

Proyecto de Innovación Docente

Metodologías docentes para el uso positivo de la inteligencia artificial generativa en asignaturas de programación

Guillermo Hernández González, Juan Andrés Hernández Simón,
Iván Álvarez Navia, María Resurrección Gutiérrez Rodríguez,
Fernando de la Prieta Pintado

Curso 2023/24

1. Introducción

Esta memoria contiene el resumen de los resultados del proyecto de innovación docente *Metodologías docentes para el uso positivo de la inteligencia artificial generativa en asignaturas de programación*, beneficiario de la Convocatoria de Ayudas a Proyectos de Innovación y Mejora Docente, Curso 2023-2024, con código ID2023/095.

El objetivo general de este proyecto era considerar cómo se debía adaptar el proceso de enseñanza en el marco de las asignaturas de programación para reflejar la realidad que supone la irrupción y la adopción de los grandes modelos de lenguaje como herramientas de inteligencia artificial y los sistemas basados en ello para su transferencia, cuyo más claro exponente, en el contexto en que nos movemos, es la herramienta ChatGPT de OpenAI.

Para ello se ha seguido un enfoque centrado en el alumnado, el colectivo cuyo uso de la herramienta tiene un impacto más significativo en el proceso de enseñanza-aprendizaje. Para dirigirnos adecuadamente a este grupo, fue necesario realizar inicialmente un análisis detallado de las posibilidades de uso de la herramienta en la materia que ocupaba el alcance del proyecto, que es la programación. Específicamente se tuvo en cuenta el contenido de la asignatura Programación I, como se recogía en la propuesta del proyecto.

En esta memoria mostraremos el proceso de realización, así como los resultados obtenidos en el proceso. La estructura del documento es la siguiente: en el apartado 2 se revisará la ejecución del proyecto, detallando especialmente el resultado del análisis realizado por el profesorado, que culmina en la ejemplificación de una serie de patrones adecuados para el uso de la IA generativa en el ámbito de la programación; en el apartado 3 se analizarán los resultados obtenidos en la encuesta dirigida a recabar la percepción del alumnado; finalmente,

en el apartado 4, se realizará un análisis crítico del resultado del proyecto y se valorará el impacto que podrá tener este en el futuro.

2. Desarrollo del proyecto

Como fruto del diseño realizado por el profesorado participante en el proyecto, se realizó una presentación sobre la herramienta, en forma de una pequeña sesión de unos 20 minutos, integrada en las clases habituales. Esta experiencia se llevó a cabo en un momento en el que ya habían tenido suficiente contacto con la programación como para entender el contenido tratado en la misma.

En dicha sesión se presentaron 6 estrategias generales para apoyarse en IA generativa en la posible preparación de la asignatura. En todo momento se trató de presentar esta como una herramienta más, intentando no interferir en la capacidad de libre decisión del estudiantado sobre si adoptar o no su uso.

Por su extensión y disponibilidad, se utilizó la versión gratuita de ChatGPT, que está basada en el modelo de lenguaje GPT3.5. Las conversaciones utilizadas, y reproducidas más adelante en este trabajo, se realizaron en octubre de 2023.

El análisis crítico de las propuestas presentadas, que se desarrolla en la siguiente sección, revela algunas cuestiones generalmente aplicables —incluyendo también más allá del ámbito de la programación— al uso de la IA generativa en el marco del aprendizaje.

La Fig. 1 muestra el primero de los patrones introducidos al alumnado: la «exploración de la teoría». En este escenario, deliberadamente abierto, pretendíamos mostrar las posibles omisiones que una respuesta de IA generativa pueda proporcionar frente a referencias más sólidas, como puedan ser las bibliográficas [6, 4] u otras más «grises» como pueden ser las webs con cierta solvencia reconocida [2, 1].

