patrones $\mathbf{y}' \in D \setminus \{\mathbf{x}\}$ se mantiene que $dist(\mathbf{x}, \mathbf{y}') < dist(\mathbf{x}, \mathbf{y})$.*

Esto quiere decir, que se toma la distancia con el vecino número k con respecto a $\mathbf{x}$ en caso de que estuvieran ordenados por cercanía, y que puede darse el caso de que existan varios vecinos k, ya que pueden estar a la misma distancia. Esta definición de $D^k(\mathbf{x})$, lleva a otra más simple.

Definition 2 (Vecindario a distancia-k de un patrón). *Sea el conjunto de patrones cuyas distancias no son superiores a $D^k(\mathbf{x})$, de modo que $L_k(\mathbf{x}) = \{\mathbf{q} \in D \setminus \{\mathbf{x}\} | dist(\mathbf{x}, \mathbf{q}) \leq D^k(\mathbf{x})\}$. Estos patrones se denominan los k vecinos más cercanos de $\mathbf{x}$.*

Una vez definido el vecindario de $\mathbf{x}$, se usa una métrica que determina como de alcanzable es con respecto a cualquier patrón dentro de ese vecindario.

Definition 3 (Distancia de accesibilidad de un patrón x con respecto a otro, y). *Sea k un número natural. La distancia de accesibilidad de un patrón $\mathbf{x}$ con respecto a $\mathbf{y}$ se define como el máximo entre la distancia entre $\mathbf{x}$ y $\mathbf{y}$ y la distancia con el vecino k de $\mathbf{y}$:*

$$R_k(\mathbf{x}, \mathbf{y}) = max\{dist(\mathbf{x}, \mathbf{y}), D^k(\mathbf{y})\} \tag{1}$$

Esta distancia $R_k(\mathbf{x}, \mathbf{y})$, no es simétrica entre $\mathbf{x}$ y $\mathbf{y}$. Además, cuando $\mathbf{y}$ se encuentra en una región densa del espacio y la distancia entre ambos patrones es grande, la distancia de accesibilidad sera la distancia entre ellos, $dist(\mathbf{x}, \mathbf{y})$. Por otra parte, cuando la distancia entre ambos es pequeña, la distancia de accesibilidad se suaviza mediante $D^k(\mathbf{y})$, por ello, cuanto mayor es el valor k, mayor es el suavizado, provocando que las distancias de accesibilidad de D, sean más similares.

Esta métrica expuesta, solo determina una distancia entre patrones y no una métrica local. Por ello, se expone la siguiente definición.

Definition 4 (Distancia de accesibilidad media de un patrón). *Sea k un número natural que representa al número de vecinos. La distancia de accesibilidad media (AR_k, Average Reachability) para un patrón, será la media (M) de las distancias de accesibilidad de ese patrón con respecto a los patrones de su vecindario.*

$$AR_k(\mathbf{x}) = M_{\mathbf{y} \in L_k(\mathbf{x})} R_k(\mathbf{x}, \mathbf{y}) \tag{2}$$

Lo obtenido es una distancia media, para obtener la densidad solo habría que calcular el inverso, sin embargo, como se verá más adelante, se puede ahorrar ese calculo a fin de acelerar el proceso y obtener los mismos resultados.

Definition 5 (*Local Outlier Factor* de un patrón). *Sea k un número natural que representa al número de vecinos. Local Outlier Factor (LOF) será un valor que determinará lo anómalo que es un patrón, obtenido al valorar las distancias medias (M) de accesibilidad de la vecindad de un patrón con respecto a la distancia media de este.*

$$LOF_k(\mathbf{x}) = M_{\mathbf{y} \in L_k(\mathbf{x})} \frac{AR_k(\mathbf{x})}{AR_k(\mathbf{y})} \tag{3}$$

Como se puede observar de la definición, $LOF_k(\mathbf{x})$ valora cómo de accesible es ese patrón con respecto a como lo es su vecindario, es decir, adaptándose a la distribución o densidad local. Si es muy similar a esta densidad local, su valor se encontrará en torno a 1, por otro lado, si es mucho mayor que uno, es porque esta muy alejado de cómo se distribuyen localmente ese grupo de patrones y será considerado anomalía. Si se hubiera utilizado densidad, la fracción presente en la formula se hubiera visto invertida y al ser, a su vez, la densidad la invertida de la distancia, hubiera producido el mismo resultado.

4.2. Implementación y desarrollo

A la hora de implementar, se ha seguido como guía principal el artículo original [27], así como implementaciones previas funcionales como la implementación de *Scikit-Learn* [19].

El cálculo de LOF, una vez se tiene la accesibilidad media, es bastante simple, ya que consiste en realizar la media de la división de la accesibilidad media del patrón actual entre los patrones de su vecindario. El valor obtenido será normal si está en torno a uno o menos, y será anómalo si es superior. Los pasos a seguir para su obtención son los siguientes:

1. Obtener los k vecinos más cercanos.
2. Calcular la accesibilidad media.
3. Calcular *Local Outlier Factor*.

El primer paso (obtener los k vecinos más cercanos) es en el que radica la mayor complejidad computacional, ya que dentro de este se engloba el cálculo de las distancias entre todos los patrones del conjunto que es de orden cuadrático con respecto al número de patrones, la propia distancia que es de orden lineal con respecto al número de variables y la búsqueda de los vecinos más cercanos que dependerá del método empleado. Para mejorar la eficiencia se han realizado las siguientes mejoras:

- Se ha calculado una matriz de distancias debido a la simetría de la distancia, logrando que se reduzcan a la mitad los cálculos de las distancias.

- A la hora de encontrar a los vecinos más cercanos se han utilizado listas, ya que la inserción ordenada en estas es óptima.
- Se ha utilizado paralelización.

Además, se ha implementado una variante del algoritmo *KDTree* [14] a fin de reducir la complejidad del cálculo de los vecinos aún más, como se ve en [22]. Este algoritmo consiste en crear un árbol binario que divide el espacio en "cuadros delimitadores" o "cajas limitantes" (*bounding boxes*). En cada profundidad del árbol se escoge una variable del conjunto de datos, y se divide el espacio en dos mitades con el mismo número de patrones, seleccionando el patrón que es la mediana según esta variable y convirtiéndolo en el patrón separador. Ahora en cada rama, se tiene la mitad y se repite el procedimiento con la siguiente variable; este procedimiento es recursivo hasta que se cumple una condición de parada que es que en una rama haya un determinado número patrones o menos, convirtiendo esta rama en hoja. Las hojas de dicho árbol son los cuadros delimitadores.

En la metodología original, se buscaban a los vecinos en el cuadrante correspondiente y luego en el resto si la distancia con el margen era inferior a la distancia con el vecino k. Esta suele emplearse cuando el algoritmo de los vecinos más cercanos funciona como clasificador, por lo que se utiliza sobre patrones nuevos y que no tienen un lugar en el árbol. Sin embargo, en este caso los patrones a analizar se encuentran ya en este árbol, por lo que cada uno tiene asignado un cuadro delimitador. Por lo que se opta por una implementación aproximada que da muy buenos resultados en muy poco tiempo. Esta consiste en calcular los vecinos más cercanos de cada patrón dentro su respectivo cuadrante, ya que estos, pese a no certificar ser sus vecinos más próximos, si que serán, igualmente, muy próximos.

Para mejorar la eficiencia, en lugar de crear una estructura arbórea, se ha optado por implementar el árbol en un vector de longitud fija, el cual se ordena por secciones según los cuadrantes del nivel actual. Esto cuadrantes son conocidos a priori debido a que se conoce el número de patrones y el número de patrones por hoja, ya que en cada nivel habrá mitad en cada cuadrante con respecto al anterior. Gracias a la ordenación por cuadrantes, se puede lograr evitar la recursividad y paralelizar. La profundidad del árbol se conoce mediante la siguiente formula:

$$p = floor(log_2(\frac{N}{N_h})) \tag{4}$$

siendo N el número de patrones, N_h el número mínimo de patrones por hoja y p la profundidad del árbol, esto es así ya que si se divide el número de patrones entre el mínimo de hoja, da cuantos grupos han de hacerse, y el número de profundidad necesario en un árbol binario para lograr ese número de hojas, se obtiene con el log_2 de este número de grupos.

Un ejemplo de la división en cuadrantes es la Tabla 3, donde en la profundidad 1, se ordenarían de forma independiente las posiciones del vector (árbol) $0 - 30$ y $31 - 61$.

0	61	-	-	-	-	-	-	-
0	30	61	-	-	-	-	-	-
0	15	30	45	61	-	-	-	-
0	7	15	22	30	37	45	52	61

Tabla 3: Ejemplo de división en cuadrantes, $N = 61$, $N_h = 7$ para *KDTree*.

La implementación de *KDTree* realizada logra:

- Utilizar un vector en vez de una estructura arbórea y evitar la recursividad.
- Reducir el cálculo de las distancias a orden cuadrático respecto del tamaño de hoja del árbol, reduciendo también el número de comparaciones para encontrar los vecinos más cercanos.

A la implementación en la que se calculan las distancias y vecinos entre todos los patrones, se le denominará "Fuerza bruta", mientras que en la que se utiliza el árbol, se le denominará "KDTree".

En los siguientes pasos (cálculo de la accesibilidad media y de LOF) solo hay una característica remarcable y es que se utiliza paralelización.

4.3. Experimentos, fuerza bruta

Cabe decir que este algoritmo es determinista, por lo que para las pruebas de la *precision* o el *recall* solo será necesario realizar una iteración, aunque para los casos del tiempo se realizarán 10 mediciones independientes y se obtendrá la media de todas ellas.

DataSet	***Precision***	**Tiempo** (s)
Arrhythmia	0.4393	0.0126
Cardio	0.2159	0.0332
ForestCover	-	-
Ionosphere	0.7857	0.0033
Lympho	0.6666	0.0010
Mnist	0.2414	1.0604
Satellite	0.3747	0.4615
Shuttle	0.1264	54.110
Vowels	0.3600	0.0199

Tabla 4: Resultado fuerza bruta, K = 20, usando contaminación (LOF).

DataSet	***Precision***	***Recall***
Arrhythmia	0.2424	0.6153
Cardio	0.1477	0.3170
ForestCover	-	-
Ionosphere	0.1111	1.0000
Lympho	0.6666	0.5714
Mnist	0.1585	0.3117
Satellite	0.0776	0.4688
Shuttle	0.0131	0.5111
Vowels	0.4800	0.2857

Tabla 5: Resultado fuerza bruta, K = 20, sin usar contaminación (LOF).

En la Tabla 4, se utilizan para los experimentos 20 vecinos (K) y se usa la contaminación especifica de cada conjunto de datos. Lo primero que llama la atención es que *ForestCover* no tiene un tiempo asignado, esto es debido a que

el algoritmo se saturaba en cuanto a tiempo debido a la longitud del conjunto de datos (300 mil patrones). En el resto, los tiempos son bastantes buenos y se podrían considerar casi instantáneos, salvo en *Shuttle* (54 seg) que el tiempo se dispara teniendo 48 mil patrones únicamente frente, por ejemplo, a *Mnist* (1 seg) que tiene 7500 patrones y no se dispara prácticamente. En cuanto a la métrica de la *precision*, se puede observar que es muy variada y que depende mucho del conjunto de datos utilizado, es decir, de las distribuciones de estos. Resaltar *Ionosphere* que con una contaminación del 36 % consigue diferenciar muy bien las anomalías, probablemente debido a que estas se encuentran en partes muy diferentes del espacio con diferentes densidades, ya que *Satellite* también tiene en torno a un 30 % y consigue menos precisión, posiblemente debido a que las anomalías se encuentran en las mismas zonas y consiguen reducirse mutuamente el LOF. Otro caso interesante, es el de *Arrhythmia*, en el cual se tienen pocos patrones y muchas variables, y aún así consigue una buena precisión; este caso es interesante puesto que al aumentar el número de variables de un conjunto de datos, los datos se tienden a volver más "juntos" según la distancia. Por lo general se podría decir que desempeñan, la mayoría, bien en cuanto a tiempo y medianamente bien en cuanto a detección.

En el siguiente experimento, Tabla 5, se observa el funcionamiento para casos reales en los que se desconoce la contaminación. Por norma general, se observa que cuanta más baja es la *precision*, más alto es el *recall*, esto es debido a que a los primeros positivos detectados, son con mucha certeza anomalías. Aunque es cierto que como se observa depende mucho del conjunto de datos. El caso más representativo es el de *Ionosphere* que detecta un 11 % de las anomalías, sin detectar ningún falso positivo; mientras que otro caso interesante es el de *Vowels* que detecta un 50 % de las anomalías, reduciendo bastante el *recall*, lo que lleva a pensar que todos los valores de LOF de ese conjunto son muy similares y tiene una desviación típica pequeña. En general y viendo los resultados de la Tabla 4 donde se conocía la contaminación, se podría decir que el margen escogido, detecta con certeza las anomalías. En este caso el tiempo no se muestra debido a que es el mismo que para el apartado anterior.

DataSet	**Prec-Sk**	**Prec-ONN**	**T-Sk**	**T-ONN**	***Speed Up***
Arrhythmia	0.4242	**0.4394**	0.2435	**0.0123**	x 3.5365
Cardio	0.1705	**0.2159**	0.3356	**0.0339**	x 4.0020
Ionosphere	0.7618	**0.7857**	0.2379	**0.0028**	x 13.546
Lympho	**0.8333**	0.6666	0.2308	**0.0008**	x 38.574
Mnist	**0.2443**	0.2414	2.5728	**1.0658**	x 2.2263
Satellite	0.3698	**0.3747**	1.6884	**0.4562**	x 3.262
Shuttle	0.1107	**0.1264**	102.375	**50.3386**	x 2.0297
Vowels	0.3400	**0.3600**	0.2874	**0.0197**	x 4.4395

Tabla 6: Comparación de *precision* y tiempo (s), Scikit-Learn vs Implementación, Fuerza Bruta (LOF).

Todo lo implementado no solo tiene el objetivo de que funcione, sino también de que sea óptimo y rentable de utilizar. Para ello lo mejor es compararlo con algo ya existente, en este caso con la librería *Scikit-Learn* (será nombrado "Sk") frente a lo implementado en OpenNN (ONN). Se va a comparar tanto la precisión que se obtiene (se espera que sea similar), y el tiempo que determinará el rendimiento. Los parámetros a utilizar en ambos algoritmos son los mismos, $k = 20$ y fuerza bruta. Asimismo, se han utilizado todos los núcleos del procesador en ambos casos. Los resultados se encuentran en la Tabla 6.

La *precision* aparece para todos los conjuntos de datos, menos para *Forest-Cover* que se satura como ocurría antes. En general las precisiones son muy similares, siendo en la mayoría de los *dataset* un poco mejores y en dos un poco peores. Ya que las anomalías se distribuyen por el espacio de diferentes formas, y que este algoritmo ha detectar un tipo concreto, se espera que ambas implementaciones más o menos las mismas, como se puede suponer al alcanzar una precisión similar. En cuanto al tiempo, aparece una columna de *Speed Up* que no solo se ha obtenido mediante el cociente de ambos resultados, si no también restando 0.2 segundos a la columna de *Scikit-Learn*, debido al tiempo de preparación de *Python* y no al algoritmo en sí. Por norma general y sobre todo para los conjuntos de datos grandes como son *shuttle* o *mnist*, en los cuales interesa más una reducción del tiempo, se ha obtenido una mejora de más del doble.

4.4. Experimentos, *KDTree*

Este apartado se ha realizado para contrastar su correcto funcionamiento y su velocidad con respecto al anterior, y con respecto a *Scikit-Learn*

Resaltar que los parámetros que influyen en este caso son el número de vecinos k como el número mínimo de patrones por hoja.

En la Tabla 7, se está utilizando $K = 20$, $patrones/hoja = 150$ y la contaminación de cada conjunto de datos. Para empezar, *ForestCover* consigue resultados al fin, aunque no muy buenos (esto se verá más adelante que es debido al algoritmo y no a la implementación). Además, los tiempo se han reducido drásticamente para casi todos los conjuntos de datos con respecto a la Tabla 4, logrando, en concreto, que el tiempo de *Shuttle* se reduzca unas 100 veces con resultados similares. En algunos casos, sobre todo en los conjuntos de datos más pequeños, se ha reducido un poco más la *precision* debido a que los patrones están espacialmente más separados. Sin embargo, el *KDTree* implementado no está pensado para conjuntos de datos relativamente pequeños y aún consigue buenos resultados. En otros casos como *Mnist* o *Cardio* incluso consigue mejores resultados, y en otros el mismo. Todo esto certifica su buen funcionamiento. El caso a destacar es *Vowels* en el que se consigue un peor desempeño.

Como en el caso anterior, se espera que este algoritmo sea usado en casos reales, por lo que se tiene la Tabla 8 sin utilizar contaminación. En este caso se reduce un poco el *recall* con respecto a 5, aunque se debe a que la precisión es menor, por lo que lleva a pensar, que se detectan menos patrones en general. En algunos casos como es *mnist* incluso aumenta. Esta implementación, consigue

DataSet	***Precision***	**Tiempo** (s)
Arrhythmia	0.4090	0.0056
Cardio	0.3125	0.0120
ForestCover	0.0731	2.7268
Ionosphere	0.6587	0.0024
Lympho	0.6666	0.0009
Mnist	0.3071	0.1314
Satellite	0.3973	0.0774
Shuttle	0.1096	0.5365
Vowels	0.2000	0.0091

Tabla 7: Resultados KDTree, K = 20, patrones/hoja = 150, usando contaminación (LOF).

DataSet	***Precision***	***Recall***
Arrhythmia	0.24242	0.5925
Cardio	0.1477	0.2574
ForestCover	0.1339	0.0387
Ionosphere	0.1111	1.0000
Lympho	0.6666	0.5714
Mnist	0.1828	0.3526
Satellite	0.0844	0.6254
Shuttle	0.0236	0.1263
Vowels	0.2200	0.1692

Tabla 8: Resultado KDTree, K = 20, patrones/hoja = 200, sin usar contaminación (LOF).

acelerar muchísimo los tiempos, a costa de aumentar un poco los falsos positivos en ocasiones, o en otras incluso mejorar.

Como en el apartado anterior, se va a hacer una comparación contra *Scikit-Learn* en el cual se van a fijar los mismos parámetros, es decir, $k = 20$ y usar $KDTree$. Los resultados se encuentran en la Tabla 9.

En cuanto a las precisiones, se observa que la mayoría son similares mientras que hay alguna que varía en pos de la implementación y un par en las que la implementación es peor. La primera es *lympho*, en este solo hay 6 anomalías, por lo que la diferencia entre clasificar una o no, es abismal. La otra es *vowels* en la que la implementación se ha observado que desempeña un poco peor.

DataSet	**Prec-Sk**	**Prec-ONN**	**T-Sk**	**T-ONN**	***Speed Up***
Arrhythmia	**0.4242**	0.4090	0.2363	**0.0095**	x 3.8294
Cardio	0.1705	**0.2670**	0.2272	**0.0131**	x 2.0809
Cover	0.0484	**0.0731**	6.3625	**2.5833**	x 2.3855
Ionosphere	0.7618	**0.7857**	0.2308	**0.0032**	x 9.6250
Lympho	**0.8333**	0.6666	0.2279	**0.0009**	x 31.044
Mnist	0.2443	**0.2828**	3.4575	**0.0883**	x 36.891
Satellite	0.3698	**0.4057**	0.7506	**0.0501**	x 10.991
Shuttle	**0.1099**	0.1097	4.5337	**0.5102**	x 8.4942
Vowels	**0.3400**	0.2200	0.2245	**0.0121**	x 2.0280

Tabla 9: Comparación de *precision* y tiempo (s), Scikit-Learn vs Implementación, KDTree (LOF).

En cuanto a los tiempos, se puede observar una drástica mejora en todos los conjuntos y se puede concluir que la implementación realizada es al menos el doble de rápida en la mayoría de las situaciones, y sobre todo en las grandes como es *ForestCover* (*cover*) o *shuttle*, siendo la media 11.93. Concluyendo que, si las

precisiones no varían tanto y los tiempos se mejoran mucho, la implementación realizada desempeña mucho mejor.

5. *Isolation Forest*

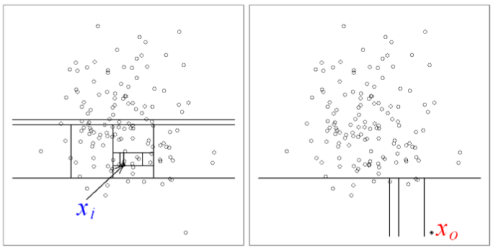


Figura 2: Ejemplo *Isolation Tree*

Isolation Forest o *IForest* [23], es un algoritmo de tipo *ensemble*, que emplea un procedimiento similar al de *RandomForest* [1], por ello, su base consiste en agrupar árboles simples, los cuales proveen información por sí mismos y crear una salida unificada de todos ellos. Ahora bien, la diferencia radica en que estos árboles (*Isolation Trees*), a diferencia de los de *RandomForest*, determinarán cómo de fácil es aislar un patrón. Por consiguiente, la idea básica de la que se parte, es que cuanto más anómalo sea un patrón más fácil será aislarlo del resto.

El aislamiento se logra particionando el espacio. Esto se puede ver reflejado en la Figura 2 (obtenido del artículo original [23]), donde el patrón x_0 es más anómalo que el patrón x_1, por lo que conlleva menos particiones o divisiones del espacio aislarlo, que a x_1 que se encuentra dentro del grupo.

5.1. Definición

El concepto de *Isolation Forest* (IForest) [23] surge del mismo punto que *LOF*, de la necesidad de clasificar instancias o patrones que no son clasificados mediante los métodos clásicos, los cuales trataban de adaptarse a distribuciones y aquellos patrones que no estuvieran dentro de estas, eran considerados anomalías. Además, se necesitaba un algoritmo que desempeñará rápido y consumiera poca memoria, por ello se tomaron las ideas de *Random Forest*:

- Construir un conjunto de árboles independientes y entrenados cada uno con una parte del conjunto de datos extraída por *Bootstrapping*.
- Cada árbol aporta las misma medida de calidad de los patrones, pero al ser entrenados con diferentes subconjuntos producen diferentes resultados.

Las ideas propias de este algoritmo son que existe una minoría de instancias, cuyos valores de sus atributos son diferentes a los de las instancias normales. Por ello, son más susceptibles a ser aisladas, lo que significa son más fácilmente separables del resto. Este aislamiento se logra mediante divisiones sucesivas del espacio como se comentaba para la Figura 2, en el que se demuestra que cuanto más aislado este un determinado patrón menos divisiones serán necesarias, siendo lo contrario también cierto.

Las divisiones o particiones se pueden lograr de forma recursiva usando una estructura de árbol en el cual se utiliza en cada nivel un atributo mediante el cual se divide el espacio. Según los patrones que haya en un determinado nodo, se busca el máximo y el mínimo del atributo elegido, para seleccionar un valor de forma aleatoria entre ese rango y dividir a los patrones en dos partes. Esto se reproduce hasta que cada patrón se encuentre aislado en el árbol, de forma que el camino seguido hasta encontrar un patrón corresponderá con el número de divisiones espaciales necesarias para aislarlo; cuanto menor sea este número más anómalo será un patrón. Este árbol se denominaría *Isolation Tree*.

Definition 6 (*Isolation Tree*).
Sea T un nodo del árbol. T es, o un nodo hoja sin hijos y test, o un nodo interno que posee dos hijos (T_l, T_r) y test. Un test consiste en un atributo q y un valor de división p de forma que el test es $q < p$ divide los datos en T_l y T_r.

De la definición dada se puede determinar que el árbol creado es un árbol binario, teniendo cada nodo de este 0 o 2 hijos. Dado un conjunto de datos $D = \{\bar{X}_1, ..., \bar{X}_n\}$, recursivamente se crea el árbol dividiendo D, seleccionando de forma aleatoria un atributo q y un valor de división p hasta que :

1. Se llega a la altura máxima.
2. Solo queda una patrón en una hoja. $|D| = 1$
3. Todos los patrones restantes de D en ese nodo, tienen los mismos valores.

Un árbol normal solo aplicaría la restricción 2, que sería el caso más óptimo ya que estaría balanceado y permitiría búsquedas más rápidas, y con el cual se tendrían $2n - 1$ nodos. Esto quiere decir que la memoria crece linealmente. Sin embargo, debido a que en ocasiones la aleatoriedad o la selección de patrones se hace en una zona densa, es conveniente escoger una profundidad límite que suele ser $log_2(N)$ siendo N el número de patrones utilizados para crear el árbol, ya que el árbol se desequilibra y el camino resultante no es representativo. Remarcar que si un patrón esta muy aislado del resto por un determinado atributo, al escoger un valor aleatorio entre el mínimo y el máximo, este patrón con una mayor probabilidad quedará aislado más prontamente del resto.

Definition 7 (Longitud del camino).
$h(\bar{X})$ de un punto $\bar{X}$ es la medida del número de aristas que $\bar{X}$ ha de atravesar en un $ITree$ desde el nodo raíz hasta que llegue a un nodo hoja.

Esta $h(\bar{X})$ permitirá determinar cómo de anómalo es ese patrón según ese determinado árbol. Sin embargo, en ocasiones como las restricciones 1 y 3 expuestas sobre la creación del árbol, la longitud del camino no es totalmente

representativa en esos casos. Por ello, según [26] la profundidad para un árbol construido con un conjunto de datos de N elementos es:

$$c(N) = log(N-1) - \frac{2(N-1)}{N} + 0{,}5772 \tag{5}$$

En el supuesto caso que un patrón llegará a un nodo en el que $|D| \neq 1$, se sumaría a la profundidad actual de ese nodo, la profundidad media $c(N)$ siendo N en este caso el número de patrones que no se dividieron originalmente. Con esto se tendría una aproximación de la distancia, además de que los patrones más interesantes son los que menos profundidad alcanzan, por lo que utilizar esto no empeora la eficacia en la detección. Además, tener una altura límite mejora la eficiencia.

La creación del bosque parte de crear cada árbol a partir de un subconjunto creado mediante *bagging* [3], el cual realiza una selección aleatoria tanto de patrones como de características para cada subconjunto, para más tarde unir los resultados producidos por cada subconjunto en único resultado. Esta metodología mejora la estabilidad y reduce la varianza y el sobre ajuste. El tamaño de este subconjunto es uno de los dos parámetros del algoritmo, siendo el otro parámetro el número de árboles o subconjuntos. Por tanto, la altura máxima de los árboles vendrá dada por el $log_2(N)$ siendo N el tamaño de subconjunto especificado.

A la hora de evaluar, cada patrón recorre su camino por cada árbol formado hasta encontrar su longitud de camino; la media de las longitudes de un patrón por cada uno de los árboles será el valor de anomalía, cuanto menor sea, más fácil será de aislar y más "diferente" o anómalo será.

5.2. Implementación y desarrollo

A la hora de implementar, se ha seguido como guía principal el artículo original [23]. También se ha utilizado la implementación de [18], sin embargo, está solo ha servido para asentar las ideas del funcionamiento ya que hacía uso de clases anteriormente implementadas como la de árbol.

Los pasos para realizar para este algoritmo son:

1. Crear un número de subconjuntos mediante *Bagging* igual al número de árboles a crear.
2. Crear *Isolation Trees* a partir de los subconjuntos.
3. Evaluar cada patrón a través de cada árbol y obtener la media.

En el primer paso, no se introduce ninguna modificación, pero sí en el segundo y en el tercero, ya que se pretende implementar el *Isolation Tree* en un vector de longitud fija, en lugar de en una estructura arbórea y evitar la recursividad, optimizando así el tiempo. Esto se logra gracias a que la profundidad o altura de los árboles es fija y a que estos son árboles binarios. La longitud máxima para un vector que contiene un árbol binario de profundidad p es:

$$Long = 2^{p+1} - 1 \tag{6}$$

Para manejar un árbol binario en un vector, los hijos se acceden mediante la posición actual multiplicada 2 más 1, si es el hijo el izquierdo y más dos si es el derecho.

Para comprenderlo mejor se va a utilizar la Figura 3 como ejemplo donde hay un árbol binario ordenado. La profundidad es de 3, y por tanto el tamaño máximo es 15. El vector que se forma es el que aparece en la Tabla 10.

Posición	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenido	5	2	6	1	3	-	-	-	-	-	4	-	-	-	-

Tabla 10: Ejemplo de vector para un árbol binario.

Si se analiza por ejemplo el nodo 3 que se encuentra en la posición 4, sus hijos posibles serían $2*4+1=9$ y $2*4+2=10$, de los cuales el único existente es el de la posición 10 que es un 4.

Se evita la recursividad, para lograr aún más eficiencia, mediante la utilización de una cola estilo FIFO, en la que se sacan los nodos (siendo estos índices del vector árbol) de la cabeza sucesivamente y se dividen en dos hijos mediante la selección aleatoria de una variable entre el máximo y mínimo de esta. Si un hijo tiene un único elemento se considera hoja, en caso de tener más, se inserta dicho nodo al final de la cola. Esto provoca que se cree un árbol en anchura.

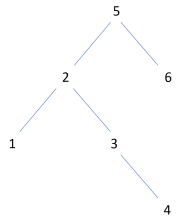


Figura 3: Ejemplo de árbol binario simple.

Es destacable que el proceso de creación de cada árbol es independiente, así como el posterior proceso de evaluación, de forma que ambos procedimiento se pueden paralelizar.

5.3. Experimentos

A diferencia de *LOF*, este algoritmo es estocástico y cada ejecución produce resultados sensiblemente diferentes, por lo que para medir correctamente las métricas, se harán 10 mediciones independientes para cada conjunto de datos.

Los experimentos han sido realizados usando en todos los casos un número de árboles de 100 y un tamaño de subconjunto de 256.

Para el caso en el que se utiliza contaminación, Tabla 11, los tiempos son realmente favorables, reduciendo aún más los que se conseguían con *KDTree*, además, se consiguen resultados bastante buenos en el promedio en cuanto a *precision*, mejorando en muchos casos a este método, siendo el caso más destacable el de *shuttle* que pasa de un 10 %, a un 90 % con *Isolation Forest*. Este conjunto de datos es relativamente grande y podría pensarse, entonces, que funciona muy bien con conjunto grandes, lo cual es en parte cierto como se expone en la teoría, empero, en el conjunto de datos de *ForestCover*, consigue peores resultados.

DataSet	***Precision***	**Tiempo** (s)
Arrhythmia	0.4621	0.0454
Cardio	0.5136	0.0503
Cover	0.0513	0.7815
Ionosphere	0.7039	0.0428
Lympho	0.8166	0.0422
Mnist	0.2695	0.0533
Satellite	0.5887	0.0634
Shuttle	0.9043	0.2006
Vowels	0.1900	0.0484

Tabla 11: Resultado árboles = 100, tamaño subconjunto = 256, usando contaminación (IForest)

DataSet	***Precision***	***Recall***
arrhythmia	0.2242	0.6016
cardio	0.2840	0.5784
cover	0.0982	0.0349
ionosphere	0.0301	1.0000
lympho	0.9333	0.6996
mnist	0.1320	0.2944
satellite	0.1586	0.9170
shuttle	0.8526	0.9426
vowels	0.2060	0.2066

Tabla 12: Resultado árboles = 100, tamaño subconjunto = 256, sin usar contaminación (IForest)

DataSet	**Prec-Sk**	**Prec-ONN**	**T-Sk**	**T-ONN**	***Speed Up***
Arrhythmia	**0.4894**	0.4621	0.4529	**0.0454**	x 5.5711
Cardio	**0.52272**	0.5136	0.4099	**0.0503**	x 4.1739
Cover	**0.0882**	0.0513	16.986	**0.7815**	x 21.480
Ionosphere	0.6627	**0.7039**	0.3389	**0.0428**	x 3.2455
Lympho	**0.8666**	0.8166	0.3131	**0.0422**	x 2.6800
Mnist	**0.3117**	0.2695	1.7198	**0.0533**	x 28.515
Satellite	0.5737	**0.5887**	0.8287	**0.0634**	x 9.9170
Shuttle	**0.9544**	0.9043	3.1688	**0.2006**	x 14.799
Vowels	0.1880	**0.1900**	0.3824	**0.0484**	x 3.7702

Tabla 13: Comparación de *precision* y tiempo (s), Scikit-Learn vs Implementación, Isolation Forest.

El otro caso de interés es cómo desempeña este algoritmo en casos reales donde no existe contaminación, Figura 12. Consigue determinar como anomalías bastantes patrones y además con un *recall* bastante alto, que es un factor fuerte, ya que, como se comentó anteriormente, esto ayuda a eliminar patrones que son

certeza anomalías, y no eliminar otros datos que podrían ser de interés en el aprendizaje.

La comparación en *precision* con *Scikit-Learn* se puede observar en la Tabla 13. Esta es necesaria dado que de nada sirve que el algoritmo sea rápido, sino alcanza un buen rendimiento a la hora de la detección. Los resultados son muy parejos para ambas implementaciones, siendo un poco mejores de promedio en *Scikit-Learn*, y en algunos algo peor este. Esto indica que el algoritmo funciona como debería.

Aquí (Tabla 13) también se le ha restado 0.2 a los tiempos de *Scikit-Learn* a la hora de calcular el aumento. Los tiempos por lo general se mejoran en con un factor muy alto, siendo la mejor más baja de 2.6, mientras que la media de 10. Incluso los conjuntos de datos muy grandes como *ForestCover* y *Shuttle* se ven acelerados considerablemente. Se podría decir que en términos de aceleración, se han conseguido unos resultados muy buenos.

Se podría concluir que la implementación realizada mejora cuantiosamente en eficiencia y se mantiene en cuanto a *precision*.

6. Conclusiones y líneas futuras

6.1. Conclusiones

- Se han diseñado e implementado dos algoritmos de detección de anomalías, ***Local Outlier Factor*** y ***Isolation Forest*** que **mejoran drásticamente** el tiempo de ejecución con sus respectivas versiones en ***Scikit-Learn***.
- Aportaciones sobre *Local Outlier Factor*:
  - Se ha conseguido realizar una implementación óptima mediante el uso de listas en el cálculo de los vecinos más cercanos.
  - El método *KDTree* para simplificar la búsqueda de los vecinos ha reducido la complejidad computacional, logrando exitosamente evitar la recursividad y la creación de estructuras auxiliares, lo cual ha mejorado aún más los tiempos.
- Aportaciones sobre *Isolation Forest*:
  - Los árboles (diferentes a los de *KDTree*) han sido creados con éxito en un vector, optimizando la búsqueda del camino más cercano.
  - Se ha evitado el uso de la recursividad mediante la utilización de una cola.

6.2. Líneas futuras

- Implementar más tipos de algoritmos en la librería OpenNN, como algoritmos basados en distribuciones estadísticas o probabilidad.
- Modificar la implementación para que puedan ser usadas operaciones vectoriales de librerías de cálculo de GPU como CUDA.

