# Diseño e Implementación de una herramienta en Studium para evitar los plagios en asignaturas de Ingeniería - ID2017/076

PROGRAMA DE MEJORA DE LA CALIDAD – PLAN ESTRATEGICO GENERAL 2013-2018

**MEMORIA DE JUSTIFICACIÓN**

26 de junio de 2018

Autor: Gabriel Villarrubia Gonzalez

# Índice

# Diseño e Implementación de una herramienta en Studium para evitar los plagios en asignaturas de Ingeniería - ID2017/076

PROGRAMA DE MEJORA DE LA CALIDAD – PLAN ESTRATEGICO GENERAL 2013-2018

## Miembros del equipo de trabajo:

| NIF | Nombre y apellidos | E-mail |
|---|---|---|
| 76125754D | Juan Francisco De Paz Santana | fcofds@usal.es |
| 52415611Z | Alfonso González Briones | alfongogb@usal.es |
| 12420865Z | Vivian Félix López Batista | vivian@usal.es |
| 70875798X | Alberto López Barriuso | albarriuso@usal.es |
| 07973126S | Ana Belén Gil | abg@usal.es |
| 70978310B | Juan Manuel Corchado Rodríguez | corchado@usal.es |
| 70903746J | María Navarro Cáceres | maria90@usal.es |
| 7833143X | Pastora Isabel Vega Cruz | pvega@usal.es |
| 07848520T | Angel Luis Sanchez Lázaro | alsl@usal.es |

## Motivación de este trabajo:

Según la Real Academia Española plagiar "es copiar en lo sustancial obras ajenas, dándolas como propias". Al igual que la Real Academia Española, la Doctrina del Tribunal Supremo entiende por plagio "todo aquello que supone copiar obras ajenas en lo sustancial". Así mismo, amplía la definición del concepto al presentar el plagio "como una actividad material mecanizada, poco creativa y carente de originalidad, aunque aporte cierta manifestación de ingenio", concluyendo que "el concepto de plagio ha de referirse a las coincidencias estructurales básicas y fundamentales y no a las accesorias, añadidas, superpuestas o modificaciones trascendentales" (STS núm. 12/1995 de 28 de enero).

El plagio abarca dos puntos fundamentales: el primero de ellos es la copia total o parcial de una obra o trabajo sin ningún tipo de autorización. Pero también debe ser considerado plagio la presentación de una obra ajena como propia, suplantando la verdadera autoría de su creador. Cada vez un mayor número de instituciones académicas recurren a nuevos programas para detectar los trabajos copiados por sus alumnos.

La herramienta que actualmente dispone la Universidad de Salamanca en el momento de elaborar este documento se llama Turnitin, una utilidad muy potente y completa para detectar el plagio de los alumnos. Cuenta con una gran base de datos que permite contrastar la información que recibe y tiene integrada una herramienta de evaluación que permite incluir rúbricas y generar informes. Cabe destacar que, actualmente, esta herramienta requiere el pago de una licencia por su uso. Además, la principal desventaja que supone para los profesores de Ingeniería es la siguiente: Turnitin es un software que únicamente puede detectar paralelismos textuales, esto es válido para buscar coincidencias o apariciones de un texto en otros trabajos o en páginas de internet, sin embargo, no es una utilidad válida para comparar ejercicios o prácticas donde intervienen los lenguajes de programación. A continuación, visualicemos un ejemplo en la Figura 1.

FIGURA 1 EJEMPLO PLAGIO

```
import java.util.*;
import java.awt.*;
import java.awt.image.*;

public class Jumpbox extends Frame implements Runnable {

    Image smile,wince,offImage;
    int count = 0,size,x0,y0,xB,yB,currentBox,result=0,colors[]=new int[4];
    Long tempTime;
    long gameTime;
    Random random = new Random();
    Thread animationThread;
    Graphics offGraphics=null;
    boolean happyFace,threadon=false;

    public Jumpbox(String title, String[] args) {
        super(title);

        if (args.length == 0){
            tempTime = new Long("15");
            gameTime = 15;
        }
        else {
            try {
                tempTime = new Long(args[0]);
                gameTime = tempTime.longValue();
            } catch (NumberFormatException ioe) {
                System.out.println("ERROR: usage: Jumpbox <integer>");
                System.exit(0);
            }
        }
    }

    gameTime = System.currentTimeMillis() + gameTime*1000;
```

```
import java.util.*;
import java.awt.*;
import java.awt.image.*;

public class Jumpbox extends Frame implements Runnable {

    Image smile, nosmile, offImage;
    Graphics offGraphics = null;
    int sizeImg;
    int nrChange = 0, xOldBox, yOldBox, xBox = 0, yBox = 0, typeBox, points = 0, cBc
    Long gTime;
    long gameTime;
    Random rnd = new Random(System.currentTimeMillis());
    Thread animationThread;
    boolean bSmile, threadRunning = false;

    public Jumpbox(String title, String[] args) {
        super(title);
        if (args.length == 0) {
            gameTime = 15;
            gTime = new Long(15);
        } else {
            try {
                gTime = new Long(args[0]);
                gameTime = gTime.longValue();
            } catch (NumberFormatException ioe) {
                System.out.println("Jumpbox: usage: Jumpbox <integer
                System.exit(0);
            }
        }

        gameTime = System.currentTimeMillis() + gameTime*1000;
```

Un software anti-copia, que realice un *matching* por palabras, daría falsos negativos, ya que las variables o los textos son diferentes. Este tipo de *matching* es cómo funciona el software ya desplegado en el moodle, no siendo útil para las asignaturas donde se evalúan ejercicios o prácticas basadas en un lenguaje de programación. Sin embargo, si se implementa una solución que interprete el código y analice su funcionalidad, se informaría al profesor de que el código es sospechoso ya que ambos ejercicios son una copia idéntica y uno de los alumnos ha intentado engañar al profesor simplemente modificando el nombre de las variables.

Existen utilidades que pueden ayudar a detectar este tipo de plagios, sin embargo, son utilidades de pago, que no están integradas en la plataforma Studium o que únicamente valen para un lenguaje de programación concreto. La diversidad de asignaturas existentes en la Universidad de Salamanca donde se explican diferentes lenguajes de programación y se requieren diferentes pruebas o ejercicios como prueba de evaluación, hace que sea realmente difícil encontrar una solución válida para todas las asignaturas.

El proyecto de innovación docente que se pretende justificar consiste en implementar una solución válida para todos los lenguajes de programación de las asignaturas de grado, con el objetivo de que se integre y se desarrolle una solución basada en Studium. El profesor al crear una tarea o entrega activará el sistema anticopia, de una forma sencilla. La solución estará basada en software libre con el objetivo de minimizar los costes. Los alumnos siempre y cuando quiera el profesor podrán obtener un informe sobre el análisis del sistema anticopia, de forma que podrían visualizar la similitud de su ejercicio respecto a sus compañeros. Esta funcionalidad será opcional y únicamente estaría disponible después del envío de la tarea. Es de vital importancia investigar y desarrollar nuevas respuestas tecnológicas para asegurar una calidad de enseñanza óptima, penalizando a la gente que realiza trampas y motivando a la gente que realiza un esfuerzo de verdad.

## Objetivos conseguidos durante la realización de este proyecto:

| # | OBJETIVO | DESCRIPCIÓN |
|---|---|---|
| **01** | Especificaciones y definición de la herramienta | *Descripción*: Concepción y elaboración del diseño de la herramienta antiplagio.<br><br>*Resultado*: Diseño y elaboración de los objetivos de la herramienta antiplagio. |
| **01.1** | Especificación de requisitos funcionales | *Descripción*: Elicitación de los contenidos funcionales que la herramienta debe cumplir en el proceso de desarrollo.<br><br>*Resultado*: Conjunto de requisitos funcionales. |
| **01.2** | Especificación de requisitos no funcionales y de interoperabilidad en Moodle | *Descripción*: Análisis de los requisitos necesarios para el correcto desarrollo de la herramienta y su incorporación al Moodle institucional - Studium.<br><br>*Resultado*: Conjunto de requisito no funcionales y medidas para la correcta incorporación de la herramienta en Moodle. |
| **01.3** | Especificación de requisitos de seguridad | *Descripción*: Análisis de requisitos de seguridad necesarios para impedir la intrusión de archivos que dañen el funcionamiento.<br><br>*Resultado*: Análisis de requisitos de seguridad para evitar amenazas y alteraciones en el sistema. |
| **01.4** | Diseño de la técnica anti-plagio | *Descripción*: Análisis de las técnicas existen y desarrollo de la técnica que permita la correcta validación de las practicas tanto en lenguajes naturales como en leguajes de programación.<br><br>*Resultado*: Estado del arte de técnicas anti-plagio y concepción del diseño aplicado en la herramienta. |
| **02** | Elaboración de la herramienta en formato plugin | *Descripción*: Incorporación de la técnica anti-plagio en una herramienta en formato plugin.<br><br>*Resultado*: Plugin con formato Moodle para su incorporación en el Campus Studium. |
| **02.1** | Interoperabilidad con el sistema existente implantado (Studium) | *Descripción:* Analizar y diseñar los archivos de configuración para la incorporación y reconocimiento en plataformas Moodle.<br><br>Resultado: Mejora en la interoperabilidad entre plataformas como moodle y software de pagio. |
| **02.2** | Desarrollo de la técnica como Plugin para su incorporación a Moodle (Studium) | *Descripción:* Desarrollo de la herramienta anti-plagio de matching en formato plugin Moodle.<br><br>*Resultado:* Archivos PHP que aplican la técnica de anti-plagio en el módulo de entrega de tareas. |
| **03** | Instalación y refinamiento | *Descripción*: Instalación del plugin en entorno Moodle con idéntica configuración al campus institucional de la Universidad de Salamanca.<br><br>*Resultado*: Versión alfa del plugin, para el testeo y realización de pruebas en entornos simulados. |

| | | |
|---|---|---|
| **03.1** | Montaje de un entorno Moodle de pruebas | *Descripción*: Descarga de paquetes de entorno Moodle, instalación y despliegue en Servido Red y aplicación accesible mediante dominio.<br><br>*Resultado*: Moodle configurado y funcional para pruebas de la herramienta. |
| **03.2** | Recreación de un entorno real de pruebas (Entrega de tareas) | *Descripción*: Diseño de un curso con características similares a algunas de las asignaturas implicadas en el proyecto.<br><br>*Resultado*: Curso de asignatura prueba, para la entrega de prácticas y obtención de resultados. |
| **03.3** | Detección de posibles errores (Instalación o funcionamiento) | *Descripción:* Ensayo del funcionamiento de plugin, para hallar posibles defectos en configuración.<br><br>*Resultado:* Estado de la configuración del plugin en el Moodle para su corrección. |
| **03.4** | Implementación de posibles mejoras en la usabilidad de la herramienta | *Descripción*: Percepción de posibles mejoras que aumenten las características de la herramienta.<br><br>*Resultado*: Incremento de la funcionalidad del sistema. |
| **04** | Verificación de la aplicación de la técnica | *Descripción*: Estudio final de la validez del sistema.<br><br>*Resultado*: Herramienta instalada, configurada y evaluada para su correcta implantación y funcionamiento. |
| **04.1** | Revisión de la funcionalidad | *Descripción*: Batería de pruebas exhaustivas del completo funcionamiento de todos los lenguajes de programación.<br><br>*Resultado*: Documentación del desarrollo de la pruebas y test unitarios. |
| **04.2** | Ejecución y monitorización del caso de estudio | *Descripción*: Simulación de prueba real con alumnos.<br><br>*Resultado*: monitorización de la usabilidad con alumnos en curso real. |
| **04.3** | Implantación en el Moodle institucional de la Usal - Studium | *Descripción*: Implantación en entorno institucional una vez realizadas las validaciones de instalación/configuración, pruebas de uso simuladas y reales.<br><br>*Resultado*: Documentación del proyecto a nivel de desarrollo, configuración y validación. |
| **05** | Aplicación de la herramienta en escenario real | Descripción: Comenzar pruebas en cursos reales en las titulaciones de ingeniería.<br><br>Resultado: Resultados general y amplio es curso académico. |
| **05.1** | Evaluación de la herramienta en un caso de estudio real | *Descripción*: Evaluar el rendimiento en las asignaturas y de la herramienta.<br><br>*Resultado*: Obtención de resultados en asignaturas de plan académico. |

| | | |
|---|---|---|
| **05.2** | Medición del impacto mediante indicadores de eficiencia | *Descripción*: Medir si la herramienta presenta una eficiencia en el rendimiento de los alumnos.<br><br>*Resultado*: Obtención de resultados académico gracias a la implantación del proyecto. |
| **05.3** | Obtención de resultados obtenidos mediante el empleo de la aplicación | *Descripción*: Generar resultados de compromiso de los estudiantes con los planes de estudio para el análisis de resultados académicos en entregas.<br><br>*Resultado*: Documentación de la evolución de la mejora en las técnicas de estudio aplicadas por los alumnos. |

## Temporización de las tareas realizadas:

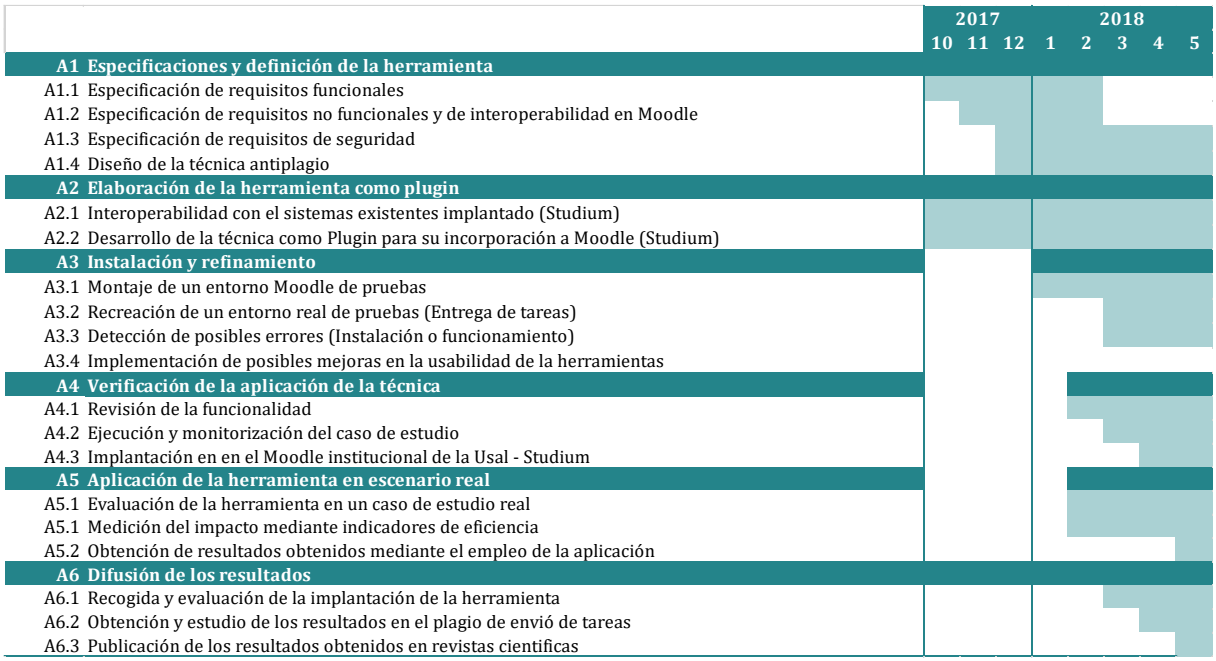



FIGURA 2 TEMPORIZACIÓN DE TAREAS

## Punto de partida, encuesta a estudiantes:

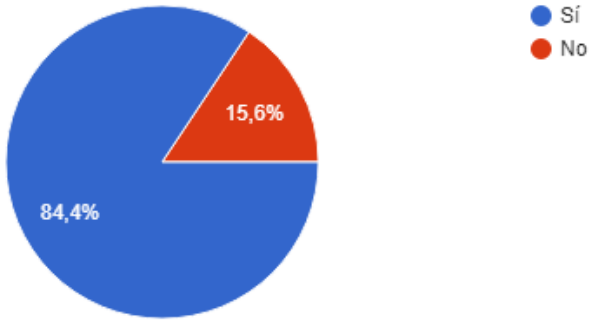
Para justificar la necesidad de implantación de una herramienta en Studium para evitar los plagios en asignaturas de Ingeniería, es de vital importancia conocer la opinión de profesores y alumnos acerca de los plagios que suceden cada día en los centros de enseñanza. Es por ello por lo que durante la realización de este proyecto se ha realizado una encuesta anónima y con carácter voluntario a personal docente y alumnos de las carreras de ingeniería con un total de 64 participantes. A continuación, se listan las preguntas que se hicieron y los resultados para cada una de ellas.

1. ¿Conoces a alguna persona que haya realizado una copia parcial o total de un trabajo?
2. ¿Has sentido que la calificación de trabajo se ha visto rebajada frente a otras person@s que se han hecho un esfuerzo menor copiando?
3. ¿Cuántos compañeros crees que copian los ejercicios prácticos de años anteriores?
4. ¿Te parecen suficientes los esfuerzos que realizan los profesores para evitar los plagios?
5. Estando en la facultad.....¿Has realizado algún plagio?
6. ¿Crees que un porcentaje de las personas que copian, no serán descubiertas y obtendrán una calificación favorable?
7. ¿Te gustaría que se mejorasen las técnicas para evitar fraudes en los trabajos de la facultad?
8. ¿Has sentido alguna vez algún tipo de frustración al conocer que otras personas que han copiado han obtenido una calificación similar a la tuya?
9. ¿Conoces alguna herramienta para comprobar si tu trabajo es parte o tiene alguna similitud con otro?
10. ¿Conoces algún compañer@ que en alguna ocasión haya pagado a un tercero para que le haga algún ejercicio/código?
11. ¿Crees que a las personas que copian se les penaliza lo suficiente al ser descubiertos?
12. ¿Has presentado alguna vez un trabajo copiado y no has sido descubierto?
13. ¿Conoces alguna estrategia para evitar ser detectado por una herramienta anti-plagio?
14. Si un profesor de la facultad implantase una nueva herramienta para detectar plagios en prácticas de informática (SQL, C, PERL.....) y le avisase al alumno de las posibles coincidencias con otros trabajos a la hora de subirla al studium..... ¿Te gustaría?

A continuación, se muestran los resultados finales de la encuesta:

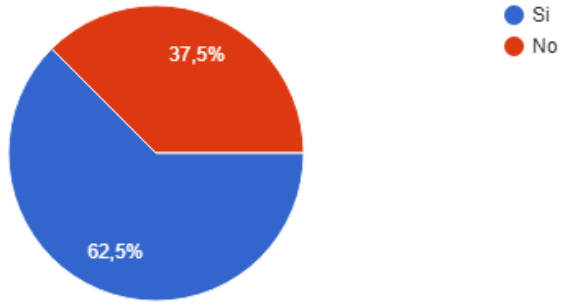
¿Conoces a alguna persona que haya realizado una copia parcial o total de un trabajo?

64 respuestas



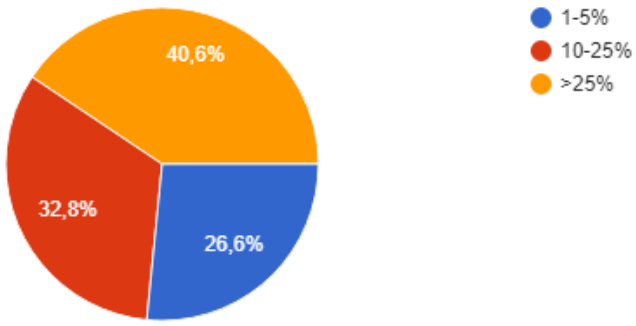

¿Has sentido que la calificación de trabajo se ha visto rebajada frente a otras person@s que se han hecho un esfuerzo menor copiando?

64 respuestas



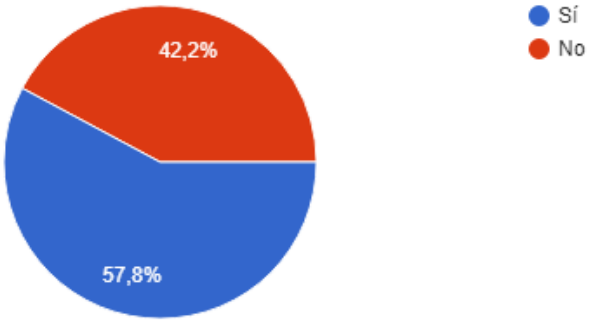

¿Cuántos compañeros crees que copian los ejercicios prácticos de años anteriores?

64 respuestas



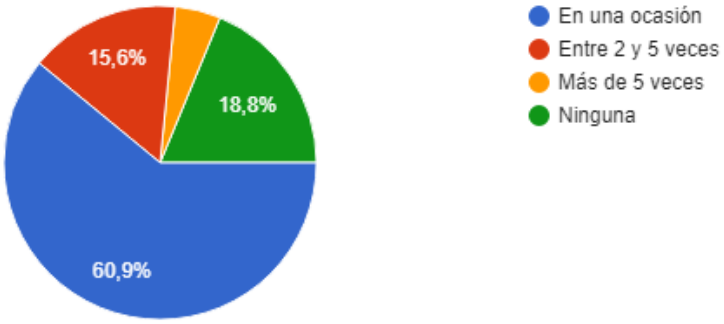
¿Te parecen suficientes los esfuerzos que realizan los profesores para evitar los plagios?

64 respuestas



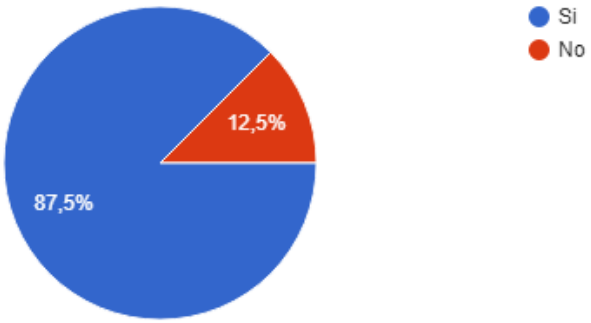

Estando en la facultad.....¿Has realizado algún plagio?

64 respuestas



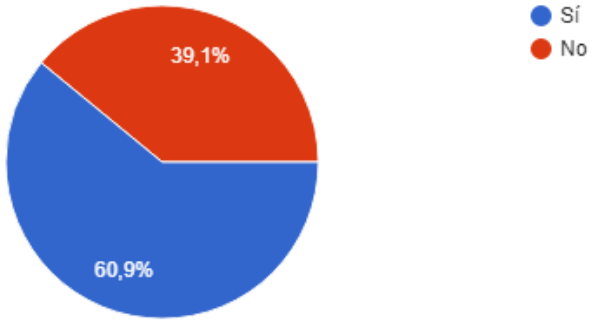

¿Crees que un porcentaje de las personas que copian, no serán descubiertas y obtendrán una calificación favorable?

64 respuestas



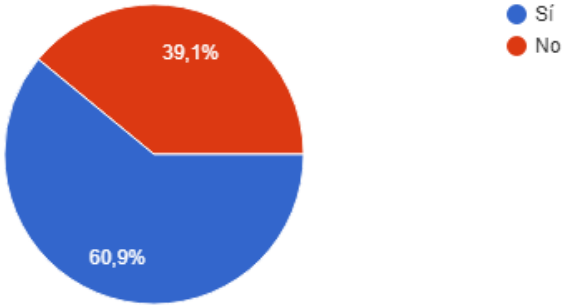
¿Te gustaría que se mejorasen las técnicas para evitar fraudes en los trabajos de la facultad?

64 respuestas



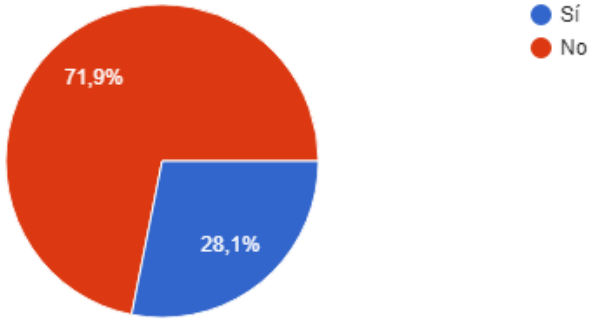

¿Has sentido alguna vez algún tipo de frustración al conocer que otras personas que han copiado han obtenido una calificación similar a la tuya?

64 respuestas



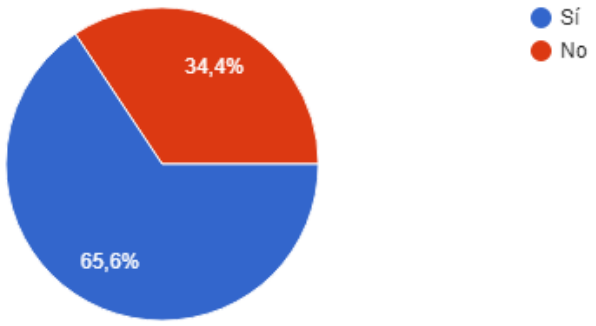

¿Conoces alguna herramienta para comprobar si tu trabajo es parte o tiene alguna similitud con otro?