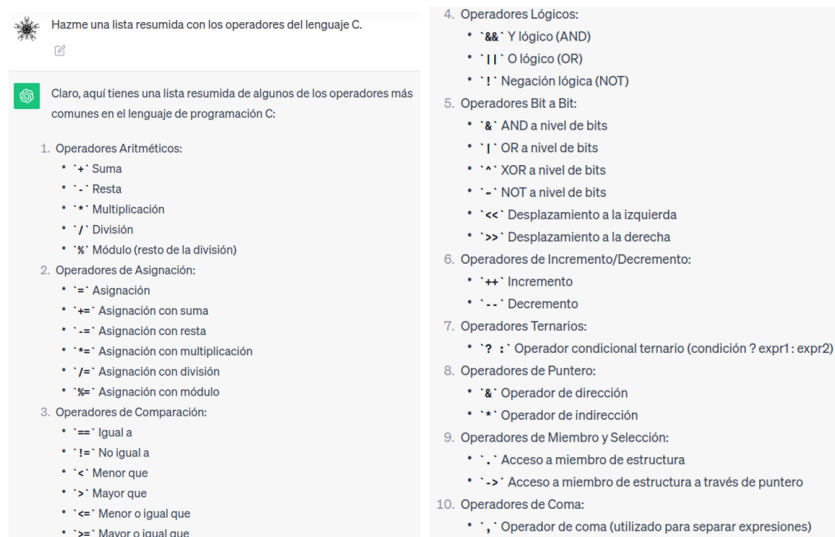


Figura 1: Exploración de la teoría, ejemplificada con la búsqueda de operadores en el lenguaje C.

En este caso particular, la respuesta de la IA generativa excluye algunos operadores que sí se pueden encontrar en la bibliografía, como son el operador unario `+` de identidad, el unario `-` de opuesto aritmético, el operador de acceso a miembro o subíndice `[]`, el operador de *casting* (`tipo`) o el operador `sizeof`. Aunque algunos puedan parecer menos importantes o secundarios, todos ellos forman parte del temario de la asignatura y son relevantes para la adquisición de, al menos, una capacidad de lectura y comprensión de los códigos en lenguaje C que el alumnado debe adquirir. Incluso, en el caso aparentemente trivial del operador unario identidad (`+`), se observa en este trabajo cómo su ignorancia puede estar relacionada con un error real del alumnado.

De este patrón se espera que el alumnado extraiga la conclusión de que es preferible abordar las consultas de carácter estructural general con criterios bibliográficos solventes, situando la IA generativa en un nivel todavía más cuestionable que las fuentes tradicionalmente denominadas «grises».

La Fig. 2 muestra otro de los patrones introducidos al alumnado: la «explicación de un contenido concreto». Se trata también de una pregunta abierta, como en el patrón anterior, aunque se diferencia en que su ámbito está mucho más acotado, lo que tiene implicaciones importantes con relación al grado de corrección que pueda tener una posible respuesta. En lugar de estructural y enumerativa, una buena respuesta tendrá un contenido selectivo, didáctico y ejemplificador. El grado de concreción de la pregunta puede escapar también, en general, a muchas de las referencias indicadas para el caso anterior. Son estas características las que motivan una consideración diferente para este tipo de patrón.

Explicame la función fabs del lenguaje C.

La función `fabs` en el lenguaje de programación C se utiliza para calcular el valor absoluto de un número de tipo flotante (por ejemplo, `float` o `double`). El valor absoluto de un número es su magnitud sin tener en cuenta su signo, es decir, convierte cualquier número negativo en su equivalente positivo y deja inalterados los números positivos.

La función `fabs` está definida en la biblioteca estándar de C, por lo que debes incluir la cabecera `<math.h>` en tu programa para poder utilizarla. Su prototipo es el siguiente:

```
c
#include <math.h>

double fabs(double x);
```

Donde:

- `x` es el número del cual deseas calcular el valor absoluto.
- La función devuelve un valor de tipo `double` que representa el valor absoluto de `x`.

Aquí tienes un ejemplo de cómo se utiliza la función `fabs` en un programa C.

```
c
#include <stdio.h>
#include <math.h>

int main() {
    double numero = -7.5;
    double valor_absoluto = fabs(numero);

    printf("El valor absoluto de %lf es %lf\n", numero, valor_absoluto);

    return 0;
}
```

Este programa imprimirá:

```
El valor absoluto de -7.500000 es 7.500000
```

Como puedes ver, la función `fabs` toma un número real como argumento y devuelve su valor absoluto, asegurándose de que siempre sea un número positivo o cero. Esto es útil en diversas situaciones, como cálculos matemáticos, manipulación de datos y control de flujo en programas C.

Figura 2: Exploración de un contenido concreto, ejemplificada con la función `fabs`.