Referencias

1. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324.
2. B. R. Preiss.Data Structures and Algorithms with Object-Oriented Design Patterns in Java. Wiley, 1999.
3. Bootstrap aggregating (bagging) | wikipedia. `https://en.wikipedia.org/wiki/Bootstrap_aggregating`. Último acceso: 20-05-2021.
4. The easy to use, online, collaborative latex editor. `https://www.overleaf.com/`. Último acceso: 20-05-2021.
5. Eigen is a c++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. `https://eigen.tuxfamily.org/index.php?title=Main_Page`. Último acceso: 20-05-2021.
6. k vecinos más próximos - wikipedia, la enciclopedia libre. `https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos`. Último acceso: 20-05-2021.
7. Novelty and outlier detection, scikit-learn. `https://scikit-learn.org/stable/modules/outlier_detection.html`. Último acceso: 20-05-2021.
8. Odds provide access to a large collection of outlier detection datasets with ground truth. `http://odds.cs.stonybrook.edu/`. Último acceso: 20-05-2021.
9. Redes neuronales desde cero (i) - introducción - iartificial.net. `https://www.iartificial.net/redes-neuronales-desde-cero-i-introduccion/`. Último acceso: 20-05-2021.
10. Artificial intelligence company | artelnics. `https://www.artelnics.com/`, 2021. Último acceso: 20-05-2021.
11. Charu C. Aggarwal. *Outlier Analysis*. Springer New York Heidelberg Dordrecht London, 2016.
12. Anava. Preprocessed version of ip-network-traffic-flows-labeled-with-87-apps. `https://www.kaggle.com/anavan/ipdata`. Último acceso: 20-05-2021.
13. LLC Axosoft. Free git gui for windows, mac and linux | gitkraken. `https://www.gitkraken.com/`. Último acceso: 20-05-2021.
14. Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Comunications of the ACM*, 18(9):509–517, 1975.
15. Xianshun Chen. Package implements a number local outlier factor algorithms for outlier detection and finding anomalous data. `https://github.com/chen0040/java-local-outlier-factor`. Último acceso: 20-05-2021.
16. Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11:1–21, 1969.
17. Qt Project; Qt Development Frameworks. Qt creator - a cross-platform ide for application development. `https://www.qt.io/product/development-tools`. Último acceso: 20-05-2021.
18. Nicolas Goix; Alexandre Gramfort. Implementación de iforest, scikit-learn - código fuente. `https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/ensemble/_iforest.py#L27`. Último acceso: 20-05-2021.
19. Nicolas Goix; Alexandre Gramfort. Implementación de lof, scikit-learn - código fuente. `https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/neighbors/_lof.py#L19`. Último acceso: 20-05-2021.
20. Longbing Cao Guansong Pang, Chunhua Shen and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Comput. Surv. 1*, Article 1, 2020.
21. Damjan Kužnar. Python implementation of local outlier factor algorithm. `https://github.com/damjankuznar/pylof`. Último acceso: 20-05-2021.

22. Wenfeng Hou; Daiwei Li; Chao Xu; Haiqing Zhang; Tianrui Li. An advanced k nearest neighbor classification algorithm based on kd-tree. *IEEE International Conference of Safety Produce Informatization*, pages 902–905, 2018.
23. Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.
24. P. Martin, C. Barranquero, J. Sanchez, G. Gestoso, J.A. Andres, D. Refoyo, C. Garcia, A. Fernadez, and R. Lopez. Opennn: Open neural network library. `https://www.opennn.net/`, 2021. Último acceso: 20-05-2021.
25. Henry Y.T. Ngan Mathew X. Ma and Wei Liu. Density-based outlier detection by local outlier factor on large-scale traffic data. *Society for Imaging Science and Technology*, 2016.
26. Joaquín Amat Rodrigo. Detección de anomalías con isolation forest y python. `https://www.cienciadedatos.net/documentos/py22-deteccion-anomalias-isolation-forest-python.html`, 2020. Último acceso: 20-05-2021.
27. Markus M. Breunig; Hans-Peter Kriegel; Raymond T. Ng; Jörg Sander. Lof: Identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, page 93–104, 2000.
28. sanjal katiyar. Binary tree (array implementation). `https://www.geeksforgeeks.org/binary-tree-array-implementation/`. Último acceso: 20-05-2021.
29. Hehe Ma; Yi Hu; Hongbo Shi. Fault detection and identification based on the neighborhood standardized local outlier factor method. *Industrial and Engineering Chemical Research*, 52:2389–2402, 2013.
30. Paul Ullrich and Colin Zarzycki. Tempestextremes: A framework for scale-insensitive pointwise feature tracking on unstructured grids. *Geoscientific Model Development*, 10:1069–1090, 03 2017.
31. Arindam Banerjee Varun Chandola and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, pages 1–72, 2009.
32. Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

Interacción gestual: desde los comienzos hasta nuestros días

Nuria Tovar Suárez, José Rafael García-Bermejo Giner

Departamento de Informática y Automática
Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
ntovar@usal.es
https://www.usal.es/

Resumen Este artículo presenta un estudio sobre la evolución de las máquinas con el tiempo, prestando especial atención a la energía que se debe aportar para realizar la interacción, además de la cantidad de información que el dispositivo puede procesar y/o generar. A lo largo del siglo XX los ordenadores sufren una continua mejora y surgen nuevos métodos de interacción, como es el caso de las Interfaces Naturales de Usuario (NUI), que facilitan la forma en que nos comunicamos con los ordenadores y hacen su manejo más simple e intuitivo.

Keywords: IPM, IPO, energía, información, GUI, CLI, NUI

1. Introducción

En este artículo se ha prestado especial atención la evolución de la interacción gestual a lo largo de los tiempos. La energía que el usuario debe aportar a las máquinas va decreciendo en todo momento. Asimismo, se puede observar un incremento en la cantidad de información que una máquina es capaz de procesar y aportar.

Se analizará la evolución de los ordenadores, así como sus métodos de interacción. Se prestará especial atención a la interacción gestual, una disciplina que estudia la forma en la que se emplean gestos que el ordenador sea capaz de reconocer, mediante los cuales ofrece una realimentación o respuesta al usuario. Los gestos que un sistema puede ser capaz de detectar son muy amplios, como la interacción oral, reconocimiento facial, interfaces gestuales, etc.

2. Máquinas calculadoras

Los primeros mecanismos utilizados para realizar cálculos tienen miles de años de antigüedad, como es el caso del ábaco, un instrumento que ha sido utilizado durante miles de años para realizar operaciones matemáticas sencillas con números naturales. La interacción con esta máquina se basa en mover cada

una de las cuentas para la representación de números y de esta forma facilitar la realización de operaciones aritméticas sencillas de forma visual. Aunque esta máquina sea de gran utilidad, cabe destacar que el usuario debe tener conocimientos previos del sistema de numeración, además de tener que realizar un esfuerzo mental para poder realizar las operaciones.

Otra máquina calculadora que tuvo especial relevancia fue la Pascalina, inventada en el año 1642 por Blaise Pascal. Este mecanismo funcionaba a base de ruedas y engranajes que permitían el cálculo de sumas y restas con una mayor rapidez y seguridad. Cada rueda estaba formada por diez pasos numerados del 0 al 9. Había un total de 8 ruedas, dos para los decimales y seis para los enteros, por lo que esta máquina tenía la capacidad de manejar números entre el 0,01 y el 999.999,99 [4]. La interacción con esta máquina se realiza mediante un punzón para hacer girar las ruedas dentadas. En este caso el usuario debe aportar energía a la máquina para que pueda realizar los cálculos que se introduzcan. Este invento ya tiene la capacidad de almacenar información, además de que facilita en gran medida la tarea que se realiza.

Como se puede ver, el trasfondo histórico de la interacción gestual es muy amplio. Los mecanismos de interacción con máquinas son muy numerosos, aunque para este artículo se han nombrado únicamente dos de las máquinas calculadoras más importantes. Como se ha observado, es el usuario el que aporta toda la energía necesaria para realizar la interacción con la máquina. Además, la información que éstas aportan y/o procesan es muy limitada.

3. Evolución de los ordenadores

La interacción entre personas y máquinas se ha producido durante miles de años y tiene un largo recorrido. Se han estudiado una serie de dispositivos de la antigüedad que se podrían considerar como los predecesores de los ordenadores, aunque posteriormente aparecería una máquina que revolucionaría aún más la computación: la máquina analítica de Babbage, un diseño del año 1837 que se convertiría en el más claro precursor del hardware computacional. La idea de la máquina de Babbage permaneció olvidada durante un siglo hasta el surgimiento de los primeros ordenadores, que usarían componentes electro-mecánico para su funcionamiento. A partir de ese momento, la evolución de los ordenadores se vería ligada en gran medida a los avances que se producían en la electrónica.

3.1. Máquina analítica de Babbage

La máquina analítica diseñada por Charles Babbage es considerada como el primer ordenador del mundo y el precursor de los ordenadores actuales, logrando un gran avance tanto conceptual como mecánico en la historia de la informática. El mecanismo debía hacer uso de un motor de vapor junto con una manivela para poder funcionar. Su forma de interacción se basó en el uso tarjetas perforadas para introducir instrucciones y datos. Se programaba mediante un lenguaje similar al lenguaje ensamblador. Su memoria consistía en 1000 columnas de 50

ruedas cada una, dándole una capacidad de almacenamiento de 1000 números de hasta 50 cifras cada uno [6].

Esta máquina no llegaría a ser construida debido a que la tecnología de esa época no estaba lo suficientemente avanzada. A pesar de ello su diseño se considera un hito en la historia de la informática, convirtiéndose en la primera máquina capaz de modificar su funcionamiento interno según las instrucciones que se le introduzcan.

3.2. Primera generación (1940-1955)

La primera generación de ordenadores se desarrolló desde 1940 hasta 1955, y la componen aquellos sistemas informáticos que hacían uso de estructuras de tubos de vacío que permitían conmutar la señal eléctrica y tambores magnéticos para la memoria [2]. Esta primera generación se basó en la arquitectura de la máquina de Babbage diseñada un siglo antes.

Los ordenadores construidos durante este periodo utilizaban una gran cantidad de energía eléctrica. Debido al coste económico que suponían tanto su fabricación como su uso, los diseños solían ser únicos y eran utilizados exclusivamente por las fuerzas militares y la industria científica.

La interacción con estas máquinas era compleja, ya que se debían de tener conocimientos sólidos de informática para poder programarlas en lenguaje de máquina a través de tarjetas perforadas.

Dentro de esta generación destacan máquinas como el Z1, construido en el año 1938, que sería el primer ordenador electromecánico programable. Funcionaba con electricidad, aunque era necesario girar una manivela para su funcionamiento. Estaba diseñado para realizar las cuatro operaciones aritméticas básicas. Su consumo era de 1 kW y podía realizar operaciones de hasta 22 bits.

El Harvard Mark I, construido en 1944, también tuvo especial relevancia. Permitía realizar las cuatro operaciones aritméticas básicas, además de tener la capacidad de hacer referencia a resultados anteriores. Poseía 72 registros mecánicos para números, cada uno con la capacidad de almacenar hasta 23 dígitos. El consumo de esta máquina era de aproximadamente 4kW.

El ENIAC fue construido en el año 1946 con el objetivo de realizar cálculos sobre las trayectorias de los proyectiles lanzados por el ejército de los Estados Unidos. Se trata de uno de los ordenadores más grandes construidos en la historia. Su consumo era de 160 kW. Su memoria tenía la capacidad de almacenar a 20 números de 10 dígitos.

El UNIVAC I, fabricado en el año 1951, tenía la capacidad de almacenar hasta 1000 secuencias en su memoria de mercurio. Cada elemento de la memoria tenía la capacidad de contener dos tipos de datos, un número de 11 cifras y signos o bien 12 caracteres alfabéticos. Su consumo era de 125 kW.

3.3. Segunda generación (1955-1964)

La segunda generación de ordenadores está comprendida aproximadamente entre los años 1956 y 1964. El cambio de generación lo marcaría la sustitución

de los tubos de vacío por transistores, ya que eran capaces de realizar la misma función pero con una mayor velocidad, reduciendo tanto el consumo de energía como el tamaño del aparato. Además, se mejoró la memoria mediante el uso de discos magnéticos como dispositivos de almacenamiento secundarios.

Durante este periodo se empezaría a usar el lenguaje ensamblador y se desarrollarían lenguajes de más alto nivel, como FORTRAN o COBOL. Esto dio a los ordenadores una mayor facilidad de uso, ya que el lenguaje ensamblador se asemeja un poco más al lenguaje natural. La interacción con la máquina se seguía realizando mediante tarjetas perforadas, aunque se produjo un gran avance respecto a la primera generación con la creación de estos nuevos lenguajes de alto nivel.

Durante esta generación la empresa IBM tuvo una gran relevancia. Lanzaría dos de los modelos más conocidos de esta generación: el IBM 1401 y el IBM 1620. Destaca especialmente el primero de estos modelos, que llegaría a tener un gran éxito con un total de 12000 unidades vendidas. Poseía una memoria de núcleo magnético de 4000 caracteres, que posteriormente se ampliaría a 16000. Su consumo energético era de 13kW.

3.4. Tercera generación (1964-1971)

La tercera generación de ordenadores se comprende entre los años 1964 y 1971. Esta generación surge tras la invención y el desarrollo del circuito integrado, consiguiendo diseños más pequeños, rápidos, con menos disipación de calor y energéticamente más eficientes.

Se comenzó a utilizar un software más complejo. Se seguirían desarrollando nuevos lenguajes de programación con una nueva sintaxis y más fáciles de comprender. Gracias al surgimiento de los sistemas operativos el manejo de ordenadores se hizo más sencillo. Además, se dio paso a la multiprogramación, una técnica mediante la cual se puede ejecutar dos o más procesos simultáneamente.

Por primera vez, los ordenadores se hicieron accesibles para la audiencia masiva, ya que eran más pequeños y baratos que sus antecesores, además de tener un menor consumo energético. Destaca el ordenador IBM 360, el primer modelo de computadora que hizo uso de circuitos integrados, dando comienzo a esta generación. Su capacidad de memoria era de 62 kB y su consumo rondaba los 200 W.

Durante esta generación se produce un gran avance en la interacción persona-maquina. Se diseñaron nuevos periféricos y se investigaron nuevas tecnologías. A pesar de la evolución en los dispositivos de entrada/salida, no comenzarían a producirse ordenadores que hagan uso de estos dispositivos hasta la cuarta generación. La gran mayoría de estos sistemas seguiría utilizando las tarjetas perforadas como método de interacción.

3.5. Cuarta generación (1971-1981)

El inicio de esta cuarta generación lo marca la introducción del primer microprocesador en el año 1971 y duraría hasta el año 1981. Este componente

electrónico permite agrupar todos los elementos del procesador en un solo circuito integrado, reduciendo considerablemente el tamaño de los ordenadores. Además, se produjo una mejora en la capacidad de la memoria debido al uso de chips de silicio, que reemplazaron a los núcleos magnéticos que se utilizaban hasta el momento como tecnología de almacenamiento.

Los primeros ordenadores de esta generación surgen a mediados de los años 70. Incorporan pantalla y teclado como dispositivos principales de interacción con la máquina, aunque algunos de ellos ya comenzaron a utilizar el ratón. Los sistemas operativos evolucionan, se consiguen gráficos con mayores velocidades y se implementa por primera vez una Interfaz Gráfica de Usuario (GUI).

Esta época se caracteriza por un considerable aumento en el uso de ordenadores personal. Además, los nuevos métodos de interacción hicieron posible que cada vez más personas fueran capaces de utilizar un ordenador. Durante esta generación surgió la empresa Apple, fundada en 1976 por Steve Jobs y Steve Wozniak. Su primer modelo, el Apple I, fue lanzado en 1976 y se convertiría el primer ordenador en combinar el uso de un microprocesador junto con teclado y ratón. También destaca el ordenador IBM PC (Personal Computer), lanzado en 1981 al límite de la quinta generación, que poseía una memoria de 16KB ampliable a 256KB [1]. Su consumo era de 63 W, un valor bastante bajo comparado con el consumo de los ordenadores actuales, que ronda sobre los 200-300W.

3.6. Quinta generación (1981-presente)

La quinta generación comienza en 1981 con el proyecto *Fifth Generation Computer Systems* de Japón, cuyo objetivo era crear ordenadores basados en inteligencia artificial. Este proyecto no se concluyó de forma satisfactoria, aunque marcaría el comienzo de esta época.

Durante esta etapa se trabaja en gran medida en el software, de forma que los ordenadores se vuelvan más versátiles y fáciles de usar parara los usuarios. Estos dispositivos pasan a tener características muy superiores a las generaciones anteriores, con un gran aumento en la velocidad y en la cantidad de memoria. Se comienza a trabajar con técnicas de inteligencia artificial, como la robótica, el procesamiento del lenguaje natural, el reconocimiento de voz y los sistemas expertos entre otros. En cuanto al hardware, se produce una mejora continua en la integración de circuitos, la cual permite introducir más transistores en un mismo espacio.

Gracias a los avances en el software aparecen nuevas formas de interacción, como las Interfaces de Voz de Usuario (VUI) o las Interfaces Naturales de Usuario (NUI), aunque la forma de interacción más común es haciendo uso de una Interfaz Gráfica de Usuario (GUI) mediante el teclado y el ratón. Con la aparición de los ordenadores portátiles en las últimas décadas se hace muy común también el uso del touchpad, una almohadilla táctil que reemplaza el ratón en estos dispositivos.

4. Evolución de las herramientas de interacción

Como se ha visto anteriormente, los primeros ordenadores se basaron en el sistema de tarjetas perforadas utilizado en el siglo XIX por Charles Babbage para el diseño de su máquina analítica. Las instrucciones se grababan en una tarjeta perforada en lenguaje máquina, un lenguaje binario que hace uso de unos y ceros (agujero y ausencia de agujero). Este método se mantuvo hasta los años 70, cuando comenzó a popularizarse el uso del ordenador y surgió la necesidad de crear herramientas de interacción más simples e intuitivas.

En los años 70 surgen los primeros ordenadores que hacen uso de pantalla y teclado, lo cual revolucionaría la forma de comunicarse con estas máquinas. Estaban dotados de sistema operativo y comenzarían a aparecer las primeras interfaces de usuario. El método de interacción más común con estos sistemas era mediante Interfaces de Líneas de Comandos (CLI), un estilo de interacción que hace uso exclusivamente del teclado para introducir instrucciones.

En el año 1968 Douglas Engelbart presentó la primera patente del ratón, creando una herramienta capaz de controlar las cosas que ocurrían en pantalla y realizar tareas que no se podrían hacer únicamente con el uso del teclado. Al principio este periférico no tuvo la aceptación esperada, y tendrían que pasar más de 10 años para que comenzara a tener éxito. El primer ordenador en hacer uso del ratón como dispositivo de entrada sería el Xerox Alto, lanzado en el año 1973, que destacaría también por ser el primero en incluir una interfaz gráfica muy rudimentaria. A pesar de ello el uso del ratón no se popularizó hasta la aparición del Macintosh en 1984. Las GUI comenzarían a ganar importancia gracias a la facilidad de interacción obtenida con el uso del ratón.

Uno de los periféricos más usados en la actualidad es la pantalla táctil, cuyos orígenes se atribuyen al británico E.A. Johnson tras el diseño del primer prototipo de una pantalla capacitiva, publicado en el año 1965. A principios de los años 70 los ingenieros del CERN Bent Stumpe y Frank Beck desarrollaron la primera pantalla táctil capacitiva totalmente transparente. Esta tecnología se ha ido mejorando a lo largo de los años incorporando más sensibilidad, más precisión y detectando varios puntos de contacto de forma simultánea. Las pantallas táctiles capacitivas son las que están presente hoy en día está presente en la gran mayoría de dispositivos táctiles (teléfonos móviles, tablets, etc.).

Una herramienta que revolucionaría la interacción en los ordenadores portátiles sería el *touchpad*, desarrollado por George E. Gerpheide en el año 1988. Cuando los ordenadores portátiles salieron al mercado en los años 80 la interacción se realizaba mediante una bola (algo como un ratón invertido) y dos botones. Esta solución no era del todo ergonómica y la bola no era muy precisa, por lo que era necesario encontrar otro solución. De esta forma surgiría el *touchpad*, que sería básicamente una fusión del ratón con la tecnología táctil, y se convertiría en un sustituto del ratón en ordenadores portátiles. La compañía Apple decidió comprar la licencia de Gerpheide y el *touchpad* aparecería por primera vez en el Macintosh Powerbook 190, en el año 1995.

A lo largo de este último siglo han ido surgiendo nuevos dispositivos que han permitido al ordenador adaptarse a su continua evolución. En sus comienzos los

ordenadores eran máquinas de grandes dimensiones diseñadas para una única tarea y debían ser operadas por expertos. Con el paso de los años cada vez se crearon ordenadores más rápidos, pequeños, baratos y con más funcionalidades, lo cual atraía a un público mayor. Se crearon nuevas herramientas de interacción, de forma que se facilitó en gran medida el uso del ordenador, haciendo posible el desarrollo y proliferación de los ordenadores personales. En la actualidad los dispositivos de interacción más usados para la entrada de datos al ordenador son el ratón, el teclado y el touchpad, aunque las pantallas táctiles han ganado importancia en las últimas décadas con el incremento en el uso de teléfonos móviles y tabletas.

5. Interfaces de Usuario (UI)

Las Interfaces de Usuario (UI) se definen como el medio utilizado para realizar la interacción entre un usuario y una máquina. Esto comprende tanto las herramientas hardware (ratón, teclado, etc.) como el software utilizado. Actualmente el paradigma de las interfaces de usuario se centra en crear nuevas formas de comunicación con el ordenador lo más fáciles y cómodas posible, capaces de adaptarse al mayor número de usuarios y asemejándose al máximo al lenguaje natural.

5.1. Interfaz de Línea de Comandos (CLI)

Antes de que se hiciera popular el uso del ratón como periférico de entrada los ordenadores únicamente contaban con pantalla y teclado. Las CLI es un método que surgió con la necesidad de crear una forma de interacción adaptada al hardware disponible en ese momento. En la pantalla únicamente se muestran caracteres ASCII y las instrucciones se introducen mediante el teclado. Se trata del tipo de interfaz más antiguo que permanece en uso, aunque en la actualidad las GUI han sustituido en gran medida a este método de interacción.

5.2. Interfaces de Texto de Usuario (TUI)

Las TUI son una evolución de las CLI, y se trata de una forma de interacción que únicamente muestra caracteres ASCII en la pantalla y la navegación se realiza mediante el teclado. Se podría decir que las TUI son un punto intermedio entre las CLI y las GUI. En la actualidad se sigue utilizando esta forma de interacción en algunas aplicaciones concretas, como en cargadores boot, para la configuración de la BIOS o para la instalación de sistemas operativos.

5.3. Interfaces Gráficas de Usuario (GUI)

Con el incremento en el uso de ordenadores personales en la década de los 70 surgiría la necesidad de crear métodos de interacción más amigables e intuitivos para así poder atraer a un mayor número de usuarios no experimentados, ya

que para el uso de las CLI era necesario tener unos conocimientos previos sobre las instrucciones que acepta el terminal o consola. La llegada del ratón como dispositivo periférico de entrada revolucionaría tanto los ordenadores como sus métodos de interacción, y daría paso a las primeras Interfaces Gráficas de Usuario (GUI), un método de interacción visual que posee una estética mejorada y cuyo objetivo es simplificar la comunicación entre el hombre y la máquina, a costa de un mayor coste computacional.

Estas interfaces se basan principalmente en el uso de imágenes y otros elementos visuales que se muestran en la pantalla del ordenador. Estos elementos se controlan mediante el uso del ratón o el touchpad, que se encargan de desplazar el puntero por la pantalla y así poder interactuar con los distintos elementos de la interfaz, como por ejemplo hacer clic en un icono para ejecutar un programa. En menor medida también se utiliza el teclado para realizar atajos mediante combinaciones de teclas.

6. Interacción gestual

La interacción gestual es una disciplina que forma parte de la IPO y estudia el uso de gestos para interaccionar con los dispositivos electrónicos. Se pueden considerar como gestos todos los movimientos corporales que puedan afectar al estado de una máquina. Actualmente el paradigma de la interacción gestual se centra en el estudio de nuevas formas de interacción lo más naturales posible y sin el uso de cables

Hoy en día las Interfaces Naturales de Usuario están ganando una especial importancia dentro de la interacción gestual, ya que dan la posibilidad de controlar las máquinas a través de comportamientos naturales e intuitivos. Dentro de este tipo de interfaces existen formas de interacción muy variadas, entre las que destacan el lenguaje oral, reconocimiento de gestos, tecnología multitáctil, etc.

6.1. Tecnología multitáctil

Una de las primeras tecnologías basadas en los principios de Interfaces Naturales de Usuario es la multitáctil, que consiste en una superficie transparente capaz de detectar la presencia de uno o más puntos de contacto, haciendo uso en conjunto de una Interfaz Gráfica de Usuario para poder visualizar las interacciones con el sistema en tiempo real. De esta forma los usuarios son capaces de interactuar directamente con los objetos de la pantalla mediante el tacto, haciendo uso de movimientos intuitivos que hoy en día ya se han convertido en un estándar de interacción en teléfonos móviles, tablets e incluso en ordenadores.

6.2. Reconocimiento de voz

El reconocimiento de voz es la capacidad que tiene una máquina o un software para reconocer palabras y frases, transformarlas a un lenguaje entendible por la

máquina y realizar las acciones que el usuario demande. Los avances en estas últimas décadas en inteligencia artificial y en concreto en el procesamiento y generación del lenguaje natural han propiciado el desarrollo de sistemas expertos que hacen uso de interfaces inteligentes para lograr una comunicación efectiva con la máquina utilizando únicamente el lenguaje oral. Esta técnica ha ganado especial importancia en los últimos años y se está convirtiendo en una nueva forma de interaccionar con nuestros dispositivos que se encuentra en pleno auge. Las Interfaces de Voz de Usuario (VUI) pueden utilizarse de dos formas: como un método de interacción complementario a los ya existentes, como es el caso de los asistentes de voz en teléfonos móviles, tablets u ordenadores; o bien como método único de interacción con la máquina, como es el caso de los altavoces inteligentes.

6.3. Reconocimiento de gestos

El gesto es una forma de comunicación no verbal en la que se realizan una serie de movimientos corporales con el objetivo de transmitir un mensaje concreto utilizando distintas partes del cuerpo, como la cara o las manos. En los últimos años la interacción basada en gestos se ha convertido en un enfoque muy prometedor dentro de las Interfaces Naturales de Usuario, dando una manera más natural e intuitiva para comunicarnos con las máquinas en una amplia variedad de aplicaciones, pudiendo controlar los dispositivos únicamente haciendo gestos con nuestro cuerpo.

En cuanto a la tecnología utilizada para el reconocimiento de gestos, existen dos tipos: aquella que funciona con un dispositivo externo que actúa como extensión del cuerpo (no óptica) y aquella que hace uso de una combinación de cámaras, sensores y técnicas de visión artificial (óptica). El uso de dispositivos externos es más costoso, puede resultar incómodo para la persona que se lo equipa y puede llegar a limitar sus movimientos, aunque da lugar a una mayor precisión. En cuanto a los métodos ópticos, se trata de una tecnología más barata, aunque su precisión puede llegar a variar en función de la luz ambiental o la falta de contraste entre la vestimenta del sujeto y el fondo.

Los primeros métodos para reconocer el movimiento del cuerpo estaban basados en dispositivos externos provistos de distintos sensores electromagnéticos (giroscopios, acelerómetros y magnetómetros) capaces de obtener la posición y la orientación en la que se encuentran, y de esta forma reconocer el gesto que se realice. Dentro de esta categoría destacan dispositivos como los trajes o los guantes inteligentes, capaces de registrar cualquier movimiento en tiempo real. Otro popular ejemplo es la tecnología de la consola Nintendo Wii (2006), que hace uso de un periférico externo con la capacidad de detectar el movimiento en el espacio y de señalar objetos en la pantalla. Años después surgirían las primeras gafas de realidad virtual, capaces de detectar la posición y orientación de la cabeza en todo momento y de esta forma realizar una inmersión más realista. Todos estos dispositivos son artefactos caros y molestos de usar. Hoy en día las investigaciones se están centrando en desarrollar interfaces gestuales que

no hagan uso de ningún tipo de equipamiento intrusivo, convirtiendo el cuerpo humano en el propio método de interacción con las máquinas

Los sistemas ópticos de captura de movimiento hacen uso de una o más cámaras cuyas imágenes son procesadas para reconocer gestos del usuario. Es común el uso de sensores de profundidad en combinación con cámaras para dar una mejor precisión al sistema. Dentro de estos sistemas destaca VideoPlace [5], un dispositivo diseñado en 1991 que permite al usuario interaccionar con objetos virtuales mediante gestos en tiempo real. El dispositivo Kinect, lanzado en el año 2010, ha tomado especial relevancia en este ámbito, ya que permite el control de un dispositivo mediante una interfaz natural de usuario que no solo reconoce gestos, sino también comandos de voz, objetos e imágenes. También destaca el dispositivo Leap Motion (2012), un dispositivo de características similares a Kinect pero con una capacidad de detección de movimientos 100 veces más exacta. Este periférico tiene la capacidad de conectarse al ordenador mediante el puerto USB y reconocer gestos de las manos y los dedos en el espacio. Otra tecnología interesante es WiSee, que hace uso de la tecnología WiFi para la detección y reconocimiento de gestos humanos en todo el hogar, lo que permite al usuario realizar gestos en cualquier dirección y sin el uso de cámaras. También son relevantes las tecnologías de seguimiento ocular, capaz de detectar a qué parte de la pantalla se está mirando y de esta forma dar mucha más facilidad y rapidez a la hora de interaccionar con las máquinas, ya que la comunicación se realizaría únicamente con la mirada.

En general, los sistemas ópticos son más naturales de usar que las interfaces que hacen uso de dispositivos externos, y son capaces de realizar un buen seguimiento de la mano y el cuerpo, pero no proporcionan la misma precisión en la determinación de la postura. Sin embargo, para muchas aplicaciones esto puede no ser importante y en un futuro veremos esta tecnología incluida en muchos de los dispositivos electrónicos que usamos a diario.

7. Métodos de interacción futuros

Se han estudiado las tecnologías de interacción más usadas hasta la actualidad, entre las que destacan las Interfaces Naturales de Usuario. Dentro de este tipo de interfaces las más populares son aquellas que hace uso de gestos sobre pantallas multitáctiles. Sin embargo, a raíz del COVID-19 la higiene pública se ha convertido en un foco de atención, dando la necesidad de crear métodos interactivos que no precisen de contacto físico con el dispositivo para generar una mayor seguridad en el usuario. Una solución a este problema son los hologramas táctiles, que consisten en una proyección de una imagen en 2D o 3D mediante la cual se puede interactuar con la máquina sin la necesidad de tocarla físicamente. Su aplicación sería interesante en hospitales, donde hay un riesgo mayor de contacto con bacterias, aunque también sería útil en otros lugares públicos como bancos, aeropuertos, restaurantes o museos. Aunque de momento solo se han conseguido crear hologramas sencillos, los resultados son bastante novedosos y podría llegar a convertirse en un gran avance en un futuro.

Otra tecnología que podría llegar a revolucionar en gran medida la interacción con las máquinas son las Interfaces Cerebro-Ordenador (BCI). En la última década se han llevado a cabo numerosas investigaciones sobre esta nueva tecnología, y todas ellas tienen características comunes: medir actividad cerebral con sensores, procesar de la señal adquirida para obtener sus características de interés e interaccionar con el dispositivo de la forma deseada por el usuario. Estas interfaces permiten la comunicación mediante un canal alternativo que conecta directamente el sistema nervioso central con dispositivos electrónicos, dando la posibilidad de controlar una máquina solo con el pensamiento. Un ejemplo exitoso de BCI es la compañía Neuralink, que en el año 2020 presentó por primera vez un prototipo de un chip que debe ser insertado en la corteza cerebral para poder realizar una conexión entre el cerebro y el ordenador.

En definitiva, los métodos de interacción futuros buscarán una experiencia que sea lo más natural posible, manos libres y personalizada. Esto está comenzando en la actualidad con las interfaces de voz, controles por gestos y seguimiento de mirada. Próximamente la interacción con gran parte de las máquinas se realizará sin contacto y en un futuro probablemente se podrán ver salir a la luz las tecnologías descritas en este apartado, aunque es muy difícil anticiparse a la aceptación que tendrán, ya que influyen muchos factores. Cabe destacar que las interfaces futuras serán multimodales, es decir, se emplearán distintos canales de comunicación de forma simultánea para poder interactuar con las máquinas.

8. Conclusiones

En el presente trabajo se ha realizado un estudio sobre la evolución de la interacción gestual desde sus orígenes hasta hoy. Para ello se han analizado una serie de mecanismos junto con sus formas de interacción. A raíz de las características de algunos de los modelos investigados, se ha realizado un estudio (Apéndice A) donde se muestra la evolución de los ordenadores en función de una serie de características.

Las máquinas y en específico los ordenadores han sufrido una grandes cambios a lo largo de este último siglo. Sus características técnicas han mejorado significativamente y se han creado métodos de interacción cada vez más simples de usar y adecuados para la mayoría de usuarios. Con el paso de los años, los ordenadores cada vez consumían menos potencia, pero realizaban más tareas y de forma más rápida, además de tener mayor memoria y capacidad de producir información. Además, la energía que debía aportar el usuario ha ido disminuyendo hasta la llegada del ratón y el teclado, el paradigma más común de interacción en la actualidad. En los últimos años con el surgimiento de las NUI la cantidad de energía que debe aportar el usuario continuará decreciendo. En un futuro, incluso esta interacción se podría realizar solo con el pensamiento. Es difícil imaginar el potencial que tiene la IPO, pero está claro que llegarán nuevos métodos de interacción más simples e intuitivos que sustituirán a los ya existentes, como ha ocurrido a lo largo de este último siglo.

Referencias

1. 4ª generación de computadoras: los primeros ordenadores personales. URL: *https://agenciab12.com/noticia/4-generacion-computadoras-primeros-ordenadores-personales*, 2021. (Consultado el 11-06-2021).
2. Juan Gomar. Las generaciones de ordenadores. URL: *https://www.profesionalreview.com/2018/10/13/generaciones-de-ordenadores/*, 2018. (Consultado el 08-06-2021).
3. Thomas Hewett, Ronald Baecker, Stuart Card, Jean Gasen, Marilyn Tremaine, Gary Perlman, Gary Strong, and William Verplank. *Curricula for Human-Computer Interaction*. 1992.
4. Alexander Iribar Ibabe. La Pascalina. URL: *http://paginaspersonales.deusto.es/airibar/Ed_ digital/INF/Intro/Pascalina.html*, 2008. (Consultado el 06-06-2021).
5. Myron W Krueger. Artificial reality II. 1991.
6. Ariel Palazzesi. La Máquina Analítica de Babbage. URL: *https://www.neoteo.com/la-maquina-analitica-de-babbage/*, 2018. (Consultado el 08-06-2021).
7. Ivan E Sutherland. Sketchpad, a man-machine graphical communication system. *Simulation*, 2(5):R–3, 1964.