64 respuestas



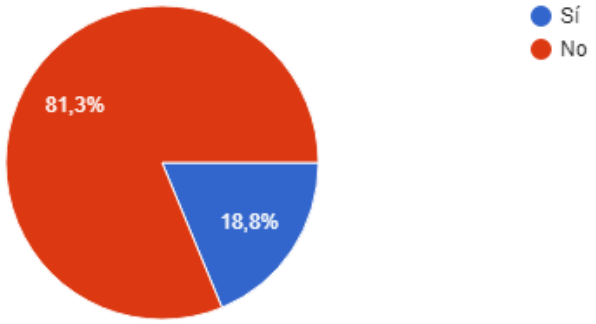
¿Conoces algún compañer@ que en alguna ocasión haya pagado a un tercero para que le haga algún ejercicio/código?

64 respuestas



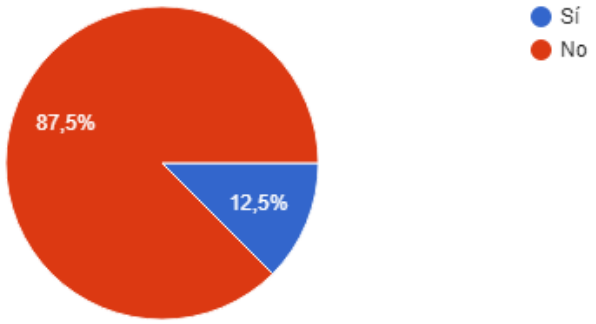

¿Has presentado alguna vez un trabajo copiado y no has sido descubierto?

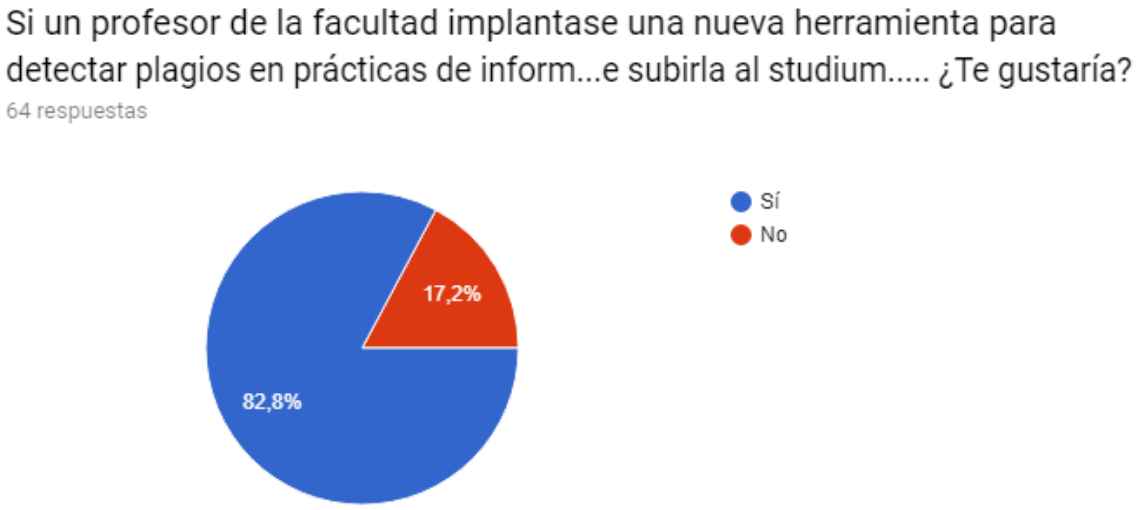
64 respuestas




¿Conoces alguna estrategia para evitar ser detectado por una herramienta anti-plagio?

64 respuestas






Observando los resultados de los encuestados, se puede argumentar que los estudiantes conocen a ciencia cierta que un gran porcentaje (85%) de trabajos son copiados y que los profesores no tienen forma de verificarlo. El (82%) está de acuerdo con que la Universidad apoye nuevas formas y técnicas que eviten estas situaciones. Además, que un alumno sea calificado positivamente sin ningún tipo de penalización al copiar un trabajo, provoca una desmotivación en los alumnos que de verdad se esfuerzan. Los estudiantes constatan que el 90% de las personas del Grado de Ingeniería Informática ha realizado algún plagio y que el 42% de los trabajos presentados en las asignaturas de prácticas son copias idénticas a compañeros de otros años.

Es por todo ello, que es de vital importancia la implantación de un sistema anti-plagios orientados a las asignaturas prácticas, donde los sistemas tradicionales basados en técnicas de matching no funcionan. La puesta en marcha de este proyecto, permite que los alumnos se encuentren más motivados y seguros al realizar sus trabajos por sus propios medios, sabiendo que, si un compañero realiza fraude, será penalizado.

## Funcionamiento Del Algoritmo:

El funcionamiento del sistema de detección de plagiarismo se basa en el trabajo realizado en el siguiente trabajo científico y que ha sido anexado al documento:

*Winnowing: Local Algorithms for Document Fingerprinting*, April 2003, DOI: 10.1145/872757.872770. Saul David SchleimerSaul David SchleimerDaniel S. WilkersonAlex Aiken

# Winnowing: Local Algorithms for Document Fingerprinting

Saul Schleimer
MSCS
University of Illinois, Chicago
saul@math.uic.edu

Daniel S. Wilkerson
Computer Science Division
UC Berkeley
dsw@cs.berkeley.edu

Alex Aiken
Computer Science Division
UC Berkeley
aiken@cs.berkeley.edu

## ABSTRACT

Digital content is for copying: quotation, revision, plagiarism, and file sharing all create copies. Document fingerprinting is concerned with accurately identifying copying, including small partial copies, within large sets of documents.

We introduce the class of *local* document fingerprinting algorithms, which seems to capture an essential property of any fingerprinting technique guaranteed to detect copies. We prove a novel lower bound on the performance of any local algorithm. We also develop *winnowing*, an efficient local fingerprinting algorithm, and show that winnowing's performance is within 33% of the lower bound. Finally, we also give experimental results on Web data, and report experience with MOSS, a widely-used plagiarism detection service.

## 1. INTRODUCTION

Digital documents are easily copied. A bit less obvious, perhaps, is the wide variety of different reasons for which digital documents are either completely or partially duplicated. People quote from each other's email and news postings in their replies. Collaborators create multiple versions of documents, each of which is closely related to its immediate predecessor. Important Web sites are mirrored. More than a few students plagiarize their homework from the Web. Many authors of conference papers engage in a similar but socially more acceptable form of text reuse in preparing journal versions of their work. Many businesses, notably in the software and entertainment industries, are based on charging for each digital copy sold.

Comparing whole document checksums is simple and suffices for reliably detecting exact copies; however, detecting partial copies is subtler. Because of its many potential applications, this second problem has received considerable attention.

Most previous techniques for detecting partial copies, which we discuss in more detail in Section 2, make use of the following idea. A *k-gram* is a contiguous substring of length $k$. Divide a document into $k$-grams, where $k$ is a parameter chosen by the user. For example, Figure 1(c) contains all the 5-grams of the string of characters in Figure 1(b). Note that there are almost as many $k$-grams as there are characters in the document, as every position in the document (except for the last $k-1$ positions) marks the beginning of a $k$-gram. Now hash each $k$-gram and select some subset of these hashes to be the document's *fingerprints*. In all practical approaches, the set of fingerprints is a small subset of the set of all $k$-gram hashes. A fingerprint also contains positional information, which we do not show, describing the document and the location within that document that the fingerprint came from. If the hash function is chosen so that the probability of collisions is very small, then whenever two documents share one or more fingerprints, it is extremely likely that they share a $k$-gram as well.

For efficiency, only a subset of the hashes should retained as the document's fingerprints. One popular approach is to choose all hashes that are *0 mod p*, for some fixed $p$. This approach is easy to implement and retains only $1/p$ of all hashes as fingerprints (Section 2). Meaningful measures of document similarity can also be derived from the number of fingerprints shared between documents [5].

A disadvantage of this method is that it gives no guarantee that matches between documents are detected: a $k$-gram shared between documents is detected only if its hash is *0 mod p*. Consider the sequence of hashes generated by hashing all $k$-grams of a file in order. Call the distance between consecutive selected fingerprints the *gap* between them. If fingerprints are selected *0 mod p*, the maximum gap between two fingerprints is unbounded and any

```
A do run run run, a do run run
```
(a) Some text from [7].

```
adorunrunrunadorunrun
```
(b) The text with irrelevant features removed.

```
adoru dorun orunr runru unrun nrunr runru
unrun nruna runad unado nador adoru dorun
orunr runru unrun
```
(c) The sequence of 5-grams derived from the text.

```
77 72 42 17 98 50 17 98 8 88 67 39 77 72 42
17 98
```
(d) A hypothetical sequence of hashes of the 5-grams.

```
72 8 88 72
```
(e) The sequence of hashes selected using *0 mod 4*.

**Figure 1: Fingerprinting some sample text.**


*SIGMOD* 2003, June 9-12, 2003, San Diego, CA.

matches inside a gap are not detected.[1] In experiments with selecting fingerprints *0 mod p* on HTML data taken from the Web, we found that gaps between fingerprints chosen *0 mod p* can be quite large. In fact, there are passages much longer than the size of the average Web page in which no hashes are *0 mod p* for reasonable values of $p$ (Section 5).

In this paper we give the first efficient algorithm for selecting the fingerprints from a sequence of hashes that guarantees that at least part of any sufficiently long match is detected. Define a *window* of size $w$ to be $w$ consecutive hashes of $k$-grams in a document ($w$ is a parameter set by the user). By selecting at least one fingerprint from every window our algorithm limits the maximum gap between fingerprints. In fact, our algorithm is guaranteed to detect at least one $k$-gram in any shared substring of length at least $w + k - 1$.

Central to our construction is the idea of a *local* algorithm (Section 4), which we believe captures the essential properties of any document fingerprinting algorithm which guarantees that matches are detected. An algorithm is local if, for every window of $w$ consecutive hashes $h_i, \ldots, h_{i+w-1}$, the algorithm decides to select one of these hashes as a fingerprint and this choice depends only on the window's contents $h_i, \ldots, h_{i+w-1}$. Note that some other of the hashes in this window may also be selected due the action of the algorithm in one of the (potentially many) overlapping windows.

With respect to a given input distribution, let the *density* of a fingerprinting algorithm be the expected proportion of hashes it selects as fingerprints. In Section 3 we introduce a particular local algorithm, the *winnowing* algorithm. We analyze its performance on random (independent uniformly distributed) data. We show that, with a given window size $w$, the density is asymptotically $2/(w + 1)$. We also prove a lower bound of $1.5/(w + 1)$ on the density of local algorithms. Thus the winnowing algorithm is within 33% of optimal.

We also report on experience with two implementations of winnowing. The first is an purely experimental framework for comparing actual performance with the theoretical predictions (Section 5). Our analysis of all fingerprinting algorithms, including that of winnowing, is based on the assumption that the sequence of hashes generated by hashing $k$-grams is independent and uniformly random. On such random data the performance of our system matches theoretical predictions perfectly. We have found, however, that there are situations where real data does not generate sufficiently random sequences of hashes. In particular, there are clusters of low-entropy strings on the Web, such as

$$000000000000000000000000\ldots$$

or more complex patterns such as

$$abbaabbaabbaabbaabbaabba\ldots$$

The important characteristic of such strings is that they have few distinct $k$-grams, and therefore few distinct hashes of $k$-grams. In the case of a long string of a single character, there is only a single $k$-gram (and hash). The straightforward winnowing algorithm selects far more fingerprints than predicted on such strings, but a simple modification of the algorithm reduces the density of fingerprints[2]. Note that other fingerprint-selection approaches are exposed to the same problem. For example, choosing hashes that are *0 mod p* means that on the string "11111111 ..." either all or none of the hashes of the string are selected as fingerprints.

We also report on experience with MOSS, a widely-used service for detecting plagiarism, primarily in programming assignments (Section 5.2).[3] Document fingerprinting has worked extremely well in this setting. Because the basic idea of hashing $k$-grams makes minimal assumptions about the form of the input, it is easy to incorporate fingerprinting for a new data format without disturbing the underlying hashing engine. False positives appear to be non-existent, and the infrequent reports of false negatives (instances of shared substrings missed by the system) have always been tracked back either to implementation bugs or user error.

[1] See Section 3.1 for a discussion of the expected gap size.
[2] See Definition 3 "Robust Winnowing" in Section 5.1

## 2. BACKGROUND AND RELATED WORK

Not all copy-detection algorithms are based on selecting fingerprints of $k$-grams. To give some basis for discussing different techniques, we list several criteria that a copy-detection algorithm should satisfy. These requirements are based on our own experience (mostly in the context of plagiarism detection), but echo many of the goals outlined in papers on related techniques.

### 2.1 Desirable properties

We believe a copy-detection algorithm should have three properties:

1. *Whitespace insensitivity* In matching text files, matches should be unaffected by such things as extra whitespace, capitalization, punctuation, etc. In other domains the notion of what strings should be equal is different—for example, in matching software text it is desirable to make matching insensitive to variable names.

2. *Noise suppression* Discovering short matches, such as the fact that the word *the* appears in two different documents, is uninteresting. Any match must be large enough to imply that the material has been copied and is not simply a common word or idiom of the language in which documents are written.

3. *Position independence* Coarse-grained permutation of the contents of a document (e.g., scrambling the order of paragraphs) should not affect the set of discovered matches. Adding to a document should not affect the set of matches in the original portion of the new document. Removing part of a document should not affect the set of matches in the portion that remains.

All schemes that we know of handle property (1) in essentially the same way. A first pass over the data transforms it to eliminate undesirable differences between documents. For example, whitespace and punctuation are removed, all letters are converted to lower case, or all variable names are replaced by the identifier "V". The exact details vary from one type of document to the next, but the essential feature is that semantic information about the document type is used to eliminate unimportant differences between documents.

Schemes based on fingerprinting $k$-grams satisfy requirement (2) by choosing $k$ to be sufficiently long that common idioms of the language have length shorter than $k$. An important assumption of this class of algorithms is that there is some threshold $k$ such that matches with length shorter than $k$ are almost always uninteresting and matches with length longer than $k$ are almost always interesting. In our experience, there is such a sharp threshold (Section 5).

The most interesting requirement is (3). Before describing methods for addressing requirement (3), we first give a short history of the use of hashes of $k$-grams in copy detection.

[3] http://www.cs.berkeley.edu/~aiken/moss.html

## 2.2 Karp-Rabin String Matching

Karp and Rabin's algorithm for fast substring matching is apparently the earliest version of fingerprinting based on $k$-grams [10]. Their problem, which was motivated by string matching problems in genetics, is to find occurrences of a particular string $s$ of length $k$ within a much longer string. The idea is to compare hashes of all $k$-grams in the long string with a hash of $s$. However, hashing strings of length $k$ is expensive for large $k$, so Karp and Rabin propose a "rolling" hash function that allows the hash for the $i+1^{st}$ $k$-gram to be computed quickly from the hash of the $i^{th}$ $k$-gram. Treat a $k$-gram $c_1 \ldots c_k$ as a $k$-digit number in some base $b$. The hash $H(c_1 \ldots c_k)$ of $c_1 \ldots c_k$ is this number:

$$c_1 * b^{k-1} + c_2 * b^{k-2} * \ldots + c_{k-1} * b + c_k$$

To compute the hash of the $k$-gram $c_2 \ldots c_{k+1}$, we need only subtract out the high-order digit, multiply by $b$, and add in the new low order digit. Thus we have the identity:

$$H(c_2 \ldots c_{k+1}) = (H(c_1 \ldots c_k) - c_1 * b^{k-1}) * b + c_{k+1}$$

Since $b^{k-1}$ is a constant, this allows each subsequent hash to be computed from the previous one with only two additions and two multiplications. Further, this identity holds when addition and multiplication are modulo some value (e.g., the size of the largest representable integer), so this method works well with standard machine arithmetic.

As an aside, this rolling hash function has a weakness. Because the values of the $c_i$ are relatively small integers, doing the addition last means that the last character only affects a few of the low-order bits of the hash. A better hash function would have each character $c_i$ potentially affect all of the hash's bits. As noted in [5], it is easy to fix this by multiplying the entire hash of the first $k$-gram by an additional $b$ and then switching the order of the multiply and add in the incremental step:

$$H'(c_2 \ldots c_{k+1}) = ((H'(c_1 \ldots c_k) - c_1 * b^k) + c_{k+1}) * b$$

## 2.3 All-to-all matching

The first scheme to apply fingerprinting to collections of documents was developed by Manber, who apparently independently discovered Karp-Rabin string matching and applied it to detecting similar files in file systems [12]. Rather than having a single candidate string to search for, in this problem we wish to compare all pairs of $k$-grams in the collection of documents.

The all-to-all nature of this comparison is a key difficulty in document fingerprinting. To illustrate, consider the problem of all-to-all matching on ASCII text. Since there is a $k$-gram for every byte of an ASCII file, and at least 4-byte hashes are needed for most interesting data sets, a naive scheme that selected all hashed $k$-grams would create an index much larger than the original documents. This is impractical for large document sets, and the obvious next step is to select some subset of the hashes to represent each document. But which hashes should be selected as fingerprints?

A simple but incorrect strategy is to select every $i^{th}$ hash of a document, but this is not robust against reordering, insertions and deletions (requirement (3) above). In fact, prepending one character to a file shifts the positions of all $k$-grams by one, which means the modified file shares none of its fingerprints with the original. Thus, any effective algorithm for choosing the fingerprints to represent a document cannot rely on the position of the fingerprints within the document.

The scheme Manber chose is to select all hashes that are *0 mod p*. In this way fingerprints are chosen independent of their position, and if two documents share a hash that is *0 mod p* it is selected in both documents. Manber found this technique worked well.

In [8], Heintze proposed choosing the $n$ smallest hashes of all $k$-grams of a document as the fingerprints of that document. By fixing the number of hashes per document, the system would be more scalable as large documents have the same number of fingerprints as small documents. This idea was later used to show that it was possible to cluster documents on the Web by similarity [6]. The price for a fixed-size fingerprint set is that only near-copies of entire documents could be detected. Documents of vastly different size could not be meaningfully compared; for example, the fingerprints of a paragraph would probably contain no fingerprints of the book that the paragraph came from. Choosing hashes *0 mod p*, on the other hand, generates variable size sets of fingerprints for documents but guarantees that all representative fingerprints for a paragraph would also be selected for the book. Broder [5] classifies these two different approaches to fingerprinting as being able to detect only *resemblance* between documents or also being able to detect *containment* between documents.

## 2.4 Other techniques

Instead of using $k$-grams, the strings to fingerprint can be chosen by looking for sentences or paragraphs, or by choosing fixed-length strings that begin with "anchor" words [4, 12]. Early versions of our system also used structure gleaned from the document to select substrings to fingerprint. The difficulty with such schemes, in our experience, is that the implementation becomes rather specific to a particular type of data. If the focus is on English text, for example, choosing sentences as the unit to hash builds in text semantics that makes it rather more difficult to later use the system to fingerprint, say, C programs, which have nothing resembling English sentences. In addition, even on text data the assumption that one can always find reasonable sentences is questionable: the input may be a document with a large table, a phone book, or Joyce's *Finnegans Wake* [9]. In our experience, using $k$-grams as the unit of hashing is much more robust than relying on common-case assumptions about the frequency of specific structure in the input.

There are approaches to copy detection not based on fingerprinting. For example, in SCAM, a well-known copy-detection system, one of the ideas that is explored is that two documents are similar if the distance between feature vectors representing the two documents is small. The features are words, and the notion of distance is a variation on standard information-retrieval measures of similarity [14].

Baker considers the problem of finding near-duplication in software and develops the notion of *parameterized matches*, or $p$-matches. Consider two strings, some letters of which are designated as parameters. The strings match if there is a renaming of parameters that makes the two strings equal. For example, if we take the parameters to be variable names, then two sections of program text could be considered equal if there was a renaming of variables that mapped one program into the other. Baker gives an algorithm for computing $p$-matches and reports on experience with an implementation in [2] and in a subsequent paper considers how to integrate these ideas with matching on $k$-grams [3].

There is an important distinction to be made between copy-detection for discrete data and for continuous data. For discrete data, such as text files and program source, after a simple suppression of the uninteresting pieces of documents, exact matching on substrings of the remainder is a useful notion. For continuous data, such as audio, video, and images, there have been a number of commercial copy-detection systems built but relatively little has been published in the open literature (an exception is [13]). The problems here are

more difficult, because very similar copies of images, for example, may have completely different bit representations, requiring a much more sophisticated first step to extract features of interest before the matching can be done.

Further afield from copy-detection, but still related, is Digital Rights Management (DRM). DRM systems seek to solve the problem of the use of intellectual property by preventing or controlling copying of documents. DRM schemes are encryption-based: the valuable content is protected by encrypting it and can only be used by those who have been granted access in the form of the decryption key. However, regardless of the copy-prevention technology chosen, users must ultimately have access to the unencrypted data somehow—otherwise they cannot use it—and as discussed in Section 1, it seems to be nearly a natural law that digital content is copied. We find ourselves in agreement with [4]: for at least some forms of digital media copy-prevention systems will have trouble ultimately succeeding. We suspect that in many environments the best one can hope for is efficient copy detection.

## 3. WINNOWING

In this section we describe and analyze the winnowing algorithm for selecting fingerprints from hashes of $k$-grams. We give an upper bound on the performance of winnowing, expressed as a trade-off between the number of fingerprints that must be selected and the shortest match that we are guaranteed to detect.

Given a set of documents, we want the find substring matches between them that satisfy two properties:

1. If there is a substring match at least as long as the *guarantee threshold*, $t$, then this match is detected, and
2. We do not detect any matches shorter than the *noise threshold*, $k$.

The constants $t$ and $k \leq t$ are chosen by the user. We avoid matching strings below the noise threshold by considering only hashes of $k$-grams. The larger $k$ is, the more confident we can be that matches between documents are not coincidental. On the other hand, larger values of $k$ also limit the sensitivity to reordering of document contents, as we cannot detect the relocation of any substring of length less than $k$. Thus, it is important to choose $k$ to be the minimum value that eliminates coincidental matches (see Section 5).

Figures 2(a)-(d) are reproduced from Figure 1 for convenience and show a sequence of hashes of 5-grams derived from some sample text.

Given a sequence of hashes $h_1 \ldots h_n$, if $n > t - k$, then at least one of the $h_i$ must be chosen to guarantee detection of all matches of length at least $t$. This suggests the following simple approach. Let the *window size* be $w = t - k + 1$. Consider the sequence of hashes $h_1 h_2 \ldots h_n$ that represents a document. Each position $1 \leq i \leq n - w + 1$ in this sequence defines a *window* of hashes $h_i \ldots h_{i+w-1}$. To maintain the guarantee it is necessary to select one hash value from every window to be a fingerprint of the document. (This is also sufficient, see Lemma 1.) We have found the following strategy works well in practice.

DEFINITION 1 (WINNOWING). *In each window select the minimum hash value. If there is more than one hash with the minimum value, select the rightmost occurrence. Now save all selected hashes as the fingerprints of the document.*

Figure 2(e) gives the windows of length four for the sequence of hashes in Figure 2(d). Each hash that is selected is shown in boldface (but only once, in the window that first selects that hash). The

```
A do run run run, a do run run
```
(a) Some text.

```
adorunrunrunadorunrun
```
(b) The text with irrelevant features removed.

```
adoru dorun orunr runru unrun nrunr runru
unrun nruna runad unado nador adoru dorun
orunr runru unrun
```
(c) The sequence of 5-grams derived from the text.

```
77 74 42 17 98 50 17 98 8 88 67 39 77 74 42
17 98
```
(d) A hypothetical sequence of hashes of the 5-grams.

```
(77, 74, 42, 17)   (74, 42, 17, 98)
(42, 17, 98, 50)   (17, 98, 50, 17)
(98, 50, 17, 98)   (50, 17, 98,  8)
(17, 98,  8, 88)   (98,  8, 88, 67)
( 8, 88, 67, 39)   (88, 67, 39, 77)
(67, 39, 77, 74)   (39, 77, 74, 42)
(77, 74, 42, 17)   (74, 42, 17, 98)
```
(e) Windows of hashes of length 4.

```
17 17 8 39 17
```
(f) Fingerprints selected by winnowing.

```
[17,3] [17,6] [8,8] [39,11] [17,15]
```
(g) Fingerprints paired with 0-base positional information.

**Figure 2: Winnowing sample text.**

intuition behind choosing the minimum hash is that the minimum hash in one window is very likely to remain the minimum hash in adjacent windows, since the odds are that the minimum of $w$ random numbers is smaller than one additional random number. Thus, many overlapping windows select the same hash, and the number of fingerprints selected is far smaller than the number of windows while still maintaining the guarantee. Figure 2(f) shows the set of fingerprints selected by winnowing in the example.

In many applications it is useful to record not only the fingerprints of a document, but also the position of the fingerprints in the document. For example, we need positional information to show the matching substrings in a user interface. An efficient implementation of winnowing also needs to retain the position of the most recently selected fingerprint. Figure 2(f) shows the set of [*fingerprint, position*] pairs for this example (the first position is numbered 0). To avoid the notational complexity of indexing all hashes with their position in the global sequence of hashes of $k$-grams of a document, we suppress most explicit references to the position of $k$-grams in documents in our presentation.

### 3.1 Expected Density

Recall that the *density* of a fingerprinting algorithm is the expected fraction of fingerprints selected from among all the hash values computed, given random input (Section 1). We now analyze the density of winnowing, which gives the trade-off between the guarantee threshold and the number of fingerprints required.