En el ejemplo se muestra una solicitud de descripción de `fabs`, una implementación de la función valor absoluto para argumentos de tipo flotante de precisión doble (`double`). En la explicación devuelta por el sistema se pueden identificar los elementos que se encuentran habitualmente en un manual, incluyendo una descripción general, la información de los parámetros de entrada y el tipo de salida, así como un ejemplo de uso. La explicación es, en líneas generales, correcta, aunque describe su uso como orientado tanto al tipo `float` como al `double`, cuando para el primero existe una función dedicada que es `fabsf`. Sería también interesante comparar con otras funciones similares como `abs` (que puede llevar a confusión al pertenecer a la biblioteca `stdlib` en lugar de `math`).

A partir de este patrón, también se espera transmitir el mensaje de prevención al alumnado. Ante este tipo de situaciones, puede ser más complejo localizar una fuente confiable, pero, en caso de recurrir a la IA generativa, es preciso analizar con juicio crítico los resultados devueltos.

La Fig. 3 muestra otro de los patrones introducidos al alumnado: la «resolución de un problema concreto». Este suele ser el patrón que más preocupa al profesorado, especialmente con relación a las actividades evaluables. La valoración de la competencia de un modelo de IA generativa en la resolución de ejercicios es una cuestión compleja, cuyo estudio sistemático queda fuera del alcance de este trabajo, pero que sería recomendable para el profesorado interesado en conocer la validez de una evaluación en el caso de que el alumnado pudiera servirse de ella para la realización de dicha actividad sin esfuerzo intelectual o creativo propio. Lo único que se anota al respecto es que, de acuerdo con las observaciones, depende del área y tipo de pregunta (por ejemplo, se observa una gran diferencia entre la implementación de programas cortos, que realiza con bastante éxito, y la resolución de preguntas conceptuales en formato de opción múltiple, especialmente en el contenido relacionado con el uso de punteros, donde el desempeño



Figura 3: Resolución de un problema concreto, ejemplificado con el problema de la suma de números enteros en un rango.

es nefasto).

La solución propuesta aquí por la IA generativa es completamente correcta. Existen algunos indicios que apuntan a un estilo de codificación que se aleja del habitualmente mostrado por los estudiantes de la asignatura, especialmente en el uso de variables locales al bucle (una característica de la versión del lenguaje C99 que no se enseña en la docencia), y la inclusión frecuente de comentarios de una línea, incluso para cuestiones triviales. Estos rasgos podrían ser objeto de estudio para el escenario antes descrito relacionado con las tareas evaluables; sin embargo, se debe notar también que son sencillos de esquivar modificando el *prompt* con que se solicitan o reescribiendo manualmente el código.

En la exposición realizada a los alumnos, se simuló dar carpetazo al patrón con solo la muestra del resultado, reflexionando justo después sobre que esto no había realizado ningún aporte real al proceso de aprendizaje. El proceso de confrontar un ejercicio concreto con una herramienta de IA generativa, procurando que esto refuerce el aprendizaje, es también posible, pero requiere de un ejercicio de análisis. Por ello, es preciso comprender todo el código propuesto por la herramienta, para lo que se pueden servir de estrategias conversacionales como la solicitud de explicaciones, el incremento del número de comentarios o la realización de pasos intermedios que faciliten la asimilación. Otras estrategias de solución, como la solicitud de una mera esquematización en lugar de solución completa o la corrección de una implementación parcial (ver más abajo), pueden resultar también beneficiosas.

En nuestra opinión, la lectura de ejercicios resueltos no puede sustituir por completo la realización autónoma de ejercicios; igual que, en la enseñanza de lenguas naturales, el desarrollo de la comprensión no garantiza por sí un desarrollo correcto de la producción oral o escrita. El riesgo de que la IA generativa transforme las actividades de producción en actividades de comprensión debe ser combatido con estrategias de aprendizaje como las presentadas.

La Fig. 4 muestra otro de los patrones introducidos al alumnado: la «búsque-

```
❄ ¿Encuentras algún fallo en el siguiente programa?  
  
#include <stdio.h>  
  
int main(){  
    int x=1;  
    x += 1;  
    printf("x debería valer 2, pero vale %d", x);  
}
```

Si, hay un fallo en el programa. La línea que dice 'x += 1;' no realiza una operación de suma, como podría parecer a primera vista. En realidad, esto está utilizando el operador de asignación '=' junto con el operador de incremento unario '+', por lo que simplemente asigna el valor positivo de 1 a la variable 'x'. Esto significa que 'x' se establecerá en 1 en lugar de 2.