Appendice A: Estudio realizado

El objetivo principal de este trabajo es mostrar el desarrollo de la interacción gestual desde sus comienzos hasta nuestros días. Para ello se han estudiado una serie de máquinas que han destacado en la historia, y dentro de ese grupo se ha seleccionado una muestra con el objetivo de realizar un análisis sobre la cantidad de energía que aporta el ser humano para realizar la interacción, la cantidad de energía que consume la máquina, la cantidad de información que podemos aportar a la máquina y la información que es capaz de aportarnos la máquina a nosotros. Gracias a ello se podrá ver de una forma más clara la evolución de estos dispositivos. Todas las máquinas seleccionadas tienen una característica común: han sido utilizadas como herramientas para realizar cálculos.

8.1. Energía consumida por la máquina

En esta primera parte del estudio se pretende analizar el consumo energético de las máquinas con el paso del tiempo. En sus comienzos las máquinas no precisaban de ninguna fuente de energía externa, sino que se trataban de dispositivos mecánicos donde el usuario aportaba toda la energía. Es por ello que en esta parte del estudio solo se tienen en cuenta las máquinas a partir del año 1837, cuando se diseñó la Máquina de Babbage. Los primeros ordenadores construidos tenían un consumo de unos pocos de miles de vatios (W). Con la llegada del ENIAC y el UNIVAC I este consumo se disparó hasta llegar a sobrepasar los 100 kW. Desde entonces la energía consumida ha ido decreciendo, llegando a su punto más bajo con el IBM PC y el Macintosh, que tenían un consumo inferior que los ordenadores actuales.

Figura 1: Energía consumida por la máquina a lo largo de los años.

Con los datos recogidos se ha obtenido la gráfica de la Figura 1. Como se puede observar, esta curva se asemeja en gran medida a una campana de Gauss cuyo pico está en el año 1946, cuando se puso en funcionamiento el ordenador ENIAC. La energía utilizada por los ordenadores crece bruscamente a lo largo de la primera generación, ya que se utilizaban tubos de vacío cuyo consumo energético era bastante superior a los componentes que se utilizarían posteriormente. El momento más bajo se encuentra a lo largo de la cuarta generación, ya que el microprocesador había logrado reducir en gran medida el consumo, y al no utilizarse muchos recursos su consumo era inferior que el de los ordenadores modernos.

Si eliminamos el ENIAC y el UNIVAC I de este estudio, ya que son ordenadores que se salieron de la norma con un consumo sorprendentemente elevado, los resultados seguirían siendo similares, obteniendo de nuevo una curva de Gauss con un punto máximo esta vez en el año 1959, con un consumo de 13 kW (Figura 2).

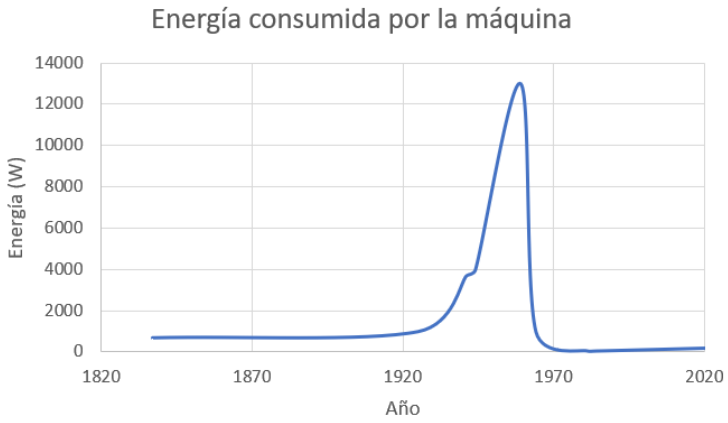


Figura 2: Energía consumida por la máquina a lo largo de los años.

Gracias a estas gráficas se puede observar cómo los componentes electrónicos utilizados en los ordenadores han afectado en gran medida en su consumo. En la actualidad la energía que utilizan los ordenadores es bastante baja en comparación con la gran variedad de tareas que son capaces de realizar.

8.2. Energía aportada por el usuario

En esta parte del estudio se pretende hacer una estimación de la cantidad de energía en vatios que debe aportar el usuario para realizar una tarea con la máquina. En el caso del ábaco, la energía que aporta el usuario es de unos pocos vatios, ya que únicamente debe mover una serie de cuentas de posición.

En cuanto a la pascalina, ésta requiere un mayor esfuerzo, ya que se deben mover las ruedas dentadas que posee mediante un punzón, aunque realizando una estimación la energía que se aporta en este caso también sería de unos pocos vatios. Con la llegada de la máquina de Babbage esto cambiaría, ya que esta máquina necesitaba que se girara una manivela para ponerla en funcionamiento, además de que se debían perforar una serie de tarjetas para introducir datos. El ordenador Z1 seguía el mismo principio que la máquina analítica, pero utilizaba dos manivelas en vez de una, por lo que el esfuerzo aportado era mayor. Los siguientes ordenadores hacían uso de corriente eléctrica para su funcionamiento, y el usuario únicamente debía perforar las tarjetas para introducir los datos o programas. Esto permanecería así hasta la llegada de los ordenadores personales en la década de los 70. Mediante estos ordenadores solo era necesario teclear la instrucción que se desea ejecutar, por lo que la energía que se aporta a esta máquina sería muy baja. Con la llegada del ratón y de las interfaces gráficas de usuario esta energía disminuiría aún más, llegando a obtener hasta GB de información únicamente con un clic del ratón. Además, con la llegada de las NUI el esfuerzo que debe realizar el ser humano se reduce considerablemente, pudiendo interactuar con la máquina sin realizar apenas esfuerzo.

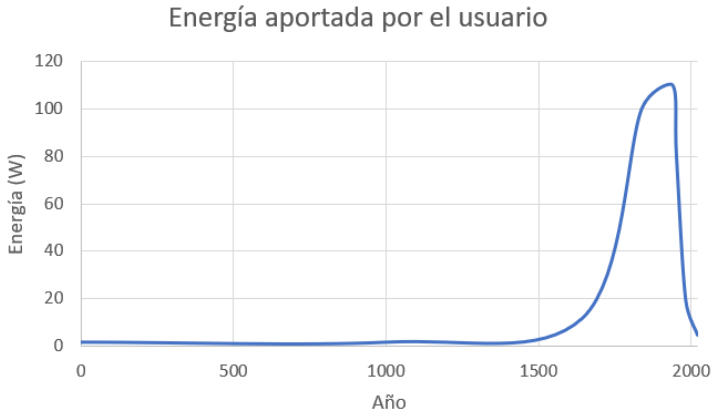


Figura 3: Energía aportada por el usuario a lo largo de los años.

En la Figura 3 se puede observar la evolución de la energía que emplea el usuario para interactuar con la máquina. De nuevo se obtiene una curva de Gauss, cuyo punto máximo se encuentra en el año 1939 con el ordenador Z1, que era el que mayor esfuerzo precisaba por parte del usuario para poder funcionar. Se puede ver que desde la máquina de Babbage en 1837 la energía que debe aportar el usuario aumenta en gran medida, hasta que sobre la década de los 80 disminuye drásticamente.

Sin embargo, en esta curva no se puede observar con claridad lo que ocurre entre los años 1837 y la actualidad, por lo que se ha realizado otra gráfica centrada en ese periodo, que se muestra en la Figura 4.

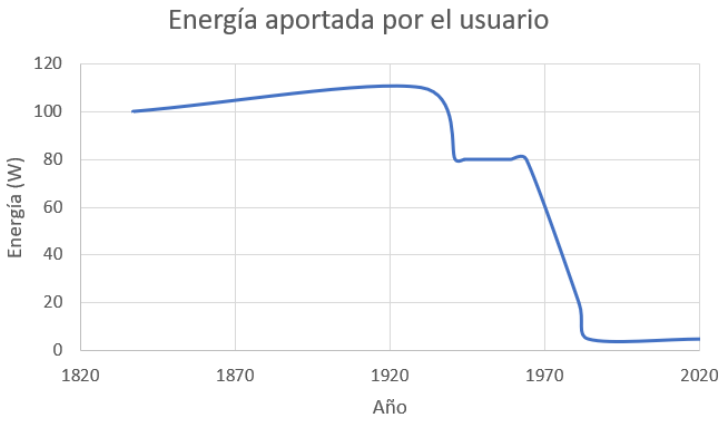


Figura 4: Energía aportada por el usuario desde 1837.

En esta segunda gráfica se puede ver un incremento desde la energía que se necesitaba aportar a la máquina de Babbage, que utilizaba una sola manivela y tarjetas perforadas, hasta la energía que utilizaba el ordenador Z1, que utilizaba un conjunto de dos manivelas, además de las tarjetas perforadas para introducir datos. En los siguientes modelos se cambiaron las manivelas por energía eléctrica, por lo que el usuario únicamente debía perforar e introducir las tarjetas con los datos e instrucciones. Este método permanecería así hasta aproximadamente los años 70, donde se comenzó a utilizar el teclado como periférico de entrada. Además, con la llegada del ratón, la energía que debe aportar el usuario a la máquina llegó a un valor mínimo, el cual se ha mantenido hasta la actualidad.

8.3. Información producida por la máquina

Como se ha visto a lo largo de este trabajo, en los comienzos de la computación las máquinas estaban diseñadas para realizar una cantidad limitada de tareas. En un principio las funciones de un ordenador eran matemáticas. Con la llegada del UNIVAC I en el año 1951 se pudo ver que los ordenadores no solo tenían la capacidad de procesar números, sino también palabras. Esto abrió un mundo de nuevas posibilidades a estos dispositivos. Desde entonces, se irían añadiendo distintas funcionalidades y de esta forma la información que podían aportar estos dispositivos incrementó drásticamente.

Para realizar una estimación de la cantidad de información que puede producir una máquina se ha utilizado la cantidad de memoria RAM que poseen

los distintos dispositivos que se han estudiado. La memoria RAM se trata de la memoria principal de un dispositivo, donde se almacenan de forma temporal los datos de los programas que están siendo utilizados. Aunque en los comienzos de la computación y hasta los años 70 no existía la memoria de acceso aleatorio, se han utilizado los datos de capacidad de memoria. Cabe destacar que los valores que se han seleccionado corresponden a la cantidad máxima de información que podrían aportar estas máquinas. Se ha decidido eliminar la máquina de Babbage de esta sección, ya que no llegó a ser construida e introduciría ruido en los datos.

La cantidad de información que pueden producir las máquinas en la actualidad es enorme comparada con las primeras generaciones, por lo que se ha estudiado su evolución desde la década de los 40 hasta mediados de los 80 con el objetivo de poder apreciar bien los cambios producidos en este periodo de tiempo. Desde la aparición del Macintosh en 1984 hasta la actualidad se ha producido un incremento muy brusco en las capacidades de las máquinas, haciendo difícil su representación en una gráfica.

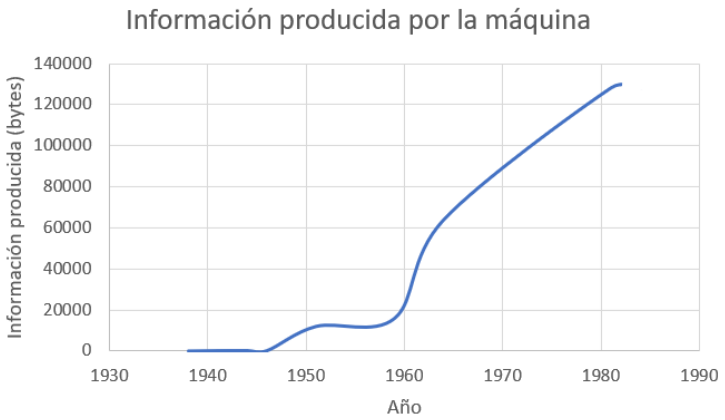


Figura 5: Información producida por el ordenador a lo largo del siglo XX.

En la Figura 5 se puede observar cómo incrementa la información que es capaz de producir una máquina desde la primera generación hasta los comienzos de la quinta. Como conclusión, En sus comienzos los ordenadores han ido progresando de forma más o menos lineal, hasta que a partir de la década de los 80 tuvieron un desarrollo casi exponencial, llegando a producir en la actualidad hasta 8 GB.

8.4. Información aportada por el usuario

En los comienzos de la computación la cantidad de información que podían aportar las máquinas era bastante limitada. En los primeros casos únicamente se introducían un par de números para realizar operaciones matemáticas básicas,

hasta que la capacidad de almacenamiento de los ordenadores fue mejorando e incluso empezaron a poder almacenar caracteres y palabras. Con el tiempo la capacidad de almacenamiento ha incrementado de manera radical. Cabe mencionar que en esta sección se ha utilizado en todo momento los valores máximos de cantidad de información que se puede aportar a la máquina, y la estimación se ha realizado en función de la capacidad de almacenamiento en memoria de los ordenadores.

En sus comienzos la cantidad de información que se podía aportar a una máquina era despreciable comparada con la capacidad de los ordenadores actuales. Hasta la aparición del Macintosh en 1984 no puede observarse ningún cambio significativo. Es por ello que se ha realizado una gráfica en el periodo de tiempo entre 1938 y 1984, con el objetivo de observar la evolución de los ordenadores a lo largo de las primeras generaciones de ordenadores, hasta la llegada del microprocesador

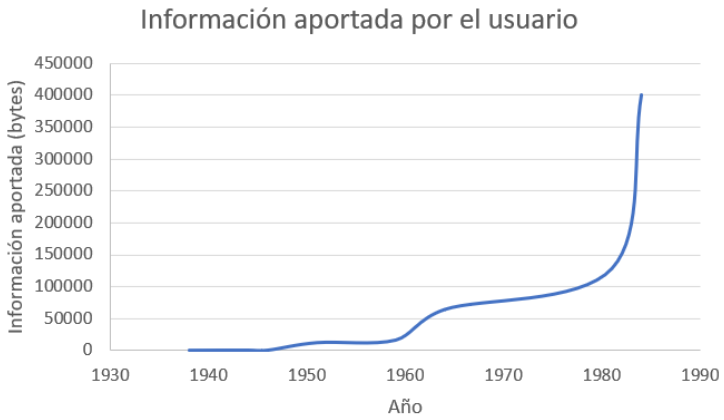


Figura 6: Información aportada por el usuario.

En la gráfica de la Figura 6 se puede observar que la cantidad de información aportada por el usuario incrementa ligeramente tras las primeras generaciones hasta llegar a la quinta generación. A partir de ese momento se produce una evolución exponencial en la cantidad de información que puede aportar el usuario.

Sistemas MAS para el control cooperativo de sistemas complejos basados en la teoría de juegos y redes neuronales

Oscar Emilio Aponte, Pastora Vega, Mario Francisco, Belén Pérez y Roberto Casado

Departamento de Informática y Automática
Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
{idu17344,pvega,mfs,lancho,rober}@usal.es
https://www.usal.es/

Resumen En este trabajo se propone una solución novedosa del uso de redes neuronales con aprendizaje por refuerzo, como una opción válida en la negociación de agentes de controladores jerárquicos distribuidos. EL método propuesto, se implementa en la capa superior de una arquitectura de control jerárquico constituido en sus niveles más bajos por un control distribuido basado en modelos locales y procesos de negociación con lógica difusa. La ventaja de la propuesta es que no requiere el uso de modelos en la negociación y facilita la minimización de cualquier índice de comportamiento dinámico y la especificación de restricciones. Concretamente, en este trabajo, se utiliza un algoritmo de gradiente de política determinista profunda del aprendizaje por refuerzo (DDPG) para el entrenamiento del proceso de consenso entre los agentes. El algoritmo resultante se aplica con éxito a un sistema de nivel compuesto por ocho tanques interconectados muy difícil de controlar por su naturaleza no lineal y la alta interacción existente entre sus subsistemas. Se plantean dos soluciones del aprendizaje por refuerzo con características distintas siendo solo una de ellas la que arroja resultados positivos.

Keywords: Redes neuronales artificiales, Aprendizaje por refuerzo, Lógica borrosa, Control Predictivo Distribuido (DMPC), Sistemas multiagentes.

1. Introducción

El momento actual, debido a la demanda de la sociedad y de la industria, la aparición de plantas industriales complejas ha crecido drásticamente por la mejora de la eficiencia económica que ofrecen. Más allá de las motivaciones económicas, la transición a modelos de producción más sostenibles implica aumentar la complejidad de los sistemas de producción, que ha conducido al desarrollo de sistemas a gran escala. Por otro lado, los grandes avances tecnológicos hacen que

el área de control de procesos se vea en la posición de implementar soluciones mejores y más sofisticadas en aquellos casos en los que la teoría de control convencional se ve limitada, como en el caso del control de sistemas complejos.

El control de estos sistemas requiere, generalmente, el uso de controladores avanzados multivariables para cumplir con los estándares de calidad y seguridad establecidos en nuestra legislación. Es bien sabido que los esquemas descentralizados y centralizados son los dos marcos de control principales utilizados en la industria. Sin embargo, aunque la implementación de un esquema descentralizado es fácil, las interacciones entre las unidades de proceso pueden conducir a la degradación del desempeño de control de toda la planta o pérdidas de estabilidad de sistema en lazo cerrado. Un esquema centralizado podría lograr el mejor rendimiento de control en toda la planta, pero la complejidad para su implementación aumenta a medida que aumenta el tamaño del problema de control y puede requerir un esfuerzo de mantenimiento sustancial que no es favorable desde el punto de vista de la tolerancia a fallos, entre otras cosas. Para abordar los problemas anteriores, han surgido las técnicas avanzadas de control distribuido que constituye una buena solución práctica, siendo el enfoque del presente proyecto. Los esquemas de supervisión y control de toda la planta basados en teorías de juegos e inteligencia artificial son enfoques atractivos para una solución óptima de este complejo problema siempre que se integren adecuadamente con las técnicas clásicas.

Por otro lado, las actuales tecnologías de la información y la comunicación permiten incrementar el tamaño y la complejidad de las aplicaciones de control con garantías prácticas e implementar metodologías de control avanzadas. Dentro de las técnicas más extendidas a nivel práctico se encuentra el control predictivo basado en modelos (MPC) siendo este tipo de metodologías de control una de las beneficiarias más visibles de estos avances. En consecuencia, el número de aplicaciones industriales de MPC ha aumentado mucho en las últimas décadas. Cuando los requisitos computacionales y de comunicación involucrados en un problema de control MPC centralizado son inviables desde un punto de vista práctico, una posible solución es utilizar MPC distribuidos (DMPC) para dividir el problema en varios más simples. Entre ellos los basados en sistemas multiagente y negociación son una alternativa útil que ha sido la empleada en este trabajo.

El DMPC se apoya en una coordinación que tiene un precio en términos de gastos generales de comunicación. En [22] algunos métodos DMPC requieren que cada agente intercambie miles e incluso millones de datos tipo float en cada instante para lograr la solución óptima. Un enfoque diferente para abordar la carga excesiva de comunicación es la lógica difusa empleada para la negociación de los agentes. En [25] se propone una arquitectura de control DMPC jerárquica de dos capas que utiliza lógica difusa en el proceso de negociación, en la primera capa los agentes negocian por parejas y, por lo general, cuantos más agentes se involucran en una negociación más pasos de comunicación deben darse para lograr un acuerdo. La tarea crítica para resolver el problema del consenso en

sistemas multiagentes es diseñar protocolos de control distribuido basados en la información de cada agente y sus vecinos, con el fin de lograr un acuerdo conjunto sobre un interés común. Este es el caso de [32], en el que se propone un enfoque DMPC para el problema de consenso óptimo cuadrático lineal de sistemas de múltiples agentes en tiempo discreto.

La teoría de juegos ha sido pionera en el campo de la economía, pero, en la actualidad, está ganando interés como uno de los posibles marcos teóricos para resolver problemas de control distribuido. Esta teoría describe el comportamiento de sistemas en los que varios actores (agentes) toman decisiones utilizando información local persiguiendo objetivos globales, siendo el comportamiento colectivo dependiente de la recopilación de las decisiones locales y del estado del medio. Desde este punto de vista, tiene relación directa con aquellos problemas de control que involucran la interconexión de múltiples tomadores de decisiones. Además, en el campo de control, es de vital importancia el uso de estrategias que permitan a los agentes cambiar sus decisiones en línea, de acuerdo con los cambios en el entorno o el comportamiento de otros agentes. Las técnicas adecuadas que se pueden aplicar en este contexto son las metodologías basadas en el aprendizaje reforzado (Reinforcement Learning o RL) que ofrecen muchas ventajas y que han motivado el desarrollo de muchos algoritmos para la toma de decisiones y el control, proporcionando el estudio de todas las propiedades de estabilidad, convergencia y viabilidad que implica un problema de control. Muchas aplicaciones prácticas exitosas en esta área también motivan el creciente interés en RL. Precisamente, uno de los principales objetivos de este proyecto es proponer nuevas técnicas basadas en RL para resolver en los juegos multijugador en tiempo real los problemas que surgen en el control distribuido cooperativo.

El aprendizaje por refuerzo se basa en la respuesta que tiene el entorno a nuestras acciones, es decir, si la acción es para el entorno incorrecta se genera una recompensa que será reconocida como inapropiada, pero si la acción es correcta se generará una recompensa que será reconocida como apropiada y, en función de esto, se reconocerá si la acción es correcta o incorrecta. Además de la recompensa el entorno devolverá un estado, consecuencia de la acción que se le ha enviado. El entorno envía la recompensa y el estado u observación al llamado agente, que es el que posee determinadas metas relacionadas con el entorno y el que genera las acciones. El RL aborda los casos de sistemas continuos buscando la política de aprendizaje, y es aquí donde el RL se plantea como un problema de optimización en la búsqueda de políticas. Por ejemplo en [29] el RL tiene como reto el diseño de algoritmos que trabajen en problemas totalmente continuos.

Por lo general, la búsqueda de las mejores acciones está basada en el envío de una sola recompensa al agente, aunque existen investigaciones en las que se trabaja con varias recompensas u objetivos, como es el caso de [5] en el que se plantea que los problemas del mundo real se describen mejor empleando múltiples objetivos, y además se propone un método para acelerar el aprendizaje y

reutilizar el conocimiento en todas las tareas.

El aprendizaje por refuerzo tiene como objetivo principal la capacitación de un agente para que así complete una tarea en un entorno incierto. Son distintos los tipos de agentes presentes en el aprendizaje por refuerzo, entre los cuales está el agente de gradiente de política determinista profunda (DDPG) que ha sido utilizado en este trabajo. Este es un método de políticas, en línea y sin modelos, en el que un agente actor-crítico busca una política óptima que maximice la recompensa acumulada a largo plazo. [20] implementa el algoritmo en una arquitectura distribuida que a su vez se integra con la reproducción de experiencias priorizadas y técnicas de modelado de recompensas para optimizar un conjunto de acciones de tipo híbrido de componentes continuos y discretos.
Las aplicaciones del aprendizaje por refuerzo en campos como la robótica o la conducción autónoma han involucrado la participación de más de un agente. En investigaciones como [23] se emplea a múltiples agentes para la regulación de enrutadores para prevenir la intrusión a una determinada red de datos, en donde el aprendizaje coordinado en equipo proporciona una respuesta coordinada descentralizada. También en [3] se aborda el uso de múltiples agentes donde cada uno se preocupa por una política de juego, planeando maximizar su recompensa que incluye un descuento esperado sobre el conjunto de estados del juego.

Uno de los problemas del control distribuido de sistemas a gran escala es el estado de las relaciones de dependencia entre subsistemas. Cuando estas conexiones representan variables de control el control distribuido tiene que ser consistente para los subsistemas, es decir, la comunicación entre los agentes debe ser consistente. En [14] se exploran las características del DMPC que combina técnicas de aprendizaje para realizar la negociación en un entorno cooperativo de múltiples agentes, en donde el agente MPC es la entidad que se encarga de controlar una parte específica del sistema y un agente negociador determina el valor de una o más variables compartidas entre dos agentes MPC empleando el algoritmo Q-Learning del aprendizaje por refuerzo.

El objetivo global de este trabajo es el desarrollo de una metodología de control multiagente que permita el uso de técnicas pertenecientes al campo del control predictivo distribuido (DMPC) y de las Redes Neuronales Artificiales (RNA) en su formato de aprendizaje por refuerzo para implementar los procesos de negociación entre agentes. Concretamente se realiza en este trabajo una propuesta novedosa de aprendizaje por refuerzo de RNA‘s en las capas superiores de la jerarquía de control y su aplicación a un sistema mutivariable de control de nivel compuesto por ocho tanques interconectados difícil de controlar debido a su naturaleza no lineal y a la alta interacción existente entre sus subsistemas.

El resto del artículo está organizado de la siguiente manera. En la sección dos y tres se describen los algoritmos de control y el método de negociación del que se parte, así como el sistema bajo estudio. El cuarto apartado contiene las soluciones planteadas, en el quinto se describe la estructura del software y en el

sexto se presentan los resultados obtenidos de aplicar las soluciones propuestas seguido de las conclusiones finales.

2. Arquitectura de control

La comunicación entre los agentes se realiza mediante una red que se puede modelar como el grafo indirecto $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ donde $\mathcal{N}$ es el conjunto de agentes y $\mathcal{L}$ es el conjunto de enlaces bidireccionales $\mathcal{L} \subseteq \mathcal{L}^{\mathcal{N}} = \{\{i, j\} : \{i, j\} \subseteq \mathcal{N},\ i \neq j\}$. Un enlace $l_{ij} \in \mathcal{L}$ conecta los agentes i y j dando un flujo de información bidireccional. Si dos agentes están conectados por un enlace de comunicación l_{ij} siguen un esquema cooperativo por pares para encontrar un consenso entre sus secuencias de control a través de un algoritmo multicapa de negociación de lógica difusa en la primera capa.

2.1. Capa de control de bajo nivel

La primera capa de control emplea un algoritmo DMPC para múltiples agentes. Las negociaciones con lógica difusa se realizan en pares teniendo en cuenta los acoplamientos con los subsistemas vecinos. Para ello, se emplea una secuencia desplazada de agente i, que es definida añadiendo $K_i x_i(k + N_\mathrm{p})$ a la secuencia elegida en el paso previo $U_i(k-1)$:

El algoritmo empleado es una extensión del esquema DMPC propuesto por [21] para n subsistemas. El algoritmo lo forman 5 pasos que se muestran a continuación:

Algoritmo 1: Algoritmo DMPC.

Para cada paso de tiempo k:

1. En cada instante k cada agente i mide $y_i(k)$ y estima el estado inicial para la predicción del MPC.
2. El agente i calcula su trayectoria desplazada y la envía a sus vecinos.
3. Cada agente i minimiza su función de costo J_i teniendo en cuenta que su vecino j aplica su trayectoria desplazada. Los otros subsistemas vecinos siguen sus trayectorias de control actuales $U_j^\mathrm{s}(k)$. Específicamente, el agente i resuelve:

$$U_i^*(k) = \arg\min_{U_i(k)} J_i\big(x_i(k), U_i(k), U_j^\mathrm{s}(k), U_l^\mathrm{s}(k)\big), \tag{1}$$

Sujetos a las siguientes restricciones sobre entradas, estados y restricción de estado terminal Ω_i:

$$x_i(k+t+1) = A_i x_i(k+t) + B_{ii} u_i(k+t) + B_{ij} u_j(k+t) + \sum_{l \in \mathcal{N}_i \setminus \{j\}} B_{il} u_l(k+t), \tag{2}$$

$$
\begin{aligned}
& x_i(k) = \tilde{x}_i(k),\ i \in \mathcal{N}, \\
& x_i(k+t) \in \mathcal{X}_i, && t = 0,\ \ldots, N_{\mathrm{p}} - 1, \\
& x_i(k+N_{\mathrm{p}}) \in \Omega_i, \\
& u_i(k+t) \in \mathcal{U}_i, && t = 0,\ \ldots, N_{\mathrm{p}} - 1, \\
& u_j(k+t) = u_j^{\mathrm{s}}(k+t), && t = 0,\ \ldots, N_{\mathrm{p}} - 1, \\
& u_l(k+t) = u_l^{\mathrm{s}}(k+t), && t = 0,\ \ldots, N_{\mathrm{p}} - 1,
\end{aligned} \tag{3}
$$

4. El agente i optimiza nuevamente su costo $J_i(\cdot)$ manteniendo su secuencia de entrada óptima $U_i^*(k)$ fija para encontrar su secuencia de entrada vecina deseada $U_j^{\mathrm{w}_i}(k)$. Aquí, también se considera que los subsistemas l siguen sus trayectorias actuales.

$$
U_j^{\mathrm{w}_i}(k) = \arg \min_{U_j(k)} J_i\big(x_i(k), U_i^*(k), U_j(k), U_l^{\mathrm{s}}(k)\big), \tag{4}
$$

5. Comunicación entre agentes: el agente i envía $U_j^{\mathrm{w}_i}(k)$ al agente j y recibe $U_i^{\mathrm{w}_j}(k)$.

2.2. Negociaciones difusas por pares

Lo siguiente al algoritmo DMPC es aplicar la negociación difusa para obtener una solución de control que garantice la estabilidad y disminuya el índice de desempeño. La negociación se realiza entre cada pareja de agentes:

$$
\{U_i^*(k), U_i^{\mathrm{s}}(k), U_i^{\mathrm{w}_j}(k)\} \text{ y } \{U_j^*(k), U_j^{\mathrm{s}}(k), U_j^{\mathrm{w}_i}(k)\}
$$

El sistema de inferencia difusa para la negociación genera la acción de control final $U_i^{\mathrm{f}}(k)$ y $U_j^{\mathrm{f}}(k)$ que llega a la segunda capa, teniendo en cuenta algunas restricciones operacionales y económicas. La siguiente figura describe el esquema de negociación difusa implementada.

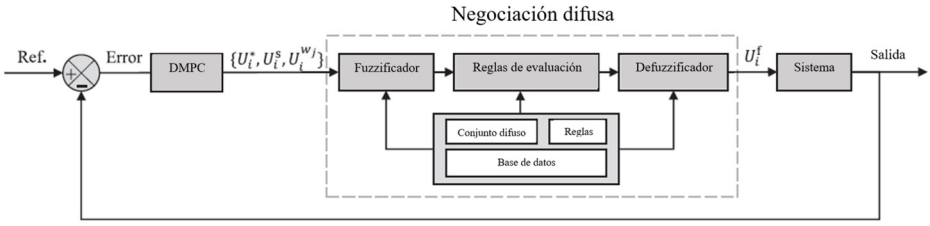


Figura 1: Esquema de la negociación difusa.

El algoritmo de la negociación difusa se muestra a continuación.

Algoritmo 2: Algoritmo de negociación difusa.

Para cada paso de tiempo k:

1. Para cada agente $i \in \mathcal{N}$, existen tres posibilidades de entradas $\{U_i^{\mathrm{s}}(k), U_i^{\mathrm{w}_j}(k), U_i^{*}(k)\}$. Desde la secuencia de control deseada $U_i^{\mathrm{w}_j}(k)$ es computada por el vecino j sin considerar las restricciones del agente i, es necesario comprobar si se satisfacen las restricciones del agente i después de aplicar $U_i^{\mathrm{w}_j}(k)$. De lo contrario, se excluye del proceso de fuzzificación. Posteriormente, se aplica una negociación difusa para calcular la secuencia final $U_i^{\mathrm{f}_m}(k)$. Del mismo modo, se calcula $U_j^{\mathrm{f}_m}(k)$.
2. Se define una secuencia de negociación difusa por pares resultante $U_{ij}^{\mathrm{f}_m}(k) = \{U_i^{\mathrm{f}_m}(k), U_j^{\mathrm{f}_m}(k), U_l^{\mathrm{s}}(k)\}$ basada en $U_i^{\mathrm{f}_m}(k)$ y $U_j^{\mathrm{f}_m}(k)$, asumiendo que el resto de subsistemas $l \in \mathcal{N} \setminus \{i, j\}$ siguen sus trayectorias predefinidas.
3. El agente i envía su costo de las entradas de control difusas y estabilizadoras a sus vecinos, y viceversa. Se define $U_{ij}^{\mathrm{s}} = \{U_i^{\mathrm{s}}(k), U_j^{\mathrm{s}}(k), U_l^{\mathrm{s}}(k)\}$; si la condición

$$\sum_{l \in \mathcal{M}_i \bigcup \mathcal{M}_j \bigcup \{i,j\}} J_l\big(x_l(k), U_{ij}^{\mathrm{f}_m}\big) \leq \sum_{l \in \mathcal{M}_i \bigcup \mathcal{M}_j \bigcup \{i,j\}} J_l\big(x_l(k), U_{ij}^{\mathrm{s}}\big) \qquad (5)$$

se mantiene, entonces la estabilidad está garantizada, y así, $U_i^{\mathrm{f}_m}(k)$ se envía a la capa de control superior. De lo contrario, se envía $U_i^{\mathrm{s}}(k)$.

2.3. Capa superior: Aprendizaje por refuerzo

Las alternativas propuestas en este trabajo utilizan Redes Neuronales para la negociación entre agentes, concretamente en la capa superior de la estructura de control jerárquica mencionada, se emplean Redes Neuronales Multicapa basadas en aprendizaje por refuerzo para la negociación entre agentes que permiten la optimización de cualquier índice de comportamiento dinámico relacionados con la estabilidad, el rendimiento, la economía y el tratamiento de restricciones adicionales no contempladas en los niveles inferiores del sistema jerárquico de control.

El algoritmo propuesto es el de gradiente de política determinista profunda [18] (DDPG), el cual es un algoritmo fuera de la política, no basado en modelo y en el marco Actor-Crítico, que emplea redes neuronales profundas tanto para la política como para la función valor, estas se corresponden a la política π_ϕ (s) y al valor $Q_\theta(s, a)$, ambas comparten los parámetros ϕ y θ. DDPG tiene como objetivo obtener una estrategia óptima π_ϕ y maximizar el valor de retorno esperado $J(\phi) = Es_i \sim p_\pi a_i \sim \pi[R_0]$.

DDPG actualiza los parámetros de la red de política a través del gradiente:

$$\bigtriangledown_\phi J(\phi) = E_{s-p_*}[\bigtriangledown_a Q^\pi(s, a)|_{a \sim \pi(s)} \bigtriangledown_\phi \pi_\phi(s)]$$

donde $Q^\pi(s, a) = E_{s_j - p_r, a_i - \pi}[R_t|s, a]$ es el valor de retorno esperado en el estado s después de que la acción a siga la condición de la estrategia π.