Consider the function $C$ that maps the position of each selected fingerprint to the position of the first (leftmost) window that se-

lected it in the sequence of all windows for a document. We say we are *charging* the cost of saving the fingerprint to the indicated window. The charge function is monotonic increasing — that is, if $p$ and $q$ are the positions of two selected fingerprints and $p < q$, then $C(p) < C(q)$.

To prove this, assume fingerprints are selected at distinct positions $p$ and $q$ where $p < q$ but $C(p) > C(q)$. Then both positions $p$ and $q$ are in both windows. Let $h_p$ be the hash at position $p$ and let $h_q$ be the hash at position $q$. There are two possibilities: If $h_p = h_q$ then, as $p < q$, the window $C(p)$ was not charged for $p$ nor for $q$, as $C(q) < C(p)$. If $h_p \neq h_q$ then one of $C(p)$ or $C(q)$ was not charged. These both contradict the hypothesis. We conclude that the charge function is monotonic increasing.

To proceed further recall that the sequence of hashes we are winnowing is random. We assume that the space of hash values is very large so that we can safely ignore the possibility that there is a tie for the minimum value for any small window size. We examine the soundness of this assumption in Section 5.

Consider an indicator random variable $X_i$ that is one iff the $i^{\text{th}}$ window $W_i$ is charged. Consider the adjacent window to the left $W_{i-1}$. The two intervals overlap except at the leftmost and rightmost positions. Their union is an interval of length $w+1$. Consider the position $p$ containing the smallest hash in that union interval. Any window that includes $p$ selects $h_p$ as a fingerprint. There are three cases:

1. If $p = i - 1$, the leftmost position in the union, then $W_{i-1}$ selects it. Since $p \notin W_i$, we know $W_i$ must select a hash in another position, $q$. This hash is charged to $W_i$ since $W_i$ selected it, $W_{i-1}$ did not select it, and the charge function is monotonic increasing. Thus in this case, $X_i = 1$.
2. If $p = i + w - 1$, the rightmost position in the union interval, then $W_i$ selects it. $W_i$ must be charged for it, as $W_i$ is also the very leftmost interval to contain $p$. Again, $X_i = 1$.
3. If $p$ is in any other position in the union interval, both $W_{i-1}$ and $W_i$ select it. No matter who is charged for it, it won't be $W_i$, since $W_{i-1}$ is further left and also selected it. Thus in this case, $X_i = 0$.

The first two cases happen with probability $1/(w+1)$, and so the expected value of $X_i$ is $2/(w+1)$. Recall that the sum of the expected values is the expected value of the sum, even if the random variables are not independent. The total expected number of intervals charged, and therefore the total number of fingerprints selected, is just this value times the document length. Thus the density is

$$d = \frac{2}{w+1}.$$

### 3.1.1 *Comparison to* 0 mod p *at same density*

Here we compare the *0 mod p* algorithm and winnowing at the same density. That is, we take $p = 1/d = (w+1)/2$. For a string of length $t = w + k - 1$ consider the event that the *0 mod p* algorithm fails to select any fingerprint at all within it. (Recall that winnowing would never fail to do so.) We now compute the probability of this event for one given string. Please note that for two overlapping such strings these events are not independent. Thus the probability we compute is not a good estimate for the fraction of all such substrings of a text that do not have a fingerprint selected using the *0 mod p* algorithm.

Again we assume independent uniformly distributed hash values. Also we assume large $w$; in our experiments $w = 100$ (see Section 5.1). Thus, the probability that the guarantee fails in a given sequence of text of length $t$, i.e. that no hash in a given sequence of $w$ hashes is *0 mod p*, is

$$(1-d)^w = \left(1 - \frac{2}{w+1}\right)^w \approx e^{\frac{-2w}{w+1}} = e^{-2+\frac{2}{w+1}} \sim 13.5\%.$$

### 3.1.2 *Comparison to* 0 mod p *with guarantee*

One may be tempted to try modifying the *0 mod p* algorithm to give a guarantee. There is one straightforward solution that we know of: In the event that a gap longer than the guarantee threshold threatens to open up, select *all* hashes as fingerprints until the next hash that is *0 mod p*.

Let the *Safe* 0 mod p algorithm be as follows. Partition hashes into:

- *Good* if a hash is *0 mod p*.
- *Bad* if it and the $w-1$ hashes to its left are not Good, and
- *Ugly* otherwise [11].

Select all non-Ugly hashes as fingerprints. As we will see in Section 4 this algorithm is *local* and is therefore correct. Note that we have chosen the parameters so that the guarantee $t$ is the same as that of winnowing. All that remains is to compute the optimal expected density.

Fix a document and consider a position $i$. Let $G_i$ and $B_i$ denote the events that the hash at $i$ is good or bad respectively. (Our notation for an event also denotes the appropriate indicator random variable (1 = true and 0 = false) depending on context.) Let $P = 1/p$. (Note that to compete with winnowing we would need $P$ to be rather small: $P \leq 2/(w+1)$; however even a slightly larger $P$ will allow for the $1 + x \approx e^x$ approximation we use below.) We have

$$\Pr[G_i] = P$$

and (except for the very first $w-1$ hashes) for small $P$

$$\Pr[B_i] = (1-P)^w \approx e^{-wP}.$$

Again, the expected value of a sum is the sum of the expected values. Ignoring the error introduced by the first $w-1$ hashes, we have that the expected value of the non-ugliness of a position is

$$\begin{aligned} \mathrm{Ex}\left[\sum G_i + B_i\right] &= \sum \mathrm{Ex}[G_i] + \sum \mathrm{Ex}[B_i] \\ &= NP + N(1-P)^w \\ &\approx N(P + e^{-wP}). \end{aligned}$$

The next step is to minimize the density. Let $f(P) = P + e^{-wP}$. Setting $f'(P_0) = 0$ and solving we have $e^{wP_0} = w$, or

$$P_0 = \frac{\ln w}{w}.$$

We check that $f''(P) = w^2 e^{-wP} > 0$ so we have found the global minimum. If we use this optimal value, $P_0$, the Safe *0 mod p* algorithm has density at least

$$f(P_0) = \frac{\ln w}{w} + e^{-\frac{w \ln w}{w}} = \frac{1 + \ln w}{w},$$

which is considerably more than that of winnowing: $2/(w+1)$.

## 3.2 Queries

This section is primarily about how to choose hashes well, but we digress a bit here to discuss how hashes can be used once selected. In a typical application, one first builds a database of fingerprints and then queries the fingerprints of individual documents against this database (see Section 5). Winnowing gives us some flexibility

to treat the two fingerprinting times (database-build time and query time) differently.

Consider a database of fingerprints (obtained from $k$-grams) generated by winnowing documents with window size $w$. Now, query documents can be fingerprinted using a different window size. Let $F_w$ be the set of fingerprints chosen for a document by winnowing with window size $w$. The advantage of winnowing query documents with a window size $w' \geq w$ is that $F_{w'} \subseteq F_w$, which means fewer memory or disk accesses to look up fingerprints. This may be useful if, for example, the system is heavily loaded and we wish to reduce the work per query, or if we are just interested in obtaining a faster but coarser estimate of the matching in a document.

We can extend this idea one step further. Fingerprint a query document with the same window $w$ used to generate the database, and then sort all of the selected fingerprints in ascending order. Next look up some number of the fingerprints in the database, starting with the smallest. If we stop after a few, fixed number of hashes, we have realized Broder's and Heintze's approach for testing document resemblance [8, 5]. If we use all of the hashes as fingerprints, we realize the standard notion of testing for document containment. There is also a spectrum where we stop anywhere in between these two extremes. Broder's paper on resemblance and containment gives distinct algorithms to compute these two properties [5]; winnowing naturally realizes both.

## 4. LOCAL ALGORITHMS

In this section we consider whether there are fingerprinting algorithms that perform better than winnowing. We introduce the notion of *local* fingerprinting algorithms. We prove a lower bound for the density of a local algorithm given uniform identically distributed random input. This lower bound does not meet the upper bound for winnowing. We suspect the lower bound can be improved.

Winnowing selects the minimum value in a window of hashes, but it is clearly just one of a family of algorithms that choose elements from a local window. Not every method for selecting hashes from a local window maintains the guarantee, however. Assume, for example, that the window size is 50 and our approach is to select every 50$^{\text{th}}$ hash as the set of fingerprints. While this method does select a hash from every window, it depends on the global position of the hash in a document, and, as discussed in Section 2, any such approach fails in the presence of insertions or deletions. The key property of winnowing is that the choice of hash depends only on the contents of the window—it does not depend on any external information about the position of the window in the file or its relationship to other windows. This motivates the following definition.

DEFINITION 2 (LOCAL ALGORITHMS). *Let $S$ be a* selection *function taking a $w$-tuple of hashes and returning an integer between zero and $w-1$, inclusive. A fingerprinting algorithm is* local *with selection function $S$, if, for every window $h_t, \ldots, h_{t+w-1}$, the hash at position $i + S(h_t, \ldots, h_{t+w-1})$ is selected as a fingerprint.*

It can be beneficial to weaken locality slightly to provide flexibility in choosing among equal fingerprints—see Section 5.1. We now show that any local algorithm is correct, in the sense that it meets the guarantee threshold $t$.

LEMMA 1 (CORRECTNESS OF LOCAL ALGORITHMS). *Any matching pair of substrings of length at least $t$ is found by any local algorithm.*

PROOF. The sequence of hashes of $k$-grams representing the each substring spans at least one window, $W$, of length $w$. Because the selection function is only a function of the contents of $W$, the same fingerprint is selected from $W$ in both copies. □

We now consider whether there is any local algorithm that is better than winnowing. We do not have a matching lower bound for winnowing, but we can show the following:

THEOREM 1 (LOWER BOUND). *Any local algorithm with noise threshold $k$ and guarantee $t = w + k - 1$ has density*

$$d \geq \frac{1.5}{w+1}.$$

Note that winnowing algorithm, with a density of $2/(w+1)$, is within 33% of this lower bound.

PROOF. Assume the hashes are independent and uniformly distributed. Consider the behavior of the algorithm on every $w+1^{\text{st}}$ window. Such windows are separated by a single position that is not part of either window. Because the windows are disjoint, their hashes and selected fingerprints are independent of each other, and each window selects a separate fingerprint.

Now consider all of the windows between the $i^{\text{th}}$ and $(i+w+1)^{\text{st}}$ windows $W_t$ and $W_{t+w+1}$; these are the $w$ windows that overlap the disjoint windows at either end. Let $Z$ be the random variable such that $Z = 0$ iff among these windows, no additional fingerprint is selected, and $Z = 1$ otherwise. We compute a lower-bound on the expected value of $Z$.

Let $X$ and $Y$ denote the random variables $S(W_t)$ and $S(W_{t+w+1})$ respectively. Again, because the windows do not overlap, $X$ and $Y$ are independent.

Now, if $Y \geq X$ then $Z = 1$, because the algorithm is required to select at least one additional fingerprint from a window in between $W_t$ and $W_{t+w+1}$. Otherwise $Z \geq 0$. Since $X$ and $Y$ are identically distributed we have $\Pr[Y > X] = \Pr[X > Y]$. Let $\Theta$ denote this quantity. Let $\Delta = \Pr[Y = X]$. We have $1 = 2\Theta + \Delta$. Thus $\Theta + \Delta = (1+\Delta)/2 > 1/2$ and

$$\mathrm{Ex}[Z] \geq \Pr[Y \geq X] = \Theta + \Delta > 1/2.$$

We thus see that in every sequence of $w+1$ windows, in addition to the fingerprint selected in the first window we expect to select an additional distinct fingerprint at least half the time for one of the subsequent windows. The density of selected points is therefore

$$d \geq \frac{1.5}{w+1}.$$

□

OBSERVATION 1. *This result can be improved slightly: As a bit of notation let $x_i = Pr[X = i]$ for $i = 0, 1, \ldots, w-1$. Of course $\sum x_i = 1$. Then $\Delta = Pr[Y = X] = \sum x_i^2$. Apply the Cauchy-Schwartz inequality to show $\Delta \geq 1/w$. The proof above then gives density*

$$d \geq \frac{1.5 + \frac{1}{2w}}{w+1}.$$

Our lower bound proof relies only on information derived from two windows that are separated sufficiently to be disjoint. We conjecture therefore that $2/(w+1)$ is a lower bound on the density of any local fingerprinting algorithm.

| | |
|---|---|
| *total bytes* | 7,182,692,852 |
| *text bytes* | 1,940,576,448 |
| *hashes computed* | 1,940,576,399 |
| *winnowing fingerprints* | 38,530,846 |
| *measured density* | 0.019855 |
| *expected density* | 0.019802 |
| *fingerprints for* 0 mod 50 | 38,761,128 |
| *measured density* | 0.019974 |
| *expected density* | 0.020000 |
| *longest run with no fingerprint* | 29983 |

**Figure 3: Results on 500,000 HTML pages**

## 5. EXPERIMENTS

In this section we report on our experience with two different implementations of winnowing. In Section 5.1 we report on a series of experiments on text data taken from the World Wide Web, and in Section 5.2 we give a more qualitative report on experience over several years with a widely-used plagiarism detection service, Moss.

### 5.1 Experiments with Web Data

Because of its size and the degree of copying, the World-Wide Web provides a readily accessible and interesting data set for document fingerprinting algorithms. For these experiments, we used 500,000 pages downloaded from the Stanford WebBase [1]. We use the rolling hash function described in Section 2. Because fingerprinting a half-million Web pages generates nearly two billion hashes and 32-bits can represent only about four billion distinct hash values, we use 64-bit hashes to avoid accidental collisions. As an aside, we have found using a rolling (or incremental) hash function is important for performance with realistic $k$-gram sizes (say $k = 50$) when using 64-bit arithmetic. Recomputing a 64-bit hash from scratch for each $k$-gram reduces the throughput of the fingerprinting algorithm by more than a factor of four.

In our first experiment we simply fingerprinted 8MB of randomly generated text. This experiment serves solely to check that our hash function is reasonably good, so that we can trust the number of matches found in experiments on real data. Strings of 50 characters were hashed and the winnowing window was set at 100. Winnowing selected 0.019902 of the hashes computed, which very closely matches the expected density of $2/(100+1) = 0.019802$. Selecting hashes equal to *0 mod 50* results in a measured density of 0.020005, which is also very close to the predicted value of 0.02. We also observed a uniform distribution of hash values; taken all together, the hash function implementation appears to be sufficient for our fingerprinting algorithm.

Our second experiment calculated the hashes for 500,000 HTML documents and measured various statistics. We again measured the density and compared it with the expected density for both winnowing and selecting fingerprints equal to *0 mod p*. Again the winnowing window size is 100 and the noise threshold is 50. The results are shown in Figure 3.

There were interesting things to note in the data. Both algorithms come close to the expected density in each case. However, the gross averages cover up some local aberrations. For example, there is a run of over 29,900 non-whitespace, non-tag characters that has no hash that is *0 mod 50*. It is easily checked that the odds of this happening on uniformly random inputs are extremely small. (The chances that a string of 29,900 characters has no hash of a substring that is *0 mod 50* is $(1-1/50)^{29,851}$, which is less than $10^{-260}$. Even in a terabyte, or $2^{40}$ bytes, of data, the chances that every substring of length 29,900 has no $k$-gram hash that is *0 mod 50*, is less than $10^{-220}$.) Clearly the data on the Web is not uniformly random.

As discussed briefly in Section 1, there are long passages on the Web of repetitive, low-entropy strings. For example, in one experiment we did (not reported here) we stumbled across a collection of pages that appear to be raw data taken from sensors in a research experiment. This data consists mostly of strings of 0's with the occasional odd character thrown in. Both winnowing as defined so far and selecting hashes equal to *0 mod p* perform poorly on such data. For the latter, if a long string has few $k$-grams, then it is very likely that none of them is *0 mod p*, and no fingerprints are selected for the region at all. This is what leads to the large gaps in fingerprints for this strategy on real data.

Winnowing, however, has a different problem. In low-entropy strings there are many equal hash values, and thus many ties for the minimum hash in a given window. To be truly local and independent of global position, it is necessary to take, say, the rightmost such hash in the winnowing window. But in the extreme case, say a long string of 0's with only one $k$-gram, nearly every single hash is selected, because there is only a single $k$-gram filling the entire winnowing window and at each step of the algorithm we must choose the rightmost copy—which is a new copy in every window.

There is, however, an easy fix for this problem. We refine winnowing as follows:

DEFINITION 3 (ROBUST WINNOWING). *In each window select the minimum hash value. If possible break ties by selecting the same hash as the window one position to the left. If not, select the rightmost minimal hash. Save all selected hashes as the fingerprints of the document.*

Robust winnowing attempts to break ties by preferring a hash that has already been chosen by a previous window. This is no longer a local algorithm, but one easily observes that for any two matching substrings of length $t = w+k-1$ we guarantee to select the same hash value and so the match is still found; we simply no longer guarantee that these fingerprints are in the same relative position in the substrings. However, the two fingerprints are close, within distance $w-1$. This technique reduces the density on a string such as "0000 ..." from asymptotically 1 to just $1/w$, one fingerprint selected per window-length. We reran the experiment in Figure 3 and found that the density of winnowing dropped from 0.019855 to 0.019829. One can imagine non-text document sets where the difference could be greater.

One may wonder why we bother worrying about low-entropy strings at all, as they are in a technical sense inherently uninteresting. But just because data is low-entropy does not mean that people are not interested in it—take the example of the sensor data given above. Such strings do exist and people may want to fingerprint a large corpus of low entropy data so that copies of it can be tracked just as they may want to fingerprint any other sort of document.

Our final experiment examines in more detail the structure of copying in 20,000 Web pages taken from our corpus of 500,000 pages. Interestingly, even though the theoretical predictions based upon an assumption that the input is uniformly random work very well, the distribution of real data is hardly uniform. We need two definitions:

- Let the *frequency* of a $k$-gram (or its hash) be the number of times it occurs.
- Sort the frequencies in monotonically decreasing order. The *rank* of a $k$-gram (or its hash) is the position of its frequency

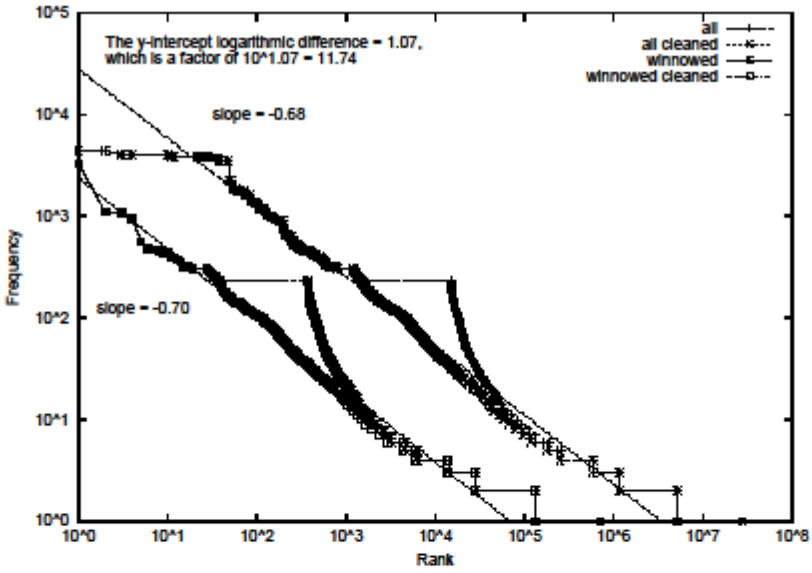



**Figure 4: Log-log plot of Frequency by Rank for all hashes (upper line) and for fingerprints (lower line) on 20,000 Web pages.**

on this list, starting with 1 for the most frequently occurring $k$-gram.

Plotting the resulting (*rank*, *frequency*) pairs on a log-log scale one obtains a line of slope about $-0.7$, demonstrating a power law relationship between frequency and rank

$$f \propto r^{-0.7}.$$

Two such plots of (*rank*, *frequency*) pairs are shown in Figure 4. The upper curve is all $k$-gram hashes computed from the entire set of 20,000 Web pages, while the lower curve is only those hashes selected as fingerprints by winnowing. (This is the reason we have limited the data set in this experiment to 20,000 pages. Even on 20,000 Web pages saving the hash of every $k$-gram requires quite a bit of storage.) Our data contains an aberrant "plateau", perhaps because of one document with a long repeating pattern of text, or one file that occurs many times in our sample. The peaked-looking curves labeled "all" show all of the data while the curves labeled "cleaned" have the aberrant plateau removed.

Since frequencies are integers, they form plateaus where a set of hashes all have the same frequency. For example, for the upper curve there are over 20 million fingerprints with one occurrence, which ties them all for the last rank. While more obvious in the lower right, these plateaus occur throughout the data. Additionally, due to the logarithmic scale, almost all points are in the lower right. Thus, if one were to actually plot all points, the line fit would be quite poor, as it would just go through the center of the last two plateaus (we tried this). It would also overwhelm the plotting program with points that do nothing other than thicken the horizontal lines drawn between points.

We plot only the left and right endpoints of each plateau (a plateau of length one is plotted twice so the weights are not biased during line fit). The lines are fitted to the cleaned plateau endpoints. The lines fit quite well, giving a slope of about $-0.7$, which corresponds to the exponent of the power law. Zipf seems to have first noticed the power law phenomenon, stating what has become known as "Zipf's Law" [16]: the frequencies of English words are proportional to the inverse of their rank when listed in decreasing order. That is, frequency and rank of English words exhibit a power law relationship with exponent $-1$.

In this experiment, 82% of the fingerprints selected by winnowing were chosen only once; 14% were selected twice; and only 2% occurred three times. At the other extreme, one $k$-gram appears 3,270 times across all the documents. The distribution of frequencies for the set of all hashes is nearly identical to the distribution for winnowed fingerprints; again the number of $k$-grams that occur once is 82%, while 14% occur twice and 2% occur three times.

We have looked at some of the most common strings and found that they are what one might expect: strings taken from menus (e.g., "English Spanish German French ..."), common legal boilerplate (e.g., disclaimers), and finally repetitive strings (for some reason the string "documentwritedocumentwritedocumentwrite" was very common in our sample). We suspect that the repetitive strings, in particular, are responsible for the most common $k$-grams. Because such strings have relatively few $k$-grams, they dramatically increase the frequency of a few $k$-grams in the overall statistics.

## 5.2 Plagiarism Detection

One of the authors has run MOSS, a widely-used plagiarism detection service, over the Internet since 1997. MOSS, which stands for Measure Of Software Similarity, accepts batches of documents and returns a set of HTML pages showing where significant sections of a pair of documents are very similar. MOSS is primarily used for detecting plagiarism in programming assignments in computer science and other engineering courses, though several text

formats are supported as well. The service currently uses robust winnowing, which is more efficient and scalable (in the sense that it selects fewer fingerprints for the same quality of results) than previous algorithms we have tried. There are a few issues involved in making such a system work well in practice.

For this application, positional information (document and line number) is stored with each selected fingerprint. The first step builds an index mapping fingerprints to locations for all documents, much like the inverted index built by search engines mapping words to positions in documents. In the second step, each document is fingerprinted a second time and the selected fingerprints are looked up in the index; this gives the list of all matching fingerprints for each document.

Now the list of matching fingerprints for a document $d$ may contain fingerprints from many different documents $d_1, d_2, \ldots$. In the next step, the list of matching fingerprints for each document $d$ is sorted by document and the matches for each pair of documents $(d, d_1), (d, d_2), \ldots$ is formed. Matches between documents are rank-ordered by size (number of fingerprints) and the largest matches are reported to the user. Note that up until this last step, no explicit consideration of pairs of documents is required. This is very important, as we could not hope to carry out copy detection by comparing each pair of documents in a large corpus. By postponing the quadratic computation to the last step, we can optimize it by never materializing the matching for a pair of documents if it falls below some user-specified threshold.

There are a number of issues in a full copy-detection system beyond how fingerprints are selected. To give the reader some sense of how winnowing fits into a complete system, we briefly discuss the most important problems.

Moss has several thousand users who wish to do copy detection for many different kinds of data. As mentioned in Section 1, we use the following architecture. For each document format, a front-end specific to that format eliminates features that should not distinguish documents (e.g., we eliminate white space in text). As output each front-end produces a string of a standard form, which is the input to the fingerprinting engine. The fingerprinting code itself knows nothing about the different kinds of documents. This architecture has proven essential to maintaining support for a wide variety of document formats. While this benefit may seem obvious, we report it because it is very tempting to put some document semantics in the fingerprinting routines, but we have always found it to be better to keep the document-specific processing separate.

Efficiency is an important consideration for fingerprinting. In Figure 5 we give code for an efficient implementation of the main winnowing loop. This implementation takes advantage of the fact that by far the most common case is that the minimum value from the preceding window is still within the current window; in this case checking to see if there is a new minimum requires only a single comparison. The only instance in which it is necessary to recompute the minimum by traversing the entire window is the case where the minimum hash of the preceding window is just outside of the current window; note that the loop that does the scan of the array works from right-to-left to ensure that the rightmost minimal hash is selected. Thus, the choice of which of several equal hashes to select is not completely arbitrary. Note the `record` function must compute the global position using the relative position, `min`. Saving this position, together with the selected hash, creates a fingerprint. This loop implements winnowing—it always selects the rightmost minimal hash in a window. To implement robust winnowing the $\leq$ comparison on line marked $(*)$ should be replaced by $<$.