Para corregir el programa y sumar 1 a la variable 'x', debes cambiar la línea problemática a 'x += 1;' o simplemente 'x++;'. Aquí tienes la corrección:

```
c  
#include <stdio.h>  
  
int main() {  
    int x = 1;  
    x += 1; // 0 x++;  
    printf("x debería valer 2, pero vale %d", x);  
}
```

Con estas correcciones, el programa imprimirá "x debería valer 2, pero vale 2" como se espera.

Figura 4: Búsqueda de errores, ejemplificada con la permutación del orden de caracteres del operador de asignación de suma (+=).

da de errores». En el contexto de la asignatura, esta constituye una de las situaciones más frecuentes durante las prácticas. Por el desconocimiento del alumnado en las etapas iniciales de formación, los errores pueden ser difíciles de localizar por deberse a situaciones que nunca le ocurrirían a un programador experimentado, en algunas ocasiones difíciles de detectar visualmente por cuestiones tipográficas o visuales.

En el ejemplo seleccionado, donde se permutaron los dos caracteres del operador de asignación de suma (+=), se produce como resultado una asignación trivial, relacionada con el operador unario identidad descrito arriba.

La Fig. 5 muestra otro de los patrones introducidos al alumnado: la «explicación de una salida del compilador». Los compiladores e intérpretes incluyen la capacidad de orientar sobre los errores en el código, pero su interpretación hasta llegar a la comprensión de la naturaleza del problema puede ser compleja. En la actualidad existe una cierta tendencia a valorar la importancia del papel que juega la notificación de errores por parte de compiladores e intérpretes para la localización de *bugs* [3], y la diferencia puede ser determinante a la hora escoger entre un sistema u otro [5].

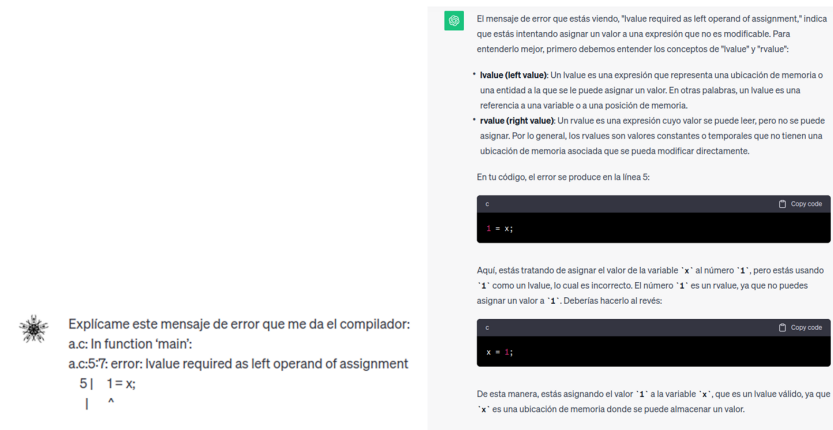


Figura 5: Explicación de una salida del compilador, ejemplificada con la permutación de los operandos en una asignación.

En el ejemplo se observa la interpretación del mensaje de error del compilador ante una situación en que se han invertido los operandos en una asignación, esto es, se ha escrito $1 = x$ en lugar de $x = 1$, lo que no tiene sentido dada la naturaleza asimétrica de la asignación. La explicación del compilador involucra los conceptos de *lvalue* y *rvalue*, demasiado avanzados para el nivel del alumnado, pero que pueden encontrarse en esta situación, y que ocasionalmente sea inevitable explicarlos en el marco de las prácticas. Para este tipo de situaciones complicadas, los estudiantes pueden servirse de las aplicaciones de IA generativa con el fin de complementar su estudio personal.

La Fig. 6 muestra el último de los patrones introducidos al alumnado: la «corrección de estilo». La adquisición de las competencias adecuadas en torno a la programación requiere no solo de la capacidad de producir código funcional, sino también legible y mantenible. Cuestiones de estilo como la indentación, la posición de delimitadores como las llaves, los espacios entre operadores o las convenciones para nombrar variables impactan esa legibilidad del código.