Los parámetros de la red de valor se actualizan de acuerdo a la función de pérdida mínima $L(\theta)$.

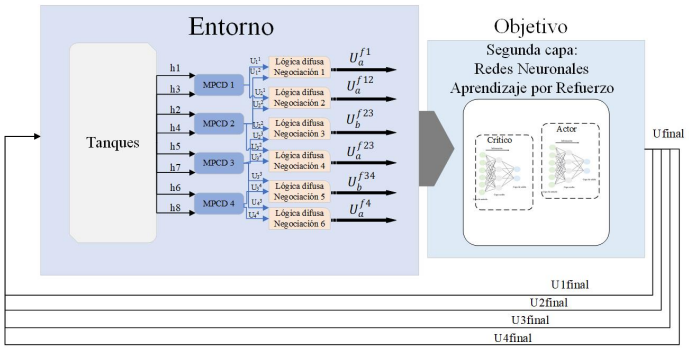


Figura 2: Objetivo - Aprendizaje por Refuerzo

$$L(\theta) = E_{s_s}, a_l, r(s_t, a_t), s_{t+1}[(y_t - Q_\theta(S_t, a_t))^2]$$

donde ϕ y θ representan respectivamente a la red de política objetiva y al valor objetivo. DDPG transfiere la información de gradiente de la función de valor Q sobre la acción a la red de estrategia por la función de red de valor. La política se lleva a cabo aumentando el valor de Q.

La estructura del algoritmo [24] se muestra a continuación.

Algoritmo 3: Algoritmo Gradiente de política determinista profunda.

1. Inicio aleatorio de la red del crítico $Q(s, a|\theta^\mu)$ y la red del actor $\mu(s|\theta^\mu)$ con pesos θ^Q y θ^μ
2. Inicio de las redes objetivo Q' y μ' con pesos $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
3. Inicio del $buffer$ de interacciones R
4. **for** $episodio = 1, M$ **do**
5. Inicio de proceso aleatorio N para la exploración del espacio de acciones.
6. Se recibe estado observacional inicial S_1
7. **for** $t = 1, T$ **do**
8. Selección de la acción $a_t = \mu(s_t|\theta^\mu) + N_t$ de acuerdo con la política actual y el ruido exploratorio.
9. Se ejecuta la acción a_t y se recibe la recompensa r_t y el nuevo estado s_{t+1}
10. Se almacena la transición (s_t, a_t, r_t, s_{i+t}) en R
11. Se muestrea el $minibatch$ aleatorio de N transiciones (s_i, a_i, r_i, s_{i+1}) de R
12. Se considera $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta_{\mu'})|\theta_{Q'})$
13. Se actualiza el crítico minimizando la pérdida: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

14. Se actualiza la política del actor empleando el gradiente de la política:
$\bigtriangledown_{\theta^\mu} J \approx \frac{1}{N} \sum_i \bigtriangledown_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \bigtriangledown_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$
15. Por último, se actualizan las redes objetivo:
$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16. **end**
17. **end**

3. Caso de estudio

La planta de laboratorio sobre la que se han probado las estrategias de control planteadas en este trabajo está formada por ocho tanques acoplados basados en el proceso de tanque cuádruple, además se encuentran interconectados como muestra la figura.

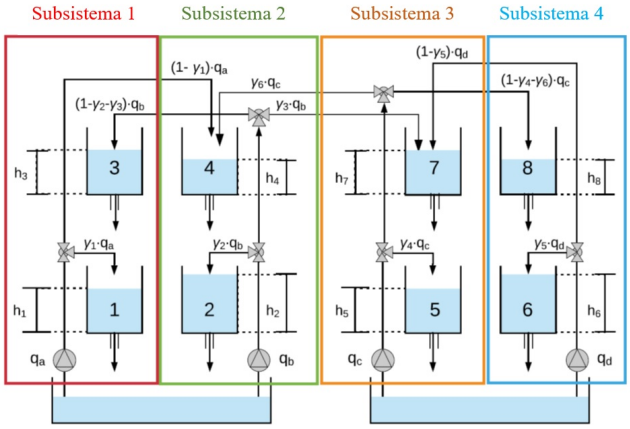


Figura 3: Sistema de 8 tanques interconectados.

Los detalles que describen a la planta se encuentran en [25] en el caso que desee profundizar en ella.

Objetivo de control

Las soluciones planteadas debe de cumplir el objetivo de control que es dirigir al sistema hacia el origen del espacio de estados garantizando que las restricciones sean cumplidas, y minimizando la suma de las funciones de costo locales. En cada instante de tiempo k la función de coste del subsistema $i \in \mathcal{N}$ se calcula en base a las trayectorias predichas de sus estados y entradas sobre el horizonte de predicción:

$$J_i\big(x_i(k), U_i(k), U_j(k)\big) = \sum_{t=0}^{N_\mathrm{p}-1} L_i\big(x_i(k+t), u_i(k+t)\big) + F_i\big(x_i(k+N_\mathrm{p})\big), \quad (6)$$

donde $L_i(\cdot)$ es la función de coste de estado, y $F_i(\cdot)$ es la función de costo terminal. Definidas como:

$$L_i\big(x_i(k+t), u_i(k+t)\big) = x_i(k+t)^\top Q_i x_i(k+t) + u_i(k+t)^\top R_i u_i(k+t),$$

$$F_i\big(x_i(k+N_\mathrm{p})\big) = x_i(k+N_\mathrm{p})^\top P_i x_i(k+N_\mathrm{p}),$$

4. Soluciones planteadas

El primer paso ha sido determinar qué variables son consideradas como observación, acción y recompensa. Se han tenido en cuenta dos opciones, en las que la observación y acción son distintas manteniendo la misma recompensa en ambas. De la capa de negociación de lógica difusa se tienen las siguientes señales resultantes:

U_a^{f1} = La señal del subsistema 1 negociado con el subsistema 2.
U_a^{f12} = La señal del subsistema 2 negociado con el subsistema 1.
U_b^{f23} = La señal del subsistema 2 negociado con el subsistema 3.
U_a^{f23}= La señal del subsistema 3 negociado con el subsistema 2.
U_b^{f34} = La señal del subsistema 3 negociado con el subsistema 4.
U_a^{f4} = La señal del subsistema 4 negociado con el subsistema 3.

Cada una de estas señales están conformadas por cinco valores correspondientes al horizonte de predicción N formado por una secuencia de cinco valores.

4.1. Opción 1

El agente necesita que el entorno le envíe una observación o estado junto con una recompensa que buscará maximizar. Teniendo en cuenta esto, se van a considerar como observaciones las señales de control que se obtienen de la primera capa de negociación difusa, pero no todas las señales sino solo aquellas pertenecientes a los subsistemas 2 y 3. Las señales del subsistema 1 y subsistema 4 no han sido consideradas como observaciones ya que se consideran señales finales de esos subsistemas. El actor del aprendizaje por refuerzo generará las señales U^{f2} y U^{f3} correspondientes de los sistemas 2 y 3, y el entorno generará sus observaciones en función de estas señales. Como recompensa se ha considerado al negativo de la suma de los errores cuadráticos, es decir, de las diferencias al cuadrado entre los niveles de referencia y los niveles reales que se obtienen de las señales U^{f1},U^{f2}, U^{f3} y U^{f4}.

4.2. Opción 2

En este caso se consideran como observación todas las señales resultantes de la primera capa de negociación. Y además el actor del aprendizaje por refuerzo no solo generará las señales U^{f2} y U^{f3} de los subsistemas 2 y 3 sino también U^{f1} y U^{f4} de los subsistemas 1 y 4. La recompensa se considera la misma que en la opción 1.

5. Software desarrollado

Está formado por las funciones *Aprendizaje_Refuerzo.m*, *Inicializa.m*, *Entorno_Reset.m*, *Entorno_Step.m*, *Calc_Ref_8.m* y otras funciones que se encargan de realizar los cálculos de la función *Entorno_Step.m*. Todas estas funciones tienen un papel que realizan en determinados momentos, La función *Inicializa.m* envía los parámetros de inicialización a la función *Entorno_Step.m* una sola vez al inicio de ejecución del software, la función *Entorno_Reset.m* envía las observaciones iniciales al inicio de cada episodio, estando formado el entrenamiento por número de episodios que a su vez están formados por número de pasos, es decir, un episodio lo forman un número determinado de pasos. En cada paso la función *Entorno_Step.m* recibe acción del agente y en función de esta envía la observación y la recompensa. La función *Calc_Ref_8.m* calcula la referencia a utilizar en cada paso mientras "Otras funciones" realizan los cálculos de las capas inferiores de control y otros cálculos.

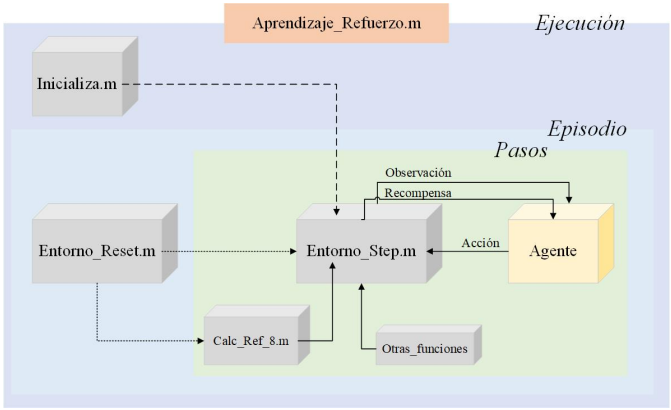


Figura 4: Esquema general del software

6. Resultados

En este apartado se describen las características conjuntas que tiene el entrenamiento de las dos opciones así como sus resultados particulares de entrenamiento y validación.

6.1. Entrenamiento

Para calcular la referencia durante el entrenamiento se utiliza una función la cual, en función del número de iteraciones n, asignará una u otra referencia, el intervalo en que se pasa de una referencia a otra dependerá del valor de la variable T que cambia de manera aleatoria en cada inicio de cada episodio.. El intervalo de los valores que pueden tomar las señales de acción durante el entrenamiento está entre -0.3 y 0.3, donde cada señal de acción y observación está formada por una secuencia de 5 valores correspondientes al horizonte de predicción N.

El entrenamiento de la opción 1 ha consistido en 6000 episodios formado cada uno por 200 pasos. Para el caso de la opción 2, esta ha sido entrenada con 8000 episodios y 200 pasos por episodio.

Las siguientes imágenes muestran el resultado del entrenamiento de las dos opciones, donde cada punto azul representa la suma de doscientas recompensas en cada episodio, teniendo en cuenta que cada recompensa es el negativo de la suma de los errores cuadráticos entre los niveles de referencia y los niveles actuales el objetivo del aprendizaje es acercar esa suma cuadrática negativa a cero, razón por la cual se ven muchos puntos azules próximos a cero. En el eje y se puede ver qué valor toma la suma de las recompensas, mientras que el eje x indica el episodio.

Para el caso de la opción 1 se ve una pequeña tendencia a obtener valores de esos puntos cada vez menos alejados del cero, tendencia que no se aprecia tan claramente en la opción 2 aunque indudablemente sucede. En cualquier caso, nunca debe ser cero puesto que entrenamos con saltos en la referencia que hacen que el error no sea cero.

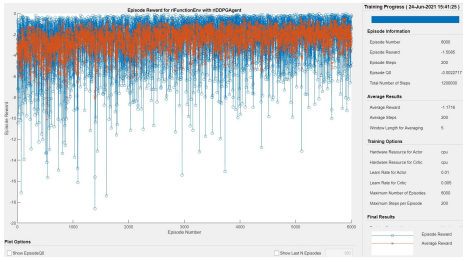


Figura 5: Entrenamiento - Opción 1.

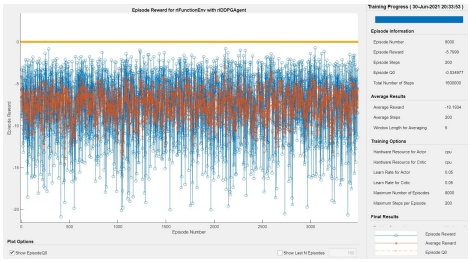


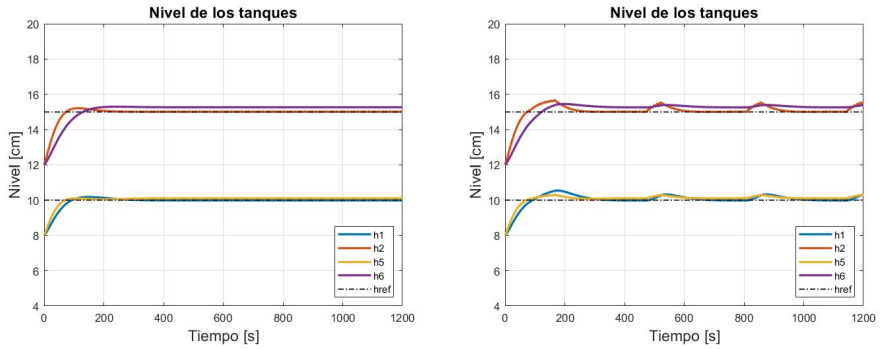
Figura 6: Entrenamiento - Opción 2.

6.2. Validación Opción 1

Una vez entrenada la red, el siguiente proceso es su validación. Conviene aquí resaltar que el entrenamiento se realiza sin imponer las condiciones de estabilidad del sistema de control mencionadas en [25]. No obstante, en la implementación del agente pueden imponerse si fuese necesario. La idea de estas condiciones es detectar aquellas situaciones en las que la función de Lyapunov elegida (suma cuadrática de los errores de predicción) no decrece y utilizar un control de refuerzo que garantiza su decrecimiento. Los resultados contemplan diferentes situaciones.

En primer lugar, analizaremos el comportamiento del sistema de control (agente entrenado) en problemas de regulación. En este caso iniciando el estado del sistema en puntos aleatorios, el agente debe llevarlos al punto de operación definido previamente. Este comportamiento garantizará un adecuado rechazo de perturbaciones.
Las imágenes que se muestran a continuación contienen dos bloques de comparación de los resultados sin condiciones de estabilidad con los de con condiciones de estabilidad, la diferencia entre el primer y segundo bloque es que cambia el inicial de operación.

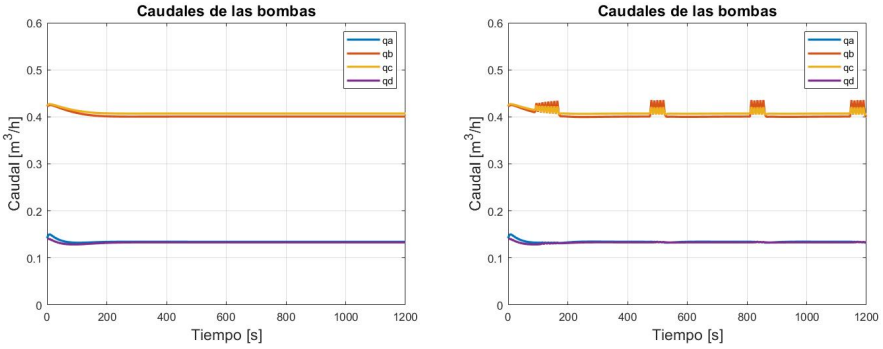
Primer bloque



(a) Niveles de los tanques controlados sin condiciones de estabilidad.

(b) Niveles de los tanques controlados con condiciones de estabilidad.

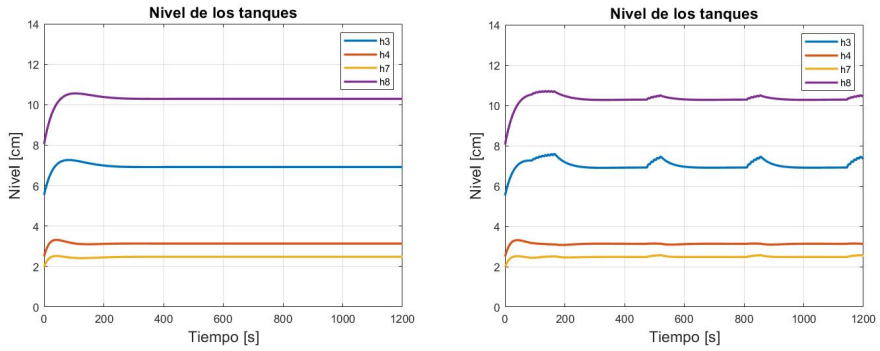
Figura 7: Validación Opción 1 - bloque 1, comparación entre niveles de los tanques controlados sin y con condiciones de estabilidad.



(a) Caudales de las bombas sin condiciones de estabilidad.

(b) Caudales de las bombas con condiciones de estabilidad.

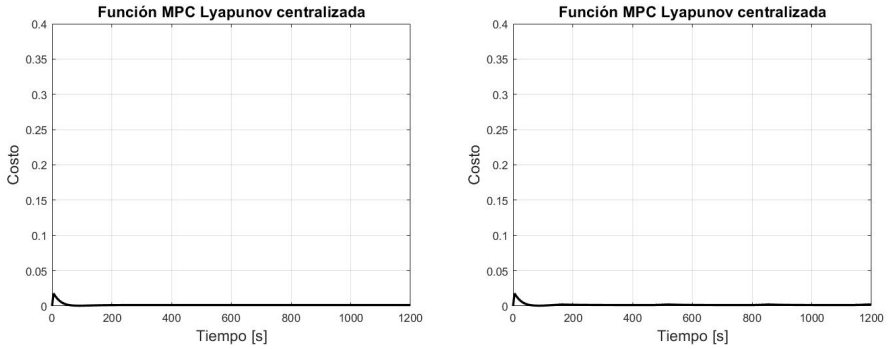
Figura 8: Validación Opción 1 - bloque 1, comparación entre los caudales de las bombas sin y con condiciones de estabilidad.



(a) Niveles de los tanques no controlados y sin condiciones de estabilidad.

(b) Niveles de los tanques no controlados y con condiciones de estabilidad.

Figura 9: Validación Opción 1 - bloque 1, comparación entre los niveles de los tanques no controlados, sin y con condiciones de estabilidad.



(a) Lyapunov sin condiciones de estabilidad.

(b) Lyapunov con condiciones de estabilidad.

Figura 10: Validación Opción 1 - bloque 1, comparación entre los Lyapunov, sin y con condiciones de estabilidad.

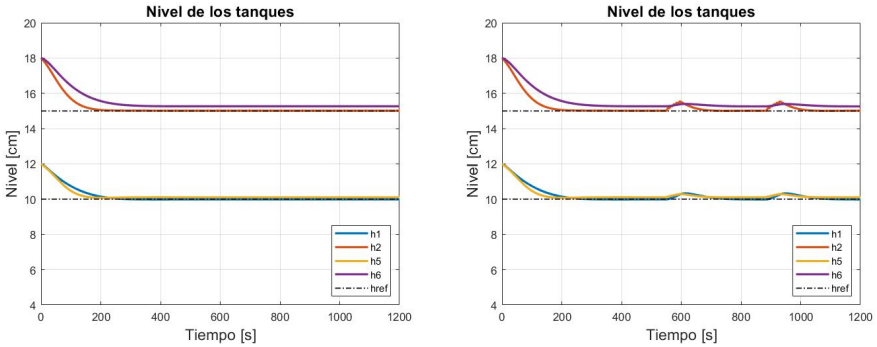
La respuesta de ambas alternativas es la misma, y es muy buena. En este caso la estabilidad está garantizada, aunque se implementan las dos opciones, sin imponer e imponiendo las condiciones de estabilidad. En la figura 7 de la derecha se aprecian ligeras oscilaciones en la respuesta debida a problemas numéricos que no deberían aparecer. Decir que es debido a la implementación software original y que serían fácilmente corregibles. Lo mismo sucede en las señales de control (Figura 8). Observamos en la derecha que se recurre al control de respaldo en situaciones no necesarias por problemas numéricos. En la figura 9 se muestran

los niveles de los depósitos de arriba que no se controlan, pero sí se limitan para evitar desbordamientos. Se aprecia una buena operación en todos ellos. Finalmente, la figura 10 muestra la evolución de la función de Lyapunov en ambos casos. Vemos la tendencia descendente que garantiza la estabilidad del sistema entrenado en las dos versiones de su implementación.

Se ha podido observar en ambos casos la presencia un pequeño error estacionario el nivel controlado del tanque 6, los otros 3 niveles controlados siguen a la referencia. Decir que es un error tolerable para este tipo de sistemas que no alcanza el valor de un cm..

Segundo bloque

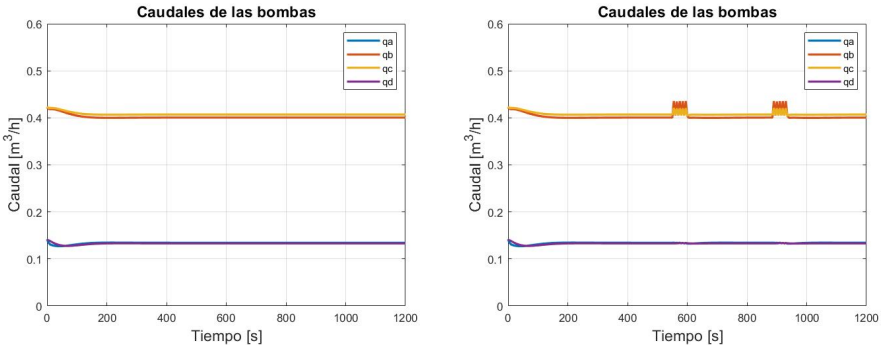
En este segundo bloque de resultados de validación se parte de punto iniciales del estado de la planta inferiores al punto de operación.



(a) Niveles de los tanques controlados sin condiciones de estabilidad.

(b) Niveles de los tanques controlados con condiciones de estabilidad.

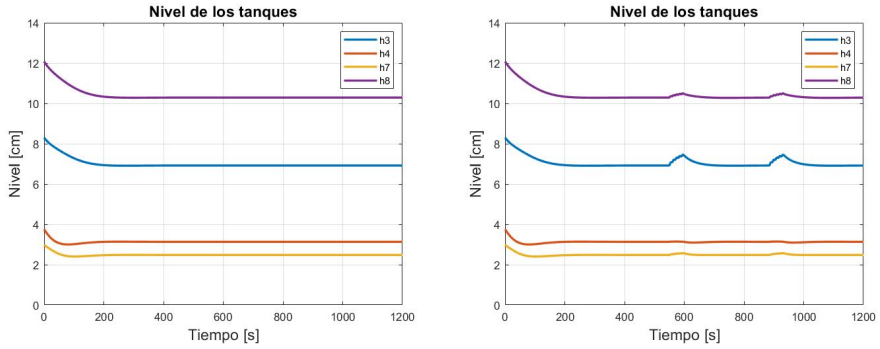
Figura 11: Validación Opción 1 - bloque 2, comparación entre niveles de los tanques controlados sin y con condiciones de estabilidad.



(a) Caudales de las bombas sin condiciones de estabilidad.

(b) Caudales de las bombas con condiciones de estabilidad.

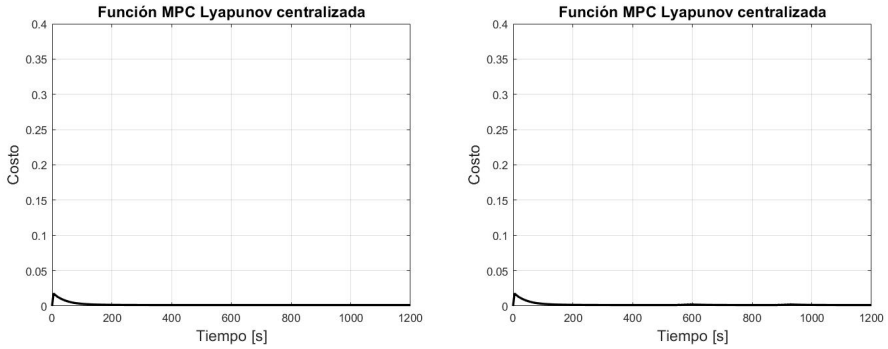
Figura 12: Validación Opción 1 - bloque 2, comparación entre los caudales de las bombas sin y con condiciones de estabilidad.



(a) Niveles de los tanques no controlados y sin condiciones de estabilidad.

(b) Niveles de los tanques no controlados y con condiciones de estabilidad.

Figura 13: Validación Opción 1 - bloque 2, comparación entre los niveles de los tanques no controlados, sin y con condiciones de estabilidad.



(a) Lyapunov sin condiciones de estabilidad.

(b) Lyapunov con condiciones de estabilidad.

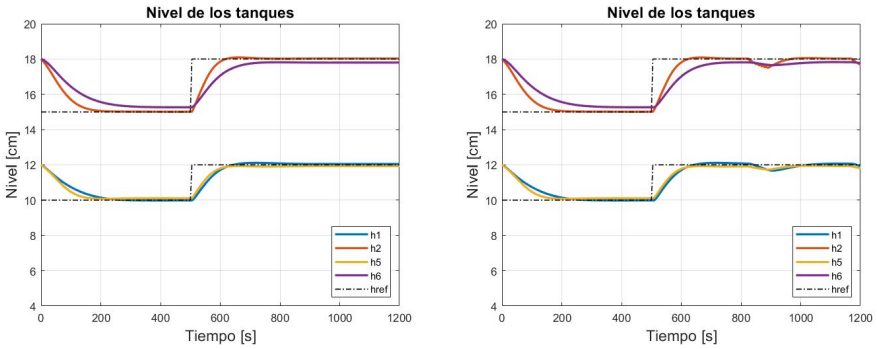
Figura 14: Validación Opción 1 - bloque 2, comparación entre los Lyapunov, sin y con condiciones de estabilidad.

En este caso pasa lo mismo que en el primer bloque donde el nivel del tanque 6 presenta un pequeño error en el estacionario, en el caso de las señales de los tanques controlados y con condiciones de estabilidad en la validación se puede observar ruido presente en dos puntos de la simulación, ruido que se ve también reflejado en los niveles de los tanques no controlados.

El sistema se comporta bien incluso cuando se coloca la referencia alejada del punto de operación, además tiene un buen rechazo a posibles perturbaciones. La estabilidad esta garantizada ya que el lyapunov decrece y se mantiene.

Complementario al problema de regulación está la validación con salto en la referencia también dividida en caso sin condiciones de estabilidad y con condiciones de estabilidad.

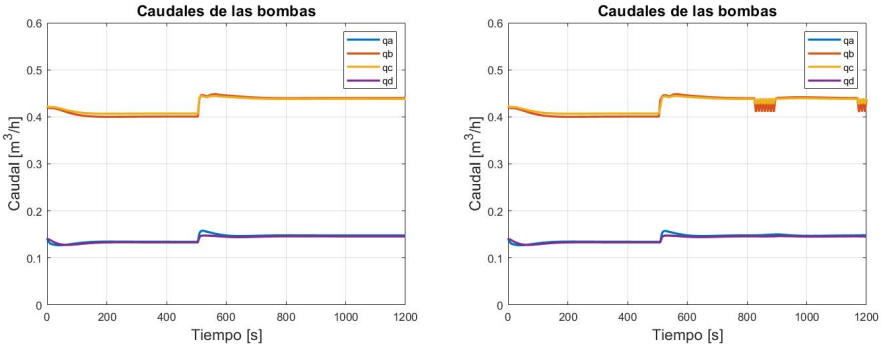
En este caso, se observa que también la presencia de un error estacionario en el nivel del tanque 6, completamente asumible y se detectan los problemas numéricos mencionados de la implementación del caso en el que se imponen condiciones de estabilidad. La figura 18 muestra las funciones de Lyapunov obtenidas en ambos casos y su comportamiento decreciente hasta cero cuando se produce el salto en la referencia.



(a) Niveles de los tanques controlados sin condiciones de estabilidad.

(b) Niveles de los tanques controlados con condiciones de estabilidad.

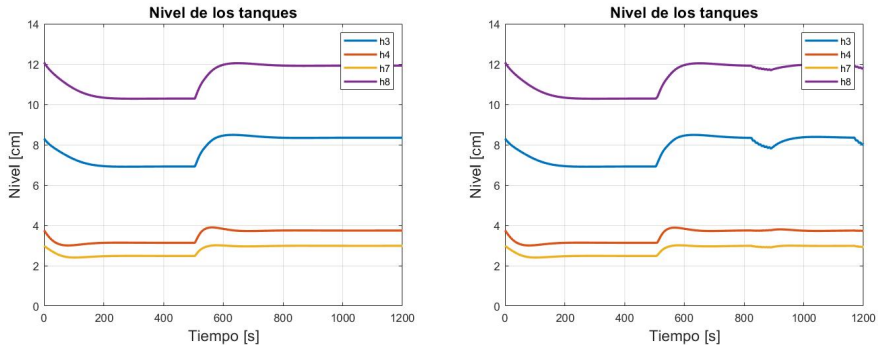
Figura 15: Validación Opción 1, comparación entre niveles de los tanques controlados sin y con condiciones de estabilidad.



(a) Caudales de las bombas sin condiciones de estabilidad.

(b) Caudales de las bombas con condiciones de estabilidad.

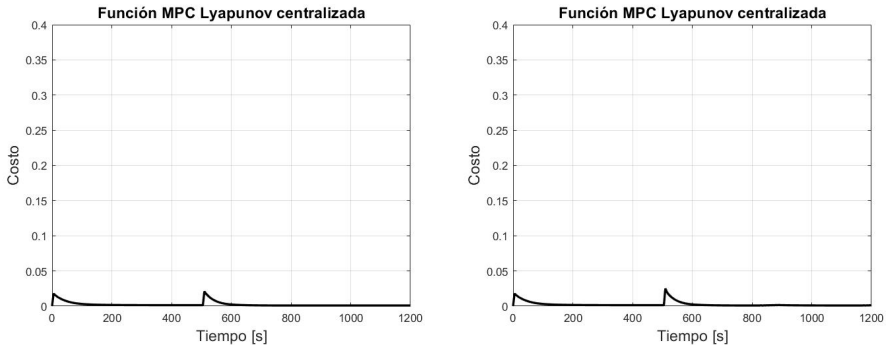
Figura 16: Validación Opción 1, comparación de caudales de las bombas sin y con condiciones de estabilidad.



(a) Niveles de los tanques no controlados y sin condiciones de estabilidad.

(b) Niveles de los tanques no controlados y con condiciones de estabilidad.

Figura 17: Validación Opción 1, comparación entre los niveles de los tanques no controlados, sin y con condiciones de estabilidad.



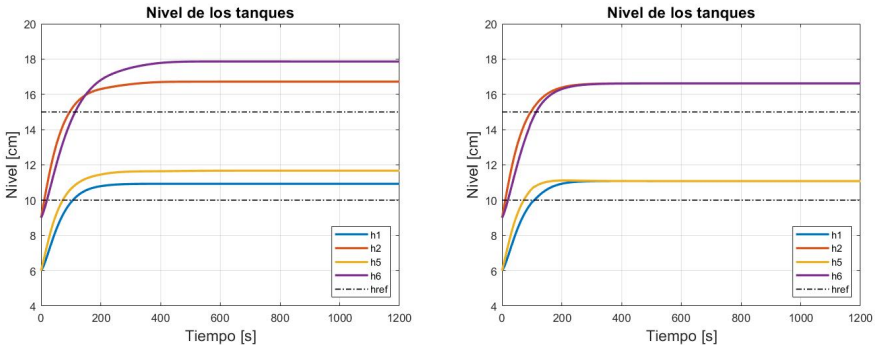
(a) Lyapunov sin condiciones de estabilidad.

(b) Lyapunov con condiciones de estabilidad.

Figura 18: Validación Opción 1, comparación entre los Lyapunov, sin y con condiciones de estabilidad.

6.3. Validación Opción 2

En el caso de la opción 2 también se parte de un problema de regulación, solo que en este caso teniendo en cuenta los resultados no se prosiguió con la validación con salto en la referencia.



(a) Niveles de los tanques controlados sin condiciones de estabilidad.

(b) Niveles de los tanques controlados con condiciones de estabilidad.

Figura 19: Validación Opción 2, comparación entre niveles de los tanques controlados sin y con condiciones de estabilidad.

Esta opción presenta un mucho mayor error estacionario en comparación con la opción 1, llega a controlar pero no alcanza el punto de equilibrio.

7. Conclusiones

El trabajo realizado propone una solución novedosa del uso de redes neuronales con aprendizaje por refuerzo y el algoritmo DDPG, como una opción válida en la negociación de sistemas multiagente de controladores jerárquicos distribuidos. La solución propuesta presenta la ventaja de que no requiere el uso de modelos en la negociación y permite la optimización de cualquier índice dinámico asociado al rendimiento del sistema de control y a la estabilidad del sistema resultante. Los resultados obtenidos son muy prometedores para el control de sistemas complejos con alta interacción entre subsistemas cuyo control centralizado, aunque más preciso, sería mucho más complejo y requiere el uso de modelos de la planta completa para su control.
Se han considerado diferentes escenarios siendo la primera opción, la negociación de las señales de dos agentes de la capa inferior, los subsistemas 2 y 3, la que proporciona mejores resultados que la opción que considera las señales que provienen de 4 subsistemas. Esto es debido a la naturaleza del problema y nos lleva a plantearnos que la definición del mismo es clave y el análisis previo es una labor muy importante para el éxito de la tarea.
El entrenamiento de la primera opción tuvo una duración de 4 días con funcionamiento ininterrumpido y la duración de la opción 2 fue de 5 días. Por ello, hay que estudiar alternativas que reduzcan estos tiempos y proporcionen más precisión. En este sentido la definición de las recompensas es clave y metodología propuesta presenta la ventaja de ser muy flexible en este sentido. El entrenamiento de las neuronales (actor y crítico) permite utilizar cualquier función dinámica

del estado y ajustar la política de control sin un conocimiento previo del sistema. En el trabajo actual se han considerado los sólo los errores cuadráticos en cada episodio. Pensamos que la función de Lyapunov que se calcula en cada paso y pesa los errores de predicción, podría incluirse como recompensa explícitamente acercándonos más al problema que queremos resolver. Esta función, además, tiene relación directa con las señales consideradas como entrada del agente y podría mejorar los tiempos de entrenamiento. Asimismo, en lo que respecta a las redes neuronales, para hacer más eficiente el entrenamiento es también posible partir de una red supervisada ya entrenada, y explorar el uso de otro tipo de redes como las redes convolucionales profundas y las recurrentes debido a sus características.