As a minor aside, because winnowing selects the minimum hash

```
void winnow(int w /*window size*/) {
  // circular buffer implementing window of size w
  hash_t h[w];
  for (int i=0; i<w; ++i) h[i] = INT_MAX;
  int r = 0;                    // window right end
  int min = 0;                  // index of minimum hash
  // At the end of each iteration, min holds the
  // position of the rightmost minimal hash in the
  // current window.  record(x) is called only the
  // first time an instance of x is selected as the
  // rightmost minimal hash of a window.
  while (true) {
    r = (r + 1) % w;            // shift the window by one
    h[r] = next_hash();         // and add one new hash
    if (min == r) {
      // The previous minimum is no longer in this
      // window.  Scan h leftward starting from r
      // for the rightmost minimal hash.  Note min
      // starts with the index of the rightmost
      // hash.
      for(int i=(r-1)%w; i!=r; i=(i-1+w)%w)
        if (h[i] < h[min]) min = i;
      record(h[min], global_pos(min, r, w));
    } else {
      // Otherwise, the previous minimum is still in
      // this window.  Compare against the new value
      // and update min if necessary.
      if (h[r] <= h[min]) {  // (*)
        min = r;
        record(h[min], global_pos(min, r, w));
      }
    }
  }
}
```

**Figure 5: Code for winnowing.**

in each window, the distribution of hashes selected is skewed. If a uniform distribution is desired, the selected hashes can be hashed a second time (not shown in Figure 5).

A very significant issue in a practical copy-detection system is the ability to ignore boiler-plate. For example, standard copyright notices, disclaimers, and other legalese would all come under the heading of material that we would not be interested in for many applications. In the case of plagiarism detection, boilerplate is usually material supplied by a course instructor that is expected to be part of the final solution—i.e., it is sanctioned copying. Excluding boilerplate is easily done by fingerprinting the boilerplate with a special document ID that indicates any match with that fingerprint should be discarded.

Presentation of the results is another important issue for users. Statistics such as reporting the percentage of overlap between two documents are useful, but not nearly as useful as actually showing the matches marked-up in the original text. Moss uses the fingerprints to determine where the longest matching sequences are; in particular, if $a_1$ in document 1 matches $a_2$ in document 2, and $b_1$ in document 1 matches $b_2$ in document 2, and furthermore $a_1$ and $b_1$ are consecutive in document 1 and $a_2$ and $b_2$ are consecutive in document 2, then we have discovered a longer match across documents consisting of $a$ followed by $b$. While this merging of matches is easy to implement, $k$-grams are naturally coarse and some of the match is usually lost at the beginning and the end of the match. It is possible that once a pair of similar documents are detected using fingerprinting that it would be better to use a suffix-tree algorithm [15] to find maximal matches in just that pair of documents.

In Section 2 we mentioned that there appears to be a sharp thresh-

old between what people consider coincidental similarity (meaning reuse of idioms, common words, etc.) and copying. We have no formal experiments on this topic, but we have informally experimented with MOSS by simply examining the results of tests on sample data. Regardless of input data type, the result is always the same: There is some value of $k$ (dependent on the document type) for which the reported matches are likely to be the result of copying, and for a slightly smaller value of $k$ significant numbers of obvious false positives appear in the results. Along the same lines, early versions of MOSS incorporated a technique similar to Baker's parameterized matches (Section 2). However, we found that replacing all of the parameters with a single constant and increasing $k$ by 1 worked just as well. This appears to be a general trick: sophisticated efforts to exploit document semantics can often be closely approximated by very simple exploits of document semantics together with a small increase in $k$.

We can report that after years of service, MOSS performs its function very well. False positives (hash collisions) have never been reported, and all the false negatives we have seen were quickly traced back to the source, which was either an implementation bug or a user misunderstanding. Furthermore, users report that copy detection does dramatically reduce the instances of plagiarism in their classes.

## 6. CONCLUSIONS

We have presented winnowing, a local document fingerprinting algorithm that is both efficient and guarantees that matches of a certain length are detected. We have also presented a non-trivial lower bound on the complexity of any local document fingerprinting algorithm. Finally, we have discussed a series of experiments that show the effectiveness of winnowing on real data, and we have reported on our experience with the use of winnowing in practice.

## 7. ACKNOWLEDGMENTS

The authors wish to thank Joel Auslander, Steve Fink, and Paul Tucker for many useful discussions and for helping make this work possible.

## 8. REFERENCES

[1] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Searching the web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, 2001.

[2] Brenda S. Baker. On finding duplication and near-duplication in large software systems. In L. Wills, P. Newcomb, and E. Chikofsky, editors, *Second Working Conference on Reverse Engineering*, pages 86–95, Los Alamitos, California, 1995. IEEE Computer Society Press.

[3] Brenda S. Baker and Udi Manber. Deducing similarities in java sources from bytecodes. In *Proc. of Usenix Annual Technical Conf.*, pages 179–190, 1998.

[4] Sergey Brin, James Davis, and Héctor García-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Conference*, pages 398–409, 1995.

[5] Andrei Broder. On the resemblance and containment of documents. In *SEQS: Sequences '91*, 1998.

[6] Andrei Broder, Steve Glassman, Mark Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Proceedings of the Sixth International World Wide Web Conference*, pages 391–404, April 1997.

[7] The Crystals. Da do run run, 1963.

[8] Nevin Heintze. Scalable document fingerprinting. In *1996 USENIX Workshop on Electronic Commerce*, November 1996.

[9] James Joyce. *Finnegans wake [1st trade ed.]*. Faber and Faber (London), 1939.

[10] Richard M. Karp and Michael O. Rabin. Pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

[11] Sergio Leone, Clint Eastwood, Eli Wallach, and Lee Van Cleef. *The Good, the Bad and the Ugly / Il Buono, Il Brutto, Il Cattivo (The Man with No Name)*. Produzioni Europee Associate (Italy) Production, Distributed by United Artists (USA), 1966.

[12] Udi Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conference*, pages 1–10, San Fransisco, CA, USA, 17–21 1994.

[13] Peter Mork, Beitao Li, Edward Chang, Junghoo Cho, Chen Li, and James Wang. Indexing tamper resistant features for image copy detection, 1999. URL: citeseer.nj.nec.com/mork99indexing.html.

[14] Narayanan Shivakumar and Héctor García-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*, 1995.

[15] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

[16] George K. Zipf. *The Psychobiology of Language*. Houghton Mifflin Co., 1935.

## Puesta en Funcionamiento:

Para probar el funcionamiento del sistema propuesto sin influir en los sistemas informáticos de la Universidad de Salamanca, se ha optado por realizar una instalación paralela de la plataforma Studium en un ordenador dedicado con las siguientes prestaciones:

Servidor dedicado: STOR-24T

CPU: Intel Xeon D-1521 - 4c/8t - 2,4GHz /2,7GHz

RAM: 16GB DDR4 ECC 2133MHz

Servicios incluidos con el servidor

500 GB de espacio de backup

API Remoto

Acceso «root» al servidor

Acceso mediante consola gráfica KVM IP

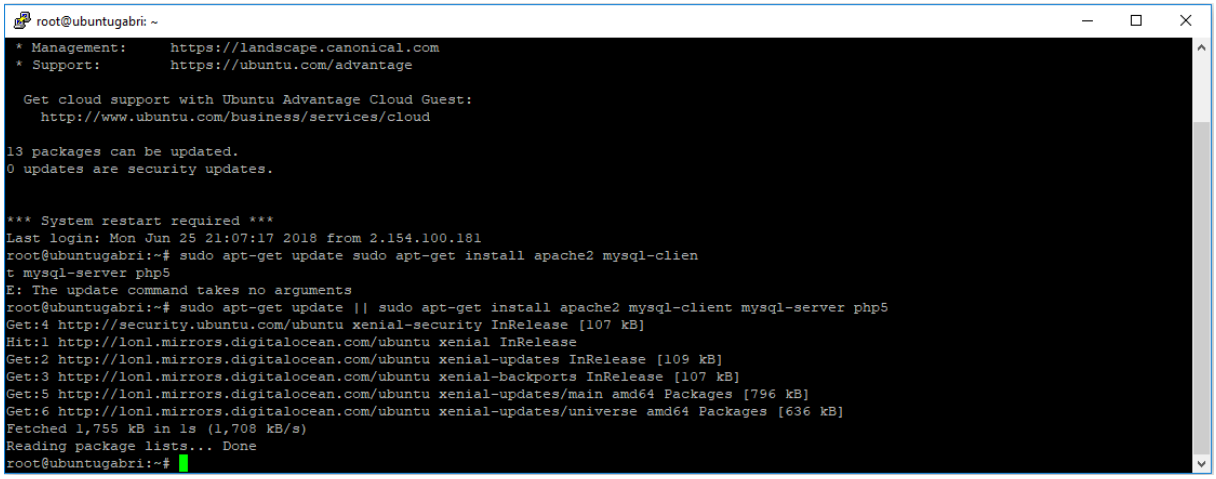
Anti-DDoS PRO incluido

A continuación, se detalla la instalación y puesta en marcha del aplicativo:

## Instalación Plataforma Studium:

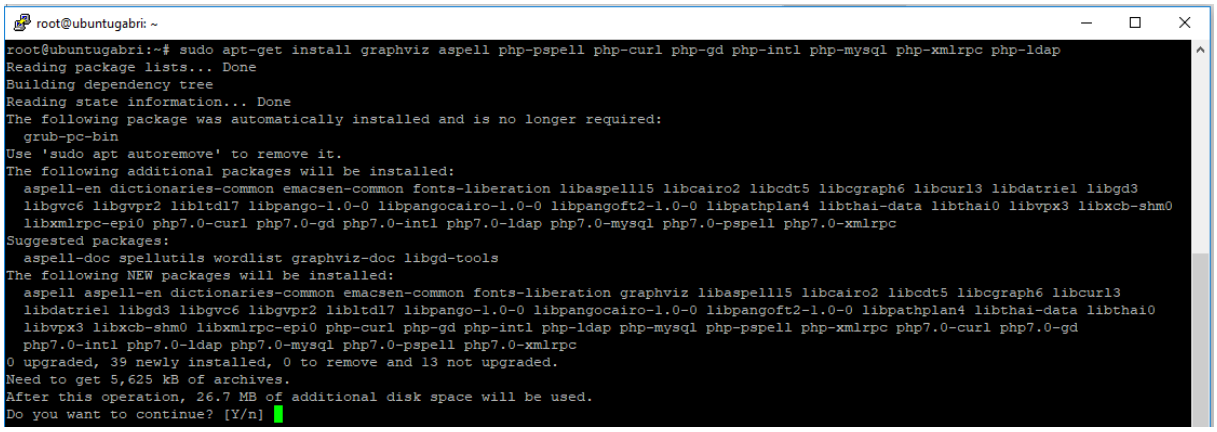
En primer lugar, nos conectamos al servidor remoto, abrimos una conexión de tipo SSH y ejecutamos el siguiente comando:

```
sudo apt-get update || sudo apt-get install apache2 mysql-client mysql-server php5
```




Durante la instalación aparecerá un asistente donde establecer la contraseña de la base de datos. Posteriormente, instalamos las dependencias que necesita la plataforma Studium.

```
sudo apt-get install graphviz aspell php-pspell php-curl php-gd php-intl php-mysql php-xmlrpc php-ldap
```




Reiniciamos el servidor Web “Apache”

```
sudo service apache2 restart
```

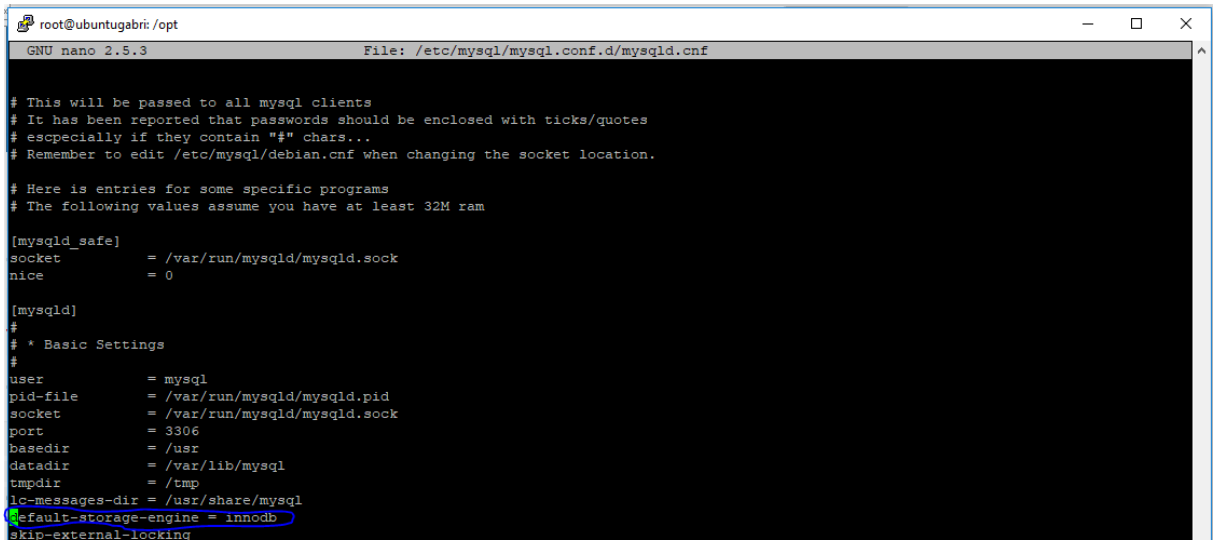
Obtenemos la última versión de la aplicación Moodle de su repositorio.

```
cd /opt sudo git clone git://git.moodle.org/moodle.git && cd moodle && sudo git branch -a  && sudo git branch --track MOODLE_26_STABLE origin/MOODLE_26_STABLE && sudo git checkout MOODLE_26_STABLE
```

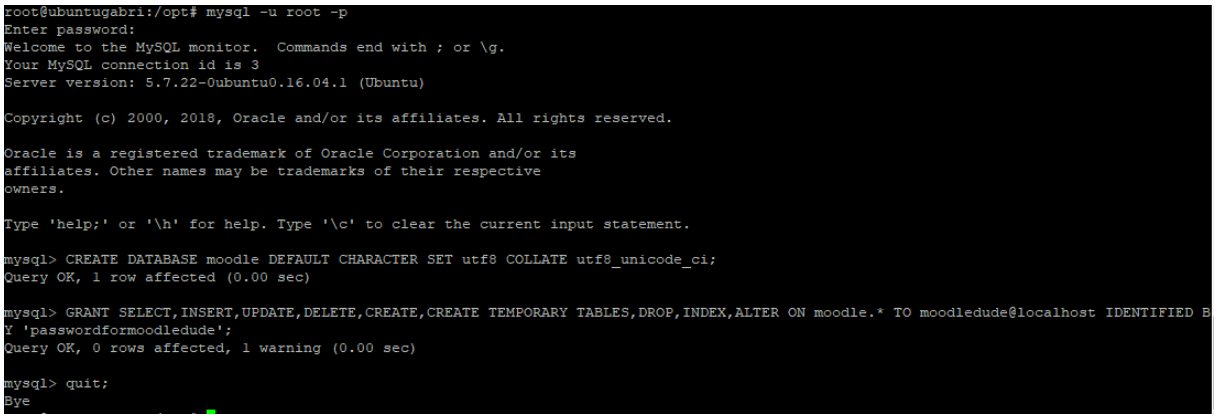
Copiamos el repositorio local al directorio público de nuestro servidor web

```
sudo cp -R /opt/moodle /var/www/html/ && sudo mkdir /var/moodledata  && sudo chown -R www-data /var/moodledata && sudo chmod -R 777 /var/moodledata && sudo chmod -R 0755 /var/www/html/Moodle
```

Cambiamos el motor de almacenamiento de la base de datos añadiendo la siguiente línea al fichero de configuración /etc/mysql/mysql.conf.d/mysqld.cnf

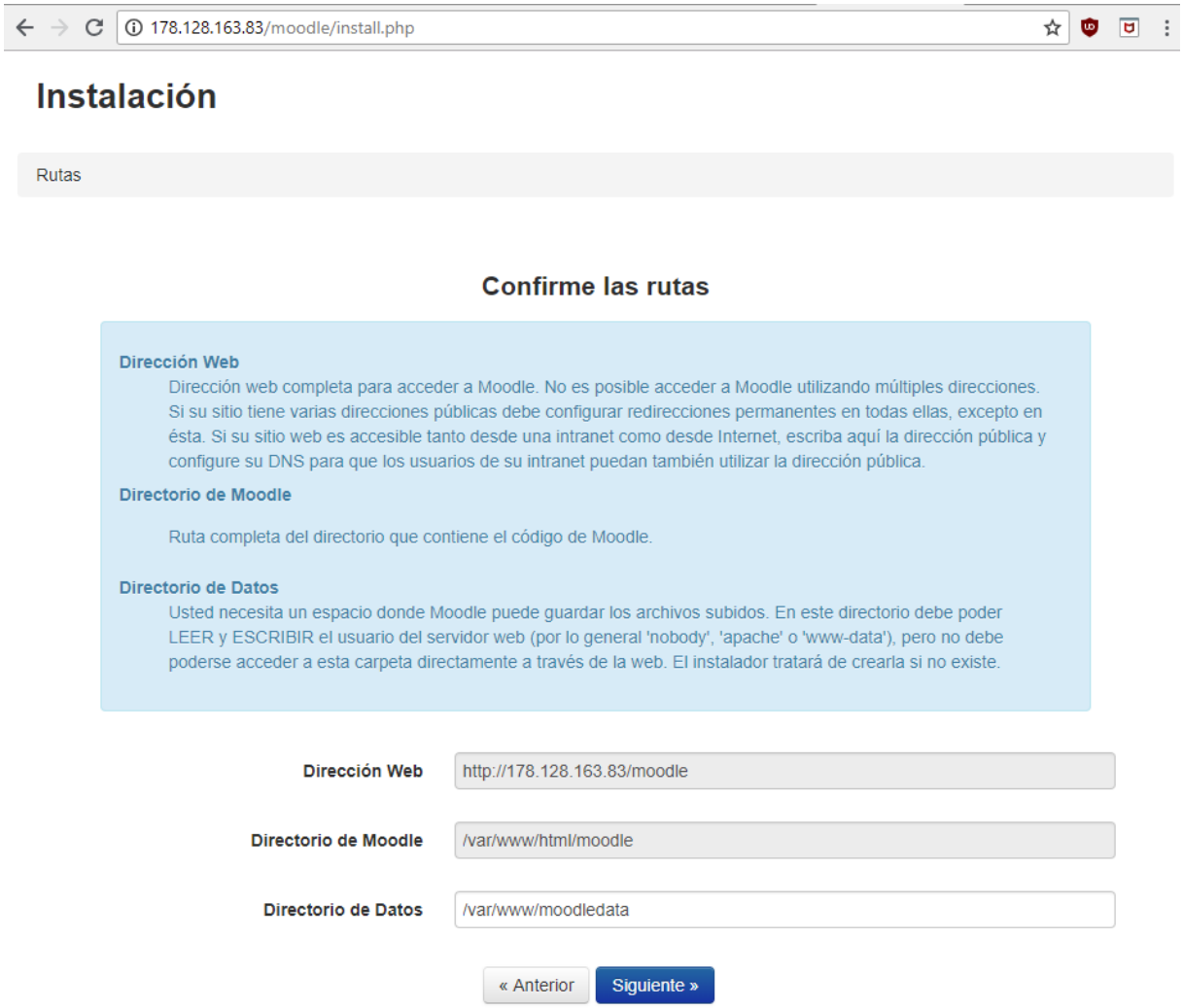



Creamos una base de datos llamada Moodle con una codificación de tipo utf8 y creamos un usuario *moodledude* con una contraseña por defecto *passwordformoodledude*.

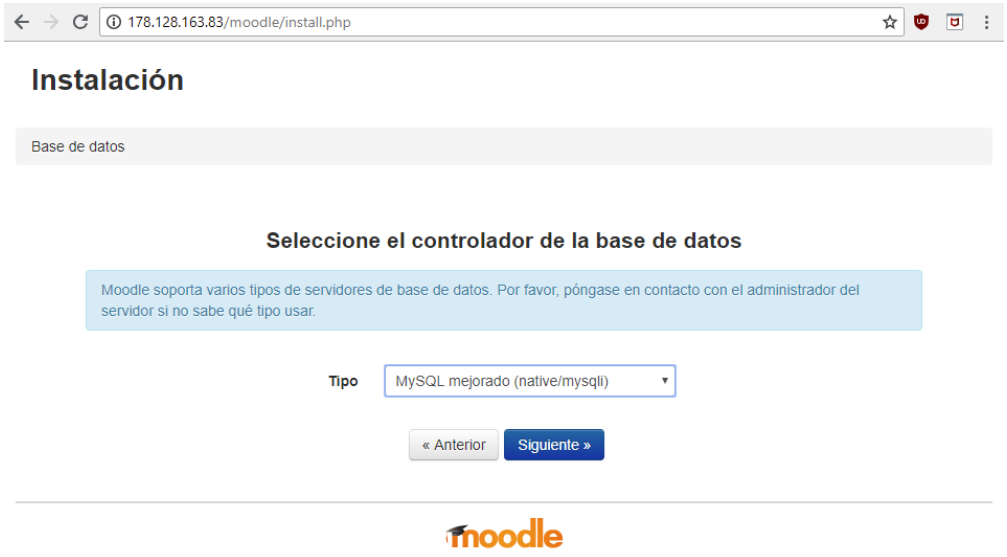



Accedemos a la url del servidor para completar la instalación del Moodle.

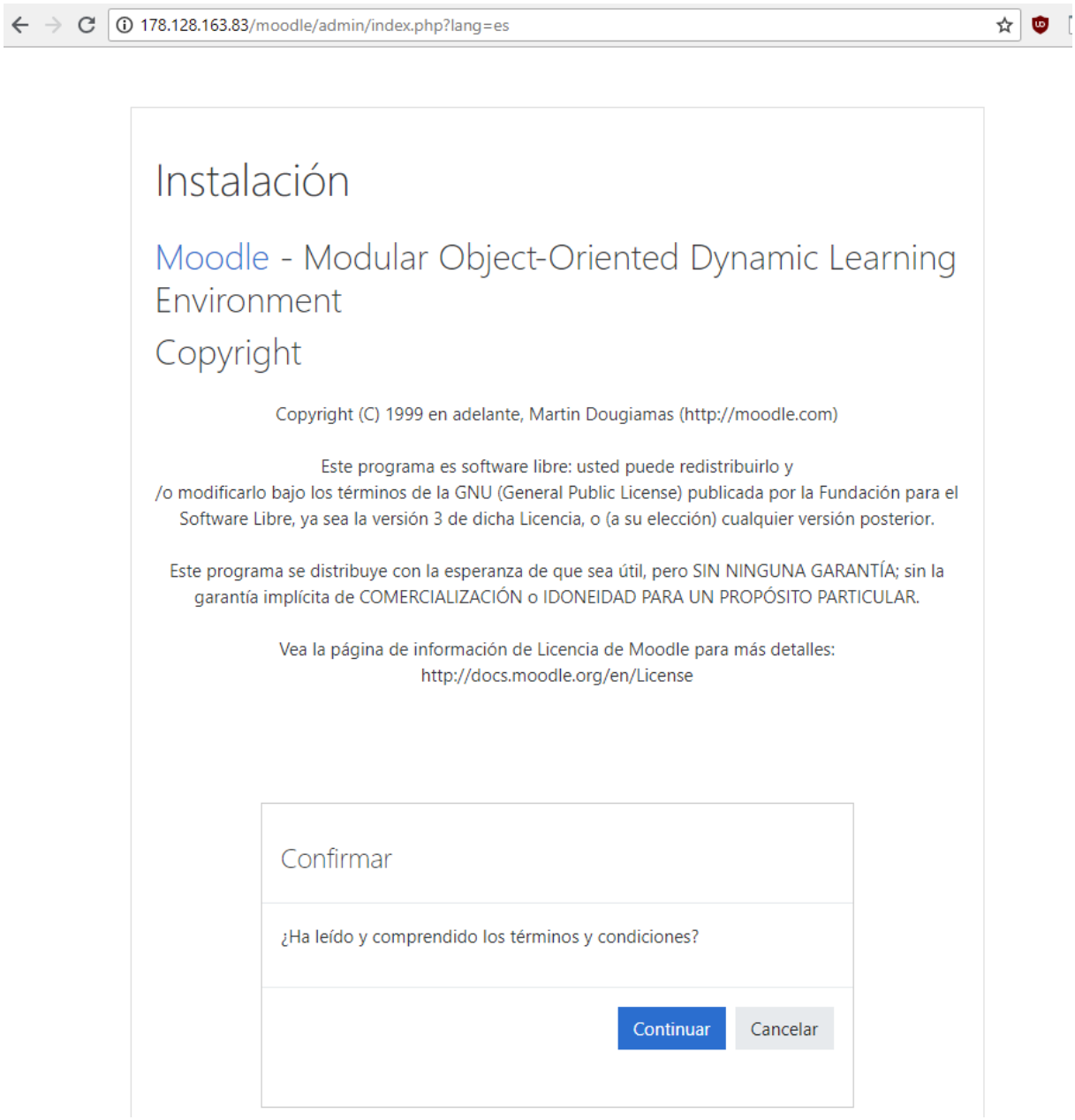
Confirmamos los datos de la instalación:




Seleccione el controlador de la base de datos:



Finalmente aparece un menú para confirmar la licencia de la plataforma Moodle:



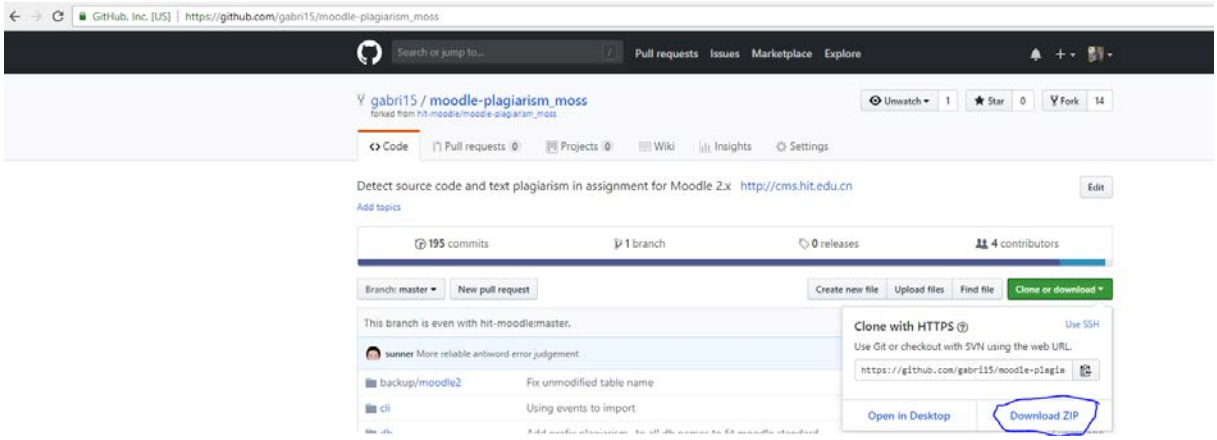

Pulsamos en “Continuar” y la plataforma está lista para ser utilizada.