Mejora el estilo del siguiente programa:

```
#include <stdio.h>
#include <math.h>
int main(){
    int x;

    puts("escribe un numero");
    scanf("%d",&x);
    printf("el numero es %d,su cuadrado %d y su cubo %d",
        x,x*x,
        x*x*x);
    // Fin del programa
}
```

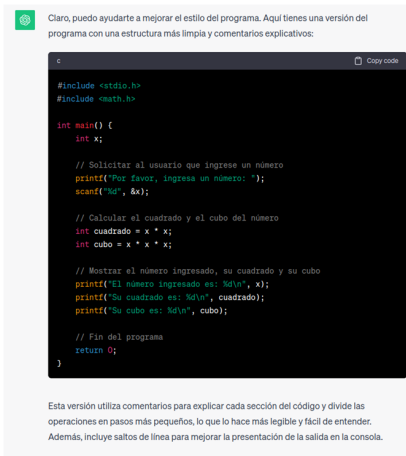


Figura 6: Corrección de estilo de código.

En el ejemplo mostrado se podían localizar defectos de estilo, que, en muchos casos, la IA generativa ha corregido, a pesar de lo generalista que ha sido el *prompt* de solicitud. El trabajo tampoco es perfecto, particularmente porque ha mantenido la inclusión de la biblioteca `math`, que no utiliza realmente. Algunas otras cuestiones también pueden estar sujetas a opinión, por ejemplo, si es preferible introducir variables intermedias para resultados que solo se utilizan una vez o mostrarlos por pantalla directamente, y si algunos comentarios son tan triviales que sería preferible obviarlos. El estilo de indentación es otra cuestión para la que muchos entornos de desarrollo integrados implementan *linters*, bien de forma directa o mediante complementos, que ajusten los programas a las normas de estilo.

El mensaje para trasladar al alumnado en torno a esta cuestión es que, en aquellas tareas mecánicas para la que ya existen programas específicos, aunque la IA generativa las integre en un marco de capacidades más general, la solución programática es más confiable y, por tanto, preferible.

3. Análisis de los resultados

Para conocer el impacto de la experiencia en los estudiantes, se les realizó una encuesta al final del año académico, cuyos resultados principales se presentan en este apartado.

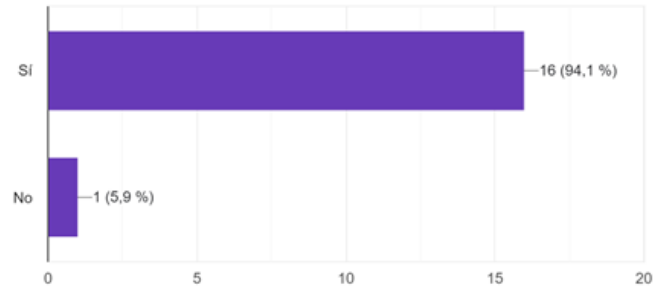
Para facilitar la lectura, alternaremos directamente cada uno de los resultados con los análisis interpretativos correspondientes. En las gráficas con escalas ordinales, el valor 1 significa mínimo grado de adscripción en la respuesta (no, nunca, totalmente en desacuerdo...) y el 5, el máximo (sí, mucho, totalmente de acuerdo...).

En primer lugar, preguntamos a los estudiantes acerca de su conocimiento

previo de la herramienta, ante lo que se obtuvieron los resultados que siguen.

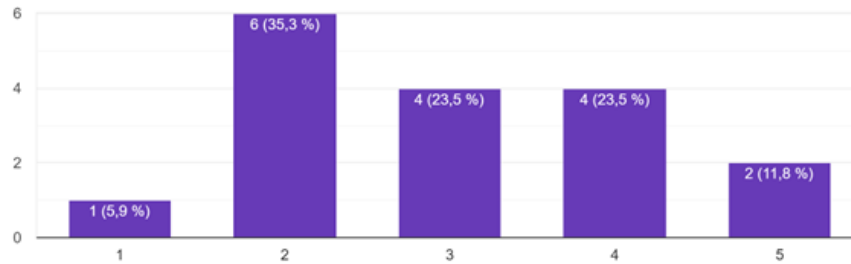
¿Conocías ChatGPT antes de iniciar este año académico?