Referencias

1. *https://es.mathworks.com/products/reinforcement-learning.html*.
2. Faraz Ahmad, Pushpendra Kumar, Anamika Bhandari, and Pravin P Patil. Simulation of the quadcopter dynamics with lqr based control. *Materials Today: Proceedings*, 24:326–332, 2020.
3. Ali Akramizadeh, Ahmad Afshar, Mohammad-B Menhaj, and Samira Jafari. Model-based reinforcement learning in multiagent systems in extensive forms with perfect information. *IFAC Proceedings Volumes*, 43(8):487–494, 2010.
4. Mostafa Al-Gabalawy. A hybrid mpc for constrained deep reinforcement learning applied for planar robotic arm. *ISA transactions*, 2021.
5. Rodrigo Cesar Bonini, Felipe Leno da Silva, and Anna Helena Reali Costa. Learning options in multiobjective reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
6. Yankai Cao and R Bhushan Gopaluni. Deep neural network approximation of nonlinear model predictive control. *IFAC-PapersOnLine*, 53(2):11319–11324, 2020.
7. GB Castro, JSC Martini, and AR Hirakawa. Multilayer distributed model predictive control of urban traffic. *WIT Transactions on Ecology and the Environment*, 179:967–976, 2013.
8. Scarlett Chen, Zhe Wu, David Rincon, and Panagiotis D Christofides. Machine learning-based distributed model predictive control of nonlinear processes. *AIChE Journal*, 66(11):e17013, 2020.
9. Zheng Chen, Hengjie Hu, Yitao Wu, Yuanjian Zhang, Guang Li, and Yonggang Liu. Stochastic model predictive control for energy management of power-split plug-in hybrid electric vehicles based on reinforcement learning. *Energy*, 211:118931, 2020.
10. Yu Ding, Liang Ma, Jian Ma, Mingliang Suo, Laifa Tao, Yujie Cheng, and Chen Lu. Intelligent fault diagnosis for rotating machinery using deep q-network based health state classification: A deep reinforcement learning approach. *Advanced Engineering Informatics*, 42:100977, 2019.
11. Xuefeng Han, Hongwen He, Jingda Wu, Jiankun Peng, and Yuecheng Li. Energy management based on reinforcement learning with double deep q-learning for a hybrid electric tracked vehicle. *Applied Energy*, 254:113708, 2019.
12. Zhenglei He, Kim Phuc Tran, Sebastien Thomassey, Xianyi Zeng, Jie Xu, and Changhai Yi. Multi-objective optimization of the textile manufacturing process using deep-q-network based multi-agent reinforcement learning. *Journal of Manufacturing Systems*, 2021.

13. Chi-Hung Hsu, Shu-Huan Chang, Jhao-Hong Liang, Hsin-Ping Chou, Chun-Hao Liu, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. Monas: Multi-objective neural architecture search using reinforcement learning. *arXiv preprint arXiv:1806.10332*, 2018.
14. Valeria Javalera, Bernardo Morcego, and Vicenç Puig. Negotiation and learning in distributed mpc of large scale systems. In *Proceedings of the 2010 American Control Conference*, pages 3168–3173. IEEE, 2010.
15. Miroslav Kubat. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. *The Knowledge Engineering Review*, 13(4):409–412, 1999.
16. Pratyush Kumar, James B Rawlings, and Stephen J Wright. Industrial, large-scale model predictive control with structured neural networks. *Computers & Chemical Engineering*, 150:107291, 2021.
17. Jay H Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415–424, 2011.
18. Jiawen Li and Tao Yu. A new adaptive controller based on distributed deep reinforcement learning for pemfc air supply system. *Energy Reports*, 7:1267–1279, 2021.
19. Ziang Li, Zhengtao Ding, and Meihong Wang. Optimal bidding and operation of a power plant with solvent-based carbon capture under a co2 allowance market: A solution with a reinforcement learning-based sarsa temporal-difference algorithm. *Engineering*, 3(2):257–265, 2017.
20. Xuze Liu, Abbas Fotouhi, and Daniel J Auger. Formula-e race strategy development using distributed policy gradient reinforcement learning. *Knowledge-Based Systems*, 216:106781, 2021.
21. JM Maestre, D Munoz De La Pena, EF Camacho, and T Alamo. Distributed model predictive control based on agent negotiation. *Journal of Process Control*, 21(5):685–697, 2011.
22. JM Maestre, MA Ridao, Attila Kozma, Carlo Savorgnan, Moritz Diehl, MD Doan, A Sadowska, T Keviczky, B De Schutter, H Scheu, et al. A comparison of distributed mpc schemes on a hydro-power plant benchmark. *Optimal Control Applications and Methods*, 36(3):306–332, 2015.
23. Kleanthis Malialis and Daniel Kudenko. Distributed response to network intrusions using multiagent reinforcement learning. *Engineering Applications of Artificial Intelligence*, 41:270–284, 2015.
24. Guillermo Urcera Martín. Generación de trayectorias robóticas mediante aprendizaje profundo por refuerzo.
25. Eva Masero, Mario Francisco, José M Maestre, Silvana Revollar, and Pastora Vega. Hierarchical distributed model predictive control based on fuzzy negotiation. *Expert Systems with Applications*, 176:114836, 2021.
26. Teresa Mendonça and Pedro Lago. Pid control strategies for the automatic control of neuromuscular blockade. *Control Engineering Practice*, 6(10):1225–1231, 1998.
27. Ahmad Mirzaei and Amin Ramezani. Cooperative optimization-based distributed model predictive control for constrained nonlinear large-scale systems with stability and feasibility guarantees. *ISA transactions*, 2021.
28. Ifrah Saeed, Tansu Alpcan, Sarah M Erfani, and M Berkay Yilmaz. Distributed nonlinear model predictive control and reinforcement learning. In *2019 Australian & New Zealand Control Conference (ANZCC)*, pages 1–3. IEEE, 2019.
29. Olivier Sigaud and Freek Stulp. Policy search in continuous action domains: an overview. *Neural Networks*, 113:28–40, 2019.

30. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction. Second edition.* The MIT Press, 2014.
31. Nathalie Vanvuchelen, Joren Gijsbrechts, and Robert Boute. Use of proximal policy optimization for the joint replenishment problem. *Computers in Industry*, 119:103239, 2020.
32. Qishao Wang, Zhisheng Duan, Yuezu Lv, Qingyun Wang, and Guanrong Chen. Linear quadratic optimal consensus of discrete-time multi-agent systems with optimal steady state: A distributed model predictive control approach. *Automatica*, 127:109505, 2021.
33. Yin-Hao Wang, Tzuu-Hseng S Li, and Chih-Jui Lin. Backward q-learning: The combination of sarsa algorithm and q-learning. *Engineering Applications of Artificial Intelligence*, 26(9):2184–2193, 2013.
34. Yinlong Yuan, Zhu Liang Yu, Zhenghui Gu, Yao Yeboah, Wu Wei, Xiaoyan Deng, Jingcong Li, and Yuanqing Li. A novel multi-step q-learning method to improve data efficiency for deep reinforcement learning. *Knowledge-Based Systems*, 175:107–117, 2019.

Robótica Educativa

Rosa Mª Folgoso Bullejos, Vidal Moreno Rodilla, Francisco Javier Blanco Rodríguez

Departamento de Informática y Automática
Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
rosafolgoso@usal.es
https://www.usal.es/

Resumen La tecnología y la robótica están avanzando a pasos agigantados y, con ello, el interés por su aplicación en colegios e institutos es cada vez mayor. Sin embargo, a día de hoy, son muy pocos los centros educativos españoles que han establecido una asignatura de Robótica Educativa en sus aulas.

Este informe es un breve resumen del Trabajo de Fin de Máster en el cual se realiza una profunda revisión del estado del arte sobre este tema, analizando la multitud de plataformas robóticas educativas que hay hoy en día en el mercado para que los niños se aproximen y conozcan mecánica, electrónica o informática. Además, se lleva a cabo una propuesta de plataforma robótica de tipo *Lego* para su implementación en las aulas de Primaria y Secundaria. Esta propuesta se adapta a las enseñanzas de cada nivel educativo y ayuda a fomentar la creatividad de los niños.

Keywords: Plataforma Robótica Educativa, Lego, Solidworks, mecánica, electrónica, informática, Arduino

1. Introducción

Actualmente la tecnología está totalmente integrada en nuestro día a día, nos servimos de ella para hacer tareas tan cotidianas como cocinar, jugar, estudiar, etc. A pesar de ello, en los primeros años de la escuela, los niños aprenden muy poco en relación a este tema, aunque luego vuelven a sus casas y se entretienen con una *tablet*, el móvil de su padre o de su madre o un juguete que se mueve por si mismo, es decir, un robot.

La disciplina de la robótica puede ofrecer a niños la oportunidad de aprender sobre la mecánica, sensores, motores, la programación y el mundo digital. Además, puede contribuir al desarrollo de la matemática aplicada, el método científico de investigación y resolución de problemas, así como invitar a los niños a participar en interacciones sociales y a colaborar al jugar para aprender y aprender a jugar [3]. Con la popularidad creciente de la robótica y dado el

escenario social actual, se están incorporando estrategias, técnicas y métodos educativos enfocados a lograr procesos de enseñanza-aprendizaje más dinámicos, colaborativos y realistas [24]. Además, el uso de paquetes instructivos de robótica se va difundiendo en centros de enseñanza secundaria media y superior y escuelas primarias, por ejemplo, los kits de Arduino. Al introducir a los menores en este mundo de la robótica, empiezan a conocer la tecnología y la ingeniería, de forma que se benefician en el desarrollo de la motricidad fina[1] y la coordinación de ojo-manos, así como a aprender ideas abstractas al jugar con piezas mecánicas para diseñar sus robots (como manivelas, palancas, motores, engranajes, etc.).

1.1. Objetivos

Por todo esto, el propósito del trabajo es realizar una exhaustiva investigación y revisión del estado del arte de todos los avances que se están produciendo en Robótica Educativa para realizar una propuesta de Plataforma Robótica (PR) destinada al aprendizaje de niños de Educación Primaria y Secundaria.

Para el desarrollo de esta Plataforma Robótica, además de una previa investigación, será necesario desarrollar su diseño en 3D para su posterior fabricación. Asimismo, es preciso tener conocimientos de electrónica para poder decidir cómo enseñarla y adaptarla a niños, así como saber escoger el microcontrolador, sensores, actuadores... atendiendo a las capacidades y al poco conocimiento de los escolares de esta rama. Y por último, habrá que llevar a cabo el código ejemplo para que el robot sea controlado, al igual que la electrónica, deberá ser un código que entiendan y puedan elaborar los alumnos, por lo que habrá que escoger un lenguaje de programación sencillo para ellos.

Para que esta enseñanza sea lo más provechosa posible, se asociarán las distintas fases del aprendizaje en Robótica con las capacidades cognitivas que puedan desarrollar los escolares, concretamente con la Teoría del Desarrollo Cognitivo de Piaget, una de las más citadas y más importantes a nivel pedagógico.

2. Revisión del estado el arte

El concepto de *Robótica Educativa* (RE) es la unión de los términos *robótica* y *educación*. En general, la RE es el concepto que abarca el uso de la robótica con fines educativos. Además, según muchos investigadores y expertos afirman que su enseñanza puede llevar a adquirir beneficios adicionales. Por ejemplo, según Eguchi, "la robótica educativa se considera una herramienta para el avance del pensamiento computacional, la codificación y la ingeniería" [4].

[1] El control de la motricidad fina es la coordinación de músculos, huesos y nervios para producir movimientos pequeños y precisos.

En el artículo *The case of prospective teachers' integration of coding-robotics practices into science teaching with STEM approach* [7], se habla también del empleo de la codificación y la robótica como herramientas de enseñanza, especialmente en educación científica al enseñarlas con un enfoque STEM que es el acrónimo de *Science, Technology, Engineering and Mathematics*. Además, se habla de que este tipo de enseñanza basada en robótica ayuda a los escolares a concretar conceptos abstractos, unificar múltiples disciplinas, asegura el aprendizaje mezclando teoría y práctica y proporciona diversión y un entorno de aprendizaje motivador. También menciona la perspectiva de los profesores al adoptar la robótica como materia educativa: "los docentes consideraron que la robótica era una herramienta eficaz para prepararse para el siglo XXI y para mejorar el trabajo en equipo, las habilidades de comunicación, las habilidades sociales, la resolución de problemas y el pensamiento crítico". De igual forma, se demuestra tras el estudio expuesto en dicho artículo que los docentes, especialmente los de primaria, deben recibir formación en nuevos campos educativos como la robótica y emplearla para mejorar las habilidades de sus alumnos en campos STEM.

Aunque haya multitud de artículos y estudios publicados que demuestran todo esto, aún no se ha consolidado esta asignatura como materia obligatoria en las enseñanzas primaria y secundaria de España a pesar de que los escolares hoy en día están estrechamente vinculados a esta disciplina desde muy pequeños. Aunque es destacable que el interés en la integración de la robótica en entornos educativos es obvio. Como dice Benitti en su artículo, "desde el jardín de infancia hasta la escuela secundaria y las actividades extraescolares, los profesionales del aprendizaje parecen adoptar la robótica educativa para enseñar diversas materias y dominios" [2].

2.1. Proceso de creación de los robots educativos

Según la RAE un *robot* es una máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones. Pero quizás, un concepto más completo para lo que se pretende enseñar en RE es el que ofrece por ejemplo Baturone en su libro *Robótica: Manipuladores y Robots Móviles* [1]: "los robots son máquinas en las que se integran componentes mecánicos, eléctricos, electrónicos y de comunicación, y dotadas de un sistema informático para su control en tiempo real, percepción del entorno y programación". Por su parte, el proyecto TERECoP (Teacher Education in Robotics – Enhanced Constructivist Pedagogical Methods), un proyecto pionero en Robótica Educativa, define robot como "una máquina inteligente, implementada mediante un sistema eléctrico y mecánico que puede ser programada para emular acciones humanas" [12].Definido este término, un robot educativo será un mecanismo físico programable, capaz de moverse con cierto grado de autonomía dentro de un entorno educativo para realizar tareas didácticas. Considerando esto, se infieren tres fases en la construcción de los robots educativos: construir el mecanismo físico, programarlo para darle cierto nivel de autonomía y comprobar que realiza las tareas [12].

- Fase de construcción: construcción de la estructura mecánica del robot y diseño de la electrónica (sensores, actuadores y computadora o microcontrolador) que va integrada en él.
- Fase de programación: fase en la que los escolares deberán llevar a cabo un programa para que su robot lleve a cabo las tareas deseadas, dotándolo así de cierto grado de autonomía.
- Fase de prueba: al unir *hardware* y *software*, es frecuente que el robot no funcione perfectamente a la primera, por ello, hay que realizar diversas pruebas para comprobar donde están los errores y solucionarlos de forma que, se estimula en los estudiantes las capacidades de análisis de problemas y dar soluciones.

2.2. Recursos para enseñar Robótica Educativa

Para aprender robótica es necesaria una plataforma robótica. Quizás, a principios de los años 2000, podía ser más complicado tener disponibles plataformas robóticas para todos los niveles de educación, ya que estaban empezando a salir al mercado. Sin embargo, hoy en día, hay multitud de recursos para enseñar robótica. Se puede realizar la siguiente distinción [12]:

- PR cerradas: su construcción está limitada a uno o algunos modelos específicos.
- PR abiertas: proporcionan un conjunto de componentes que se puede utilizar para diseñar y crear tantos modelos como la imaginación del estudiante le permita.

Además, según las área que abarquen, las PR pueden clasificare en:

- EIM (Electrónica, Informática y Mecánica): el alumno aplica la robótica de forma integral al tocar las tres áreas.
- IM (Informática y Mecánica): el alumno cuenta con piezas prediseñadas de fácil conexión para construir su robot, por lo que no es necesario tener muchos conceptos electrónicos.
- I (Informática): robots con una morfología determinada siendo su uso principal el desarrollo del Pensamiento Computacional.

Cabe destacar, que en los últimos años gracias al auge de las impresoras 3D se están desarrollando multitud de plataformas robóticas del tipo EIM, conocidas como robots imprimibles. En ese tipo de robots, se lleva a cabo todo el diseño de todas sus piezas o parte de ellas y se imprimen con las impresoras 3D. A continuación se decide la electrónica que se encargará de controlar el robot, normalmente *Arduino*, *GoGo Board* o *micro:bit* y los sensores y actuadores que permitirán al robot interactuar con el entorno. Y finalmente, se lleva a cabo toda la programación que permitirá darle a robot cierto grado de autonomía.

2.3. Robots educativos en el mercado

A continuación, se exponen algunos ejemplos de PR que hay en el mercado actual:

- Bee-Bot: ha sido diseñado especialmente para niños pequeños, de 3 a 7 años. Se trata de un robot con forma de abeja que posee 7 botones para programarlo con un máximo de 40 instrucciones de tipo: avanza, retrocede, gira a la izquierda y gira a la derecha.
- Blue Bot: Blue Bot se considera el hermano mayor de Bee Bot, pues en apariencia son prácticamente iguales pero a nivel interno posee una electrónica algo más compleja. A diferencia de Bee Bot, incluye Bluetooth que permite que se pueda programar desde móviles o ordenadores a través de su app disponible tanto para Android como para iOS.
- LEGO Mindstorms: Se trata de una línea de robots para niños fabricada por LEGO que permiten, por lo general, que los niños aprendan nociones de mecánica e informática sin necesidad de conocimientos electrónicos. Estos robots constan de un bloque inteligente desde donde se controla el robot y, a menudo, de una serie de sensores (de luz, de temperatura, de contacto, de giro o ultrasónico) que vienen encapsulados en unas piezas similares a las LEGO. Los microcontroladores Minsdtorms vienen con un software de programación basado en GUI de Robolab, un entorno de programación gráfico. Estos recursos LEGO están especialmente destinados a niños de 6 a 16 años.

3. Propuesta de Plataforma Robótica: EduROLO

Al comenzar con la parte de diseño para la propuesta de plataforma robótica de este Trabajo de Fin de Máster, se optó por llevar a cabo un robot Lego. Esta decisión se basa sobre todo en la importancia de fomentar la capacidad cognitiva humana de la creatividad en los niños, pues es uno de los elementos esenciales a tener en cuenta dentro de los procesos de enseñanza en el sistema escolar. Además, algunos estudios en los que las plataformas robóticas educativas de Lego se utilizan como actividad docente, muestran que estos conjuntos educativos de Lego afectan positivamente las habilidades del proceso académico de los estudiantes; su creatividad académica; y su actitud hacia los cursos de ciencias [7].

La creatividad es algo que poseemos todas las personas, es la capacidad de producir algo cuando surge un problema o una misión en un ámbito específico. Esta capacidad se aviva gracias a una serie de actitudes y habilidades que los niños por su inocencia o poca experiencia las tienen bastante más desarrolladas que los adultos:

- Espontaneidad e improvisación
- Afrontar los propios errores
- Actitud lúdica
- Curiosidad

Teniendo en cuenta todo esto, se ha desarrollado EduROLO, un robot Lego orientado a la educación cuyo diseño ha sido meticulosamente pensado para cumplir con todos estos puntos a conseguir: el desarrollo de la creatividad, que sea flexible en cuanto a diseño y que ayude a aprender Robótica Educativa.

3.1. Diseño mecánico de EduROLO

EduROLO es un robot de piezas Lego imprimible. El diseño de las piezas que lo conforman han sido elaboradas a partir de Solidworks, software de diseño asistido por ordenador (CAD) 3D. Con este software puede elaborarse de forma bastante detallada cada una de las piezas del robot y posteriormente permite unir todas estas piezas en un mismo ensamblaje de forma que puede verse el diseño final completo de EduROLO (figura 1. Además permite multitud de soluciones para abarcar diferentes aspectos del desarrollo del producto: diseño, simulación, estudio de propiedades físicas, obtención de croquis, etc.

Elaboradas cada una de las piezas que forman a EduROLO se procede a la

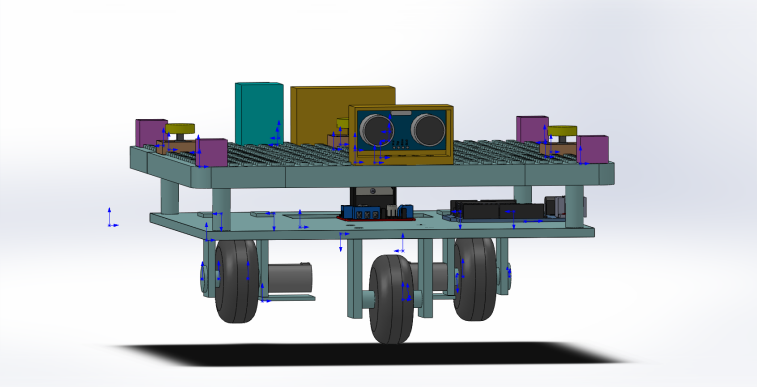


Figura 1: Diseño de EduROLO.

impresión 3D de cada una de ellas. Para la impresión del robot se ha empleado una impresora TEVO Tornado y plástico PLA. El diseño de las piezas del robot se ha elaborado siguiendo las medidas oficiales de las piezas Lego con un diámetros en sus círculo de 5 milímetros. Debido a la necesidad de conectar los sensores a una serie de cables que conectan al Arduino, se ha incorporado pasta de estaño en los orificios necesarios tanto de las piezas Lego como de la base para que al poner las piezas hagan contacto y los sensores queden conectados. Este, ha sido un paso fundamental en el diseño del robot, pues era necesario tener en cuenta que un niño de 7 años que está empezando a aprender, puede conectar las piezas de los sensores como quiera. Por consiguiente, podría conectar el pin de 5 voltio al de masa o el de masa al pin digital de Arduino, etc. Para evitarlo, se tomó la decisión de hacer los anclajes de las piezas de los sensores con una forma asimétrica para que sólo haya una forma de conectar esos sensores dentro de la base Lego de soporte de EduROLO.

Finalmente destacar que, era necesario un robot con una plataforma grande ya que una de los objetivos iniciales con este robot era dar libertad de creatividad en el diseño del robot a los niños para fomentar esta cualidad tan característica

de los escolares y que va desapareciendo en la vida adulta.

3.2. Hardware de EduROLO

EduROLO es un robot basado en Arduino. También consta de actuadores, concretamente de dos motores DC que convierten la energía eléctrica que les llega desde el Arduino en un movimiento rotatorio que acciona las ruedas motrices del robot, permitiendo de esta forma que el robot pueda desplazarse según sea su configuración en el lenguaje de programación. La tercera rueda del robot es una "rueda caster" simple, sin tracción. Cabe destacar que los dos motores de corriente continua están controlados por el *driver* L298N, que se trata de un puente H Dual.

Otro subsistema que forma parte de EduROLO es el sensórico. Para EduROLO se han incorporado hasta cuatro tipo de sensores diferentes que explicaré a continuación:

- En primer lugar, se han incorporado hasta cuatro botones o pulsadores. Estos dispositivos son interruptores capaces de detectar la presión ejercida sobre ellos para controlar alguna acción que se desea que haga el robot. Los cuatro botones que incorpora EduROLO poseen una resistencia de *pull-down* y esto habrá que tenerlo en cuenta a la hora de realizar el código de programación, pues se tratará de un botón normalmente cerrado que se activa solo al presionarlo.
- También está equipado por un sensor de distancia de ultrasonidos denominado HC-SR04. Observando su *datasheet*[2], sabemos que se trata de un sensor capaz de medir distancias de entre 2cm y 4m.
- El último sensor que se ha incorporado a EduROLO es el DHT11. Se trata de un sensor de temperatura y humedad que consta de sólo tres pines: alimentación (entre 3,5 y 5V según *datasheet*), masa y un pin digital que va a Arduino para enviar la información obtenida del entorno. En el caso de EduROLO, se emplea el módulo CNT5 que es una pequeña PCB[3] que incorpora este sensor DHT11 con la resistencia de *pull-up* de 5KΩ que recomienda su *datasheet*. Cabe destacar que este sensor es capaz de detectar temperaturas de entre 0 y 50 ^{o}C con un margen de error de $\pm 2^{o}$C.

Además de todos estos sensores también se ha implantado la posibilidad de poner 4 LEDs en EduROLO para que los niños puedan activarlos para diversas funciones, según su imaginación les permita.

Hay multitud de formas de alimentar al sistema electrónico de EduROLO: pilas, baterías... El Arduino puede alimentarse con el ordenador gracias al cable

[2] Hoja técnica de un componente electrónico que posee todas sus características de diseño y de funcionamiento, para que el usuario sepa cómo trabaja dicho componente.
[3] *Printed Circuit Board*

USB que lo conecta, sin embargo, si optamos por este sistema de alimentación EduROLO siempre estaría conectado por un cable privándole así de cierta libertad de movimiento, por lo que esta opción se descartó rápidamente. La opción elegida ha sido el uso de una *power bank*, es decir, una batería externa portátil con una salida USB de 5V y 1A.

3.3. Software para EduROLO

Para que los alumnos puedan programar sus robots EduROLO, se usarán dos lenguajes de programación diferentes, según esté el trabajo orientado a los alumnos de Primaria a Secundaria. Para los alumnos de Primaria, se recomienda el uso de S4A (*Scratch for Arduino*) que facilita una programación por bloques. Cabe destacar que para poder cargar este tipo de código al Arduino, es necesario establecer previamente la comunicación SPI con Arduino. Para ello hay que descargarse el código de la página web `http://s4a.cat/index_es.html` y cargarlo en el IDE de Arduino. Para los alumnos de Secundaria, se introducirá el lenguaje de programación de tipo textual con el propio IDE de Arduino. Al haber trabajado previamente con S4A, les será más sencillo entender este entorno.

4. Aplicación de EduROLO en el sistema educativo

Muchos conceptos relacionados con EduROLO son totalmente desconocidos para los niños, a pesar de emplearlos o usarlos en su día a día, es decir, todo niño de 6 años sabe apagar y encender las luces de sus casas, sin embargo, ¿saben verdaderamente cuál es el mecanismo del interruptor que las enciende y apaga?

Para esta sección se ha recurrido a artículos del ámbito de la pedagogía [6] [23] [10] [9] [8]. A partir de ellos, se ha extraído información fundamental sobre una de las Teorías del Desarrollo Cognitivo más importantes en la que se basa el plan de enseñanza de EduROLO. Dentro de esta Teoría, pueden distinguirse cuatro etapas entre los niños de 2 a 12 años:

- Etapa sensorio-motora (desde el nacimiento a los 2 años): se considera la etapa del "niño activo", relacionada con los sentidos y la acción [9], en la cual comienza a interactuar con el medio que le rodea, a manipular objetos y a percibir con los sentidos.
- Etapa preoperacional (niños de 2 a 7 años): etapa del "niño intuitivo". Comienzan a usar palabras y a reconocer símbolos (gestos, imágenes...) con los que representan objetos o cosas reales.
- Etapa de operaciones concretas (desde los 7 a los 11 años): se trata de la etapa del "niño práctico" que aprende operaciones lógicas de seriación, de clasificación y de conservación. Empiezan a utilizar operaciones mentales y la lógica para reflexionar sobre sucesos de la vida real [9].

- Etapa de operaciones formales (en niños de 12 años en adelante): los niños y adolescentes de estas edades se clasifican como "reflexivos", pues empiezan a ser capaces de aprender sistemas abstractos que les permite usar el razonamiento científico y proporcional.

4.1. EduROLO en Educación Primaria

En el primer ciclo de Educación Primaria, hay que tener en cuenta que los alumnos tienen entre 6 y 8 años. Es la edad en la que los niños ya saben reconocer ciertos símbolos y palabras, expresan sus ideas por medio de dibujos y se considera la edad de oro del "juego simbólico" [9] y la creatividad. Sin embargo, hay que tener en cuenta que los alumnos a estas edades aún no saben leer ni escribir con destreza. Por todo esto, son una edades que sirven para tener un primer contacto con EduROLO en forma de juego, es decir, darle la base del robot y piezas Lego y que su creatividad les lleve a crear con ello lo que más le guste, un coche, un animal, un muñeco... Introducir poco a poco algunas de las piezas que contienen los botones o los LEDs, podría ser interesante después de estar familiarizados con la PR.

Los alumnos del segundo ciclo de enseñanza Primaria se encuentran ya inmersos en la etapa del "niño práctico". A estas edades, son capaces de ordenar o clasificar objetos de forma lógica por su tamaño, color o cualquier otra relación de semejanza. Además, según Piaget la capacidad de razonar problemas de conservación es una de las principales característica de esta etapa, y gracias a ellos empiezan a adquirir términos como longitud, volumen o masa. Por esto, son capaces de entender conceptos numéricos, de tiempo y de medición. [9]. Esto permitirá que se pueda empezar a introducir conceptos más técnicos de los sensores que posee EduROLO o de programación, como los que se exponen a continuación:

- En primer lugar, podrá enseñarse el concepto de sensor digital, pues serán capaces de entender que pueden tener dos estados: encendido o apagado.
- Podrán enseñarse conceptos electrónicos básicos (LED e interruptor). Se podrá empezar a introducir el sensor DHT11 para la medición de temperatura y humedad, ya que serán conceptos que les será fácil de entender a estas edades por sus capacidades cognitivas.
- Se introduce el concepto del motor, aunque el puente H aún será algo complejo para ellos.
- Serán capaces de dar sus primeros pasos en programación usando *Scratch for Arduino*.

En los escolares del tercer ciclo de primaria (11 y 12 años), comienza a crearse el pensamiento coherente que les permite entender las relaciones conceptuales, ya que empiezan a dejar la etapa de operaciones concretas y empiezan la de operaciones formales. Se les puede introducir nuevas lecciones como las siguientes:

- Hasta entonces sólo habrán trabajado con pines digitales, a partir del tercer ciclo de Primaria es recomendable explicar los pines analógicos de entrada y salida.
- Conocidos todos los pines de la placa Arduino, empezarán a ver qué es el puente H y qué puede conseguirse con él en EduROLO.
- Al incorporar el concepto de puente H y el L298N, tendrán que desarrollar lenguajes de programación en *Scratch* algo más largos y que requieren de haber entendido correctamente los conceptos electrónicos. Por ello, se fomentará esa capacidad de relación conceptual que desarrollan los alumnos a estas edades enlazando términos de electrónica e informática, empezando a ver así como se complementan la una con la otra.
- Concepto de PWM (*pulse-width modulation*).

Sería interesante introducir a EduROLO en el aula como parte de trabajo en equipo, es decir, que en grupos reducidos intenten llevar una creación para que los niños desde pequeños aprendan a trabajar en equipo, cooperar, compartir ideas con los demás y escuchar a sus compañeros.

Con el artículo de Romero Carrasquero y Tapia Luzardo [23] en el que se comprueba con un caso práctico las habilidades cognitivas desarrolladas al final de la Educación Primaria, se obtiene que los niños al finalizar esta etapa escolar, tienen finalizadas las tres fases de adquisición: comprensión, retención y transformación. Sin embargo, las habilidades de descripción y relación no tuvieron tan buenos resultados [23]. Por ello, el hecho de trabajar en las aulas con EduROLO donde se combina una enseñanza teórico-práctica de conceptos en mecánica, electrónica e informática principalmente, puede ayudar a mejorar esas dos habilidades de descripción y relación al ver como ponen en práctica lo que han aprendido en las lecciones teóricas.

4.2. EduROLO en Educación Secundaria

Según Piaget, los adolescentes se encuentran en el estadio de operaciones formales, donde ya tienen criterio y capacidades de seriación, clasificación y conservación. En esta etapa, son capaces de desarrollar operaciones mentales de más complejidad y poseen un pensamiento abstracto [10], lo que les lleva a tener ideas que no se basan en la realidad, dejando atrás el razonamiento lógico y basado en lo real que caracteriza las etapas anteriores del desarrollo cognitivo. A partir de estas edades, los adolescentes empiezan a poseer una serie de operaciones mentales que se corresponde a cierto tipo de operación lógica denominada lógica proposicional [9], es decir, son capaces de extraer una conclusión lógica a partir de dos afirmaciones antecedentes. Esto, junto al desarrollo de un pensamiento más científico, les ayudará a iniciarse con EduROLO de una forma algo más compleja que en Primaria.

En los primeros cursos de Secundaria (1º y 2º), el desarrollo cognitivo de esta etapa comprende el comienzo del surgimiento del pensamiento abstracto

o formal, por ello, la toma de decisiones empieza a involucrar habilidades más complejas, que son esenciales para la creatividad y el rendimiento académico de un nivel superior [6]. Podrán iniciarse con un entorno de programación que emplee lenguaje textual usando el propio IDE de Arduino, como se comentó anteriormente en otras secciones.

En la adolescencia media, cuando tienen entre los 14 y 15 años de edad, mejoran las habilidades de pensamiento abstracto y razonamiento y la creatividad. El adolescente ya no se conforma con lo que dice la norma hasta conocer el principio que la rige [6]. Esto puede hacerlos más ambiciosos y el docente podrá aprovechar para enseñarles nuevos conceptos como por ejemplo el sensor ultrasónico, el último que se ha incorporado hasta el momento en EduROLO. Al conocer ya todos los componentes electrónicos que posee esta plataforma robótica, se les puede empezar a dar libertad para crear y así analizar hasta donde puede llevarlos su imaginación y conocimientos.

Finalmente, en la adolescencia tardía, aproximadamente a los 16 años, los alumnos ya poseerán todos los conocimientos necesarios para trabajar en un alto nivel con EduROLO. A estas edades aumenta la capacidad de resolución de problemas [6] y con ello, podrán enseñarse otros sensores que no posea EduROLO y retarlos a integrarlos de alguna forma en la plataforma robótica, gracias a la flexibilidad en el montaje que ofrece EduROLO.

5. Conclusiones

Son innumerable los robots educativos que han salido al mercado en los último años, sobretodo, desde el 2014. Al mismo tiempo, dada esta gran variedad de plataformas robóticas y la multitud de tecnologías que nos rodean, es destacable lo poco integrados que están en los Colegios de Educación Primaria y Secundaria e Institutos una asignatura como Robótica. Llama la atención que los niños de 6 años sepan manejar a la perfección un teléfono móvil, hablar con un asistente de voz o activar un robot aspiradora y no tengan ningún tipo de conocimiento tecnológico. Además, es indudable que la Robótica ha llegado a multitud de campos para quedarse, como por ejemplo, en Medicina, con el robot Da Vinci o en Industria con los cobots[4]. Por todo esto, sería un acierto empezar a insertar esta enseñanza en las aulas, para preparar desde pequeños a los niños a trabajar con robots ya que, seguramente, dentro de 10, 15 o incluso menos años, puede que en un gran porcentaje de puestos de trabajo deban saber trabajar codo con codo con algún tipo de robot.

Otro aspecto importante a destacar tras la elaboración de este trabajo es que si finalmente la Robótica o alguna asignatura de ciencias tecnológicas se integra próximamente en las aulas, sería conveniente llevar a cabo algún curso u otro

[4] Brazo robótico creado para trabajar junto a los humanos en una cadena de producción

tipo de preparación para los docentes.