## Instalación Módulo Anti-plagio:

Lo primero que vamos a hacer, es bajarnos el código fuente de las librerías necesarias para activar el sistema. El código implementado ha sido adjuntado en esta memoria en un Anexo, pero también puede ser consultado en la siguiente dirección.

https://github.com/gabri15/moodle-plagiarism_moss

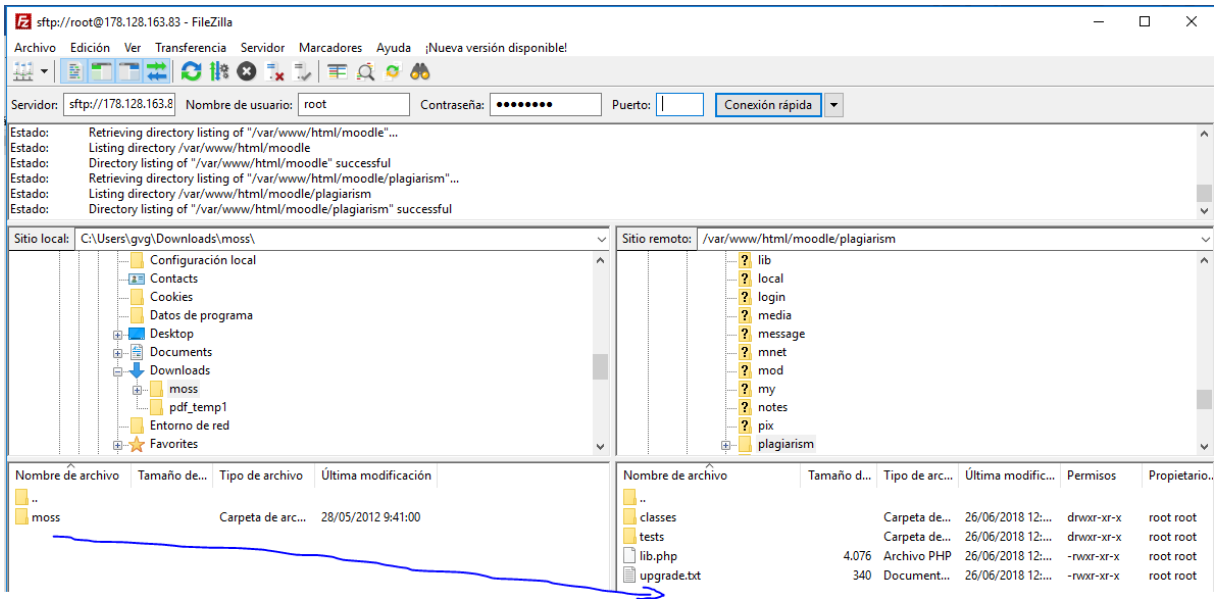
Para descargar el código del repositorio, pulsamos en el botón señalado en la imagen.



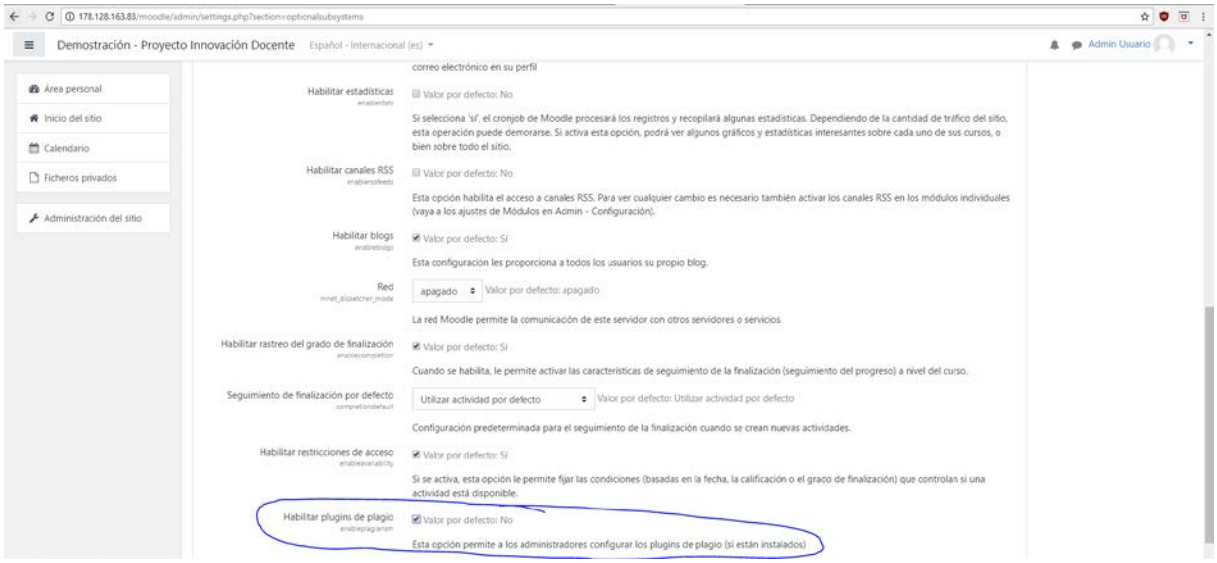

Una vez descargado el código necesario, los pasos de instalación se enumeran a continuación:

1. Si el directorio MOODLE_PATH/plagiarism/moss/ existe, **ES OBLIGATORIO ELIMINARLO**.
2. Asegurarse de que el código una vez descomprimido se encuentra contenido en un directorio llamado moss. En caso contrario, renombrarlo.
3. Subir el directorio al servidor remoto y situarlo bajo el directorio MOODLE_PATH/plagiarism/.
4. Loguearse en la plataforma Moodle como administrador para activar el plugin.
5. Acceder a http://IP/admin/settings.php?section=optionalsubsystems para activar la librería.
6. Acceder a http://IP/plagiarism/moss/settings.php para configurar el anti-plagio.

A continuación, se muestra una imagen resumen de los pasos dados.

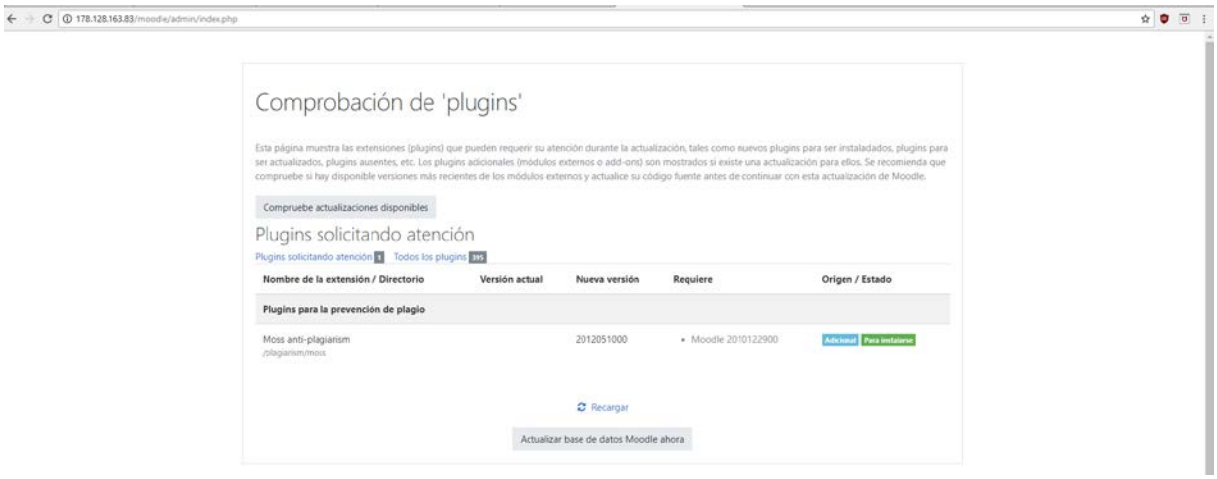



SUBIMOS EL CÓDIGO DEL SISTEMA ANTI-PLAGIO PARA ACTIVARLO EN LA PLATAFORMA

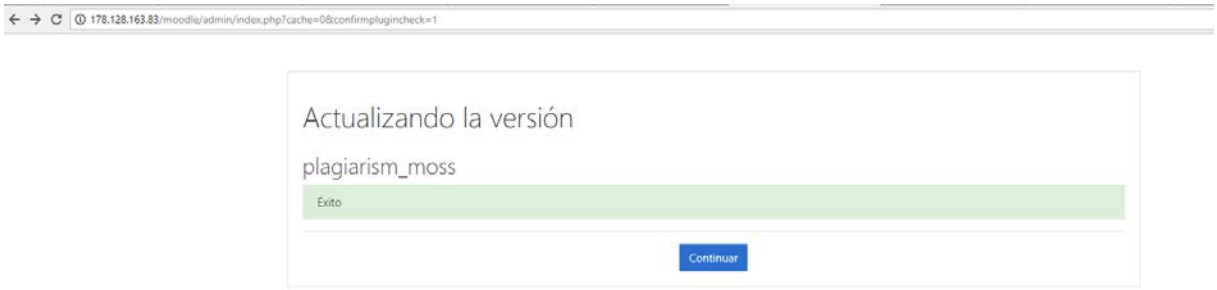



ACTIVAMOS EL SISTEMA ANTI-PLAGIO

Al guardar los cambios, saldrá un mensaje diciendo que el sistema se ha activado correctamente

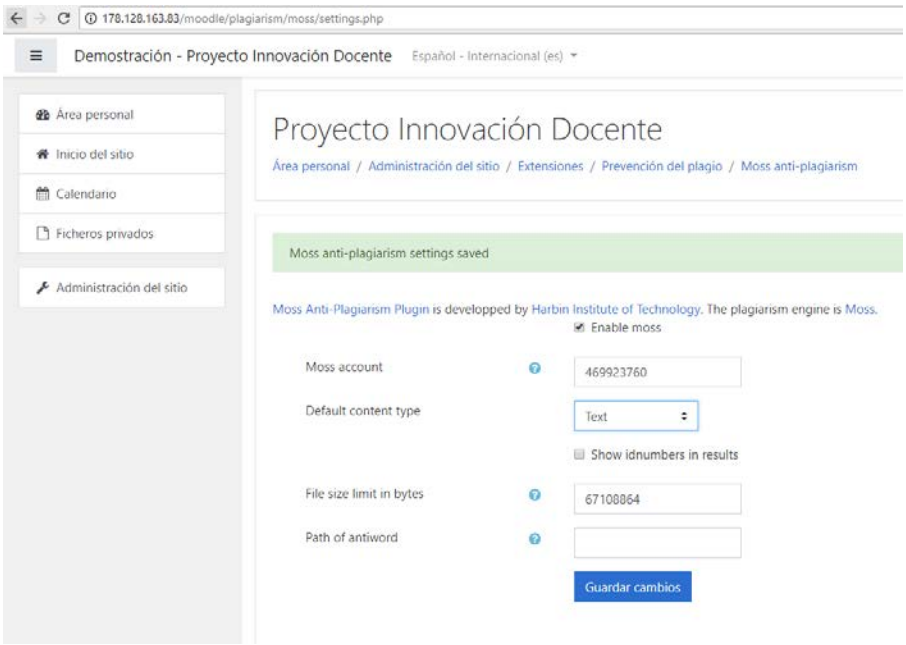



ACTIVAMOS EL SISTEMA ANTI-PLAGIO




ACTIVACIÓN CORRECTA

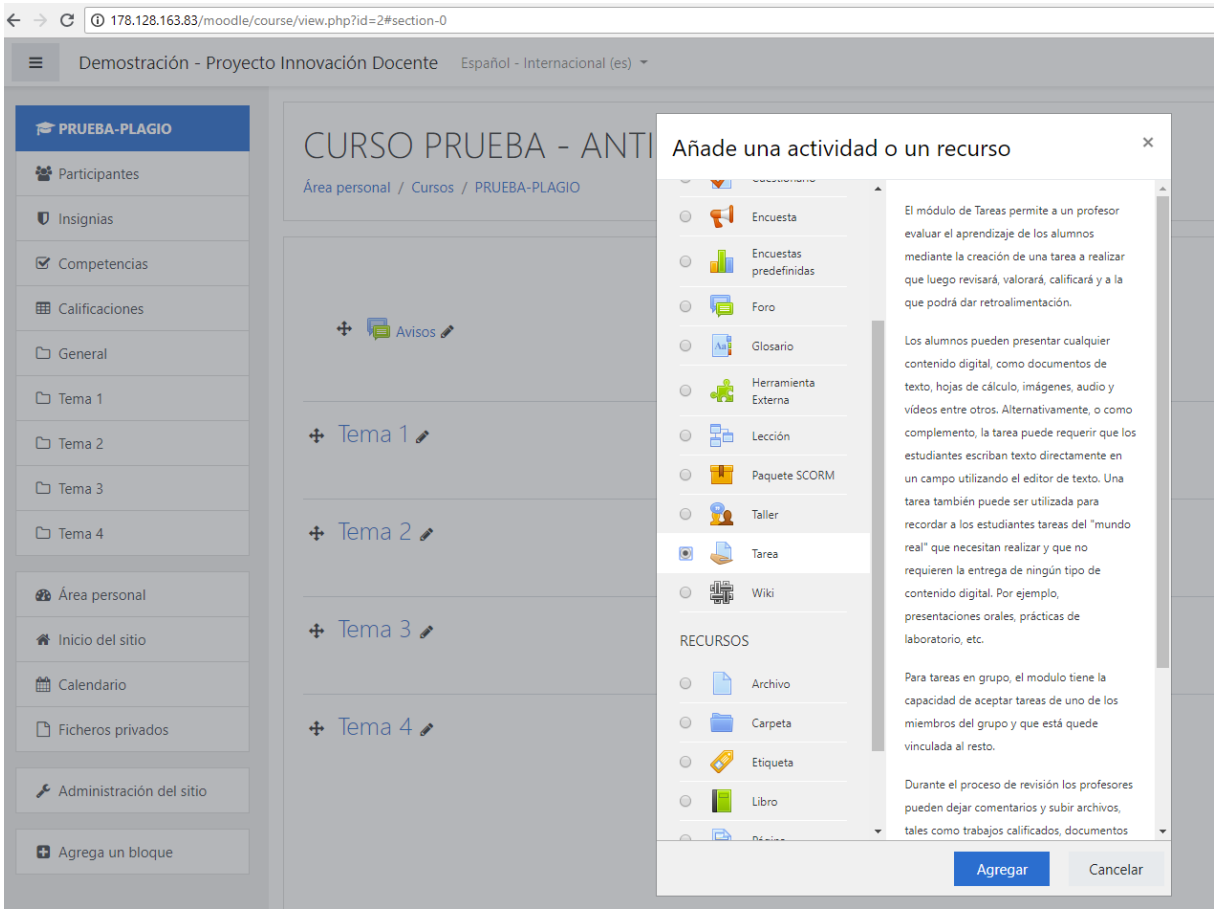
Una vez el sistema se ha activado, es necesario configurarlo, mediante el siguiente enlace:




ACTIVACIÓN ANTI-PLAGIO

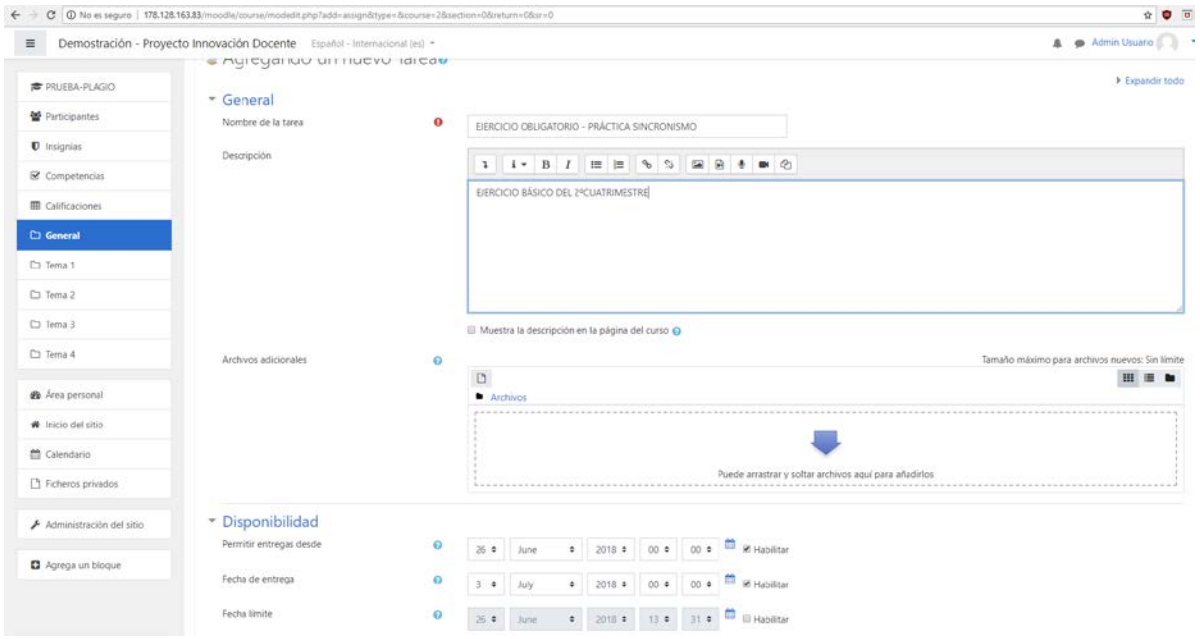
## Configuración Anti-Plagio en una Tarea:

Una vez realizado los pasos anteriores descritos en esta memoria, el trabajo que debe realizar un profesor para utilizar el sistema anti-plagio es muy sencillo. Lo primero que debe hacer el profesor es crear una tarea. En nuestro caso de prueba, vamos a crear una entrega para la asignatura de Sistemas Operativos II, donde los alumnos entregan prácticas realizadas en C.



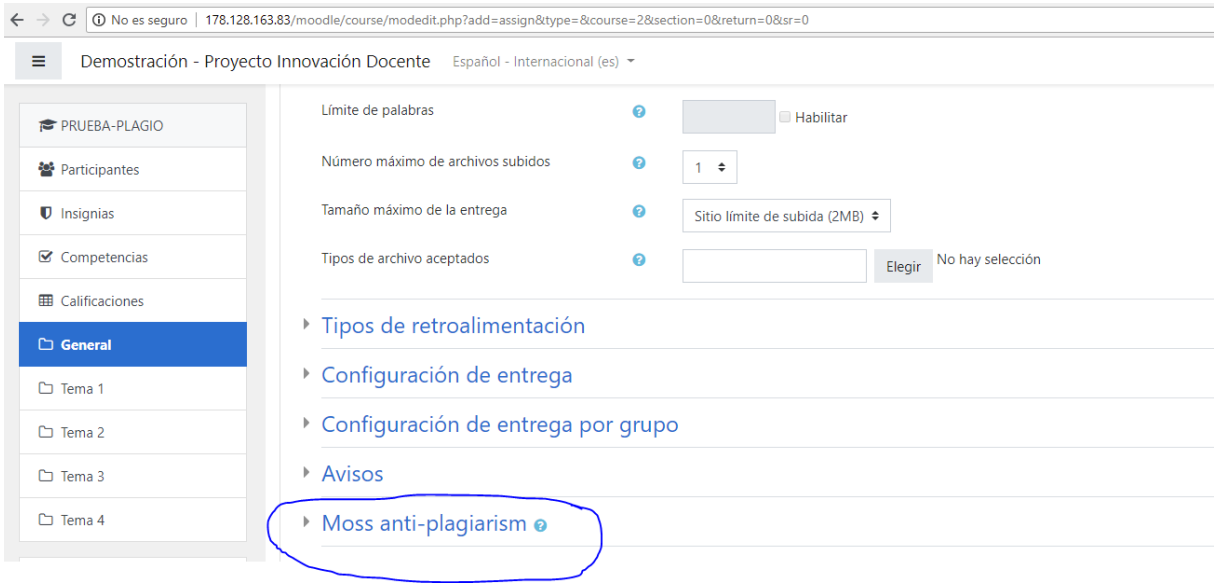

Creación de una Tarea

Se cumplimentarán los campos como se hace de forma habitual:

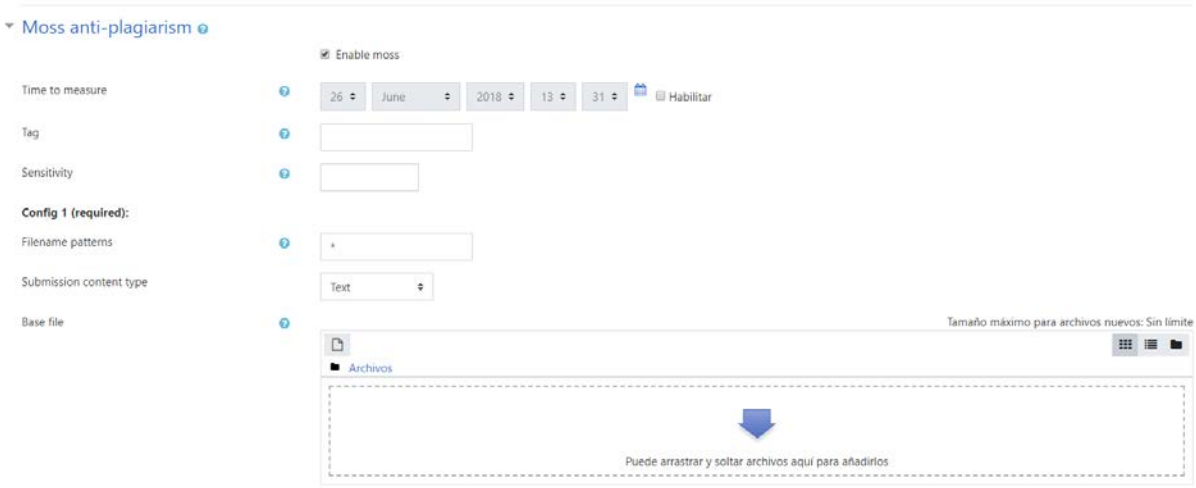



CREACIÓN DE UNA TAREA II

Ahora viene la parte más importante, en la parte de creación de una tarea, se habrá creado un submenú para configurar el sistema anti-plagio acorde a las necesidades particulares del profesor:




CREACIÓN DE UNA TAREA III




OPCIONES ANTI-PLAGIO DEL PROFESOR

El campo ***Time To Measure*** sirve para establecer el tiempo donde se ejecutarán las comprobaciones pertinentes que midan el grado de similitud de los trabajos entregados en la tarea. Si el campo no se establece, este chequeo se realizará una vez haya finalizado el tiempo límite para subir el ejercicio.