17 respuestas



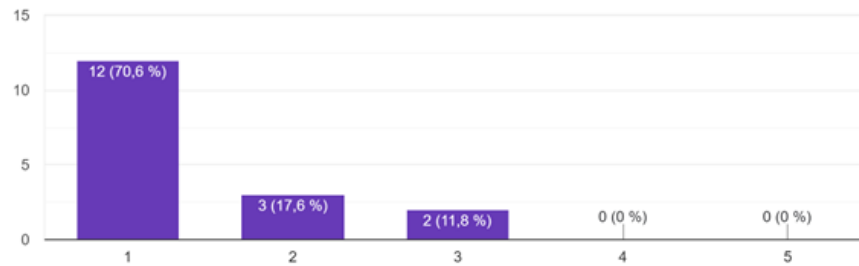
¿Habías usado ChatGPT antes de iniciar este año académico?

17 respuestas



¿Habías usado ChatGPT para generar código antes de iniciar este año académico?

17 respuestas



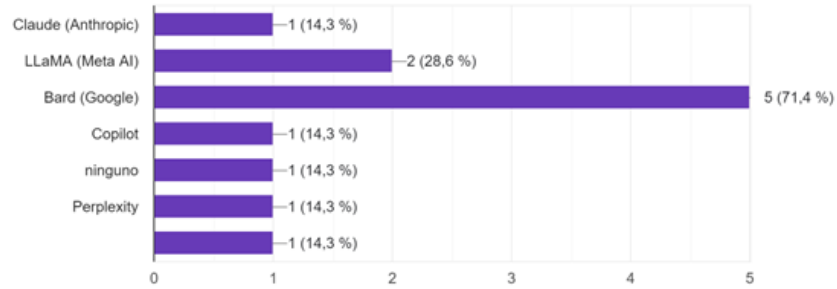
En conclusión, casi todos los alumnos conocían la herramienta ChatGPT y la habían probado con anterioridad, aunque generalmente poco. Ninguno de los casos parecía haberlo empleado para la tarea específica de generación de código,

con alguna excepción que lo hizo de manera todavía infrecuente.

A continuación se encuestó a los alumnos acerca de las diferentes alternativas, incluyendo opciones de respuesta abierta. Los resultados fueron los que siguen.

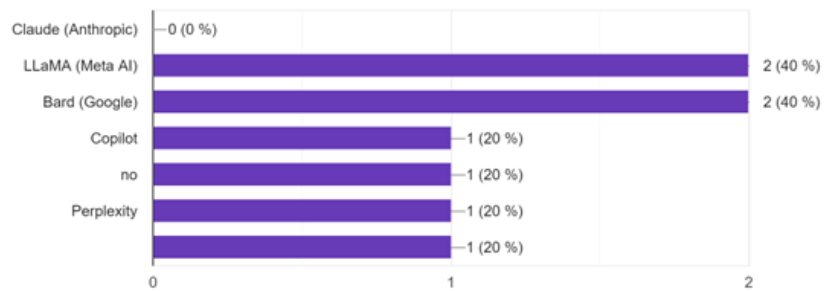
¿Conocías alguno de estos otros chatbots basados en grandes modelos del lenguaje?

7 respuestas



¿Habías usado alguno de estos otros chatbots basados en grandes modelos del lenguaje?

5 respuestas

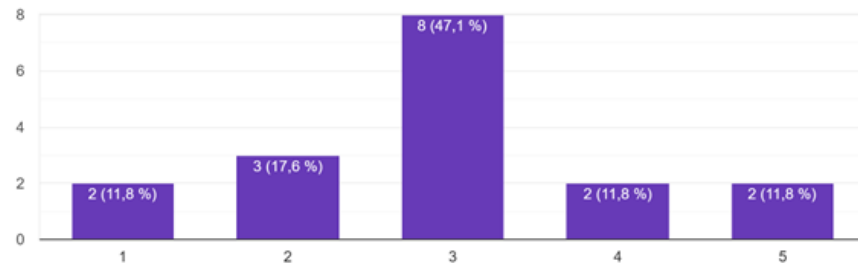


En conclusión, el conocimiento y uso de modelos alternativos era muy reducido.

Respecto al uso de la herramienta durante el año académico, centrándonos en el caso de la generación de código, se encuestó como sigue al alumnado.