A pesar de todas las ventajas que puede tener el incorporar la Robótica en los centros educativos, hay algunos inconvenientes. En primer lugar, a pesar de la gran variedad que hay ya hoy en día de robots para la enseñanza, la inversión económica que debe hacerse es considerable, por ejemplo, el famoso *Lego Mindstorms* tiene un valor aproximado de 480 euros. No obstante, con propuestas de robots imprimibles como la de este trabajo, se muestra que hay formas más económicas de incorporar una nueva rama a la Educación que preparará mucho mejor a los alumnos para la vida futura y los cambio tecnológicos que quedan por llegar.

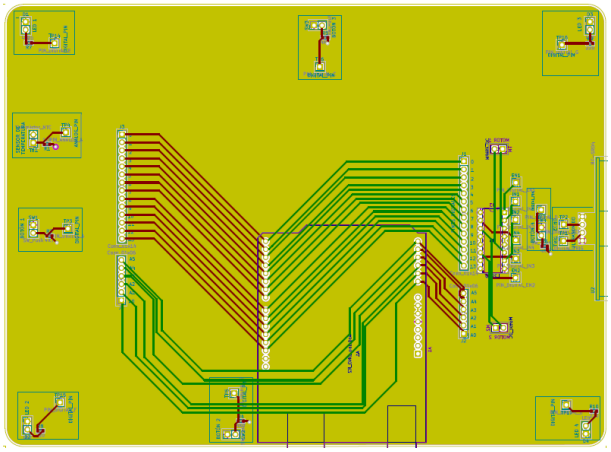


Figura 2: PCB para mejorar el diseño electrónico de EduROLO.

Al pensar el diseño de EduROLO y, especialmente, a la hora de llevarlo a cabo, han sido algunas las ideas que me han surgido como propuestas de mejora de esta plataforma robótica educativa:

- En primer lugar, se podría estudiar e investigar el poder incorporar más tipo de sensores al robot, especialmente de lectura analógica, ya que de los sensores propuestos ninguno necesita los pines de entrada analógica y por las cualidades y capacidades de los niños de Educación Secundaria, considero que podría ser una nueva lección a incorporar en líneas futuras de mejora.
- El hecho de querer conectar todos los sensores en Arduino o incluso incorporar alguno más es una de las principales limitaciones que presenta EduROLO al usar el hardware Arduino que tiene tan solo 14 pines digitales. Por ello, sería interesante estudiar la posibilidad de usar otro microcontrolador que ofrezca más pines para conectar más sensores pero a la vez que permita ser programado con *Scratch* y el IDE de Arduino ya que considero que son dos

entornos de programación fáciles de usar y acordes a los niveles de enseñanza en los que se va a enseñar. Alguno microcontrolador interesante a usar puede ser el ESP32 que además permite conectividad Wi-Fi y Bluethoot por si quiere incorporarse algún tipo de aplicación para controlar al robot con un dispositivo móvil.

- Al llevar a cabo el prototipo físico de EduROLO, se han soldado cables a los pines de los sensores ya que uno de los propósitos a lograr con este robot es que los alumnos se aproximen tanto a la mecánica como a la electrónica y la informática. Por ello, se propone el diseño de una PCB (figura 2) que incluye muchas de las conexiones hechas de forma que se eliminen los cables excesivamente largos y con ello que quede un diseño bastante más limpio y ordenado.

Referencias

1. Aníbal Ollero Baturone. *Robótica: manipuladores y robots móviles*. Marcombo, 2005.
2. Fabiane Barreto Vavassori Benitti. Exploring the educational potential of robotics in schools: A systematic review. *Computers and Education*, 58(3):978–988, 2012.
3. Marina Umaschi Bers. The TangibleK Robotics Program: Applied Computational Thinking for Young Children, 2010.
4. Amy Eguchi. Educational robotics for promoting 21st century skills. *Journal of Automation Mobile Robotics and Intelligent Systems*, 8(1):5–11, 2014.
5. Gabriela Krumm, Jael Vargas Rubilar, Viviana Lemos y Laura Oros. Percepción de la creatividad en niños, padres y pares. *Pensamiento Psicológico*, 13(2):21–32, 2015.
6. Verónica Gaete. Desarrollo psicosocial del adolescente. *Revista chilena de pediatría*, 86(6):436–443, 2015.
7. Gülfem Muslu Kaygisiz, Özlem Üzümcü, Melike Uçar. The case of prospective teachers' integration of coding-robotics practices into science teaching with stem approach. *Elementary Education Online*, 19(3):1200–1213, 2020.
8. Iveth Moreno, Lilia Muñoz, José Rolando Serracín, Jacqueline Quintero, Kathia Pittí Patiño y Juan Quiel. La robótica educativa, una herramienta para la enseñanza-aprendizaje de las ciencias y las tecnologías. *Teoría de la Educación. Educación y Cultura en la Sociedad de la Información*, 13(2):74–90, 2012.
9. Aurèlia Rafael Linares. Desarrollo cognitivo: Las teorías de Piaget y de Vygotsky. Master's thesis, Universidad Autónoma de Barcelona, 2007-2008.
10. Pierre Mounoud. El desarrollo cognitivo del niño: desde los descubrimientos de Piaget hasta las investigaciones actuales. *Contextos educativos*, 4:53–77, 2001.
11. Nelson David Muñoz, Carlos Andrés Andrade y Nelson Londoño Ospina. Diseño y construcción de un robot móvil orientado a la enseñanza e investigación. *Ingeniería y Desarrollo-Universidad del Norte*, (19):114–127, 2006.
12. Kathia Pittí Patiño. Caracterización de Entornos de Aprendizaje basados en Robótica en el ámbito preuniversitario de Iberoamérica y España. Master's thesis, Universidad de Salamanca, 2021.
13. Liliana Patricia Quiroga. La robótica: otra forma de aprender. *Revista de Educación y Pensamiento*, 25:51–64, 2018.
14. Sergi Resa Blanquez, Ernesto Nogueira Rodríguez, Ramón Martínez López. *1 ESO. Tecnología, programación y robótica.* Editorial Teide, S.A., 2016.

15. Sergi Resa Blanquez, Ernesto Nogueira Rodríguez, Ramón Martínez López. *2 ESO. Tecnología, programación y robótica.* Editorial Teide, S.A., 2016.
16. Sergi Resa Blanquez, Ernesto Nogueira Rodríguez, Ramón Martínez López. *3 ESO. Tecnología, programación y robótica.* Editorial Teide, S.A., 2016.
17. William Bojorquez Neyra. *1º de Primaria. Robótica Educativa: máquinas simples.* Editorial Libru, S.A.C., 2019.
18. William Bojorquez Neyra. *2º de Primaria. Robótica Educativa: máquinas simples.* Editorial Libru, S.A.C., 2019.
19. William Bojorquez Neyra. *3º de Primaria. Robótica Educativa: WeDo 2.0.* Editorial Libru, S.A.C., 2019.
20. William Bojorquez Neyra. *4º de Primaria. Robótica Educativa: WeDo 2.0.* Editorial Libru, S.A.C., 2019.
21. William Bojorquez Neyra. *5º de Primaria. Robótica Educativa: WeDo 2.0.* Editorial Libru, S.A.C., 2019.
22. William Bojorquez Neyra. *6º de Primaria. Robótica Educativa: WeDo 2.0.* Editorial Libru, S.A.C., 2019.
23. Yanice Romero Carrasquero y Fernando Tapia Luzardo. Desarrollo de las habilidades cognitivas en niños de edad escolar. *Multiciencias*, 14(3):297–303, 2014.
24. Yen Air Caballero-González and Ana García-Valcárcel. ¿Aprender con robótica en Educación Primaria? : un medio de estimular el pensamiento computacional. *Education in the Knowledge Society*, 21(21):15–15, 2020.

Autogenerated Human Machine Interface for supervision and control

Tobias Tønnessen, Mario Francisco Sutil, and Pastora Isabel Vega Cruz

Departamento de Informática y Automática
Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
idu20091@usal.es
https://www.usal.es/

Resumen This paper presents a new approach to developing Human Machine Interfaces (HMI) for control systems by creating an application that automatically generates a universal HMI that adapts its characteristics to the system to be supervised and controlled. The graphical user interface is flexible depending on the number of inputs and outputs, and it can communicate with a Programmable Logic Controller (PLC) that is located remotely. Further, it describes how such a system can be implemented and potential real-world use for such a system. Furthermore, the type of system proposed shows great potential in testing hardware, monitoring, and simulating parts of a control system, especially when the hardware is unavailable. Finally, the system's modularity makes it possible to switch between different system types, personalize the user interface or extend the capabilities to support additional custom functionality.

Keywords: Autogenerated, Adaptable, Control System, HMI, PLC

1. Introduction

Human Machine Interfaces (HMI) are an essential part of any control system. An HMI is a device or interface that allows the user to interact with a physical system. It helps the operator to monitor and visualize the process to make critical decisions. The definition of an HMI includes everything from a button to a graphical user interface (GUI) as long as they are used to control a machine or piece of equipment. The complexity of the HMI can vary profoundly. Where some HMIs are standardized for a specific product, others combine multiple systems into one cohesive application so that the operator can get an overview of the entire production from one centralized location. These more complex systems, where many different devices are integrated into one system, require a lot of custom logic. The development of such systems can take a long time and needs rigorous testing.

HMIs are also a vital part of system testing to validate that the system works as intended. Testing the different parts of the system using the HMI itself makes

it possible to validate everything from the software to the wiring of the equipment. However this requires the HMI to be ready, to test critical components, which is not always possible. A developer might have to rush to create a minimalistic HMI that is good enough for testing but not for the final product. The developer loses critical time that could otherwise be used to complete the final product or other projects.

Another challenge is that the systems can change throughout development. New requirements can be added or equipment renamed. Such changes add complexity to the development process, where the user interface (UI) must be kept up to date with the latest changes to the control system. A minor change in a name might lead to hours of work to implement the changes correctly. Testing may have to be postponed and projects delayed.

This paper proposes a new process for creating an HMI for a control system. Instead of carefully designing the HMI systems in advance, the proposed system will connect directly to a Programmable Logic Controller (PLC), an industrial computer used to control most control systems. It will read out the data structure and generate an HMI based on this. This approach gives the HMI a lot more flexibility because it adapts the UI to the system it is controlling. This allows it to monitor and control whatever control system it is connected to, be it a control loop, production plant, or any other type of control system.

2. Existing solutions

All the major PLC vendors provide programs for designing and creating production-ready HMIs. For example, Rockwell Automation has FactoryTalk View [4]. Siemens has the SIMATIC WinCC [17]. These programs integrate well into the PLC development environments and create a streamlined way of creating HMI for all types of control systems. They are excellent tools and have good integration with their specific PLC brand, but they require significant work to establish a fully working HMI. They are advanced tools requiring specialized developers with experience with control systems to both build and perform any changes to the HMI. This requirement creates a divide between the user of a system and the designer. If anything needs to be changed, the user can not perform the changes themselves. They need to communicate the information to an engineer so he can perform the changes. If the HMI is developed by another company or is located in another country, it can take a long time before new changes are implemented.

There have been several papers attempting to generate adaptive HMI systems. For example, the paper [11] proposes a way of using XML files to create the user interface. These configuration files are used to build preconfigured visualizations. This system is both modular and extendable but is limited to monitoring and parameter visualization. Another approach is described in [1] where a system was created to modernize and improve existing applications. As part of the process, a simplification of the UI is performed. By hiding unused parts of the UI, the application is simplified, which leads to a faster workflow for the end-

user. This approach has a streamlined way of modernizing existing applications, but it still requires multiple steps and is targetting enterprise applications.

A way for personalized information visualization for control and monitoring is discussed in [10]. The system is based on information received from web services. It uses the information more intelligently by tailoring the information to different users based on their position and the monitored system. This way, a manager can get an overview of the entire production, where an engineer would see more fine-grained detail about equipment. In [20] the authors attempted to generate an HMI based on the Piping and Instrumentation Diagrams (P&ID), which is a diagram that shows piping and process equipment together with the instrumentation. Their system extracts information from the P&ID, like the position and the connections between elements. This is used to generate the user interface. However, in the generated output, the alignment of these elements is not at an acceptable level. Having a person go over the generated UI and correct misalignment would greatly improve the quality of the result. When comparing the UI to one designed by a human, we see that humans usually hide or rearrange elements to improve the user experience.

A combination of autogeneration and human intervention seems to be the best approach. Using computers to generate the elements and humans to place the elements logically.

3. Implementation

Our goal was to create an application that is not hardcoded to a specific system or use case but can adapt to the user's needs. If additional functionalities are required in the future, a developer should be able to add them to the system with ease. This approach should extend throughout the entire architecture of the application. To achieve this goal, our Adaptive Control System (ACS) was separated into two main parts: a Universal HMI (UHMI) and a *PLC reader*. The UHMI is the main application with which the operator interacts, while the *PLC reader* handles the communication with the PLC.

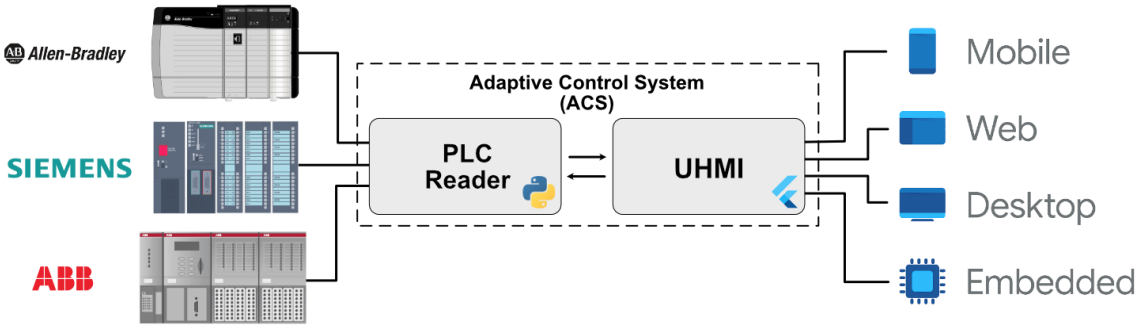


Figura 1: System architecture

Splitting these two tasks into two different applications makes the ACS very flexible because the applications can run on two different devices. Since each part handles a different set of tasks, the same framework and programming language need not be used for both. Instead, more specialized frameworks that are better suited for each task can be used. In this case, Flutter [7] has been chosen to build the UHMI application and Python [21] for the PLC reader. Flutter is a cross-platform framework, making it possible to run the UHMI on any operating system or device. The program reading and writing to the PLC must connect directly to the PLC, but the UHMI can be free to move around, making it possible to run the application on mobile devices such as an iPad. In addition, the UHMI can be located anywhere in the world, as long as it has a connection to the PLC reader.

3.1. Project-specific input

The use of only the data read from PLC directly without any project-specific input from the operator has a limited use case. Even though the operator can manipulate individual inputs and output, the operator has no visual feedback in the UI to see what the signal represents because the PLC typically only handles numerical values and not strings. The tag's description cannot be read from the PLC to know what it represents, only the tag number. The operator can continue without a project-specific input, but they need to refer to the documentation at all times. This issue can be solved by identifying the minimum project-specific inputs needed to create a good and usable HMI system using a document that is already part of any control system. This avoids introducing yet another document that the operator needs to keep updated and improves the possibility of creating a product that is usable in any system.

One document that is part of any control system is an IO list. This document describes all inputs and outputs used in the project and additional information such as tag description and sensor ranges. This document should always be up to date, and by using this document as an input, we can make sure that the information in our UHMI application is also up to date.

To include information from both the PLC and the IO list in the UI, the *PLC reader* parses the content of the IO list and combines it with the information read from the PLC (fig. 2) when it first connects. Even though IO lists are part of any type of control system, there is no defined standard for this document, so the IO list's content and layout can vary dramatically. The IO list parser is, therefore, made to be adaptable.

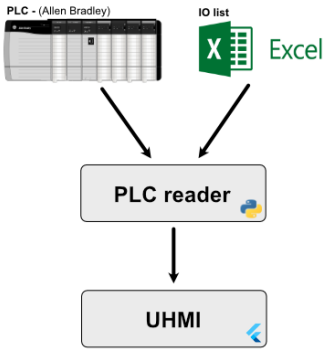


Figura 2: Combining data

A good UI requires the extraction of a lot of information from the IO list. In table 1 the information to be extracted is defined together with the source. When combining the data from the IO list and the PLC, the *"PLC tag number"* is used, since it will be defined in both places. The slot and channel properties are only used for physical signals to show the tag's description on each IO channel.

Tabla 1: Combined content from PLC and IO list

Property	Description	Source
Writable	Determines if it's possible to write to the tag.	PLC
PLC tag number	The tag number defined on the PLC.	PLC + IO list
PLC tag type	The tag data type defined in the PLC.	PLC
HMI tag number	Tag number shown in the HMI.	IO list
HMI tag type	Used to override the default representation of the tag in the HMI.	IO list
Description	Description describing the tag.	IO list
Slot	The physical slot it is connected to, if it is a wired signal.	IO list
Channel	The physical channel it is connected to, if it is a wired signal.	IO list
Range high	The high range of the senor for analog values. Definition of a high signal for digital signals.	IO list
Range low	The low range of the senor for analog values. Definition of a low signal for digital signals.	IO list
Unit	The unit of the signal.	IO list
Signal type	Describes of a signal it is. Hardwired, 4-20 mA, Software, etc. Only used for info.	IO list
Screen	Defines what screen the tag should be presented on.	IO list
Position	Describes the x and y coordinates of the tag in the HMI.	IO list (Optional)

3.2. PLC reader

The PLC reader will handle the communication with the PLC itself. The PLC reader should support control systems with different designs and not be locked into a specific brand of PLCs. However, since not all vendors will support the same functionality, an explicit data structure must be defined for the communication between the PLC reader and the UHMI. Therefore regardless of what the PLC vendor supports, we need to convert the data read from the PLC to a standardized format that the UHMI can interpret.

All PLC manufactures have different processes for reading and writing values. Since an Allen Bradley PLC is used in this project, the open-source project pycomm3 is used for communication. An abstract class has been created to enable any developer to add support for additional PLC brands easily. This abstract class defines the methods a custom implementation needs to implement and what the return type should for be each method. As long as the implementation inherits this abstract class, it should be compatible with the UHMI application.

3.3. Universal Human Machine Interface

The UHMI application is the only part of the ACS that the user normally interacts with. This tool should be approachable for non-programmers, but they will still be required to have a good understanding of the system they are controlling. The user interface should give the operator a good overview of what is going on in the system and make it possible to operate it. When the PLC reader first connects to the PLC, a list of all the tags and input modules connected to the PLC are created. This information is used to generate the user interface. The UHMI consists of different sections to target different aspects of control systems.

The first part is the signal section, which can manipulate the physical IO signals on the PLC. This is useful when the hardware is unavailable, or the underlying software is not yet finished. The operator can manipulate any signal by simulating a value to and from the PLC itself, making it possible to perform any testing needed. This section is possible to use without an IO list, but the user will not know what tag is connected to each channel.

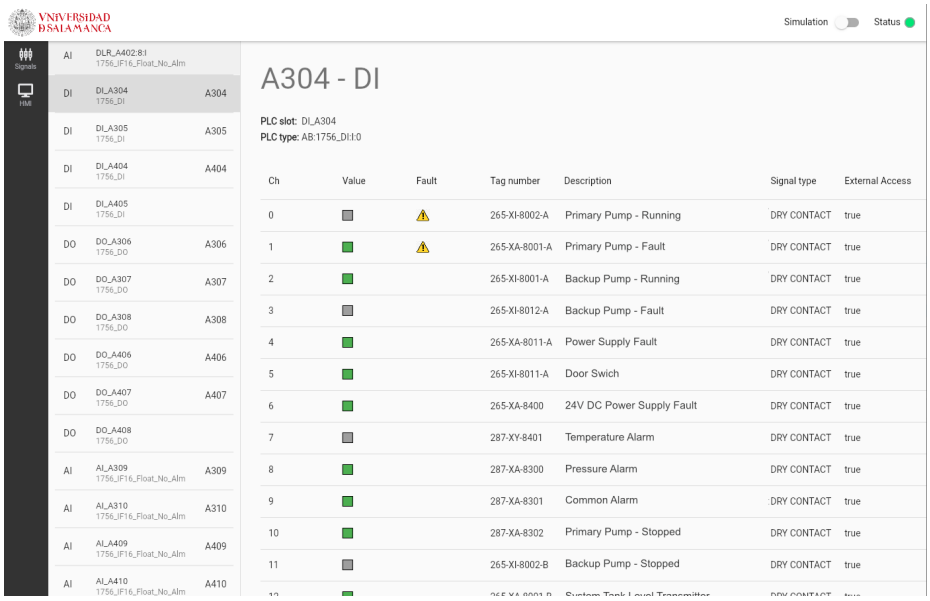


Figura 3: Signal section for digital input card

The second part, the HMI section, attempts to make a fully-fledged HMI system based on minimum input parameters and requires the use of an IO list. On this screen, the user can see what is going on in the control system and manipulate actuators, like valves and pumps. There is a tab bar on the top of the screen, allowing the user to navigate to the screen of his choice. The screen itself contains all the tags related to the screen as defined in the IO list. When the operator first connects to a new PLC, all the tags will be stacked on top of each other. Since the position of each tag on the screen is usually not part of the IO list. The operator then needs to drag each tag to a logical position. When the operator is finished manipulating the UI, he can export the new configuration to not reorder it again the next time he connects.

When the UI is all laid out, the operator can start operating the system. When a tag is selected, a bigger window appears with additional information about the tag and potential actions which the operator can perform. The visualization of this window will depend on the "PLC tag type.ºf the tag. To visualize tags with the same "PLC tag type"differently, the. "HMI tag typeçan be used to override the default behavior. If the "PLC tag typeïs not defined in UHMI, the user will be notified in the UI. Thus, prompting them to update the UHMI application.

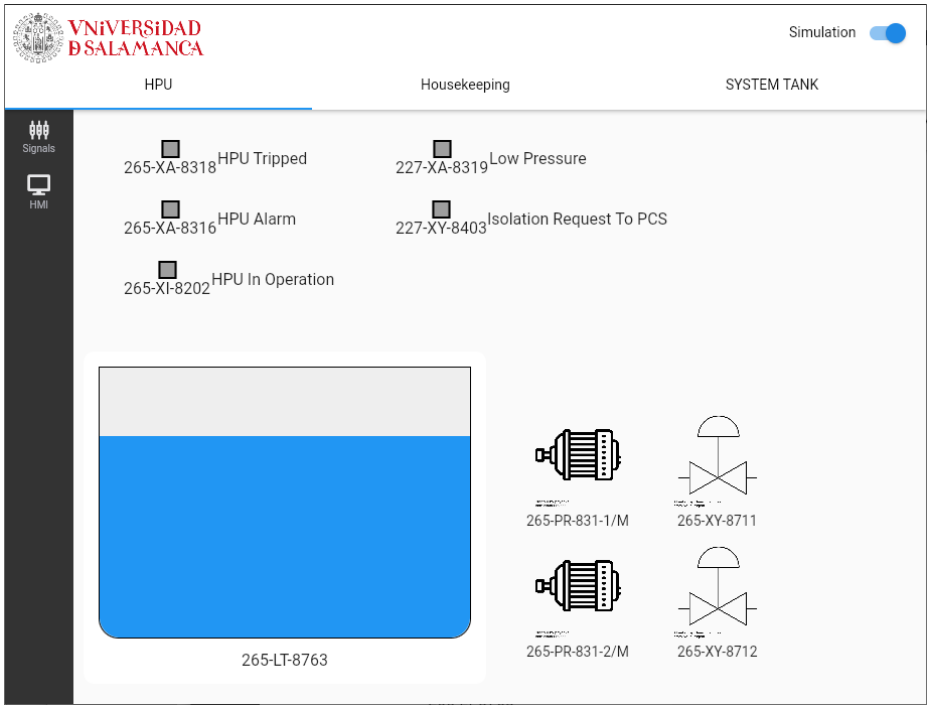


Figura 4: HMI section

4. Testing

To validate that the ACS works as intended, it has been tested on an existing system. The system consisted of multiple pumps, actuators, valves, pressure transmitters and switches. The project's IO list consists of 274 tags. The PLC includes these and, in addition, 122 local tags. To prevent damage to physical hardware, a simulator has been used to test the ACS. Even though no physical actuator was used, we still had access to the IO cards and could measure the signals with a multimeter. The testing shows that the system works and that it is fast to set up. As a result, the ACS can save much time for the end-user. When simulating a value, a latency of two seconds was observed. More testing is needed to validate that the ACS works consistently and with different hardware configurations. Nevertheless, the initial results of the testing are very promising.

5. Conclusion

This paper has resulted in the successful creation of an autogenerated HMI system. Eliminating the need to design an HMI in advance. Instead, the HMI is generated at the time of connection to the PLC. The proposed system can potentially reduce the time it takes to get a system up and running from weeks to minutes. The result of the testing on a real-world system is very promising. Furthermore, the IO signals can be manipulated without any setup, and it does not require any logic on the PLC itself to be ready. This makes it an excellent tool for testing equipment before the software is completed.

It was concluded that integration with an IO list is the best way to create a more general-purpose HMI system. This approach ensures that the information in the UI is always up to date and if any changes need to be performed. It can be done easily in the official documentation itself using Microsoft Excel [13], greatly reducing the skill level need to use the program. Users without any prior knowledge can get the software up and running and perform any changes, in stark contrast to programs usually used to develop HMI systems. The use of the project's IO list directly makes it easier to find errors in the documentation. These errors can be easily fixed, and importing the IO list again updates the HMI. This creates a fast feedback loop when encountering errors.

There are many similarities between the generated UHMI and the production HMI. However, some core features need to be added before the UHMI can be used in production. The UHMI does not have a way of showing automated sequences, where multiple devices are used in combination. The UHMI only allows for controlling individual components. There is no way of visualizing interlock conditions, where a condition must be fulfilled before something else can happen. This problem might prevent the operator from starting a motor without knowing why. The UHMI is also missing a way of handling alarms. This is harder to remedy since alarms are not part of the IO list and can be handled differently between PLC vendors and companies. Some companies handle alarms on the PLC itself, where others handle them in the HMI. By design, the ACS only

looks at real-time values. In a control system, however, it might be beneficial to see changing values over time in the form of a graph. This feature can help the operator to see tendencies and perform mitigations before an unwanted situation occur. Lastly, the response time experienced during the testing is around two seconds. If speed is essential when controlling the control system, our solution might not be sufficient.

The ACS shows great potential with many use cases. For example, when equipment is not yet ready or not available for testing, this application can manipulate hardware IOs to perform the necessary integration testing without needing access to the actual hardware. The application can also be used to show more detail than is normally part of a production HMI. The operator could use the two HMI solutions in combination and gain additional debugging capabilities without having to debug on the PLC itself. The operator can run an additional HMI on a tablet and be freer to move around. This allows the operator to be next to the equipment he is operating and still see the live values in the HMI.

5.1. Future work

Autogenerated HMI based on real-time data is a new area of research, and there is, therefore, a lot of further research that can be performed to further improve upon the system. The author has identified three main categories where more research is beneficial. These are the performance, trending and simulation.

The response time of the application is a little higher than desirable. Additional research can be done to try to minimize this latency, either by optimizing the code or by switching out a *PLC reader* entirely. The Flutter framework does support foreign function interface (FFI) [6], which allows native C APIs to be called. This can potentially dramatically speed up the latency of the system.

Implementing a data trend function would be beneficial in the UHMI. This implementation could be achieved by logging all the values read while connected to the PLC or by changing the back-end with a database.

The system has the potential of adding more advanced simulation to the HMI. It has been demonstrated that different types of PLC data types can be handled differently. If advanced simulations are defined for each PLC data type, the system can create a fully simulated system in about the same time as it takes to get the HMI fully functional. Research can also be performed to integrate external simulators, where parts of the system use machine learning models to calculate some values.

Referencias

1. Pierre A. Akiki, Arosha K. Bandara, and Yijun Yu. Integrating adaptive user interface capabilities in enterprise applications. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, page 712–723, New York, NY, USA, 2014. Association for Computing Machinery.
2. Annerose Braune, Stefan Hennig, and Torsten Schaft. Xml-based monitoring and operating for web services in automation. In *2007 5th IEEE International Conference on Industrial Informatics*, volume 2, pages 797–802, 2007.

3. T. Bray. The javascript object notation (json) data interchange format. *RFC*, 7158:1–16, 2014.
4. FactoryTalk View-HMI Software | Factory Talk United States.
5. FastAPI documentation (0.65.2). url: *https://fastapi.tiangolo.com/*. (accessed 22.06.2021).
6. Binding to native code using dart:ffi. url: *https://flutter.dev/docs/development/platform-integration/c-interop*.
7. Flutter - UI toolkit for building native crossplatform applications. url: *https://flutter.dev/*. (accessed 22.06.2021).
8. Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems (8th Edition) (What's New in Engineering)*. Pearson, 2018.
9. Karl Heinz John and Michael Tiegelkamp. *IEC 61131-3: Programming Industrial Automation Systems Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids*. Springer Publishing Company, Incorporated, 2nd edition, 2010.
10. Norma Angelica Nieto Lee, Luis E. Gonzalez Moctezuma, and Jose L. Martinez Lastra. Visualization of information in a service-oriented production control system. In *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pages 4422–4428, 2013.
11. Lilija I. Martinova, Sergey S. Sokolov, and Petr A. Nikishechkin. Tools for monitoring and parameter visualization in computer control systems of industrial robots. In Ying Tan, Yuhui Shi, Fernando Buarque, Alexander Gelbukh, Swagatam Das, and Andries Engelbrecht, editors, *Advances in Swarm and Computational Intelligence*, pages 200–207, Cham, 2015. Springer International Publishing.
12. Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
13. Microsoft Corporation. Microsoft excel. url: *https://office.microsoft.com/excel*.
14. pyads's documentation (3.3.7). url: *https://pyads.readthedocs.io/en/latest/index.html*. (accessed 22.06.2021).
15. pycomm3 documentation (1.1.1). url: *https://docs.pycomm3.dev/en/latest/*. (accessed 22.06.2021).
16. python-snap7's documentation. url: *https://python-snap7.readthedocs.io/en/latest/*. (accessed 22.06.2021).
17. Siemens. Wincc. url: *https://www.siemens.com*.
18. Steven. PLC Manufacturers: The Latest PLC Brands, Rankings & Revenues, June 2020. url: *https://ladderlogicworld.com/plc-manufacturers/*.
19. Studio 5000 Design Software | Factory Talk Puerto Rico. url: *https://www.rockwellautomation.com/en-pr/products/software/factorytalk/designsuite/studio-5000.html*.
20. Leon Urbas, Stefan Hennig, Henning Hager, Falk Doherr, and Annerose Braune. Towards context adaptive hmis in process industries. In *2011 9th IEEE International Conference on Industrial Informatics*, pages 244–249, 2011.
21. Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

Pentóminos: Interacción y Realimentación Táctil

Xuzeng Mao y José Rafael García-Bermejo Giner

Departamento de Informática y Automática
Facultad de Ciencias
Plaza de los Caídos s/n, 37008, Salamanca, España
Universidad de Salamanca
{xuzengmao,coti}@usal.es
https://www.usal.es/

Resumen La interacción y realimentación táctil es una de las ramas de la interacción gestual así como de la interacción persona-ordenador. Esta rama se ha vuelto una parte importante de la interacción gestual cuando en 2007 empezaron a salir al mercado los primeros modelos de teléfonos inteligentes. Principalmente, fue gracias al lanzamiento del iPhone de Apple en el día 29 de junio de 2007 cuando marcó una nueva era para la tecnología de las pantallas táctiles. Antes de esta fecha, los dispositivos personales más utilizados eran los ordenadores y los portátiles. Estos dispositivos no contaban con una pantalla táctil por lo que la interacción táctil y las pantallas táctiles no estaban extendidos sobre la población. Sin embargo, después de esta fecha, la población en general empezaron a utilizar los teléfonos inteligentes para muchas tareas cotidianas y una de las novedades principales de estos dispositivos era la sustitución de los botones por una pantalla táctil. En este artículo se presenta unas propuestas de interacción gestuales de la aplicación desarrollada en este trabajo. Principalmente, se propone gestos táctiles para las transformaciones de las piezas de los pentóminos.

1. Introducción

La interacción y realimentación táctil se ha convertido en los últimos años en una de las ramas más importantes de la interacción gestual así como de la interacción persona-ordenador. Para la interacción táctil es necesario un dispositivo de entrada del ordenador, la pantalla táctil. Este dispositivo muestra, a través de una pantalla, una interfaz gráfica al usuario que permite comunicar información desde el ordenador a la persona (dispositivo de salida). Por otro lado, tiene reconocedores del toque físico en la pantalla (dispositivo de entrada) que permite recoger órdenes del usuario para su posterior realimentación en la interacción. Además, como el dispositivo ofrece una pantalla permite asociar las interacciones táctiles de la persona sobre la vista o interfaz que contenga el punto de contacto.

Antes de que se inventara las pantallas táctiles, tal como lo conocemos ahora, el primer instrumento que utilizó la tecnología táctil fue Eletronic Sackbut [52]. Es un instrumento electrónico musical diseñado y construido por Hugh Le Caine

entre los años 1945-1948 [29]. Este instrumento consta un dispositivo con tecnología táctil que permite al usuario con su mano izquierda controlar el timbre. Cada dedo de la mano izquierda permite manejar un control sensible a la presión por separado.

La pantalla táctil fue descrito por primera vez por Erich Jhonson en un artículo publicado en 1965 [40]. Su trabajo describía teóricamente las pantallas táctiles capacitivas donde en 1967 publicó otro artículo más completo [41]. La idea desarrollada sólo se consideraba un toque al mismo tiempo y no detectaba la presión, solamente detectaba si ha realizado el toque o no, es decir, una información binaria. Las pantallas táctiles capacitivas constan de sensores capacitivos. Estos sensores constan de una capa aislante, en este caso el cristal, recubierto con un conductor. Por naturaleza, el cuerpo humano es también un conductor eléctrico, por lo tanto, al tocar el dedo sobre la pantalla distorsiona el campo electrostático de la pantalla (Figura 1). Los sensores son capaces de detectar la posición de toque del dedo gracias a la distorsión. La primera vez que se puso en práctica las pantallas táctiles capacitivas para su uso fue por la *Organización Europea para la Investigación Nuclear (CERN)* en su acelerador de partículas en el año 1976 [27].