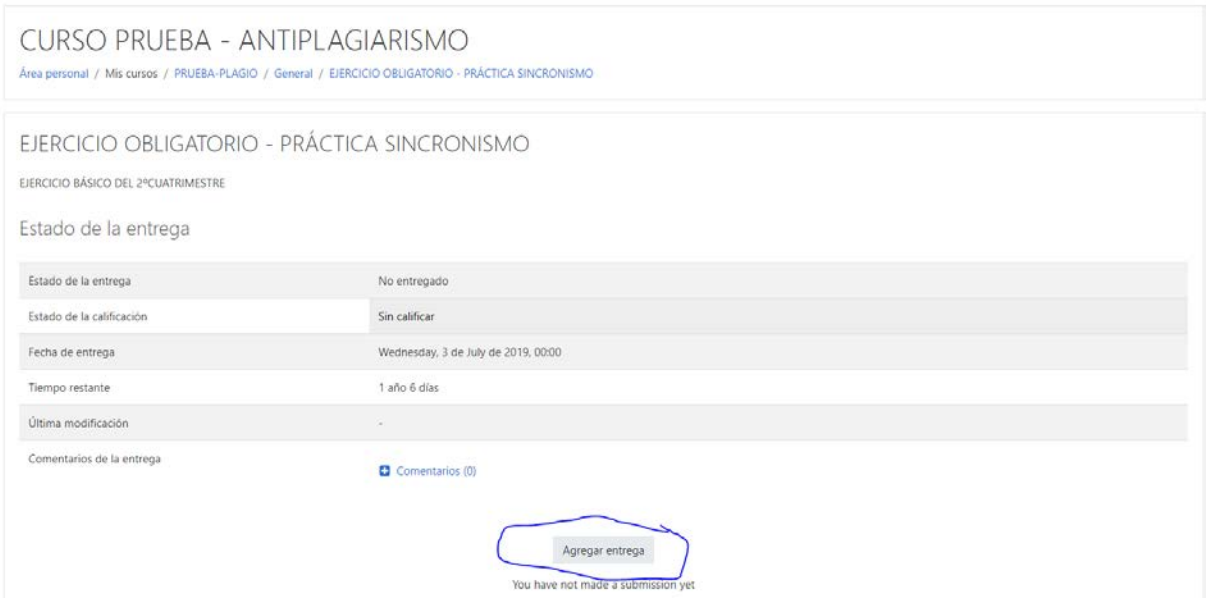
El campo **Enable moss** permite al profesor habilitar o deshabilitar la comprobación de plagios para una determinada tarea.

El campo **Sensitivity** permite al profesor ajustar el grado de permisividad del detector de plagios. Por defecto se recomienda el valor numérico 2.0, donde se establece que el código particular no aparezca en dos trabajos distintos. Sin embargo, si el número de trabajos es elevado y se utiliza un código base que va a ser utilizado por muchos alumnos, se recomienda aumentar este valor.

El campo **Filename patterns** permite establecer los ficheros que van a ser analizados para comprobar si hay fraude o no. Es decir, como el ejercicio de un alumno puede estar compuesto por una multitud de ficheros (PDF, CPP,.DOC...etc.), este campo sirve para indicar que ficheros serán analizados por el anti-plagio. Por defecto, el valor * indica que toda la totalidad de los ficheros serán analizados.

El campo **Submission content type** permite establecer el lenguaje de programación que se emplea en el ejercicio. Este campo resulta muy importante ya que el sistema realizará un análisis semántico y funcional de los códigos entregados con el objetivo de detectar un posible fraude. El número de lenguajes soportados para identificar los plagios se enumera a continuación: **C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, a8086 assembly, HCL2**.

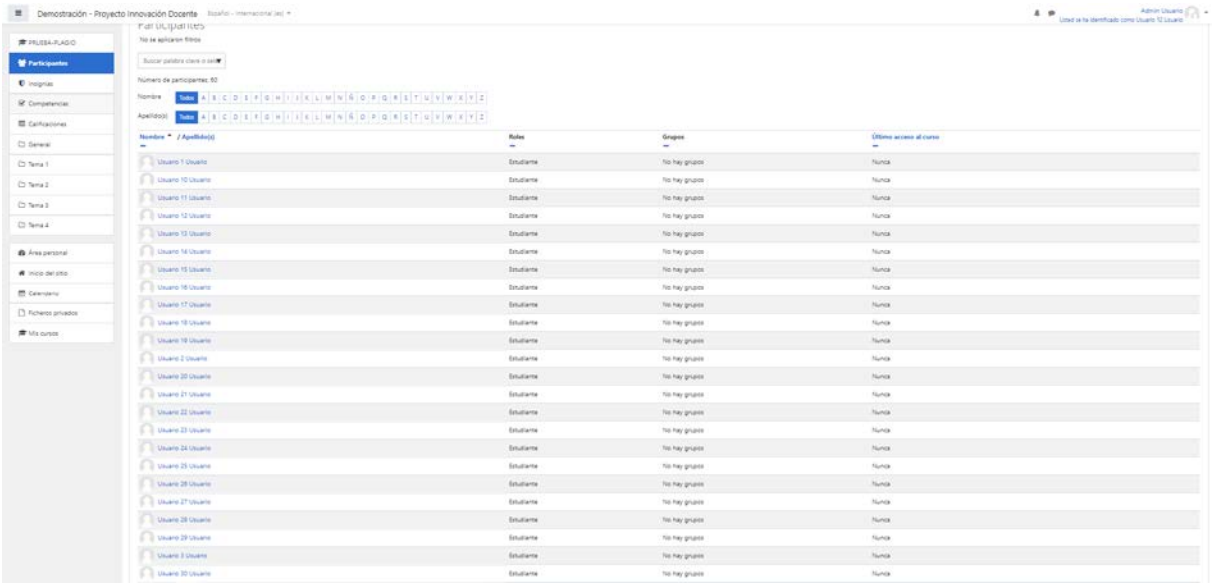
Finalmente, la tarea se ha creado con éxito y los alumnos pueden subir sus trabajos de forma habitual.




SUBIR TAREA PLATAFORMA

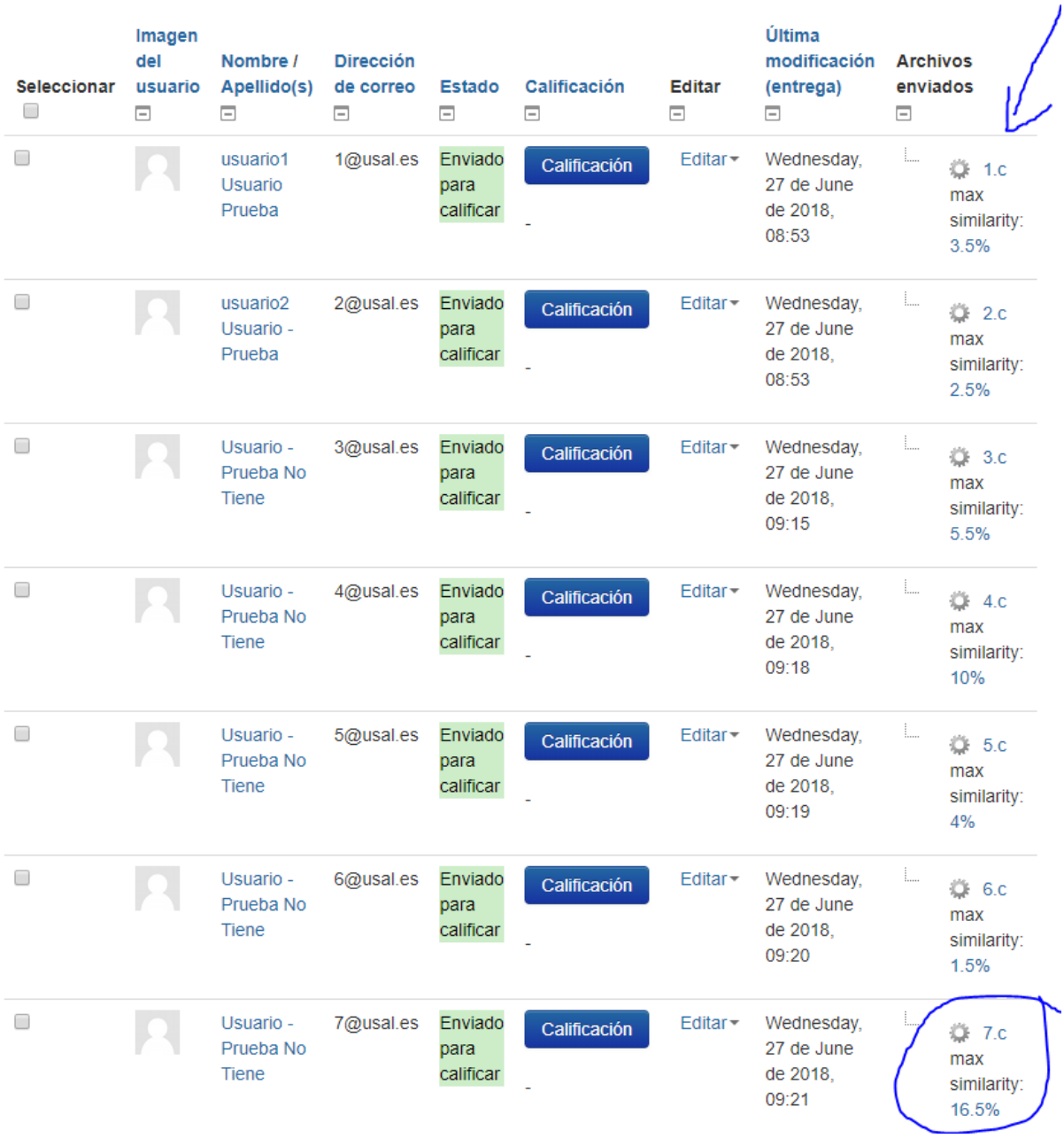
## Resultados:

Para el testeo y la verificación del sistema se ha realizado una entrega como prueba de la asignatura de Sistemas Operativos II del grado de Ingeniería Informática. Para salvaguardar la confidencialidad de los alumnos, el nombre se ha sustituido por “Alumno1, Alumno2… y así sucesivamente”. En total, la muestra de trabajos que ha participado en la tarea de validación es de 51. El profesor ha establecido que el sistema de plagiarismo debe de buscar posibles conflictos entre los ejercicios realizados en lenguaje de programación C.



LISTADO DE USUARIOS

Como puede apreciarse en la siguiente imagen el sistema es capaz de calcular el porcentaje de similitud entre ejercicios:

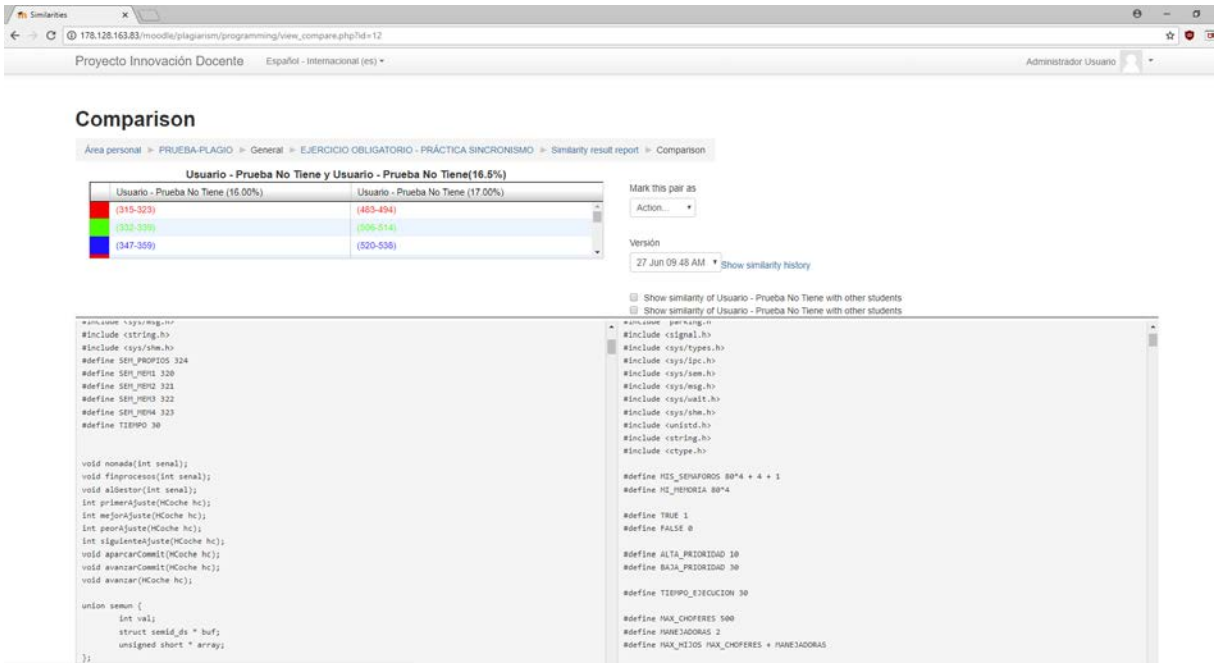


| Seleccionar | Imagen del usuario | Nombre / Apellido(s) | Dirección de correo | Estado | Calificación | Editar | Última modificación (entrega) | Archivos enviados |
|---|---|---|---|---|---|---|---|---|
| ☐ | | usuario1 Usuario Prueba | 1@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 08:53 | 1.c max similarity: 3.5% |
| ☐ | | usuario2 Usuario - Prueba | 2@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 08:53 | 2.c max similarity: 2.5% |
| ☐ | | Usuario - Prueba No Tiene | 3@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 09:15 | 3.c max similarity: 5.5% |
| ☐ | | Usuario - Prueba No Tiene | 4@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 09:18 | 4.c max similarity: 10% |
| ☐ | | Usuario - Prueba No Tiene | 5@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 09:19 | 5.c max similarity: 4% |
| ☐ | | Usuario - Prueba No Tiene | 6@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 09:20 | 6.c max similarity: 1.5% |
| ☐ | | Usuario - Prueba No Tiene | 7@usal.es | Enviado para calificar | Calificación - | Editar | Wednesday, 27 de June de 2018, 09:21 | 7.c max similarity: 16.5% |

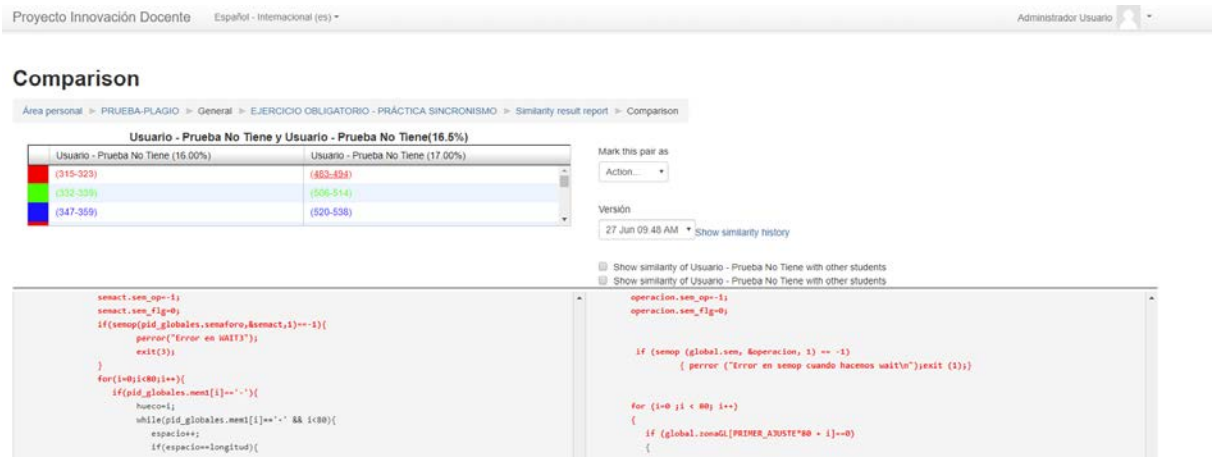
Si pulsamos sobre el porcentaje de similitud el sistema nos indica con que alumnos se ha encontrado algún tipo de coincidencia.


| | | | |
|---|---|---|---|
| 1 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 16.5% |
| 2 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 10.5% |
| 3 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 7.5% |
| 4 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 7.5% |
| 5 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 7.5% |
| 6 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 4% |
| 7 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 4% |
| 8 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 3.5% |
| 9 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 3.5% |
| 10 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 2% |
| 11 | Usuario - Prueba No Tiene | usuario2 Usuario - Prueba | 1.5% |
| 12 | Usuario - Prueba No Tiene | Usuario - Prueba No Tiene | 1% |

A continuación, se muestra un reporte entre dos alumnos cuyo porcentaje de similitud es de un 16,5%.




Si pulsamos sobre el icono rojo, el sistema nos indicará la parte del programa que ha sido considerada como plagio. Debemos recordar que el sistema es inteligente e interpreta los pseudocódigos de forma programática, es decir, no realiza un matching sintáctico, sino que es capaz de analizar el código y averiguar si ha existido un posible fraude como el que se muestra a continuación.




## Conclusiones:

Una vez finalizado este proyecto, se ha diseñado e implementado una utilidad que permite detectar los plagios en asignaturas de ingeniería. La plataforma studium carecía de un sistema con estas prestaciones y se ha conseguido integrar una solución compuesta de librerías y complementos de tipo open-source. Durante el desarrollo de este proyecto, se ha producido alguna dificultad en el despliegue de la solución en versiones de Moodle 3.X que al final han sido solventadas mediante la adaptación del código php. Los alumnos, tal y como refleja la encuesta recogida en este documento, se encuentran más tranquilos y se sienten más valorados conociendo que los profesores investigan y ponen en marcha soluciones y sistemas que penalizan a los tramposos. Sin la implantación de estos sistemas, los alumnos se desmotivan al comprobar que los plagios no son identificados y penalizados. La implantación de este sistema tiene un coste cero para la Universidad, lo que le permite ahorrarse costosas licencias en software propietario para uso similar. Los profesores pueden realizar una evaluación mucho más justa, evitando posibles enfrentamientos con el alumnado, ya que este software proporciona una prueba tecnológica fehaciente a la hora de demostrar que un trabajo tiene un grado de similitud muy grande con otro. Anteriormente, este proceso era manual, y hace que el profesor tenga que comparar trabajos de forma individual, siendo realmente complejo detectar un plagio. Tanto profesores como alumnos han valorado positivamente la facilidad de uso del sistema, así como su grado de utilidad. Todo el código utilizado en el desarrollo de este proyecto, ha sido publicado en un repositorio para que la comunidad académica pueda hacer uso de esta solución.

# CODIGO ADJUNTADO

File: /home/gabri/moodle/view.php

```php
<?php
require_once(dirname(dirname(dirname(__FILE__))).'/config.php');
require_once($CFG->dirroot.'/plagiarism/moss/locallib.php');

$cmid = optional_param('id', 0, PARAM_INT);  // Course Module ID
$mossid = optional_param('moss', 0, PARAM_INT);  // Moss ID
$userid  = optional_param('user', 0, PARAM_INT);   // User ID
$tab = optional_param('tab', 0, PARAM_INT);
$from = optional_param('from', 0, PARAM_INT);
$num = optional_param('num', 30, PARAM_INT);

if ($cmid) {
    if (! $cm = get_coursemodule_from_id('', $cmid)) {
        print_error('invalidcoursemodule');
    }
    if (! $moss = $DB->get_record("plagiarism_moss", array('cmid'=>$cmid))) {
        print_error('unsupportedmodule', 'plagiarism_moss');
    }
    if (! $course = $DB->get_record("course", array("id"=>$cm->course))) {
        print_error('coursemisconf', 'assignment');
    }
} else if ($moss) {
    if (! $moss = $DB->get_record("plagiarism_moss", array('cmid'=>$cmid))) {
        print_error('unsupportedmodule', 'plagiarism_moss');
    }
    if (! $cm = get_coursemodule_from_id('', $moss->cmid)) {
        print_error('invalidcoursemodule');
    }
    if (! $course = $DB->get_record("course", array("id"=>$cm->course))) {
        print_error('coursemisconf', 'assignment');
    }
} else {
    require_param('id', PARAM_INT);
}

$url = new moodle_url('/plagiarism/moss/view.php', array('id' => $cmid, 'tab' => $tab, 'from' =>
$from, 'num' => $num));
if ($userid != 0) {
    $url->param('user', $userid);
}

$PAGE->set_url($url);
require_login($course, true, $cm);

$context = get_context_instance(CONTEXT_MODULE, $cmid);

if ($userid != $USER->id) {
    require_capability('plagiarism/moss:viewallresults', $context);
}

$PAGE->set_context($context);
$PAGE->set_pagelayout('standard');

$modname = get_string('modulename', $cm->modname);
$activityname = $DB->get_field($cm->modname, 'name', array('id' => $cm->instance));
$pagetitle = strip_tags(format_string($activityname, true).': '.get_string('moss',
'plagiarism_moss'));
$heading = $course->fullname.': '.get_string('moss', 'plagiarism_moss');
$PAGE->set_title($pagetitle);
$PAGE->set_heading($heading);

$PAGE->navbar->add(get_string('allresults', 'plagiarism_moss'), new moodle_url('/plagiarism/moss/
view.php', array('id' => $cmid)));
if ($userid != 0) {
    $PAGE->navbar->add(get_string('personalresults', 'plagiarism_moss'));
}

$output = $PAGE->get_renderer('plagiarism_moss');
$output->moss = $moss;
$output->cm = $cm;

// confirm
$result  = optional_param('result', 0, PARAM_INT);
if ($result) {
    require_sesskey();
```

File: /home/gabri/moodle/view.php

```
73          require_capability('plagiarism/moss:confirm', $context);
74          $r = $DB->get_record('plagiarism_moss_results', array('id' => $result), 'moss, config, userid');
75          $r->confirmed = required_param('confirm', PARAM_BOOL);
76          $r->confirmer = $USER->id;
77          $r->timeconfirmed = time();
78          $results = $DB->get_records('plagiarism_moss_results', array('moss' => $r->moss, 'config' => $r-
    >config, 'userid' => $r->userid), 'id');
79          foreach ($results as $o) {
80              $r->id = $o->id;
81              $DB->update_record('plagiarism_moss_results', $r);
82          }
83
84          moss_message_send($r);
85
86          if (optional_param('ajax', 0, PARAM_BOOL)) {
87              echo $output->confirm_button($r, true);
88              die;
89          }
90      }
91
92      $jsmodule = array(
93          'name'     => 'plagiarism_moss',
94          'fullpath' => '/plagiarism/moss/module.js',
95          'requires' => array('base', 'io', 'node', 'json'),
96          'strings' => array(
97              array('confirmmessage', 'plagiarism_moss')
98          )
99      );
100     $updating_html = $output->pix_icon('i/loading_small', get_string('updating', 'plagiarism_moss'));
101     $PAGE->requires->js_init_call('M.plagiarism_moss.init', array($updating_html), false, $jsmodule);
102
103     /// Output starts here
104     echo $output->header();
105
106     if ($userid) {
107         $user = $DB->get_record('user', array('id' => $userid));
108         echo $output->user_result($user);
109     } else {
110         echo $output->cm_result($tab, $from, $num);
111     }
112
113     echo $output->footer();
```

File: /home/gabri/moodle/version.php

```php
<?php
$plugin->version   = 2012051000; // The current module version (Date: YYYYMMDDXX)
$plugin->requires  = 2010122900; // Requires this Moodle version

$module->maturity  = MATURITY_BETA;
$module->release   = "2.1 beta"; // User-friendly version number
```

File: /home/gabri/moodle/textlib.php

```php
<?php
function getTextFromZippedXML($archiveFile, $contentFile) {
        // create zip archive in the memory
        $zip = new ZipArchive;
        // open zip file
        if ($zip->open($archiveFile)) {
                // check the file in the archive
                if (($index = $zip->locateName($contentFile)) !== false) {
                        // if found read in text variable
                        $content = $zip->getFromIndex($index);
                        // close the archive, we don't need it anymore
                        $zip->close();
                        // sw
                        $content=str_replace("<w:p ","\n<w:p ",$content);
                        // TODO add all entities and includes
                        // skip all errors and warnings
                        $xml = DOMDocument::loadXML($content, LIBXML_NOENT | LIBXML_XINCLUDE | LIBXML_NOERROR | LIBXML_NOWARNING);
                        // return data without wml tags
                        return strip_tags($xml->saveXML());
                } else {echo "Not found!";}
                $zip->close();
        }
     // if something wron return ERROR text
     return "ERROR in text Tokenization";
}

function rtf_isPlainText($s) {
    $failAt = array("*", "fonttbl", "colortbl", "datastore", "themedata");
    for ($i = 0; $i < count($failAt); $i++)
        if (!empty($s[$failAt[$i]])) return false;
    return true;
}

function rtf2text($filename) {
    $text = file_get_contents($filename);
    if (!strlen($text))
        return "";


    // start with empty stack of modifiers
    $document = "";
    $stack = array();
    $j = -1;

    // read chars from buffer...
    for ($i = 0, $len = strlen($text); $i < $len; $i++) {
        $c = $text[$i];

        // select what to do with the current char
        switch ($c) {
            // the most important key \
            case "\\":
                // read the next char
                $nc = $text[$i + 1];

                // put into the out stream
                if ($nc == '\\' && rtf_isPlainText($stack[$j])) $document .= '\\';
                elseif ($nc == '~' && rtf_isPlainText($stack[$j])) $document .= ' ';
                elseif ($nc == '_' && rtf_isPlainText($stack[$j])) $document .= '-';
                // * goes to stack
                elseif ($nc == '*') $stack[$j]["*"] = true;
                elseif ($nc == "'") {
                    $hex = substr($text, $i + 2, 2);
                    if (rtf_isPlainText($stack[$j]))
                        $document .= html_entity_decode("&#".hexdec($hex).";");
                    // move the index
                    $i += 2;
                // read the key symbol
                } elseif ($nc >= 'a' && $nc <= 'z' || $nc >= 'A' && $nc <= 'Z') {
                    $word = "";
                    $param = null;

                    // read after \
                    for ($k = $i + 1, $m = 0; $k < strlen($text); $k++, $m++) {
```