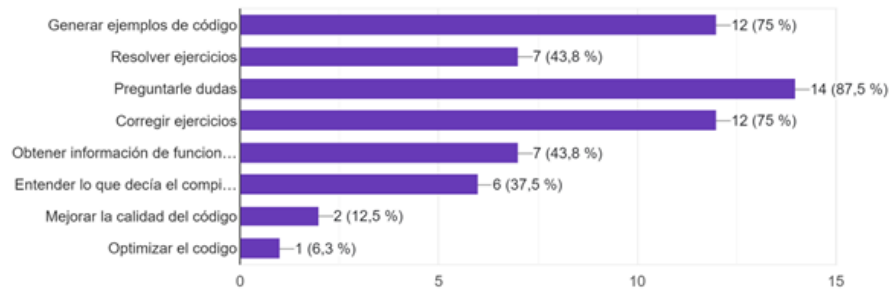
¿Has usado ChatGPT para generar código durante las asignaturas de este año académico?

17 respuestas



¿Qué usos de ChatGPT has realizado?

16 respuestas

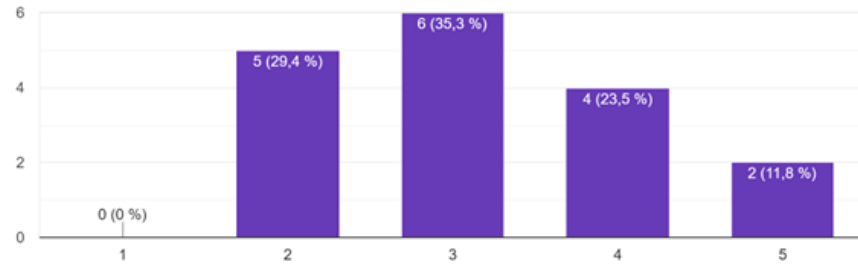


Una muy pequeña parte de los encuestados indica no haber llegado a emplear la herramienta nunca, aunque tampoco ha sido común el uso frecuente. La mayoría de alumnos indican un nivel de uso intermedio de la herramienta. Los usos más usados han sido la generación de ejemplos de código, la consulta de dudas y la solicitud de corrección de ejercicios.

Finalmente, las restantes preguntas, cuyos resultados se muestran a continuación, han servido para conocer la opinión final de la herramienta en el contexto de la programación, tras la experiencia en el año académico.

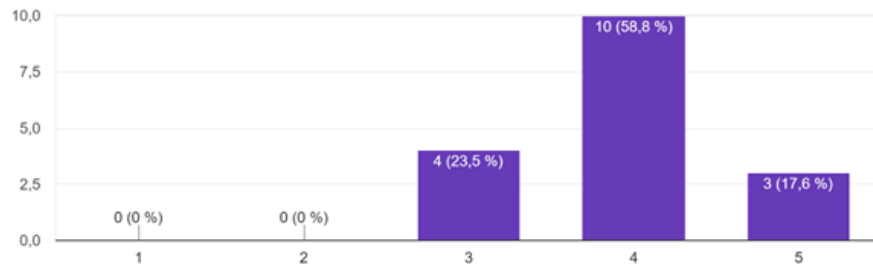
¿Crees que ChatGPT es adecuado para generar código?

17 respuestas



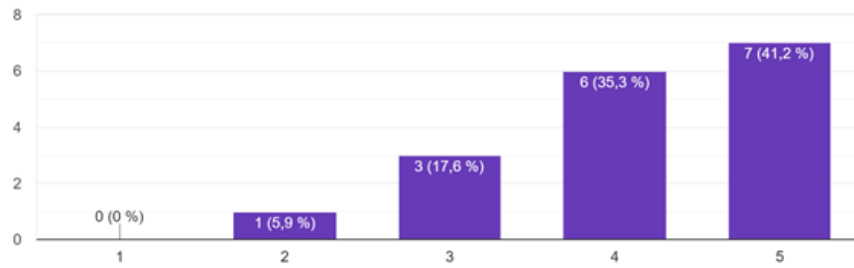
¿Crees que ChatGPT es adecuado para aprender programación?