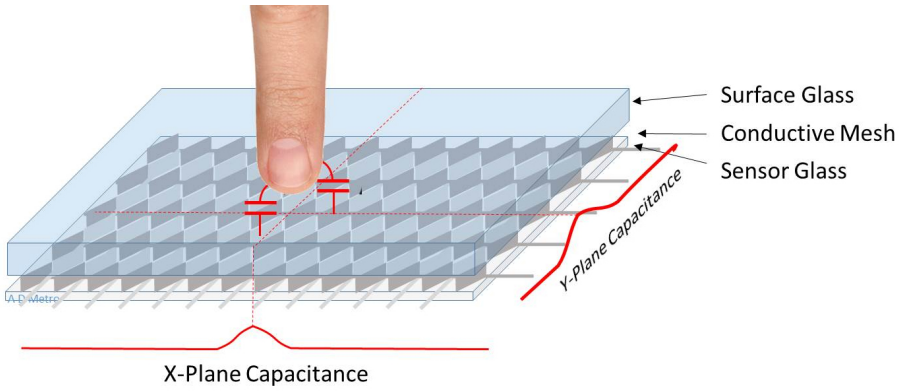


Figura 1: Pantalla táctil capacitiva (https://admetro.com)

En 1971, George Samuel Hurst [32], desarrolló la tecnología pantalla táctil resistiva, que gracias a este descubrimiento accidentado, fundó la empresa Elographics (actualmente, Elotouch) para refinar, mejorar y vender el nuevo producto. No fue hasta en 1982, cuando se produce la primera versión de la pantalla táctil resistiva. Esta pantalla táctil no funciona por la conductividad eléctrica del cuerpo humano sino por la presión que se ejerce en la pantalla haciendo que entren en contacto las dos capas conductivas separadas por un aislante (Figura 2). Por esta razón, se le llama resistiva ya que es una pantalla en que hay que ejercer más presión que una pantalla capacitiva.

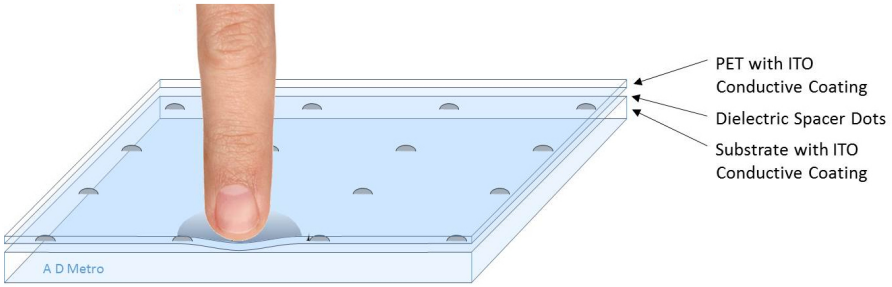


Figura 2: Pantalla táctil resistiva (https://admetro.com)

En 1972, se desarrolla el ordenador PLATO IV considerado como uno de los primeros ordenadores con pantalla táctil. Este ordenador utiliza la tecnología de pantalla táctil por infrarrojos (Figura 3). Consta de un conjunto de diodos emisores de luz y un conjunto de detectores de luz. Cuando el usuario pasa el dedo cerca de la pantalla, el dedo bloquea algunos haces de luz causando la perdida de señal en los receptores de luz. La pérdida de la señal permite al ordenador saber en qué posición está el dedo sobre la pantalla. En 1983, la compañía *Hewlett-Packard (HP)* ofreció el primer ordenador comercial con una pantalla táctil, el ordenador personal HP-150 [30]. Este primer ordenador comercial utilizaba la tecnología de infrarrojos.

A comienzos de la década de 1980, el grupo de investigación Input Research Group de la Universidad de Toronto empezó a explorar sistemas con entrada multitáctil. Esta tecnología permite reconocer simultáneamente múltiples puntos de contacto en la pantalla táctil. Es necesario también, para su funcionamiento, un software adecuado para interpretar toques simultáneos. En el grupo de investigación, Nimish Mehta logró desarrollar el primer dispositivo con tecnología multitáctil en 1982 [44]. Este dispositivo consistía en una lámina de vidrio con múltiples puntos de contacto. Detrás de ese vidrio, había una cámara capaz de capturar cualquier tipo de presión sobre la lámina de vidrio. Tras su descubrimiento, Bob Boie de la empresa Bells Labs desarrolló la primera pantalla multitáctil permitiendo a los usuarios manipular gráficos con los dedos.

Hasta 2007, la tecnología táctil no era una tecnología muy extendida. Fue gracias a la llegada de los teléfonos inteligentes cuando su popularidad y extensión de uso en la población aumentó drásticamente. Este crecimiento, fue debido principalmente al lanzamiento del iPhone de Apple en el día 29 de junio de 2007 [22] La pantalla del iPhone es una pantalla multitáctil capacitativa. Marcó una nueva era de la tecnología de la pantalla táctil. Antes de 2007, el mercado lo dominaba las pantallas táctiles resistivas como se menciona en [50]. Tras el año 2007, las pantallas táctiles capacitivas empezaron a aumentar hasta que en 2011 era la tecnología táctil más comprada. Aunque es el iPhone el que empezó

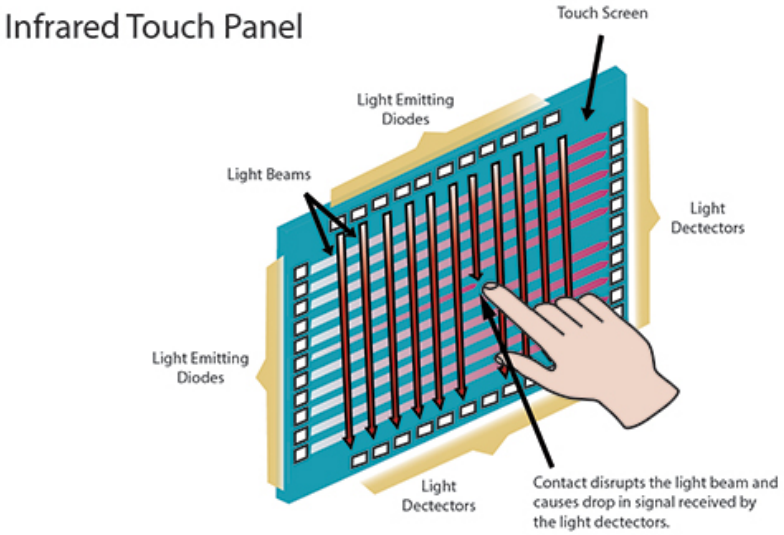


Figura 3: Pantalla táctil de infrarrojos (https://www.epectec.com)

la nueva era de la tecnología táctil, el primer teléfono móvil con la tecnología capacitiva fue el LG Prada lanzado en mayo de 2007 [38]. En abril 2010, Apple lanzó al mercado la primera generación de los iPad [23]. Las tabletas ya existían anteriormente, pero no estaba extendido su uso en la población. Fue gracias dicho lanzamiento que hizo aumentar su popularidad. Entre la aparición de los teléfonos inteligentes y la revolución de las tabletas, hicieron que el mercado de las computadoras fueron dominados por dispositivos con pantallas táctiles.

2. Estado del Arte

2.1. Reconocedores Táctiles

Los reconocedores táctiles son objetos de un entorno de desarrollo que reconocen un gesto realizado sobre la pantalla táctil. En el presente artículo, el objetivo de estudio es el sistema operativo iOS [25]. Este sistema operativo originalmente estaba desarrollado para los iPhone pero posteriormente se incorporó en los dispositivos iPod Touch y los iPad. La aplicación desarrollado y relacionado con el artículo es para iOS en los dispositivos iPad. Actualmente, hay una versión especial del sistema operativo iOS para los iPad, llamado iPadOS [26]. En cuanto a los reconocedores gestuales, no hay diferencia entre el sistema operativo iOS y iPadOS, por lo tanto, el presente estudio servirá para ambos dispositivos. A partir de ahora, se nombrará iOS a los dos sistemas operativos. A parte del sistema operativo, el estudio se centra en el marco de trabajo UIKit [14]. Es un marco de trabajo que permite construir y gestionar una interfaz gráfica de usuario basado en eventos para aplicaciones en iOS. En contrapartida, tenemos

SwiftUI [12] otro marco de trabajo para la construcción de interfaces gráficas de usuario, pero su presentación fue el 3 de junio de 2019 [24], es decir, no es un marco de trabajo maduro comparando con UIKit.

Los reconocedores de gestos táctiles en el marco de trabajo UIKit heredan de la clase UIGestureRecognizer [13]. Este es un objeto reconocedor de gestos que permite desacoplar la lógica de reconocimiento de una secuencia de toques en la pantalla multitáctil y la acción o realimentación de ese reconocimiento. Al reconocer un gesto, envía un evento de acción al objeto de destino asignado. Además, cada reconocedor gestual conlleva un estado. Los diferentes estados de los reconocedores táctiles que ofrece UIKit son: ha comenzado el gesto (began), ha cambiado el gesto (changed), ha terminado el gesto (ended) y ha sido cancelado el gesto (cancelled). A continuación, se estudiará los diferentes reconocedores gestuales que ofrece el marco de trabajo.

UITapGestureRecognizer. El reconocedor gestual UITapGestureRecognizer ([21] y [9]) permite captar los gestos de toque que el usuario realiza sobre la pantalla. Permite especificar el número de dedos que es necesario para considerarse un toque. Además, permite especificar el número de toques que debe realizar el usuario en la pantalla. El gesto de toque es un gesto discreto, es decir, el método de acción sólo se llama cuando el gesto termina con éxito. Los manejadores de la acción puede obtener la posición del gesto. Si hay múltiples toques, la ubicación es del primer toque. En caso de que el toque sea con varios dedos, la posición es el centroide de los toques de cada dedo. Si, tenemos varios reconocedores UITapGestureRecognizer hay que tener en cuenta que es necesario controlar que alguno de los reconocedores falle para que empiece a reconocer otro reconocedor de gestos. En la documentación de Apple Developers [11] nos ofrece la solución para elegir un reconocedor gestual antes que otro. El código de ejemplo que nos ofrece es el siguiente:

```
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
        shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool
{
    // Don't recognize a single tap until a double-tap fails.
    if gestureRecognizer == self.tapGesture &&
            otherGestureRecognizer == self.doubleTapGesture
    {
        return true
    }

    return false
}
```

Es necesario asignar al reconocedor de gesto un delegado que implemente el protocolo UIGestureRecognizerDelegate, este protocolo incluye la función anterior. En este código, nos indica el reconocedor de gesto de un toque no funcionará si no falla el reconocedor de gesto de dos toques. Aunque se explique aquí, esta funcionalidad se puede realizar conjuntamente con otros reconocedores de gestos y no sólo los reconocedores de toque. Además, el protocolo permite definir otras funciones que permite, por ejemplo, un reconocimiento simultáneo de varios gestos.

UIPinchGestureRecognizer. El reconocedor de gestos UIPinchGestureRecognizer ([17] y [6]) permite reconocer gestos de pellizco que implican el toque de dos dedos. Es un gesto continuo, es decir, el método de acción es llamado cada vez que la distancia entre los dedos cambie. Este gesto es muy utilizado para el zoom. Cuando el usuario acerca los dos dedos, el significado convencional es alejar el zoom, mientras que si aleja los dedos, el significado convencional es acercar el zoom. Permite definir el factor escala relativo a los puntos de los dos toques sobre la pantalla. Cada vez que el usuario aumenta o disminuya la distancia entre los dos dedos, el reconocedor de gestos envía el evento de acción. Hay que tener en cuenta que para que el gesto empiece, es necesario que la distancia entre los dos dedos cambie. El manejador del evento de la acción que emite el reconocedor, puede obtener información sobre el factor escala relativo y la velocidad en que se aleja o se acerca los dedos.

UIRotationGestureRecognizer. El reconocedor de gestos UIRotationGestureRecognizer ([18] y [7]) permite reconocer gestos de rotación que implican el toque de dos dedos. El gesto de rotación es un gesto continuo. El usuario debe presionar los dos dedos sobre una misma vista o interfaz gráfica mientras la rota. Cuando el usuario mueve los dedos enfrentados en un movimiento circular, el funcionamiento convencional es que la vista gire a la vez en la misma dirección y velocidad que los dedos. El gesto no comienza hasta que los dedos se mueva lo suficiente. El manejador del evento de la acción puede obtener información sobre el giro realizado en radianes y la velocidad de la rotación.

UISwipeGestureRecognizer. El reconocedor de gestos UISwipeGestureRecognizer ([20] y [8]) permite reconocer gestos de deslizamiento hacia una dirección. Este gesto es discreto. Esto quiere decir que el evento de la acción sólo se manda después de que el gesto termina con éxito. Permite especificar el número de dedos que es necesario para considerar el toque. Al definir el reconocedor de gestos es necesario definir la dirección del gesto que puede reconocer (hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha). Por defecto, reconoce el gesto de deslizamiento hacia la derecha. Además, permite recuperar la posición en que comienza el toque (el centroide si el toque es de más de un dedo) y la posición de cada dedo por separado. Es necesario que el dedo del usuario se mueva en una dirección específica y no se desvía para que el gesto no falle.

UIPanGestureRecognizer. El reconocedor de gestos UIPanGestureRecognizer ([16] y [5]) permite reconocer gestos de desplazamiento. Este gesto es continuo. Llama al método de acción cada vez que la posición del dedo cambie. Permite definir un número mínimo y un número máximo de dedos para que reconozca el gesto de desplazamiento. El método de la acción puede recuperar información sobre la traslación y la velocidad. Además, permite cambiar la traslación a un valor deseado. Normalmente, cuando el método de acción ha realizado los cambios de las vistas o interfaces gráficas establece la traslación a cero, ya que si no, se acumula la traslación. En el funcionamiento interno del

reconocedor de gesto, la traslación no se establece a cero cada vez que se llame al método de acción.

UILongPressGestureRecognizer. El reconocedor de gestos UILongPress-GestureRecognizer ([15] y [4]) permite reconocer gestos continuos de pulsación larga. Permite establecer el mínimo número de dedos necesarios para el gesto. También permite establecer el mínimo número de toques que debe realizar el usuario. Además, se puede establecer la duración mínima de la pulsación para que empiece el gesto. Su valor por defecto es de 0,5 segundos. Por último, se puede establecer una distancia máxima de movimiento mientras que no haya empezado el gesto. Si sobrepasa dicha distancia, el reconocimiento de gesto falla y no comienza el gesto de pulsación larga. Una vez que se reconozca el gesto de pulsación larga, el usuario puede desplazar el dedo y el reconocedor de gestos llama al método de acción cada vez que cambie de posición. Esto permite tener una funcionalidad parecida al reconocedor de gestos de desplazamiento. En este caso, el método de acción puede recuperar la posición inicial en el comienzo del gesto, la posición cada vez que el usuario cambie la posición del/de los dedo/s, así como, la posición final cuando levanta el/los dedo/s.

UIScreenEdgePanGestureRecognizer. El reconocedor de gestos UIScree-nEdgePangestureRecognizer [19] permite reconocer gestos discretos que comienzan en un borde de la pantalla y se desliza hacia dentro. Se debe establecer sólo un borde de la pantalla para cada reconocedor. Los bordes que se pueden establece son arriba, izquierda, abajo, y derecha. Por defecto, tiene asignado el de la derecha.

2.2. Realimentación Táctil

Gestos estándares en iOS. Los gestos estándares en el sistema operativo iOS se describe en el documento *Human Interface Guidelines* [10]. Es importante asegurarnos de que, para las realimentaciones comunes, se utilice los gestos estándares debido a que la gente está familiarizada a dichos gestos y sus realimentaciones. Si utilizamos otros gestos hace que las personas se sientan obligados a aprender diferentes formas para la misma realimentación lo que puede llevar a la confusión, aumentar el coste cognitivo del usuario, bajar la usabilidad de la aplicación y aumentar la complejidad en el manejo de la aplicación. También hay que evitar utilizar gestos estándares para realizar acciones no comunes. En el documento nombrado anteriormente, comentan que se debe evitar interferir los gestos de borde de pantalla del sistema ya que el usuario está acostumbrado a acceder de manera rápida a funcionalidades básicas del sistema operativo, como por ejemplo, el *Control Center*. Además, también comenta que la aplicación a desarrollar debe ofrecer gestos de acceso directo para complementar, pero no sustituir la navegación ni las acciones basadas en la interfaz. Esto quiere decir, por ejemplo, que si una pantalla puede volver a la pantalla anterior, puedes ofrecer un gesto para volver, pero nunca quitar el botón para volver hacia atrás ya

que el usuario que no esté familiarizado con ese gesto, no sería capaz de volver a la pantalla anterior. Uno de los problemas de las acciones de los gestos, es que el usuario no puede observar dicha funcionalidad de la aplicación, todo al contrario que las acciones basadas en la interfaz. Por esta razón, se debe utilizar los gestos estándares o, si en caso de que no son gestos estándares, es necesario ofrecer una ayuda para que el usuario pueda aprender de los gestos permitidos en la aplicación.

Control de Gestos para la Navegación de Huawei. Con la llegada de la versión 9 del sistema operativo EMUI basado en Android Pie los dispositivos Huawei trae con una opción de control de navegación basado en gestos [39]. El sistema operativo permite cambiar entre la navegación tradicional de tres teclas y el nuevo control de navegación basado en gestos. Este sistema de navegación incluye las siguientes acciones:

- **Volver:** Al realizar un desplazamiento desde el borde de la pantalla izquierdo o derecho hacia dentro permite volver a la pantalla anterior.
- **Inicio:** Al realizar un desplazamiento desde el borde inferior de la pantalla hacia dentro permite ir a la pantalla principal.
- **Tareas recientes:** Si se desliza desde el borde de la pantalla inferior hacia dentro y posteriormente manteniéndolo pulsado despliega las tareas recientes.
- **Cambiar de aplicación:** Al realizar un deslizamiento en el borde de pantalla inferior permite cambiar a una aplicación en segundo plano. Si se hace un deslizamiento hacia la izquierda, se cambia a la aplicación de la derecha (la siguiente más recientemente abierto) y si se hace un deslizamiento hacia la derecha, se cambia a la aplicación de la izquierda (la siguiente menos recientemente abierto).
- **Navegación dentro de la aplicación:** Cuando se hace un deslizamiento desde el borde de la pantalla izquierda o derecha, en la parte superior abre el menú de la aplicación.

En cuanto a Android comenzó a añadir gestos para la navegación en la versión Android 9 (Pie) en 2018 [1], pero ha ido poco a poco. Por ejemplo, en la versión Android 10 [2] añadieron la acción volver a partir del gesto de deslizamiento desde el borde izquierdo o derecho de la pantalla hacia dentro.

2.3. Pentominós

El pentominó es un juego que utiliza las piezas del poliominó de 5 cuadrados. Fue formalmente definido por Solomon W. Golomb en 1953 que posteriormente lo publicó en su libro en 1965 [36]. El rompecabezas original ideado por Golomb consistía en un tablero de 8x8 y las 12 piezas libres del pentominó para dos o tres jugadores. Los jugadores se tornan de turno donde en cada turno, el jugador tiene que colocar una de las 12 piezas en el tablero sin solapar otra pieza y sin repetir pieza. Se permitía realizar las transformaciones de traslación, rotación y

volteo. El objetivo del juego es ser el último jugador en colocar una ficha en el tablero.

La versión del rompecabezas que vamos a estudiar consta de las 12 piezas libres del pentominó [31]. Permite realizar las transformaciones de traslación, rotación y volteo sobre las piezas. Existen versiones modificadas del juego en que no se puede realizar alguna de las transformaciones, pero constan siempre de una variante de cada pieza libre. El objetivo del juego es completar un tablero de 60 cuadrados, utilizando las 12 piezas que ofrece el juego, sin dejar ningún hueco en el tablero.

Existen tableros rectangulares normales, tableros degenerados y tableros con agujeros. Los tableros degenerados son aquellos que tienen una área menor de 60 cuadrados, es decir, no tienen espacio suficiente para colocar las 12 piezas, 12x5=60 cuadrados. En cuanto a los tableros rectangulares normales existen 4 tableros diferentes: 3x20, 4x15, 5x12 y 6x10.

El pentominó consta de 12 piezas libres, 18 piezas unilaterales o 63 piezas fijas. Se consideran dos piezas libres distintas si no coinciden al realizar transformaciones de traslación, rotación o volteo. Por otro lado, consideran dos piezas unilaterales distintas si no coinciden al realizar transformaciones de traslación o rotación. Por último, se consideran dos piezas fijas distintas si no coinciden al realizar una traslación. Esto nos permite también sacar la conclusión de que las transformaciones que se pueden realizar en las piezas de los poliominós son traslación, rotación y volteo. El volteo lo vamos a dividir en dos nuevas transformaciones. El volteo de modo espejo horizontal, voltea la pieza según el eje horizontal en el centro de la pieza. El volteo de modo espejo vertical, voltea la pieza según el eje vertical en el centro de la pieza. Cada una de las 12 piezas libres tienen asignado una letra, según el parecido de su forma.

Tablero	**Soluciones**
3x20	2
4x15	368
5x12	1.010
6x10	2.339

Tabla 1: Número de soluciones de cada tablero

En la referencia [46] podemos encontrar todas las soluciones de los tableros rectangulares normales, así como de otros tableros como degenerados o agujereados. En este trabajo, nos centraremos en los tableros normales rectangulares de área de 60 cuadrados. En la Tabla 1 se muestran el número de soluciones de cada uno de los tableros normales rectangulares. No se consideran aquellas soluciones que coinciden con otra solución por rotación o simetría del tablero. Como se puede observar, la dificultad de encontrar una solución es más difícil cuanto más ancho es el tablero. Además, es evidente que no existe solución para los tableros 1x60 ni 2x30 por la geometría de las piezas. Los demás dimensiones de tableros rectangulares no construye un tablero con área de 60 cuadrados o son giros de los tableros anteriores (10x6, 12x5, 15x4 y 20x3).

3. Propuestas de Interacción Táctil

Tras realizar un estudio del estado del arte se realiza una serie de propuestas de interacción táctil para la aplicación desarrollada. Esta aplicación está destinado para las tabletas de la serie iPad con sistema operativo iPadOS 14.x. Permite seleccionar uno de los tableros normales rectangulares. No se consideran los tableros degenerados ni con agujeros. Las transformaciones de las piezas se realizan a través de interacciones táctiles. Permite al usuario guardar y recuperar partidas. Además, la aplicación tiene dos modos de juego, un jugador y dos jugadores.

3.1. Interacción Táctil en las Piezas

Como hemos estudiado anteriormente, se permite realizar 3 diferentes transformaciones a cada una de las piezas. Dichas transformaciones son traslación, rotación y volteo.

Para la elección de las interacciones táctiles sobre las piezas se debe considerar las siguientes condiciones:

- El juego debe permitir dos jugadores por lo que la pantalla se divide en dos. Esto hace que tanto el espacio de juego como las piezas sean más pequeñas.
- El jugador utiliza un lápiz táctil para jugar la partida.

A continuación, se describe los reconocedores gestuales añadidos a la aplicación como propuesta final para la interacción con las piezas.

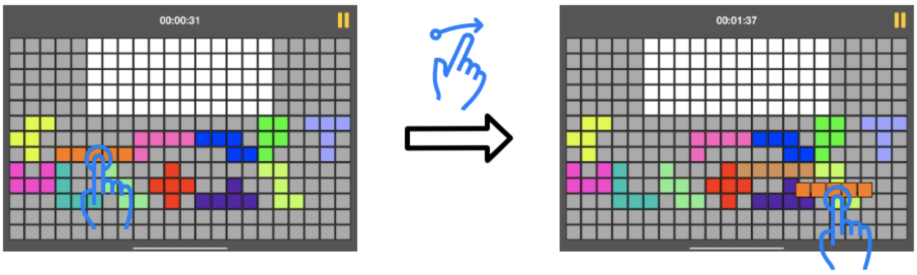


Figura 4: Traslación de una pieza

- **UIPanGestureRecognizer:** Este gesto permitirá realizar la traslación de las piezas (Figura 4). Evidentemente, este es el gesto más natural para el desplazamiento de la pieza. En cuanto al número de dedos a utilizar para el desplazamiento, se ha establecido como máximo 1 dedo. En funcionamiento de reconocedor es el siguiente:

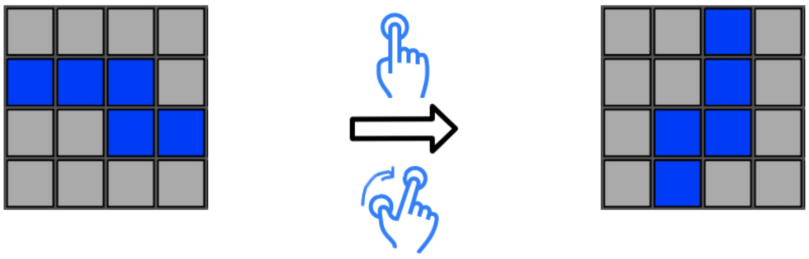


Figura 5: Rotación en sentido horario de una pieza

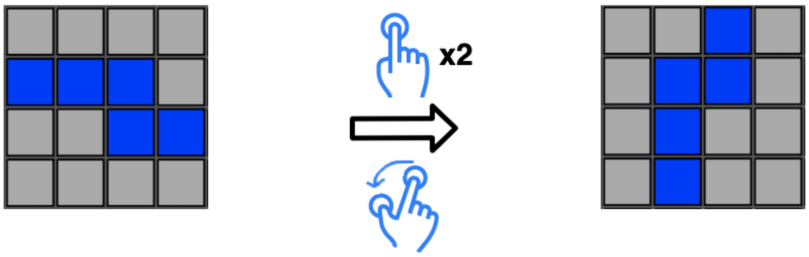


Figura 6: Rotación en sentido antihorario de una pieza

  - Cuando el usuario empieza el desplazamiento, se crea una sombra temporal de la pieza.
  - Mientras desplaza con el dedo, la pieza original seguirá el dedo mientras que la sombra se colocará alineado sobre el tablero en caso de que se puede colocar en dicha posición. Si no puede colocarse en dicha posición, se queda con la posición anterior.
  - Cuando el usuario levante el dedo, la pieza se colocará donde está la sombra mientras que ésta desaparece. Esto nos asegura que la pieza se coloque en una posición alineada con el tablero, sin salirse del espacio de juego y sin estar en conflicto con ninguna otra pieza.
- **UIRotationGestureRecognizer:** Este gesto permitirá girar las piezas 90^{o} tanto en sentido horario como en sentido antihorario (Figuras 5 y 6). La pieza no se girará hasta que el usuario gire con los dedos 90 grados en un sentido u en otro. Este reconocedor de gesto es evidentemente para realizar rotaciones de vistas.
- **UITapGestureRecognizer - gesto de un toque con un dedo:** Cuando el usuario realiza un toque con un dedo sobre una pieza, la pieza se girará 90^{o} en sentido horario (Figura 5). Se ha elegido un toque con un dedo para un giro de 90^{o} en sentido horario y no antihorario porque la mayoría de las aplicaciones de pentominós que se ha estudiado, si utilizan reconocedores gestuales, asignan un toque con un dedo para el giro de 90^{o} grados en sentido horario. Además, normalmente, un toque se le asocia a algo principal o bueno

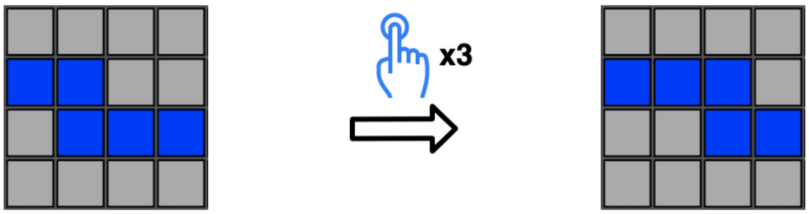


Figura 7: Reestablecer a la variante por defecto de la pieza

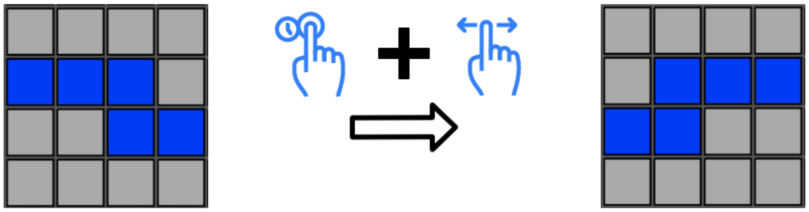


Figura 8: Volteo en el eje vertical de la pieza

y dos toques a algo secundario o contrario. El giro de 90º grados en sentido horario se le puede asignar como algo principal mientras que el giro en sentido antihorario como algo secundario. Además, en las aplicaciones estudiadas, no ofrecen un giro de 90º grados en sentido antihorario, sólo giros en sentido horario.

- **UITapGestureRecognizer - gesto de dos toques con un dedo:** Cuando el usuario realiza dos toques con un dedo sobre una pieza, la pieza se girará 90º en sentido antihorario (Figura 6).
- **UITapGestureRecognizer - gesto de tres toques con un dedo:** Cuando el usuario realiza tres toques con un dedo sobre una pieza, se restablece la variante de la pieza a la variante por defecto (Figura 7).
- **UILongPressGestureRecognizer:** Cuando el usuario realiza una pulsación larga y posteriormente, un deslizamiento hacia la izquierda o hacia la derecha, volteará la pieza en el eje vertical (Figura 8). Si en vez de un deslizamiento hacia la izquierda o hacia la derecha, si el usuario realiza el deslizamiento hacia arriba o hacia abajo, entonces, volteará la pieza en el eje horizontal (Figura 9). En este intento, en vez de usar dos reconocedores táctiles, sólo se usa UILongPressGestureRecognizer. Este reconocedor permite saber la posición inicial cuando reconoce con éxito el gesto. Una vez que reconoce el gesto, sin levantar el dedo, el usuario puede deslizar o desplazar el dedo sobre la pantalla. Por lo tanto, cuando levanta el dedo, se puede

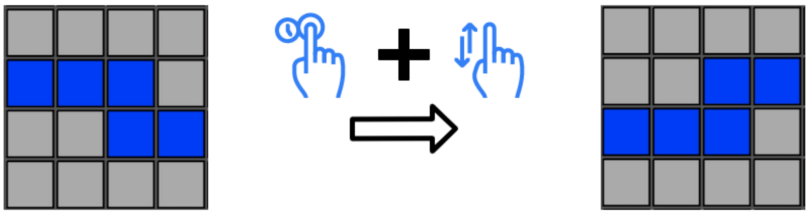


Figura 9: Volteo en el eje horizontal de la pieza

recuperar la posición final. Sabiendo la posición inicial y la posición final, podemos saber la dirección que ha realizado el usuario. Además, se puede establecer una distancia mínima entre la posición inicial y final para que se considere que se ha realizado un deslizamiento. Si coincide con una de las direcciones sin un desvío muy grande, se realizará la acción. Es evidente que para realizar volteos horizontales o verticales lo más recomendable es utilizar deslizamientos direccionales. En cuanto a la utilización de pulsación larga, permite diferenciar con éxito entre los gestos para el desplazamiento y los gestos para los volteos.

3.2. Easter Egg

En la pantalla del menú principal se ha añadido un Easter Egg con reconocedores táctiles. Si el usuario realiza una secuencia de gestos adecuados sin fallar, se mostrará el Easter Egg. Uno de los Easter Egg más famosos es el Código Konami [35]. Este truco fue introducido por primera vez en 1986 en el juego Gradius para la videoconsola NES. La secuencia para el truco es al pulsar los botones de la NES en la siguiente secuencia *Arriba-Arriba-Abajo-Abajo-Izquierda-Derecha-Izquierda-Derecha-B-A-Start*. El jugador obtenía todos los potenciadores del juego. Si el usuario realizaba la misma secuencia pero en el sentido contrario, es decir, *A-B-Derecha-Izquierda-Derecha-Izquierda-Abajo-Abajo-Arriba-Arriba-Start* recibía 30 vidas extras.

Podemos transformar ésta secuencia de botones del mando de la NES en una secuencia de gestos táctiles. Por lo tanto, podemos sustituir cada uno de los botones por un gesto. En la Tabla 2 se muestra las equivalencias propuestas entre los botones de la NES y gestos táctiles.

Botón en la NES	**Interacción Táctil**
Arriba	Deslizamiento hacia arriba
Abajo	Deslizamiento hacia abajo
Derecha	Deslizamiento hacia la derecha
Izquierda	Deslizamiento hacia la izquierda

A	Un toque con un dedo
B	Un toque con dos dedos

Tabla 2: Propuesta de equivalencia entre botones y gestos

Es evidente que para los botones de las flechas se le asigne las distintas direcciones del gesto de deslizamiento. Por otro lado, el botón A para la NES significaba *aceptar* mientras que el botón B para la NES significaba *cancelar*. Si tenemos en cuenta que con un toque con un dedo es equivalente a un clic izquierdo del ratón (principal) y el toque con dos dedos es equivalente a un clic derecho del ratón (secundario), entonces se ha decido las siguientes asignaciones: para el botón A un toque con un dedo y para el botón B un toque con dos dedos.

3.3. Otras Interacciones Gestuales

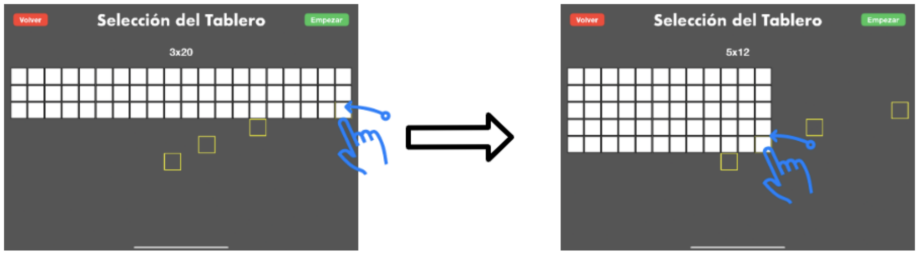


Figura 10: Interacción gestual en la selección del tablero

Selección del Tablero. En un principio, en la pantalla de selección del tablero de un jugador o de dos jugadores se había decidido realizar por interacciones basados en interfaz. Se quería poner todos los tableros, donde el jugador tenía que elegir qué tablero utilizar. Sin embargo, al final se ha decidido mejor utilizar un gesto táctil para la selección del tablero. Por lo tanto, es necesario pensar en un gesto táctil para modificar el tablero. Como todos los tableros de la aplicación son rectángulos, entonces el gesto táctil más adecuado es tocar la esquina inferior derecha del tablero y arrastrar a otra dimensión del tablero (Figura 10). Esto permite que se pueda elegir entre todas las opciones de dimensiones de tableros rectangulares con un gesto táctil.

Pantalla Ayuda. En esta pantalla existe una lista de imágenes de ayuda que se debe mostrar. Sólo se puede mostrar una imagen de ayuda en cada instante. Por lo tanto, es necesario añadir reconocedores de gestos táctiles que permite pasar

de una imagen a otra. En la Tabla 3 se muestra los gestos táctiles añadidos para esta funcionalidad.