File: /home/gabri/moodle/textlib.php

```
                        $nc = $text[$k];

                        if ($nc >= 'a' && $nc <= 'z' || $nc >= 'A' && $nc <= 'Z') {
                            if (empty($param))
                                $word .= $nc;
                            else
                                break;

                        } elseif ($nc >= '0' && $nc <= '9')
                            $param .= $nc;

                        elseif ($nc == '-') {
                            if (empty($param))
                                $param .= $nc;
                            else
                                break;
                        // end
                        } else
                            break;
                    }
                    // move the index
                    $i += $m - 1;

                    // read the word
                    $toText = "";
                    switch (strtolower($word)) {

                        case "u":
                            $toText .= html_entity_decode("&#x".dechex($param).";");
                            $ucDelta = @$stack[$j]["uc"];
                            if ($ucDelta > 0)
                                $i += $ucDelta;
                        break;

                        case "par": case "page": case "column": case "line": case "lbr":
                            $toText .= "\n";
                        break;
                        case "emspace": case "enspace": case "qmspace":
                            $toText .= " ";
                        break;
                        case "tab": $toText .= "\t"; break;

                        case "chdate": $toText .= date("m.d.Y"); break;
                        case "chdpl": $toText .= date("l, j F Y"); break;
                        case "chdpa": $toText .= date("D, j M Y"); break;
                        case "chtime": $toText .= date("H:i:s"); break;

                        case "emdash": $toText .= html_entity_decode("&mdash;"); break;
                        case "endash": $toText .= html_entity_decode("&ndash;"); break;
                        case "bullet": $toText .= html_entity_decode("&#149;"); break;
                        case "lquote": $toText .= html_entity_decode("&lsquo;"); break;
                        case "rquote": $toText .= html_entity_decode("&rsquo;"); break;
                        case "ldblquote": $toText .= html_entity_decode("&laquo;"); break;
                        case "rdblquote": $toText .= html_entity_decode("&raquo;"); break;

                        default:
                            $stack[$j][strtolower($word)] = empty($param) ? true : $param;
                        break;
                    }

                    if (rtf_isPlainText($stack[$j]))
                        $document .= $toText;
                }

                $i++;
            break;

            case "{":
                array_push($stack, $stack[$j++]);
            break;
            // } removes current stack. Group is over.
            case "}":
                array_pop($stack);
                $j--;
            break;
```

File: /home/gabri/moodle/textlib.php

```
            //
            case '\0': case '\r': case '\f': case '\n': break;
            //
            default:
                if (rtf_isPlainText($stack[$j]))
                    $document .= $c;
            break;
        }
    }
    //
    return $document;
}

function decodeAsciiHex($input) {
    $output = "";

    $isOdd = true;
    $isComment = false;

    for($i = 0, $codeHigh = -1; $i < strlen($input) && $input[$i] != '>'; $i++) {
        $c = $input[$i];

        if($isComment) {
            if ($c == '\r' || $c == '\n')
                $isComment = false;
            continue;
        }

        switch($c) {
            case '\0': case '\t': case '\r': case '\f': case '\n': case ' ': break;
            case '%':
                $isComment = true;
            break;

            default:
                $code = hexdec($c);
                if($code === 0 && $c != '0')
                    return "";

                if($isOdd)
                    $codeHigh = $code;
                else
                    $output .= chr($codeHigh * 16 + $code);

                $isOdd = !$isOdd;
            break;
        }
    }

    if($input[$i] != '>')
        return "";

    if($isOdd)
        $output .= chr($codeHigh * 16);

    return $output;
}
function decodeAscii85($input) {
    $output = "";

    $isComment = false;
    $ords = array();

    for($i = 0, $state = 0; $i < strlen($input) && $input[$i] != '~'; $i++) {
        $c = $input[$i];

        if($isComment) {
            if ($c == '\r' || $c == '\n')
                $isComment = false;
            continue;
        }

        if ($c == '\0' || $c == '\t' || $c == '\r' || $c == '\f' || $c == '\n' || $c == ' ')
            continue;
        if ($c == '%') {
```

File: /home/gabri/moodle/textlib.php

```
225                 $isComment = true;
226                 continue;
227             }
228             if ($c == 'z' && $state === 0) {
229                 $output .= str_repeat(chr(0), 4);
230                 continue;
231             }
232             if ($c < '!' || $c > 'u')
233                 return "";
234
235             $code = ord($input[$i]) & 0xff;
236             $ords[$state++] = $code - ord('!');
237
238             if ($state == 5) {
239                 $state = 0;
240                 for ($sum = 0, $j = 0; $j < 5; $j++)
241                     $sum = $sum * 85 + $ords[$j];
242                 for ($j = 3; $j >= 0; $j--)
243                     $output .= chr($sum >> ($j * 8));
244             }
245         }
246         if ($state === 1)
247             return "";
248         elseif ($state > 1) {
249             for ($i = 0, $sum = 0; $i < $state; $i++)
250                 $sum += ($ords[$i] + ($i == $state - 1)) * pow(85, 4 - $i);
251             for ($i = 0; $i < $state - 1; $i++)
252                 $output .= chr($sum >> ((3 - $i) * 8));
253         }
254
255         return $output;
256     }
257     function decodeFlate($input) {
258         // The most common compression method for data streams in PDF.
259         // Very easy to deal with using libraries.
260         return @gzuncompress($input);
261     }
262
263     function getObjectOptions($object) {
264         // We need to get current object attrbutes. These attributes are
265         // located between << and >>. Each option starts with /.
266         $options = array();
267         if (preg_match("#<<(.*)>>#ismU", $object, $options)) {
268             // Separate options from each other using /. First empty one should be removed from the
    array.
269             $options = explode("/", $options[1]);
270             @array_shift($options);
271
272             // Create handy array for current object attributes
273             // Attributs that look like "/Option N" will be written to hash
274             // as "Option" => N, and properties like "/Param", will be written as
275             // "Param" => true.
276             $o = array();
277             for ($j = 0; $j < @count($options); $j++) {
278                 $options[$j] = preg_replace("#\s+#", " ", trim($options[$j]));
279                 if (strpos($options[$j], " ") !== false) {
280                     $parts = explode(" ", $options[$j]);
281                     $o[$parts[0]] = $parts[1];
282                 } else
283                     $o[$options[$j]] = true;
284             }
285             $options = $o;
286             unset($o);
287         }
288
289         // Return an array of parameters we found
290         return $options;
291     }
292     function getDecodedStream($stream, $options) {
293         // Now we have a stream that is possibly coded with some compression method(s)
294         // Lets try to decode it.
295         $data = "";
296         // If current stream has Filter attribute, then is is definately compressed or en coded
297         // Otherwise just return the content
298         if (empty($options["Filter"]))
```

File: /home/gabri/moodle/textlib.php

```
299             $data = $stream;
300         else {
301             // If we know the size of data stream from options then we need to cut the data
302             // using this size, or we may not be able to decode it or maybe something else will go
    wring
303             $length = !empty($options["Length"]) ? $options["Length"] : strlen($stream);
304             $_stream = substr($stream, 0, $length);
305 
306             // Looping through options looking for indicatiors of data compression in the current
    stream.
307             // PDF supprts many different stuff, but text can be coded either by ASCII Hex, or ASCII
    85-base or GZ/Deflate
308             // We need to look for these keys and apply respecrtive functions for decoding.
309             // There is another option: Crypt, but we are not going to work with encrypted PDF's.
310             foreach ($options as $key => $value) {
311                 if ($key == "ASCIIHexDecode")
312                     $_stream = decodeAsciiHex($_stream);
313                 if ($key == "ASCII85Decode")
314                     $_stream = decodeAscii85($_stream);
315                 if ($key == "FlateDecode")
316                     $_stream = decodeFlate($_stream);
317             }
318             $data = $_stream;
319         }
320         // Return the result
321         return $data;
322     }
323     function getDirtyTexts(&$texts, $textContainers) {
324         // So we have an array of text contatiners that were taken from both  BT and ET.
325         // Our new task is to find a text in them that would be displayed by viewers
326         // on the screen. There are many options to do that, Lets check the pair: [...] TJ and Td
    (...) Tj
327         for ($j = 0; $j < count($textContainers); $j++) {
328             // Add the pieces of row data the we found to the general array of text objects.
329             if (preg_match_all("#\[(.*)\]\s*TJ#ismU", $textContainers[$j], $parts))
330                 $texts = array_merge($texts, @$parts[1]);
331             elseif(preg_match_all("#Td\s*(\(.*\))\s*Tj#ismU", $textContainers[$j], $parts))
332                 $texts = array_merge($texts, @$parts[1]);
333         }
334     }
335     function getCharTransformations(&$transformations, $stream) {
336         preg_match_all("#([0-9]+)\s+beginbfchar(.*)endbfchar#ismU", $stream, $chars, PREG_SET_ORDER);
337         preg_match_all("#([0-9]+)\s+beginbfrange(.*)endbfrange#ismU", $stream, $ranges,
    PREG_SET_ORDER);
338 
339         // First of all process separate symbols. Transformaiton string looks as follows:
340         // - <0123> <abcd> -> 0123 should be transformed to abcd;
341         // - <0123> <abcd6789> -> 0123 should be transformed to many symbols (abcd and 6789 in this
    case)
342         for ($j = 0; $j < count($chars); $j++) {
343             // There is a number of strings before data list that we are going ot read. We gonna use
    it later on.
344             $count = $chars[$j][1];
345             $current = explode("\n", trim($chars[$j][2]));
346             // Read data from each string.
347             for ($k = 0; $k < $count && $k < count($current); $k++) {
348                 // Wrute the transformation we just found. Don't forget about writing leading zeros if
    there are less then 4 digits..
349                 if (preg_match("#<([0-9a-f]{2,4})>\s+<([0-9a-f]{4,512})>#is", trim($current[$k]),
    $map))
350                     $transformations[str_pad($map[1], 4, "0")] = $map[2];
351             }
352         }
353         // Now we can deal with sequences. Manuals are saying that they can be one of two possible
    types
354         // - <0000> <0020> <0a00> -> in this case  <0000> will be substituted with <0a00>, <0001> with
    <0a01> and so on
355         //   till  <0020>, that will be substituted with <0a20>.
356         // OR
357         // - <0000> <0002> [<abcd> <01234567> <8900>] -> here it works in a bit different way. We need
    to look how
358         //   many elemants are located between  <0000> and <0002> (its actually three including 0001).
    After it we assign to each element
359         //   a corresponding value from [ ]: 0000 -> abcd, 0001 -> 0123 4567, a 0002 -> 8900.
360         for ($j = 0; $j < count($ranges); $j++) {
361             // We need to cross check the number of elements for transofrmation.
362             $count = $ranges[$j][1];
363             $current = explode("\n", trim($ranges[$j][2]));
```

File: /home/gabri/moodle/textlib.php

```
364          // Working with each string
365          for ($k = 0; $k < $count && $k < count($current); $k++) {
366              // This is first type sequence.
367              if (preg_match("#<([0-9a-f]{4})>\s+<([0-9a-f]{4})>\s+<([0-9a-f]{4})>#is",
     trim($current[$k]), $map)) {
368                  // Convert data into decimal system: looping will be easier.
369                  $from = hexdec($map[1]);
370                  $to = hexdec($map[2]);
371                  $_from = hexdec($map[3]);
372
373                  // We put all the elements from the sequence into transformations array.
374                  // According to manuals we need also to ass leading zeros if hex-code size is less
     than 4 symbols.
375                  for ($m = $from, $n = 0; $m <= $to; $m++, $n++)
376                      $transformations[sprintf("%04X", $m)] = sprintf("%04X", $_from + $n);
377              // Second option.
378              } elseif (preg_match("#<([0-9a-f]{4})>\s+<([0-9a-f]{4})>\s+\[(.*)\]#ismU",
     trim($current[$k]), $map)) {
379                  // This is also beginnigna nd end of the sequence. Split data in [ ] by symbols
     located near to spaces.
380                  $from = hexdec($map[1]);
381                  $to = hexdec($map[2]);
382                  $parts = preg_split("#\s+#", trim($map[3]));
383
384                  // Loop through data and assign the new values accordingly.
385                  for ($m = $from, $n = 0; $m <= $to && $n < count($parts); $m++, $n++)
386                      $transformations[sprintf("%04X", $m)] = sprintf("%04X", hexdec($parts[$n]));
387              }
388          }
389      }
390  }
391  function getTextUsingTransformations($texts, $transformations) {
392
393      // Lets go. We are accumulating text data processign "raw" pieces of text
394      $document = "";
395      for ($i = 0; $i < count($texts); $i++) {
396          // 2 cases are possible: text can be either in <> (hex) or in () (plain).
397          $isHex = false;
398          $isPlain = false;
399
400          $hex = "";
401          $plain = "";
402          // scan current piece of text.
403          for ($j = 0; $j < strlen($texts[$i]); $j++) {
404              // get current char
405              $c = $texts[$i][$j];
406              // ...and decide what to do with it.
407              switch($c) {
408                  // We have hex data in front of us
409                  case "<":
410                      $hex = "";
411                      $isHex = true;
412                  break;
413                  // Hex data are over. Lets parse them.
414                  case ">":
415                      // split the string into chunks of 4 chars...
416                      $hexs = str_split($hex, 4);
417                      // ...and cheking what we can do with each chunk
418                      for ($k = 0; $k < count($hexs); $k++) {
419                          // if there are less then 4 symbols then the manual says that we need to
     add zeros after them
420                          $chex = str_pad($hexs[$k], 4, "0");
421                          // Checking if current hex-code is already in transformations.
422                          // If this is the case change this piece to the required.
423                          if (isset($transformations[$chex]))
424                              $chex = $transformations[$chex];
425                          // Write a new Unicode symbol into the output .
426                          $document .= html_entity_decode("&#x".$chex.";");
427                      }
428                      // Hex-sata are over. Need to say it.
429                      $isHex = false;
430                  break;
431                  // There is a piece of "plain" text
432                  case "(":
433                      $plain = "";
434                      $isPlain = true;
```

```
435                 break;
436                 // Well... this piece will be over sometime.
437                 case ")":
438                     // Get the text we just got into the output stream.
439                     $document .= $plain;
440                     $isPlain = false;
441                 break;
442                 // Specail symbol. Lets see what is located after it.
443                 case "\\":
444                     $c2 = $texts[$i][$j + 1];
445                     // If it is  \ ot either one of ( or ), then print them as it is.
446                     if (in_array($c2, array("\\", "(", ")"))) $plain .= $c2;
447                     // If it is empty space of EOL then process it.
448                     elseif ($c2 == "n") $plain .= '\n';
449                     elseif ($c2 == "r") $plain .= '\r';
450                     elseif ($c2 == "t") $plain .= '\t';
451                     elseif ($c2 == "b") $plain .= '\b';
452                     elseif ($c2 == "f") $plain .= '\f';
453                     // It might happen that a digit follows after \ . It may be up to 3 of them.
454                     // They represent sybmol code in octal system. Lets parse them.
455                     elseif ($c2 >= '0' && $c2 <= '9') {
456                         // We need 3 digits. No more than 3. Digits only.
457                         $oct = preg_replace("#[^0-9]#", "", substr($texts[$i], $j + 1, 3));
458                         // Getting the number of characters we already have taken. We need it to
    shift the position of current char properly.
459                         $j += strlen($oct) - 1;
460                         // Put the respective char into "plain" text.
461                         $plain .= html_entity_decode("&#".octdec($oct).";");
462                     }
463                     // We increased the position of current symbol at least by one. Need to inform
    parser about that.
464                     $j++;
465                 break;
466
467                 // If we have something else then write current symbol into temporaty hex string
    (if we had < before),
468                 default:
469                     if ($isHex)
470                         $hex .= $c;
471                     // or into "plain" string if ( was opeon.
472                     if ($isPlain)
473                         $plain .= $c;
474                 break;
475             }
476         }
477         // Define text blocks by EOL
478         $document .= "\n";
479     }
480
481     // Return text.
482     return $document;
483 }
484
485 function pdf2text($filename) {
486     // Read from the pdf file into string keeping in mind that file may contain binary streams
487     $infile = @file_get_contents($filename, FILE_BINARY);
488     if (empty($infile))
489         return "";
490
491     // First iteration. We need to get all the text data from file.
492     // We'll get only "raw" data after the firs iteration. These data will include positioning,
493     // hex entries, etc.
494     $transformations = array();
495     $texts = array();
496
497     // Get list of all files from pdf file.
498     preg_match_all("#obj(.*)endobj#ismU", $infile, $objects);
499     $objects = @$objects[1];
500
501     // Let start the crawling. Apart fromthe text we can meet some other stuff including fonts.
502     for ($i = 0; $i < count($objects); $i++) {
503         $currentObject = $objects[$i];
504
505         // Check if there is data stream in the current object.
506         // Almost all the time it will be compressed with gzip.
```

File: /home/gabri/moodle/textlib.php

```
507             if (preg_match("#stream(.*)endstream#ismU", $currentObject, $stream)) {
508                 $stream = ltrim($stream[1]);
509 
510                 // Read the attributes of this object. We are looking only
511                 // for text, so we have to do minimal cuts to improve the speed
512                 $options = getObjectOptions($currentObject);
513                 if (!(empty($options["Length1"]) && empty($options["Type"]) &&
    empty($options["Subtype"])))
514                     continue;
515 
516                 // So, we "may" have text in from of us. Lets decode it from binary file to get the
    plain text.
517                 $data = getDecodedStream($stream, $options);
518                 if (strlen($data)) {
519                     // We need to find text container in the current stream.
520                     // If we will be able to get it the raw text we found will be added to the
    previous findings.
521                     if (preg_match_all("#BT(.*)ET#ismU", $data, $textContainers)) {
522                         $textContainers = @$textContainers[1];
523                         getDirtyTexts($texts, $textContainers);
524                     // Otherwise we'll try to use symbol transformations that we gonna use on the 2nd
    step.
525                     } else
526                         getCharTransformations($transformations, $data);
527                 }
528             }
529         }
530 
531         // After the preliminary parsing of  pdf-document we need to parse
532         // the text blocks we got in the context of simbolic transformations. Return the result after
    we done.
533         return getTextUsingTransformations($texts, $transformations);
534     }
535 
536     class cfb {
537         // We gonna read the content of the file we need to decode into this variable.
538         protected $data = "";
539 
540         // Sizes of FAT sector (1 << 9 = 512), Mini FAT sector (1 << 6 = 64) and maximum size
541         // of the stream that could be written into a miniFAT.
542         protected $sectorShift = 9;
543         protected $miniSectorShift = 6;
544         protected $miniSectorCutoff = 4096;
545 
546         // FAT-sector sequence array and Array of "files" belonging to this file structure
547         protected $fatChains = array();
548         protected $fatEntries = array();
549 
550         // Array of sequences of  Mini FAT-sectors and the whole Mini FAT of our file
551         protected $miniFATChains = array();
552         protected $miniFAT = "";
553 
554         // Version (3 or 4), and way to write numbers (little-endian)
555         private $version = 3;
556         private $isLittleEndian = true;
557 
558         // The number of "files" and the position fo the first "file" in FAT
559         private $cDir = 0;
560         private $fDir = 0;
561 
562         // The number of FAT sectors in the file
563         private $cFAT = 0;
564 
565         // The number of miniFAT-sectors and position of sequences of miniFAT-csectors in the file
566         private $cMiniFAT = 0;
567         private $fMiniFAT = 0;
568 
569         // DIFAT: number of such sectors and offset to sector 110 (first 109 sectors are located in
    the header)
570         private $DIFAT = array();
571         private $cDIFAT = 0;
572         private $fDIFAT = 0;
573 
574         // Constants: end of sequence and empty sector (4 bytes each)
575         const ENDOFCHAIN = 0xFFFFFFFE;
576         const FREESECT   = 0xFFFFFFFF;
```

File: /home/gabri/moodle/textlib.php

```
577
578         // Read the file into internal variable
579         public function read($filename) {
580             $this->data = file_get_contents($filename);
581         }
582
583         public function parse() {
584             // First of all we need to check weither we really have CFB in front of us.?
585             // To do it we read the first 8 bytes and compare them with 2 patterns: common and the old
    one
586             $abSig = strtoupper(bin2hex(substr($this->data, 0, 8)));
587             if ($abSig != "D0CF11E0A1B11AE1" && $abSig != "0E11FC0DD0CF11E0") { return false; }
588
589             // Read the file header;
590             $this->readHeader();
591             // get the remaining DIFAT sectors if any;
592             $this->readDIFAT();
593             // read the sequence of FAT sectors
594             $this->readFATChains();
595             // read the sequence of MiniFAT-sectors
596             $this->readMiniFATChains();
597             // read the structure of "directories" within the file
598             $this->readDirectoryStructure();
599
600             // Finally we need to check the root entry in the file structure.
601             // This stream is required ot be in a file at least because it has a link
602             // to file's miniFAT that we gonna read into $this->miniFAT
603
604             $reStreamID = $this->getStreamIdByName("Root Entry");
605             if ($reStreamID === false) { return false; }
606             $this->miniFAT = $this->getStreamById($reStreamID, true);
607
608             // Remove the unnecessary link to DIFAT-sectors, we have "stolen" complete FAT sequences
    instead of them.
609             unset($this->DIFAT);
610
611             // After all this we should be able to work with any of the "upper" formats from Microsoft
    such as doc, xls или ppt.
612         }
613
614         // Function that looks for stream number in the directory structure by its name.
615         // It returns false if nothing was found.
616         public function getStreamIdByName($name) {
617             for($i = 0; $i < count($this->fatEntries); $i++) {
618                 if ($this->fatEntries[$i]["name"] == $name)
619                     return $i;
620             }
621             return false;
622         }
623         // Function gets the stream number ($id) and a second parameter (second perameter is required
    for the root entry only).
624         // It returns the binary content fo this stream.
625         public function getStreamById($id, $isRoot = false) {
626             $entry = $this->fatEntries[$id];
627             // Get the size and offset position to the content of "current" file.
628             $from = $entry["start"];
629             $size = $entry["size"];
630
631             // Now 2 options are possible: is size is less than  4096 byte, then we need ot read data
632             // from MiniFAT. If more than 4096 read from the common FAT. RootEntry is an exclusion:
633             // we need ot read contents from FAT as miniFAT is located there.
634
635             $stream = "";
636             // So, here is the 1st option: small size and not root.
637             if ($size < $this->miniSectorCutoff && !$isRoot) {
638                 // Get the miniFAT sector size - 64 bytes
639                 $ssize = 1 << $this->miniSectorShift;
640
641                 do {
642                     // Get the offset in miniFAT
643                     $start = $from << $this->miniSectorShift;
644                     // Read miniFAT-sector
645                     $stream .= substr($this->miniFAT, $start, $ssize);
646                     // Get the next piece of miniFAT in the array of chains
647                     $from = $this->miniFATChains[$from];
648                     // While not end of chain (sequence).
```