17 respuestas



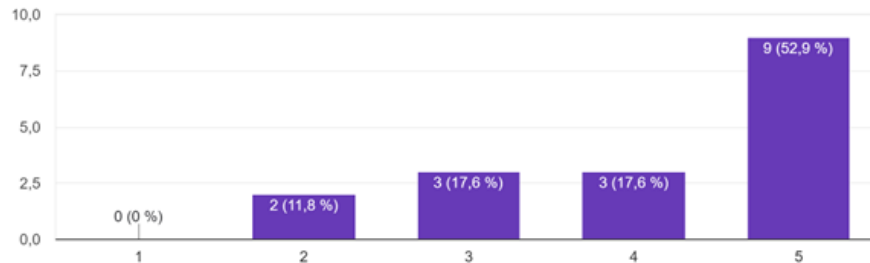
¿Crees que los profesores deberían dar formación sobre ChatGPT y otros sistemas similares en el contexto de la programación?

17 respuestas



¿Crees que usarás ChatGPT u otros sistemas similares para programar en el futuro?

17 respuestas



Como se puede observar, son minoría los que piensan que no es muy adecuado el empleo de ChatGPT para la generación de código, aunque la mayoría de las opiniones se agrupan en la región intermedia, sin pleno convencimiento. Esta observación debe contextualizarse en que el alumnado se ha visto expuesto principalmente a ejercicios sencillos de programación en C, por lo que los resultados no se deben extrapolar sin cuidado a otros dominios.

Respecto a la utilidad para el aprendizaje en sí de programación, sí es mucho más positivo el resultado. Esto evidencia que, al menos en esta etapa de la formación educativa, lo perciben más como un medio didáctico al servicio del aprendizaje que como un «solucionador de problemas». Esta consideración también se evidencia en su claro posicionamiento a favor de que el profesorado de programación proporcione formación específica para este contexto.

Finalmente, también es mayoritaria la expresión de intención de futuro uso de la herramienta. Esto muestra que su presencia en las aulas es una cuestión inevitable a la que el profesorado ha de adaptarse.

4. Conclusiones

Como conclusión general, el proyecto ha permitido llevar a los estudiantes herramientas para manejar de forma más adecuada la inteligencia artificial generativa. El éxito está corroborado por la percepción de los alumnos, según los datos de la encuesta recogida. Es especialmente importante destacar, dentro de estos resultados, que la mayoría del alumnado ha indicado que espera usar mucho la herramienta ChatGPT. Además, la mayoría también destaca el valor didáctico de la herramienta por encima del de la capacidad de generación de código. También la mayoría reporta, con bastante o mucho convencimiento, que el profesorado debería proporcionar formación acerca de este tipo de herramientas.

Habida cuenta de los resultados de la aplicación en esta anualidad, el equipo de profesores tiene previsto mantener el seminario de formación en la próxima anualidad. Los resultados obtenidos también han sido interesantes, y se pueden

confrontar con los que se han obtenido en contribuciones en otros campos con las que sería interesante la comparación. Si las circunstancias económicas nos lo permiten, nos gustaría producir alguna contribución con los resultados del estudio, lo que, lamentablemente, no ha sido posible al no recibir financiación el proyecto. También tenemos intención de realizar un estudio longitudinal durante los cursos venideros para entender cómo los cambios generacionales y la evolución de las herramientas de IA generativa afectarán a su uso en nuestro contexto.

Referencias

- [1] cplusplus.com. <https://cplusplus.com>. (Último acceso: 28/06/2024).
- [2] cppreference.com. <https://en.cppreference.com/w/>. (Último acceso: 28/06/2024).
- [3] Pablo Galindo, Batuhan Taskaya, and Ammar Askar. Pep 657 – include fine grained error locations in tracebacks — peps.python.org. <https://peps.python.org/pep-0657/>. (Último acceso: 28/06/2024).
- [4] Francisco J. García Peñalvo, Juan Andrés Hernández Simón, Roberto Therón Sánchez, Vivian Félix López Batista, and Iván Álvarez Navia. *Programación en C*. Departamento de Informática y Automática, Universidad de Salamanca, 2007.
- [5] GCC Wiki. ClangDiagnosticsComparison - GCC Wiki. <https://gcc.gnu.org/wiki/ClangDiagnosticsComparison>. (Último acceso: 28/06/2024).
- [6] Brian W Kernighan and Dennis M Ritchie. *The C programming language*. Pearson Education, 2002.