Gesto Táctil	**Acción**
Deslizamiento a la izquierda	Imagen siguiente (Imagen de la derecha)
Deslizamiento a la derecha	Imagen anterior (Imagen de la izquierda)
Un toque con un dedo	Imagen siguiente (Imagen de la derecha)
Un toque con dos dedos	Imagen anterior (Imagen de la izquierda)

Tabla 3: Propuesta gestos táctiles para la pantalla ayuda

Volver, Pausa, Salir. En varios sistemas operativos tienen habilitado el gesto de deslizamiento desde el borde de la izquierda o de la derecha de la pantalla hacia dentro para volver hacia atrás. Sin embargo, en iOS no tiene incluido dicho gesto táctil para volver. En la aplicación de este trabajo se quiere incluir esta funcionalidad. Antes de añadir, como dice en los documentos de *Human Interface Guidelines* [10] hay que comprobar que los gestos de desplazamiento desde el borde de la pantalla no entre en conflicto con los reconocedores de dichos gestos del sistema operativo. En la referencia [3] vemos que los gestos de deslizamiento desde el borde de la pantalla utilizados por el sistema operativo iOS sólo son de borde superior e inferior de la pantalla. Por lo tanto, se puede añadir el reconocedor de deslizamientos de borde izquierdo y derecho de la pantalla sin entrar en conflicto. Se va a añadir este gesto para todas las pantallas que se puede volver a la pantalla anterior. Además, en las pantallas de las partidas de un jugador o dos jugadores, al realizar este gesto permite mostrar rápidamente el menú de pausa. Por último, en la pantalla de menú principal, también se ha añadido estos reconocedores de gesto para salir del juego, aunque en este caso, muestra primero la alerta de si quiere cerrar la aplicación.

4. Observaciones

A continuación se detalla algunas observaciones interesantes en relación a la interacción táctil observados en el desarrollo y pruebas de la aplicación.

- En un usuario que no se le ha explicado cómo funciona el juego ni tampoco estaba habilitada la pantalla de ayuda, sólo realizaba traslaciones sobre las piezas. No realizaba las transformaciones de rotación ni de volteó. Tampoco he visto que hiciera gestos para intentar girar o voltear, por lo que simplemente no sabía ni se intuía que se podía girar ni voltear.
- Este mismo usuario, tampoco sabía cómo cambiar las dimensiones del tablero.
- En un segundo usuario, sin tener habilitada la pantalla de la ayuda, pero sí que se le ha explicado el funcionamiento de los pentóminos hizo el gestos de rotación. Sin embargo, como no había realimentación hasta girar 90^{o} grados, se pensó que no se podía realizar dicho giro mediante este gesto.

Probó pulsando la pieza y al ver que se giraba empezó a utilizar dicho gesto para los giros. Por otro lado, no supo como voltear las piezas y no hacia rotaciones en sentido antihorario. Tampoco hacía el gesto para restablecer la variante por defecto de la pieza.

- Cuando se le habilitó la ayuda a este segundo usuario, se quejó de la usabilidad de dicha ayuda. La queja en cuestión era que había implementado mal el cambio de imagen de las ayudas ya que cuando intentaba realizar el deslizamiento hacia la izquierda con la espera de ver la siguiente imagen, la realimentación no era la correcta, ya que estaba implementado para este gesto, ver la imagen anterior. Este fallo de implementación ya está corregido al terminar este artículo.
- Al habilitar la ayuda los jugadores si supieron realizar todas las transformaciones de las piezas.

5. Conclusiones

Como conclusión, es necesario intentar usar gestos para la misma realimentación que realice el sistema operativo u otras aplicaciones de éxito. Esto es importante, ya que permite al usuario, sin una ayuda, realizar algunas acciones mediante gestos táctiles de manera intuitiva y rápida, sin necesidad de un coste cognitivo. Normalmente, un usuario cuando persigue un objetivo realiza la misma acción que le permitía el sistema operativo u otras aplicaciones para lograr dicho objetivo.

También, si se va a utilizar nuevos gestos táctiles asociados a acciones que el usuario no ha realizado nunca es muy necesario añadir una ayuda para que comprenda y entienda los nuevos gestos. Sin esta ayuda, normalmente los usuarios no consiguen descubrir dichos gestos. En cambio, en la interacción basado en interfaz no es muy necesario dicha ayuda, ya que es un elemento visible. Todo lo contrario que un gesto táctil, es decir, no es visible. Por esta razón es difícil que un usuario descubra el nuevo gesto sin una ayuda. Tanto los sistemas operativos como las aplicaciones, cuando añaden nuevos gestos, siempre introducen una ayuda inicial para explicar dichos gestos o una opción que te permita abrir la ayuda.

Además, los gestos táctiles no deberían sustituir a las interfaces gráficas, solo añadir como complemento. Por ejemplo, si sustituimos el botón *Volver* por el gesto de deslizamiento desde el borde izquierdo o derecho de la pantalla hacia dentro, los usuarios que no conocen este gesto, si quitamos la interfaz gráfica *Volver*, no sabría cómo volver a la pantalla anterior, por lo que pueden quedarse atrapados en algunas pantallas. Aunque los nuevos gestos táctiles suponen un coste cognitivo inicial, permite de cara a futuro mejorar su eficiencia en el uso de la aplicación. Por ejemplo, cuando se realiza las transformaciones sobre las piezas con los gestos propuestos, todas estas interacciones se realizan alrededor de la pieza. En cambio, en las aplicaciones que en vez de utilizar gestos táctiles, el usuario para girar o voltear tiene que desplazar el dedo desde la posición de la pieza hasta el botón designado para el giro o el volteo. Además, normalmente

estos botones están colocados arriba de la pantalla, lo que dificulta aún más la realización de las transformaciones.

Como se ha comentado en las observaciones, si implementados un reconocedor de gestos cuya realimentación es contrario a la realimentación que intuitivamente debería tener, no sólo genera un coste cognitivo al usuario, sino que además baja mucho la satisfacción del usuario sobre la aplicación.

Referencias

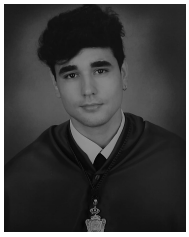
1. Android. Android 9. https://www.android.com/versions/pie-9-0/, 2018. Last Visit: 2021-07-03.
2. Android. Android 10. https://www.android.com/intl/es_es/android-10/, 2019. Last Visit: 2021-07-03.
3. Apple. Aprender gestos avanzados para interactuar con el ipad. https://support.apple.com/es-es/guide/ipad/ipadab6772b8/ipados. Last Visit: 2021-07-05.
4. Apple. Handling long-press gestures. https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_long-press_gestures. Last Visit: 2021-07-02.
5. Apple. Handling pan gestures. https://developer.apple.com/documentation/ uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_pan_gestures. Last Visit: 2021-07-02.
6. Apple. Handling pinch gestures. https://developer.apple.com/documentation/ uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_pinch_gestures. Last Visit: 2021-07-02.
7. Apple. Handling rotation gestures. https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_rotation_gestures. Last Visit: 2021-07-02.
8. Apple. Handling swipe gestures. https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_swipe_gestures. Last Visit: 2021-07-02.
9. Apple. Handling tap gestures. https://developer.apple.com/documentation/ uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_tap_gestures. Last Visit: 2021-07-02.
10. Apple. Human interface guidelines: Gestures. https://developer.apple.com/design/human-interface-guidelines/ios/user-interaction/gestures/. Last Visit: 2021-07-03.
11. Apple. Preferring one gesture over another. https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/coordinating_multiple_gesture_recognizers/preferring_one_gesture_over_another. Last Visit: 2021-07-02.
12. Apple. Swiftui. https://developer.apple.com/xcode/swiftui/. Last Visit: 2021-07-02.
13. Apple. Uigesturerecognizer. https://developer.apple.com/documentation/uikit/uigesturerecognizer. Last Visit: 2021-07-02.
14. Apple. Uikit. https://developer.apple.com/documentation/uikit. Last Visit: 2021-07-02.
15. Apple. Uilongpressgesturerecognizer. https://developer.apple.com/documentation/uikit/uilongpressgesturerecognizer. Last Visit: 2021-07-02.

16. Apple. Uipangesturerecognizer. https://developer.apple.com/documentation/ uikit/uipangesturerecognizer. Last Visit: 2021-07-02.
17. Apple. Uipinchgesturerecognizer. https://developer.apple.com/documentation/ uikit/uipinchgesturerecognizer. Last Visit: 2021-07-02.
18. Apple. Uirotationgesturerecognizer. https://developer.apple.com/ documentation/uikit/uirotationgesturerecognizer. Last Visit: 2021-07-02.
19. Apple. Uiscreenedgepangesturerecognizer. https://developer.apple.com/ documentation/uikit/uiscreenedgepangesturerecognizer. Last Visit: 2021-07-02.
20. Apple. Uiswipegesturerecognizer. https://developer.apple.com/documentation/ uikit/uiswipegesturerecognizer. Last Visit: 2021-07-02.
21. Apple. Uitapgesturerecognizer. https://developer.apple.com/documentation/ uikit/uitapgesturerecognizer. Last Visit: 2021-07-02.
22. Apple. Apple reinvents the phone with iphone. https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/, 2007. Last Visit: 2021-07-02.
23. Apple. Apple launches ipad. https://www.apple.com/newsroom/2010/01/27 Apple-Launches-iPad/, 2010. Last Visit: 2021-07-02.
24. Apple. Swiftui. https://developer.apple.com/news/?id=06032019b, 2019. Last Visit: 2021-07-02.
25. Apple. ios 14. https://www.apple.com/es/ios/ios-14/, 2020. Last Visit: 2021-07-02.
26. Apple. ipados 14. https://www.apple.com/es/ipados/ipados-14/, 2020. Last Visit: 2021-07-02.
27. Frank Beck and Bent Stumpe. *Two devices for operator interaction in the central control of the new CERN accelerator*. CERN Yellow Reports: Monographs. CERN, Geneva, 1973.
28. Julian M Bucknall. Learn to solve pentominoes. https://boyetblog.s3.amazonaws.com/PCPlus/294.pentominoes.pdf. Last Visit: 2021-07-06.
29. Maker Hugh Le Caine. Electronic sackbut. https://soundandscience.de/instrument/ electronic-sackbut. Last Visit: 2021-07-02.
30. Hewlett-Packard Development Company. Hp-150 touchscreen personal computer with hp 9121 dual drives, 1983. http://www.hp.com/hpinfo/abouthp/histnfacts/museum/personalsystems/ 0031/, 2012. Last Visit: 2021-07-02.
31. Isomer Design. Pentominios. http://isomerdesign.com/Pentomino/. Last Visit: 2021-07-03.
32. Elotouch. https://www.elotouch.com/about. Last Visit: 2021-07-02.
33. Yuki Fujimoto. Pentomino. https://apps.apple.com/us/app/pentomino/ id1116468490. Last Visit: 2021-07-04.
34. Weeny Brain's Game. Pentomino classic. https://apps.apple.com/us/app/pentomino-classic/id1219129174. Last Visit: 2021-07-04.
35. Jay Garmon. Geek trivia: The cheat goes on. https://www.techrepublic.com/article/geek-trivia-the-cheat-goes-on/, 2007. Last Visit: 2021-07-04.
36. Solomon W. Golomb. *Polyominoes*. Scribner, New York, 1965.
37. Solomon W. Golomb. *Polyominoes*. Princeton University Press, 2020.
38. GSMArena. Lg prada. https://www.gsmarena.com/lg_ke850_prada-1828.php, 2007. Last Visit: 2021-07-02.

39. Huawei. Navigate your phone using gesture control. https://consumer.huawei.com/uk/support/faq/navigate-your-phone-using-gesture-control/, 2018. Last Visit: 2021-07-03.
40. E.A. Johnson. Touch display—a novel input/output device for computers. *Electronics Letters*, 1:219–220(1), October 1965.
41. E.A. Johnson. Touch displays: A programmed man-machine interface. *Ergonomics*, 10(2):271–277, 1967.
42. Ricardo León. Pentomino. https://play.google.com/store/apps/details?id=com.ralr.pentominon. Last Visit: 2021-07-04.
43. W.F. Lunnon. *Counting Polyominoes*. Computers in Number Theory, 1971.
44. Nimish Mehta. *A Flexible Machine Interface*. PhD thesis, University of Toronto, 01 1982.
45. Pamun. Pentomino basic. https://play.google.com/store/apps/details?id=com.gmkwak.PentominoBasic. Last Visit: 2021-07-04.
46. Gerard Putter. Gerard's universal polyomino solver. https://gp.home.xs4all.nl/PolyominoSolver/Polyomino.html. Last Visit: 2021-07-03.
47. D.Hugh Redelmeier. Counting polyominoes: Yet another attack. *Discrete Mathematics*, 36(2):191–203, 1981.
48. Familion Studio. Pentamino. https://play.google.com/store/apps/details?id=ru.familion.pentamino. Last Visit: 2021-07-04.
49. Tetris. History of tetris. https://tetris.com/history-of-tetris. Last Visit: 2021-07-03.
50. Geoff Walker. A review of technologies for sensing contact location on the surface of a display. *Journal of the Society for Information Display*, 20(8):413–440, 2012.
51. Ray Wenderlich. High quality programming tutorials: ios, android, swift, kotlin, flutter, server side swift, unity, and more! https://www.raywenderlich.com. Last Visit: 2021-07-06.
52. Gayle Young. Electronic sackbut (1945-1973). `http://hughlecaine.com/en/sackbut.html`, 1999. Last Visit: 2021-07-02.

Autores

Adrián Valera Roman es graduado en ingeniería informática por la universidad de Salamanca, donde también curso el máster en sistemas inteligentes, durante el cual se centró en la investigación sobre los sistemas de recomendación para grupos basados en el filtrado colaborativo. Como apasionado de la informática desde pequeño, sus áreas de interés son muy heterogéneas: Ciberseguridad, Machine Learning, ingeniería del software o blockchain. Actualmente se encuentra centrado en el desarrollo de software y el análisis de datos.



Alberto Montero Fernández es técnico e ingeniero informático graduado en la Universidad de Salamanca, donde terminó su estudios de máster sobre sistemas inteligentes. Actualmente forma parte del grupo de investigación BISITE donde realiza proyectos relacionados con la inteligencia artificial, extracción de datos y big data. Donde la mayor parte de su tiempo se centra en tareas relacionadas con el backend de las aplicaciones. Busca siempre formarse en las nuevas tecnologías relacionadas con estas ramas.

Álvaro Lozano Murciego es Ingeniero Técnico en Informática de Sistemas y Graduado en Ingeniería Informática y Doctor en Ingeniería Informática por la Universidad de Salamanca en la que también realizó el Máster en Sistemas Inteligentes. Su investigación durante el Máster ha abordado el área de los Sistemas de Recomendación y el desarrollo de aplicaciones móviles para bicicletas eléctricas. Durante 3 años ha trabajado como desarrollador Android en proyectos junto con la empresa Ebikemotion y posee la certificación de Associate Android Developer. En 2017 obtuvo una beca predoctoral del Banco Santander y la Universidad de Salamanca y curso los estudios de doctorado en dicha universidad. Posteriormente ha trabajado como investigador postdoctoral en el área de los Sistemas de Recomendación. Actualmente es Profesor Ayudante Doctor en el Departamento de Informática de la Universidad de Salamanca donde imparte docencia en el Grado en Ingeniería Informática Grado en Física y Grado de Ingeniería Química. Sus líneas de investigación están directamente relacionadas con el Internet de las Cosas, Sensores Inteligentes, Sistemas de recomendación, Machine Learning, Sistemas de Predicción y Optimización de rutas.

Angélica González Arrieta es doctora en Informática por la Universidad de Salamanca. Cuenta con una amplia experiencia investigadora en el campo de sistemas conexionistas. Es Profesora Titular del Departamento de Informática y Automática de dicha Universidad, del área de Ciencia de la Computación e Inteligencia Artificial. Actualmente compatibiliza su labor docente e investigadora con la dirección de diversas actividades formativas sobre recursos tecnológicos, pericia y seguridad informática, colaborando activamente con las Fuerzas y Cuerpos de Seguridad.

Belén Pérez Lancho es Licenciada y Doctora en Ciencias Físicas por la Universidad de Salamanca. Realizó su formación postdoctoral en la Universidad Paris VI (curso 1996-97) y desde 1998 es Profesora Titular en el Área de Ingeniería de Sistemas y Automática de la Universidad de Salamanca. Imparte o ha impartido docencia en las titulaciones de Ingeniería Informática, Física, Estadística y en el Máster en Sistemas Inteligentes. Pertenece al grupo de investigación BISITE y ha participado en más de 20 proyectos de investigación, principalmente en aplicaciones de sistemas de control y sistemas multiagente, colaborando en varias publicaciones científicas, en la dirección de tres tesis y de numerosos proyectos de fin de carrera. Ha ocupado cargos de gestión en la Facultad de Ciencias como Secretaria Académica (años 2000 a 2004) y como Vicedecana (años 2008 a 2016).

Benjamin Begic es licenciado en Ingeniería Industrial junto a una *Master of Science*en Ingeniería Mecánica con especialización en Automotores. Además, ha realizado en Máster en Sistemas Inteligentes por la Universidad de Salamanca. Actualmente trabaja para Arriva plc en Italia siendo el ingeniero de la planificación y conservación de los buses del transporte público de los municipios de Bérgamo y Lecco. Entre las tareas que lleva a cabo se encuentra seguir con la instalación de los dispositivos electrónicos sobre la flota de automóviles y otras actividades del grupo para el norte de Italia como el sistema GPS. Al mismo tiempo se está formando en el motorsport con técnicos e ingenieros de varias empresas como Brembo, Regina Chain, Magneti Marelli y Ducati. Sus pasiones son la ingeniería aplicada a los vehículos, la robótica, el *management* y la historia.

Denis Pato Martínez es ingeniero informático graduado en la Universidad de Salamanca, donde también terminó sus estudios de máster sobre sistemas inteligentes. Actualmente forma parte del grupo de investigación ESALab, donde trabaja como programador en proyectos dentro del ámbito de la inteligencia artificial y la automatización de procesos informáticos de obtención de la información. Otras áreas de su interés son el diseño web y la ciberseguridad.

Francisco Javier Blanco Rodríguez es Licenciado en Ciencias Físicas y Doctor por la Universidad de Salamanca desde el año 2003. En la actualidad es Profesor Contratado Doctor en el Área de Ingeniería de Sistemas y Automática en el Departamento de Informática y Automática de la Universidad de Salamanca. Imparte varias asignaturas de robótica tanto en titulación de Grado como de Máster. En cuanto a la investigación está interesado en temas de Robots autónomos, robótica educativa y nuevas tecnologías aplicadas en la educación.

Guillermo Hernández González es Doctor en Física Fundamental y Matemáticas además de poseer el Máster en en Sistemas inteligentes por la Universidad de Salamanca. Cuenta también con otras titulaciones en los ámbitos de Física, Informática y Matemáticas. Su investigación actual se centra en las líneas del aprendizaje automático, el aprendizaje profundo, el procesamiento del lenguaje natural y la analítica visual. En este momento disfruta un contrato postdoctoral en la Universidad de Salamanca en torno a los sistemas inteligentes autoexplicativos. Adicionalmente ha trabajado en las aplicaciones de los sistemas de aceleración láser a la irradiación de tejido biológico, donde ha realizado varias contribuciones a campos como la interacción radiación-materia y la caracterización de emisiones generadas por pulsos ultracortos. También ha trabajado en el campo de la radioprotección y el diseño de blindajes.

Héctor Sánchez San Blas es Personal Investigador en Formación por la Universidad de Salamanca, siendo beneficiario de un contrato predoctoral FPU en la convocatoria 2020-2021. Es estudiante de doctorado en ingeniería informática en la misma universidad, así mismo, ha realizado en dicho centro la carrera de Ingeniería Informática y el Máster en Sistemas Inteligentes. Durante el máster, fue investigador colaborador en el Laboratorio de Sistemas Expertos y Aplicaciones (ESALab) de esta universidad, colaborando con proyectos de investigación relacionados con el Internet de las Cosas, aplicaciones de Realidad Virtual y aprendizaje automático. Actualmente, investiga el desarrollo de IoT y redes neuronales junto a proyectos de investigación centrados en la visión artificial y Smart Cities.

Juan Francisco de Paz Santana, Doctor en Informática y Automática por la Universidad de Salamanca en 2010. Ingeniero técnico en Informática y de Sistemas (2003), Ingeniero Informático (2005) y Diplomado en Estadística (2007) por la Universidad de Salamanca. Actualmente es Catedrático de Universidad, director de departamento en el Departamento de Informática y Automática de la Universidad de Salamanca y coordinador del progarma de doctorado en Ingeniería Informática por la Universidad de Salamanca. Además, ha realizado el Máster Universitario en el Desarrollo de Sistemas para el Comercio Electrónico y el Máster Universitario en Animación Digital. Ha obtenido diferentes premios a nivel académico como el Premio de Grado, Premio de doctorado y en otros ámbitos como un premio de SUN Microsystems y otro de la Junta de Castilla y León. En la actualidad trabaja en el campo de los sistemas distribuidos basados en sistemas multi-agente, la Inteligencia Artificial, la bioinformática, la inteligencia ambiental, internet de las cosas, entre otros. Ha participado en 48 proyectos de investigación a nivel regional, nacional e internacional. Es coautor de más de 180 artículos publicados en diversas revistas de ámbito nacional e internacional, workshops y simposios, 50 de estas publicaciones en revistas con índice de impacto. Ha sido organizador de numerosas conferencias y además ha sido revisor en diferentes conferencias y revistas de ámbito internacional en el área de las Ciencias de la Computación e Inteligencia Artificial. Es actualmente el director del Departamento de Informática y Automática, director del Grupo de Investigación Reconocido ESAL, miembro del grupo IBSAL y socio de la AEPIA. Evaluador de la ANECA desde 2010 y FONCyT

Juan M. Corchado Rodríguez es Catedrático en la Universidad de Salamanca. Ha sido Vicerrector de Investigación desde el 2013 hasta el 2017 y Director del Parque Científico de la Universidad de Salamanca. Elegido dos veces como Decano de la Facultad de Ciencias, es Doctor en Ciencias de la Computación por la Universidad de Salamanca y, además, es Doctor en Inteligencia Artificial por la University of the West of Scotland. Dirige el Grupo de Investigación Reconocido BISITE (Bioinformática, Sistemas Inteligentes y Tecnología Educativa), creado en el año 2000. Director del IOT Digital Innovation Hub y presidente del AIR Institute, J. M. Corchado también es Profesor Visitante en el Instituto Tecnológico de Osaka desde enero de 2015, Profesor visitante en la Universiti Malaysia Kelantan y ha sido profesor visitante en la Universiti Teknologi Malaysia y Miembro del Advisory Group on Online Terrorist Propaganda of the European Counter Terrorism Centre (EUROPOL). Ha sido presidente de la asociación IEEE Systems, Man and Cybernetics, y coordinador académico del Instituto Universitario de Investigación en Arte y Tecnología de la Animación de la Universidad de Salamanca e investigador en las Universidades de Paisley (UK), Vigo (Spain) y en el Plymouth Marine Laboratory (UK). En la actualidad compagina toda su actividad con la dirección de los programas de Máster en Seguridad, Animación Digital, Desarrollo de Aplicaciones Móviles Multiplataforma, Inteligencia Digital, Fintech, Transferencia de Conocimiento y Gestión de la I+D+i, Transformación Digital, Dirección de Sistemas de Información, Internet de las Cosas, Social Media, Diseño e Impresión 3D, Blockchain, Industria 4.0 y Smart Cities Intelligent Buildings, en la Universidad de Salamanca y su trabajo como editor jefe de las revistas ADCAIJ (Advances in Distributed Computing and Artificial Intelligence Journal), OJCST (Oriental Journal of Computer Science and Technology) o Electronics MDPI (Computer Science & Engineering section). Actualmente, desarrolla principalmente trabajos en proyectos relacionados con Inteligencia Artificial, Machine Learning, Blockchain, IoT, Fog Computing, Edge Computing, Smart Cities, Smart Grids y Análisis de sentimiento.

José Rafael García-Bermejo Giner posee un doctorado en Físicas (1989, Universidad de Salamanca, España) y la Certificación de Apple ACTC T3 (Snow 100, Snow 101, Snow 201). Actualmente desarrolla su actividad académica como Profesor Titular del Departamento de Informática y Automática de la Universidad de Salamanca en el área de Lenguajes y Sistemas Informáticos. Ha realizado estancias como docente e investigador en diferentes universidades de Alemania y Finlandia. Es coordinador Erasmus de los estudiantes de Ingeniería Informática. Sus principales líneas de investigación incluyen programación

estructurada, programación orientada a objetos, interfaces Hombre-Máquina, interfaces de usuario, dispositivos móviles y administración de sistemas. Es autor y traductor técnico de un gran número de libros.

Luis Carlos Arroyo Vicente es físico graduado en la Universidad de Salamanca, y postgraduado en el máster de Sistemas Inteligentes. Actualmente continúa su formación académica en el Máster Universitario de Profesorado, también en la USAL.

Manuel Luque Cuesta es un ingeniero informático graduado en la Universidad de Córdoba, realizando un TFG sobre la construcción de una biblioteca de algoritmos evolutivos. Además, finalizó sus estudios con un Máster en la Universidad de Salamanca sobre sistemas inteligentes, realizando un TFM sobre el desarrollo de algoritmos de detección de anomalías.

María N. Moreno García es Catedrática de Universidad del Departamento de Informática y Automática de la Universidad de Salamanca, directora del grupo de investigación en Minería de Datos y Coordinadora del Programa de Doctorado de Ingeniería Informática. Sus áreas de investigación de interés se centran en el desarrollo y aplicación de algoritmos de minería de datos en diferentes dominios como minería web y de medios sociales o apoyo a las decisiones en medicina.

Mario Francisco Sutil es licenciado en Ciencias Físicas, especialidad Electrónica, por la Universidad de Salamanca (España) en 1997, y Doctor en Ciencias en 2011. Trabaja en el Departamento de Informática y Automática de la Universidad de Salamanca desde 2000, y actualmente es Profesor Contratado Doctor. Sus áreas de investigación de interés son el Control Avanzado de Procesos, en particular Control Predictivo Basado en Modelos, Control Distribuido basado en sistemas multiagente, y Control Inteligente. Las aplicaciones son

principalmente sistemas integrados de aguas residuales y otros procesos de gran escala. Ha trabajado en más de 20 proyectos de investigación, con numerosas publicaciones en revistas indexadas SCI.

Nuria Tovar Suárez es graduada en ingeniería electrónica, industrial y automática por la Universidad de León. Tras finalizar el grado decidió cursar el máster en Sistemas Inteligentes en la Universidad de Salamanca. Actualmente trabaja en el departamento de Software Embebido en el CTAG (Centro Tecnológico de Automoción de Galicia), una entidad dedicada al desarrollo, la investigación y a la innovación tecnológica en el sector automovilístico, centrándose en el desarrollo de vehículos autónomos y ADAS

Oscar Emilio Aponte Rengifo es graduado en Electrónica Industrial y Automática por la Universidad de Valladolid. Ha realizado el Máster en Sistemas Inteligentes por la Universidad de Salamanca en la cual se encuentra ahora mismo realizando el doctorado en Ingeniería informática siguiendo como línea de investigación el Control Inteligente donde destaca la aplicación del Aprendizaje por Refuerzo en sistemas de control.

Pablo García Ullán es físico graduado en la Universidad de Salamanca, donde también realizo sus estudios de Máster en Sistemas Inteligentes. Para llevar a cabo su TFM sobre redes LSTM trabajó como desarrollador de software en Artelnics. Actualmente continua su formación académica en el Máster Universitario de Profesorado a la vez que trabaja como desarrollador de software.

Pastora Isabel Vega Cruz es Catedrática de Universidad del área de Ingeniería de Sistemas y Automática y lidera el Grupo de Investigación Reconocido de la USAL denominado "Supervisión y Control de Procesos". La actividad del mismo se centra en proponer soluciones eficientes para la optimización y el control de procesos industriales, integradas en los niveles más altos de la jerarquía del control y con el objeto de incrementar la seguridad, la fiabilidad y la calidad en la operación y en el control en las industrias de procesos.

Concretamente, se investigan soluciones teóricas y prácticas, para la optimización global de la operación, el control avanzado de los procesos y la supervisión inteligente de los mismos. Entre las líneas de investigación más destacadas se encuentran: Control Predictivo Distribuido basado en sistemas multiagente, Optimización, supervisión y control de EDARs y Síntesis y diseño integrado de procesos. Las aplicaciones industriales más importantes incluyen los Sistemas Integrados de Aguas y los Sistemas de Poligeneración.

Roberto Casado Vara es doctor en Ingeniería Informática por la Universidad de Salamanca desde 2019 (con premio extraordinario al doctorado). Es graduado en Matemáticas por la Universidad de Salamanca, y cuenta con un máster en Big Data & Visual Analytics por la Universidad Internacional de la Rioja. Ha trabajado en la compañía Viewnext como data scientist y consultor Powercenter para importantes clientes en el sector farmacéutico y de la administración pública. Desde 2017, ha trabajado en diferentes centros públicos y privados de investigación, entre ellos: el Grupo de investigación BISITE de la Universidad de Salamanca, donde trabaja como investigador en Deep Learning, modelos matemáticos avanzados para la monitorización y control no lineal inteligente, blockchain, tecnología en la cual ha obtenido una certificación oficial.

Roberto López es el fundador y director de Artelnics. Es Licenciado en Física por la Universidad de Salamanca, Máster en Física Computacional de la Universidad de Salford (Reino Unido) y Doctor en Inteligencia Artificial por la Universidad Politécnica de Cataluña. Tiene 20 años de experiencia desarrollando algoritmos y aplicando inteligencia artificial en Tata Steel, el Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE), la Universidad Politécnica de Cataluña y Artelnics. Roberto es el desarrollador principal de la biblioteca de redes neuronales abiertas OpenNN y la plataforma de aprendizaje automático Neural Designer.

Roberto Therón Sánchez cursó sus estudios de Informática en la Universidad de Salamanca (Diplomatura) y la Universidad de la Coruña (Licenciatura).Tras entrar a formar parte del Grupo de Investigación en Robótica de la Universidad de Salamanca, presentó su trabajo de Tesis, Çálculo paralelo del espacio de las configuraciones para robots redundantes", recibiendo el Premio Extraordinario

de Doctorado.Posteriormente ha obtenido los títulos de
Licenciado en Comunicación Audiovisual (Universidad de Salamanca) y Licenciado en Humanidades (Universidad de Salamanca). En la misma Universidad de Salamanca continúa realizando su trabajo de investigador, como encargado del grupo VisUsal (dentro del Grupo de Investigación GRIAL) que se centra en la combinación de enfoques procedentes de la Informática, Estadística, Diseño Gráfico y Visualización de Información, para obtener una adecuada comprensión de conjuntos de datos complejos.En los últimos años, se ha dedicado al desarrollo de herramientas de visualización avanzada para datos multidimensionales, como por ejemplo datos genéticos o paeloclimáticos. En el área de Analítica Visual desarrolla productivas colaboraciones con grupos e instituciones de reconocido prestigio internacional, como el Laboratorio de Ciencias del Clima y del Medio Ambiente (París), la Real Academia Española o la Academia de Ciencias de Austria. Ha sido coordinador del Programa de Doctorado en Ingeniería Informática de la USAL (2012-2017) y actualmente es el Director del Máster Universitario en Sistemas Inteligentes (desde 2012).

Rosa Folgoso Bullejos es ingeniera electrónica industrial graduada por la Universidad de Granada y máster en Sistemas Inteligentes por la Universidad de Salamanca. Actualmente trabaja para DHV technology (Málaga) en el departamento de desarrollo eléctrico de paneles solares para Small Satellites & Cubesats. Otras áreas de interés son la robótica y la domótica.

Sara Rodríguez González es Profesora Contratada Doctor en el Departamento de Informática y Automática de la Universidad de Salamanca. Obtuvo su doctorado en Informática en la misma Universidad en 2010. Recibió el título de Ingeniería Técnica en Informática de Sistemas en 2004 y de Ingeniería en Informática en 2007. Por otra parte, ha llevado a cabo otros estudios de postgrado como el Master en el desarrollo de sistemas de comercio electrónico y el Master en animación de digital. Ha participado como co-autora en artículos publicados en revistas internacionales de reconocido prestigio. También ha participado como comité de programa en conferencias como PAAMS, IWANN, PACBB, ISAMI, HAIS o DCAI. En la actualidad, es miembro de la Asociación Española para la Inteligencia Artificial (AEPIA), del Instituto de Investigación de Arte y Tecnologías de la Animación de la Universidad de Salamanca y del grupo de investigación BISITE (Bioinformática, Sistemas Inteligentes y Tecnología) en el que sigue su labor investigadora.

Tobias Tønnessen es Ingeniero en Tecnologías de Automatización e Ingeniería de Control por la Universidad Noruega Occidental de Ciencias Aplicadas. Ha realizado un máster en Sistemas Inteligentes por la Universidad de Salamanca, España. Actualmente trabaja en una empresa internacional como desarrollador de software.

Vidal Moreno Rodilla es Licenciado en Ciencias Físicas en la rama de Electrónica y Doctor en Ciencias en 1996. Actualmente es Profesor Titular de Universidad del área de Ingeniería de Sistemas y Automática de la Universidad de Salamanca. Ha impartido docencia sobre Robótica, Inteligencia Artificial y Control en niveles de Grado, Máster y Doctorado tanto en Salamanca como en el extranjero. Miembro fundador del Grupo de Investigación Reconocido "Robótica y Sociedad", es autor de varias decenas de publicaciones en revistas internacionales fruto de la dirección de una centena trabajos de investigación que incluye tesis doctorales, Trabajos de Fin de Master, etc.

Vivian Félix López Baptista es profesora titular de la Universidad de Salamanca en el área de Ciencias de la Computación e Inteligencia Artificial. Doctorada en Informática por la Universidad de Valladolid en 1996. Miembro del Grupo de Minería de Datos. Ha realizado investigación en diferentes campos como procesamiento del lenguaje natural, redes neuronales y minería de datos. Tiene 80 artículos publicados en revistas de reconocido prestigio, talleres y actas de conferencias, 20 libros y capítulos de libros y 20 informes técnicos, la mayoría de ellos en estos temas. Miembro del comité organizador y científico de varios simposios internacionales. Fue directora del Máster en Sistema Inteligente y del Programa de Doctorado en Informática y Automática de la Universidad de Salamanca desde junio de 2010 hasta octubre de 2012.

Xuzeng Mao realizó los estudios en el Grado en Ingeniería Informática entre los años 2016 y 2020 en la Universidad de Salamanca. Así mismo durante el curso 2020-2021 obtuvo el Máster en Sistemas Inteligentes por la Universidad de Salamanca. En el verano de 2019 realizó las prácticas externas curriculares en el grupo de investigación ESALab y posteriormente siguió trabajando en dicho grupo. Su

investigación se ha centrado en el desarrollo de algoritmo
de visión artificial e IoT. Actualmente, se centra en el desarrollo Full Stack.