File: /home/gabri/moodle/textlib.php

```
649             } while ($from != self::ENDOFCHAIN);
650         } else {
651             // Second option - large piece - read it from FAT.
652             // Get the sector size  - 512 (or 4096 for new versions)
653             $ssize = 1 << $this->sectorShift;
654 
655             do {
656                 // Getting the offset in the file (taking into account that there is a header of
    512 bytes in the begining)
657                 $start = ($from + 1) << $this->sectorShift;
658                 // Read a sector
659                 $stream .= substr($this->data, $start, $ssize);
660                 // Get the next sector inthe array of FAT chains
661                 $from = $this->fatChains[$from];
662                 // While not end of chain (sequence).
663             } while ($from != self::ENDOFCHAIN);
664         }
665         // Return the stream content accrding to its size.
666         return substr($stream, 0, $size);
667     }
668 
669     // This function reads data from file header
670     private function readHeader() {
671         // We need to get the information about the data format in the file
672         $uByteOrder = strtoupper(bin2hex(substr($this->data, 0x1C, 2)));
673         // We need to check if it is  little-endian record
674         $this->isLittleEndian = $uByteOrder == "FEFF";
675 
676         // Version 3 or 4 (never actually met 4th, but its description appears in the manual)
677         $this->version = $this->getShort(0x1A);
678 
679         // Offsets for FAT and miniFAT
680         $this->sectorShift = $this->getShort(0x1E);
681         $this->miniSectorShift = $this->getShort(0x20);
682         $this->miniSectorCutoff = $this->getLong(0x38);
683 
684         // Number of entries in the directory and offset to the first description in the file
685         if ($this->version == 4)
686             $this->cDir = $this->getLong(0x28);
687         $this->fDir = $this->getLong(0x30);
688 
689         // Number of FAT sectors in the file
690         $this->cFAT = $this->getLong(0x2C);
691 
692         // Number and position of hte 1st miniFAT-sector of sequences.
693         $this->cMiniFAT = $this->getLong(0x40);
694         $this->fMiniFAT = $this->getLong(0x3C);
695 
696         // Where are the FAT sector chains and how many of them are there.
697         $this->cDIFAT = $this->getLong(0x48);
698         $this->fDIFAT = $this->getLong(0x44);
699     }
700 
701     // So.... DIFAT. DIFAT shows in which sectors we can find descriptions of FAT sector chains
702     // Without these chains we won't be able to get stream contents in fragmented files
703     private function readDIFAT() {
704         $this->DIFAT = array();
705         // First 109 links to sequences are being stored in the header of our file
706         for ($i = 0; $i < 109; $i++)
707             $this->DIFAT[$i] = $this->getLong(0x4C + $i * 4);
708 
709         // we also check if there are other links to chains. in small (upto 8.5MB) there is no such
710         // links but in larger files we have to read them.
711         if ($this->fDIFAT != self::ENDOFCHAIN) {
712             // Sector size and start position to read links.
713             $size = 1 << $this->sectorShift;
714             $from = $this->fDIFAT;
715             $j = 0;
716 
717             do {
718                 // Get the position in the file considering header
719                 $start = ($from + 1) << $this->sectorShift;
720                 // Read the links to sequences' sectors
721                 for ($i = 0; $i < ($size - 4); $i += 4)
722                     $this->DIFAT[] = $this->getLong($start + $i);
```

```
723                 // Getting the next  DIFAT-sector. Link to this sector is written
724                 // as the last "word" in the current  DIFAT-sector
725                 $from = $this->getLong($start + $i);
726                 // Ef sector exists we need to move there
727             } while ($from != self::ENDOFCHAIN && ++$j < $this->cDIFAT);
728         }
729 
730         // Remove the unnecessary links.
731         while($this->DIFAT[count($this->DIFAT) - 1] == self::FREESECT)
732             array_pop($this->DIFAT);
733     }
734     // So, we done with reading DIFAT. Now chains of FAT sectors should be converted
735     // Lets go further.
736     private function readFATChains() {
737         // Sector size
738         $size = 1 << $this->sectorShift;
739         $this->fatChains = array();
740 
741         // Going through  DIFAT array.
742         for ($i = 0; $i < count($this->DIFAT); $i++) {
743             // Go to the sector that we were looking for (with the header)
744             $from = ($this->DIFAT[$i] + 1) << $this->sectorShift;
745             // Getting the FAT chain: array index is a current sector,
746             // value from an array s index of the next element or
747             // ENDOFCHAIN - if it is last element in the chain.
748             for ($j = 0; $j < $size; $j += 4)
749                 $this->fatChains[] = $this->getLong($from + $j);
750         }
751     }
752     // We done with reading of FAT sequences. Now heed to read MiniFAT-sequences exaactly the same
    way.
753     private function readMiniFATChains() {
754         // Sector size
755         $size = 1 << $this->sectorShift;
756         $this->miniFATChains = array();
757 
758         // Looking for the first sector with MiniFAT- sequences
759         $from = $this->fMiniFAT;
760         // If MiniFAT appears to be in file then
761         while ($from != self::ENDOFCHAIN) {
762             // Looking for the offset to the sector with MiniFat-sequence
763             $start = ($from + 1) << $this->sectorShift;
764             // Read the sequence from the current sector
765             for ($i = 0; $i < $size; $i += 4)
766                 $this->miniFATChains[] = $this->getLong($start + $i);
767             // If this is notthe last sector in the chain we need to move forward
768             $from = $this->fatChains[$from];
769         }
770     }
771 
772     // The most important functions that reads structure of "files" of such a type
773     // All the FS objects are written into this structure.
774     private function readDirectoryStructure() {
775         // get the 1st sector with "files" in file system
776         $from = $this->fDir;
777         // Get the sector size
778         $size = 1 << $this->sectorShift;
779         $this->fatEntries = array();
780         do {
781             // get sector in the file
782             $start = ($from + 1) << $this->sectorShift;
783             // Let go through the content of this sector. One sector contains up to 4  (or 128 for
    version 4)
784             // entries to FS. Lets read them.
785             for ($i = 0; $i < $size; $i += 128) {
786                 // Get the binary data
787                 $entry = substr($this->data, $start + $i, 128);
788                 // and prcess these data:
789                 $this->fatEntries[] = array(
790                     // get the entry name
791                     "name" => $this->utf16_to_ansi(substr($entry, 0, $this->getShort(0x40,
    $entry))),
792                     // and its type: either stream, or user data, or empty sector, etc.
793                     "type" => ord($entry[0x42]),
794                     // its color in the Red-Black tree
```

File: /home/gabri/moodle/textlib.php

```
                    "color" => ord($entry[0x43]),
                    // its "left" siblings
                    "left" => $this->getLong(0x44, $entry),
                    // its "right" siblings
                    "right" => $this->getLong(0x48, $entry),
                    // its child
                    "child" => $this->getLong(0x4C, $entry),
                    // offset to the content in FAT or miniFAT
                    "start" => $this->getLong(0x74, $entry),
                    // size of the content
                    "size" => $this->getSomeBytes($entry, 0x78, 8),
                );
            }

            // get the next sector with descriptions and jump there
            $from = $this->fatChains[$from];
            // Of course if such a sector exists
        } while ($from != self::ENDOFCHAIN);

        // remove "empty" entries  at the end if any.
        while($this->fatEntries[count($this->fatEntries) - 1]["type"] == 0)
            array_pop($this->fatEntries);
    }

    // Support function to get the adequate name of the current entrie in FS.
    // Note: names are written in the Unicode.
    private function utf16_to_ansi($in) {
        $out = "";
        for ($i = 0; $i < strlen($in); $i += 2)
            $out .= chr($this->getShort($i, $in));
        return trim($out);
    }

    protected function unicode_to_utf8($in, $check = false) {
        $out = "";
        if ($check && strpos($in, chr(0)) !== 1) {
            while (($i = strpos($in, chr(0x13))) !== false) {
                $j = strpos($in, chr(0x15), $i + 1);
                if ($j === false)
                    break;

                $in = substr_replace($in, "", $i, $j - $i);
            }
            for ($i = 0; $i < strlen($in); $i++) {
                if (ord($in[$i]) >= 32) {}
                elseif ($in[$i] == ' ' || $in[$i] == '\n') {}
                else
                    $in = substr_replace($in, "", $i, 1);
            }
            $in = str_replace(chr(0), "", $in);

            return $in;
        } elseif ($check) {
            while (($i = strpos($in, chr(0x13).chr(0))) !== false) {
                $j = strpos($in, chr(0x15).chr(0), $i + 1);
                if ($j === false)
                    break;

                $in = substr_replace($in, "", $i, $j - $i);
            }
            $in = str_replace(chr(0).chr(0), "", $in);
        }

        // Loop thriugh 2 byte words
        $skip = false;
        for ($i = 0; $i < strlen($in); $i += 2) {
            $cd = substr($in, $i, 2);
            if ($skip) {
                if (ord($cd[1]) == 0x15 || ord($cd[0]) == 0x15)
                    $skip = false;
                continue;
            }

            // If upper byte is  0 then this is ANSI
           if (ord($cd[1]) == 0) {
```

File: /home/gabri/moodle/textlib.php

```
870                 // If ASCII value is higher than 32 we will write it as it is.
871                 if (ord($cd[0]) >= 32)
872                     $out .= $cd[0];
873                 elseif ($cd[0] == ' ' || $cd[0] == '\n')
874                     $out .= $cd[0];
875                 elseif (ord($cd[0]) == 0x13)
876                     $skip = true;
877                 else {
878                     continue;
879                     // В противном случае проверяем символы на внедрённые команды (список можно
880                     // дополнить и пополнить).
881                     switch (ord($cd[0])) {
882                         case 0x0D: case 0x07: $out .= "\n"; break;
883                         case 0x08: case 0x01: $out .= ""; break;
884                         case 0x13: $out .= "HYPER13"; break;
885                         case 0x14: $out .= "HYPER14"; break;
886                         case 0x15: $out .= "HYPER15"; break;
887                         default: $out .= " "; break;
888                     }
889                 }
890             } else { // Иначе преобразовываем в HTML entity
891                 if (ord($cd[1]) == 0x13) {
892                     echo "@";
893                     $skip = true;
894                     continue;
895                 }
896                 $out .= "&#x".sprintf("%04x", $this->getShort(0, $cd)).";";
897             }
898         }
899 
900         // and return the results
901         return $out;
902     }
903 
904     // Support function to geto some bytes from the string
905     // taking into account order of bytes and converting values into a number.
906     protected function getSomeBytes($data, $from, $count) {
907         // Read data from  $data by default.
908         if ($data === null)
909             $data = $this->data;
910 
911         // Read a piece
912         $string = substr($data, $from, $count);
913         // in case of backward order reverse it
914         if ($this->isLittleEndian)
915             $string = strrev($string);
916 
917         // encode from binary to hex and to a number.
918         return hexdec(bin2hex($string));
919     }
920     // Read a word from the variable (by default from this->data)
921     protected function getShort($from, $data = null) {
922         return $this->getSomeBytes($data, $from, 2);
923     }
924     // read a double word from the variable (by default from this->data)
925     protected function getLong($from, $data = null) {
926         return $this->getSomeBytes($data, $from, 4);
927     }
928 }
929 
930 // Class to work with Microsoft Word Document (or just doc). It extends
931 // Windows Compound Binary File Format. Lets try to find text here
932 
933 class doc extends cfb {
934     // This function extends parse funciton and returns text from the file.
935     // If returns flase if something went wrong.
936     public function parse() {
937         parent::parse();
938 
939         // To read a DOC file we need 2 streams - WordDocument and 0Table or
940         // 1Table depending on the situation. Lets get hte first stream.
941         // It contains pieces of text we need to collect.
942         $wdStreamID = $this->getStreamIdByName("WordDocument");
943         if ($wdStreamID === false) { return false; }
944 
```

```
945         // We got the stream. Lets read it into a variable
946         $wdStream = $this->getStreamById($wdStreamID);
947 
948         // Next we need to get something from  FIB - special block named
949         // File Information Block that is located in the beginning of WordDocument stream.
950         $bytes = $this->getShort(0x000A, $wdStream);
951 
952         // Read which table we need to read: number 0 or number 1.
953         // To do so we need to read a small bit from the header.
954         $fWhichTblStm = ($bytes & 0x0200) == 0x0200;
955 
956         //Now we need to get the position of  CLX in the table stream. And the size of CLX itself.
957         $fcClx = $this->getLong(0x01A2, $wdStream);
958         $lcbClx = $this->getLong(0x01A6, $wdStream);
959 
960         // Conting few values to separate positions from the size in  clx
961         $ccpText = $this->getLong(0x004C, $wdStream);
962         $ccpFtn = $this->getLong(0x0050, $wdStream);
963         $ccpHdd = $this->getLong(0x0054, $wdStream);
964         $ccpMcr = $this->getLong(0x0058, $wdStream);
965         $ccpAtn = $this->getLong(0x005C, $wdStream);
966         $ccpEdn = $this->getLong(0x0060, $wdStream);
967         $ccpTxbx = $this->getLong(0x0064, $wdStream);
968         $ccpHdrTxbx = $this->getLong(0x0068, $wdStream);
969 
970         // Using the value that we just got we can look for the value of the last CP - character
    position
971         $lastCP = $ccpFtn + $ccpHdd + $ccpMcr + $ccpAtn + $ccpEdn + $ccpTxbx + $ccpHdrTxbx;
972         $lastCP += ($lastCP != 0) + $ccpText;
973 
974         // Get the required table in the file.
975         $tStreamID = $this->getStreamIdByName(intval($fWhichTblStm)."Table");
976         if ($tStreamID === false) { return false; }
977 
978         // And read the stream to a variable
979         $tStream = $this->getStreamById($tStreamID);
980         // Потом находим в потоке CLX
981         $clx = substr($tStream, $fcClx, $lcbClx);
982 
983         // Now we need to go through  CLX (yes... its complex) looking for piece with offsets and
    sizes of text pieces
984         $lcbPieceTable = 0;
985         $pieceTable = "";
986 
987         // Well... this is the most exciting part. There is not too much of documentation on the
    web site about
988         // what can be found before  pieceTable in the  CLX. So we will do the total search
    looking
989         // for the possible beginning of pieceTable (it must start with  0x02), and read the
    following 4 bytes
990         // - size of pieceTable. If the actual size equial to size writtent in the offset then
    Bingo! we found pieceTable.
991         // If not continue the search.
992 
993         $from = 0;
994         // Looking for  0x02 in CLX starting from the current offset
995         while (($i = strpos($clx, chr(0x02), $from)) !== false) {
996             // Get the pieceTable size
997             $lcbPieceTable = $this->getLong($i + 1, $clx);
998             // Get the  pieceTable
999             $pieceTable = substr($clx, $i + 5);
1000 
1001             // If the real size differs from required then this is not what we are lloking for.
1002             // Skip it.
1003             if (strlen($pieceTable) != $lcbPieceTable) {
1004                 $from = $i + 1;
1005                 continue;
1006             }
1007             // Oh.... we got it!!! its break time  my littel friends!
1008             break;
1009         }
1010 
1011         // Now we need to fill the array of  character positions, until we got the last  CP.
1012         $cp = array(); $i = 0;
1013         while (($cp[] = $this->getLong($i, $pieceTable)) != $lastCP)
1014             $i += 4;
```

```php
        // The rest will go as PCD (piece descriptors)
        $pcd = str_split(substr($pieceTable, $i + 4), 8);

        $text = "";
        // Yes! we came to our main goal - reading text from file.
        // Go through the descriptors of such pieces
        for ($i = 0; $i < count($pcd); $i++) {
            // Get the word with offset and  compression flag
            $fcValue = $this->getLong(2, $pcd[$i]);
            // Check what do we have: simple ANSI or Unicode
            $isANSI = ($fcValue & 0x40000000) == 0x40000000;
            // The rest without top will go as an offset
            $fc = $fcValue & 0x3FFFFFFF;

            // Get the piece of text
            $lcb = $cp[$i + 1] - $cp[$i];
            // if htis is Unicode, then lets read twice more bytes.
            if (!$isANSI)
                $lcb *= 2;
            // If ANSI - start twice earlier.
            else
                $fc /= 2;

            // Read a piece from Worddocument stream considering the offset
            $part = substr($wdStream, $fc, $lcb);
            // If this is a Unicode text then decode it to the regular text
            if (!$isANSI)
                $part = $this->unicode_to_utf8($part);

            // add a piece
            $text .= $part;
        }

        // Remove entries with embedded objects from the file
        $text = preg_replace("/HYPER13 *(INCLUDEPICTURE|HTMLCONTROL)(.*)HYPER15/iU", "", $text);
        $text = preg_replace("/HYPER13(.*)HYPER14(.*)HYPER15/iU", "$2", $text);
        // Return the results
        return $text;
    }
    // Function to convert from Unicode to UTF8
    protected function unicode_to_utf8($in) {
        $out = "";
        // Loop through 2-byte sequences
        for ($i = 0; $i < strlen($in); $i += 2) {
            $cd = substr($in, $i, 2);

            // If the first byte is 0 then this is  ANSI
            if (ord($cd[1]) == 0) {
                // If ASCII value of the low byte is higher than 32 then write it as it is.
                if (ord($cd[0]) >= 32)
                    $out .= $cd[0];

                // Otherwise check symbols against embedded commands. Please extend the list ;)
                switch (ord($cd[0])) {
                    case 0x0D: case 0x07: $out .= "\n"; break;
                    case 0x08: case 0x01: $out .= ""; break;
                    case 0x13: $out .= "HYPER13"; break;
                    case 0x14: $out .= "HYPER14"; break;
                    case 0x15: $out .= "HYPER15"; break;
                }
            } else // Otherwise convert to  HTML entity
                $out .= html_entity_decode("&#x".sprintf("%04x", $this->getShort(0, $cd)).";");
        }

        // And... return the result
        return $out;
    }
}

// Function to convert doc to plain-text. For those who "don't need classes".
function doc2text($filename) {
    $doc = new doc;
    $doc->read($filename);
    return $doc->parse();
}
```

```php


class ppt extends cfb {
    public function parse() {
        parent::parse();

        // File must have  Current User stream.
        $cuStreamID = $this->getStreamIdByName("Current User");
        if ($cuStreamID === false) { return false; }

        // Get this stream and check hash (do we really have PowerPoint-presentation?)
        // and read the offset to the first ocurence of UserEditAtom
        $cuStream = $this->getStreamById($cuStreamID);
        if ($this->getLong(12, $cuStream) == 0xF3D1C4DF) { return false; }
        $offsetToCurrentEdit = $this->getLong(16, $cuStream);

        // Getting stream named PowerPoint Document.
        $ppdStreamID = $this->getStreamIdByName("PowerPoint Document");
        if ($ppdStreamID === false) { return false; }
        $ppdStream = $this->getStreamById($ppdStreamID);

        // Look for all UserEditAtoms in PPT document. We need UserEditAtoms to get offsets to PersistDirectory.
        $offsetLastEdit = $offsetToCurrentEdit;
        $persistDirEntry = array();
        $live = null;
        $offsetPersistDirectory = array();
        do {
            $userEditAtom = $this->getRecord($ppdStream, $offsetLastEdit, 0x0FF5);
            $live = &$userEditAtom;
            array_unshift($offsetPersistDirectory, $this->getLong(12, $userEditAtom));
            $offsetLastEdit = $this->getLong(8, $userEditAtom);
        } while ($offsetLastEdit != 0x00000000);

        // Looping through all the offsets.
        for ($j = 0; $j < count($offsetPersistDirectory); $j++) {
            $rgPersistDirEntry = $this->getRecord($ppdStream, $offsetPersistDirectory[$j], 0x1772);
            if ($rgPersistDirEntry === false) { return false; }

            // Read 4-byte words: first 20 bit represent the initial ID of this entry in PersistDirectory,
            // next 12 bytes - number of subsequent offsets                                .
            for ($k = 0; $k < strlen($rgPersistDirEntry); ) {
                $persist = $this->getLong($k, $rgPersistDirEntry);
                $persistId = $persist & 0x000FFFFF;
                $cPersist = (($persist & 0xFFF00000) >> 20) & 0x00000FFF;
                $k += 4;

                // Based on the results we got we need to populate PersistDirectory array.
                for ($i = 0; $i < $cPersist; $i++) {
                    $offset = $this->getLong($k + $i * 4, $rgPersistDirEntry);
                    $persistDirEntry[$persistId + $i] = $this->getLong($k + $i * 4, $rgPersistDirEntry);
                }
                $k += $cPersist * 4;
            }
        }

        // In the last record we need to fined the ID of the entry with DocumentContainer.
        $docPersistIdRef = $this->getLong(16, $live);
        $documentContainer = $this->getRecord($ppdStream, $persistDirEntry[$docPersistIdRef], 0x03E8);

        // No we need to skip a lot of gqarbage to SlideList.
        $offset = 40 + 8;
        $exObjList = $this->getRecord($documentContainer, $offset, 0x0409);
        if ($exObjList) $offset += strlen($exObjList) + 8;
        $documentTextInfo = $this->getRecord($documentContainer, $offset, 0x03F2);
        $offset += strlen($documentTextInfo) + 8;
        $soundCollection = $this->getRecord($documentContainer, $offset, 0x07E4);
        if ($soundCollection) $offset += strlen($soundCollection) + 8;
        $drawingGroup = $this->getRecord($documentContainer, $offset, 0x040B);
        $offset += strlen($drawingGroup) + 8;
        $masterList = $this->getRecord($documentContainer, $offset, 0x0FF0);
        $offset += strlen($masterList) + 8;
        $docInfoList = $this->getRecord($documentContainer, $offset, 0x07D0);
```

```php
        if ($docInfoList) $offset += strlen($docInfoList) + 8;
        $slideHF = $this->getRecord($documentContainer, $offset, 0x0FD9);
        if ($slideHF) $offset += strlen($slideHF) + 8;
        $notesHF = $this->getRecord($documentContainer, $offset, 0x0FD9);
        if ($notesHF) $offset += strlen($notesHF) + 8;

        // Clean up the garbage.
        unset($exObjList, $documentTextInfo, $soundCollection, $drawingGroup, $masterList, $docInfoList, $slideHF, $notesHF);

        // Reading SlideList structure.
        $slideList = $this->getRecord($documentContainer, $offset, 0x0FF0);
        $out = "";
        for ($i = 0; $i < strlen($slideList); ) {
            // Read the current block and use its type to decide how to process it.
            $block = $this->getRecord($slideList, $i);
            switch($this->getRecordType($slideList, $i)) {
                case 0x03F3: # RT_SlidePersistAtom
                    // The worst case: we have pointer to a slide. If this is the case
                    // we have to get this slide from PersistDirectory.
                    $pid = $this->getLong(0, $block);
                    $slide = $this->getRecord($ppdStream, @$persistDirEntry[$pid], 0x03EE);

                    // Again skip lots of different stuff looking for Drawing structure.
                    $offset = 32;
                    $slideShowSlideInfoAtom = $this->getRecord($slide, $offset, 0x03F9);
                    if ($slideShowSlideInfoAtom) $offset += strlen($slideShowSlideInfoAtom) + 8;
                    $perSlideHFContainer = $this->getRecord($slide, $offset, 0x0FD9);
                    if ($perSlideHFContainer) $offset += strlen($perSlideHFContainer) + 8;
                    $rtSlideSyncInfo12 = $this->getRecord($slide, $offset, 0x3714);
                    if ($rtSlideSyncInfo12) $offset += strlen($rtSlideSyncInfo12) + 8;

                    // Drawing is MS Drawing object that has header structure similar to PPT .
                    // To avoid possible parsing of complicated nested structures we can search the text directly.
                    $drawing = $this->getRecord($slide, $offset, 0x040C);
                    $from = 0;
                    while(preg_match("#(\xA8|\xA0)\x0F#", $drawing, $pocket, PREG_OFFSET_CAPTURE, $from)) {
                        $pocket = @$pocket[1];
                        // We must check that block header starts with 00, otherwise it may happen that we found
                        //  something in the middle of other data
                        if (substr($drawing, $pocket[1] - 2, 2) == "\x00\x00") {
                            // Read either plain or Unicode text.
                            if (ord($pocket[0]) == 0xA8)
                                $out .= htmlspecialchars($this->getRecord($drawing, $pocket[1] - 2, 0x0FA8))." ";
                            else
                                $out .= $this->unicode_to_utf8($this->getRecord($drawing, $pocket[1] - 2, 0x0FA0))." ";
                        }
                        // Read the next entry
                        $from = $pocket[1] + 2;
                    }
                break;
                case 0x0FA0: # RT_TextCharsAtom
                // Simple option: we've found Unicode-text
                    $out .= $this->unicode_to_utf8($block)." ";
                break;
                case 0x0FA8: # RT_TextBytesAtom
                // Or regular plain text.
                    $out .= htmlspecialchars($block)." ";
                break;
                # skip other "options"
            }

            // Move by the length of the block with header.
            $i += strlen($block) + 8;
        }

        // Return UTF-8 text.
        return html_entity_decode(iconv("windows-1251", "utf-8", $out), ENT_QUOTES, "UTF-8");
    }

    // Additional funciton that defines the lingth of the current internal structure.
```

File: /home/gabri/moodle/textlib.php

```
1232        // It gets the input stream, offset and type of the structure to read.
1233        // Type will be used ot check the actual structure we read.
1234        private function getRecordLength($stream, $offset, $recType = null) {
1235            $rh = substr($stream, $offset, 8);
1236            if (!is_null($recType) && $recType != $this->getShort(2, $rh))
1237                return false;
1238            return $this->getLong(4, $rh);
1239        }
1240        // Get the type of the current structure according to MS manual.
1241        private function getRecordType($stream, $offset) {
1242            $rh = substr($stream, $offset, 8);
1243            return $this->getShort(2, $rh);
1244        }
1245        // Get the record by its offset. Attention, the header doesn't go back
1246
1247        private function getRecord($stream, $offset, $recType = null) {
1248            $length = $this->getRecordLength($stream, $offset, $recType);
1249            if ($length === false)
1250                return false;
1251            return substr($stream, $offset + 8, $length);
1252        }
1253    }
1254
1255    // For those ones who do not need classes :)
1256    function ppt2text($filename) {
1257        $ppt = new ppt;
1258        $ppt->read($filename);
1259        return $ppt->parse();
1260    }
1261
1262    ?>
```

File: /home/gabri/moodle/styles.css

```
1   .pager {
2       text-align: right
3   }
```

File: /home/gabri/moodle/settings.php

```php
<?php
require_once(dirname(dirname(dirname(__FILE__))) . '/config.php');
require_once($CFG->libdir.'/adminlib.php');
require_once($CFG->dirroot.'/plagiarism/moss/locallib.php');
require_once($CFG->dirroot.'/lib/formslib.php');

class moss_global_settings_form extends moodleform {

    function definition() {
        global $CFG;

        $mform =& $this->_form;

        $helplink = get_string('mossexplain', 'plagiarism_moss');
        $mform->addElement('html', $helplink);

        $mform->addElement('checkbox', 'mossenabled', get_string('mossenabled', 'plagiarism_moss'));
        $mform->setDefault('mossenabled', false);

        $mform->addElement('text', 'mossuserid', get_string('mossuserid', 'plagiarism_moss'));
        $mform->addHelpButton('mossuserid', 'mossuserid', 'plagiarism_moss');
        $mform->setDefault('mossuserid', '');
        $mform->disabledIf('mossuserid', 'mossenabled');

        $choices = moss_get_supported_languages();
        $mform->addElement('select', 'defaultlanguage', get_string('defaultlanguage',
'plagiarism_moss'), $choices);
        $mform->setDefault('defaultlanguage', 'ascii');
        $mform->disabledIf('defaultlanguage', 'mossenabled');

        $mform->addElement('checkbox', 'showidnumber', get_string('showidnumber',
'plagiarism_moss'));
        $mform->setDefault('showidnumber', false);
        $mform->disabledIf('showidnumber', 'mossenabled');

        $mform->addElement('text', 'maxfilesize', get_string('maxfilesize', 'plagiarism_moss'));
        $mform->addHelpButton('maxfilesize', 'maxfilesize', 'plagiarism_moss');
        $mform->setDefault('maxfilesize', MOSS_DEFAULT_MAXFILESIZE);
        $mform->setType('maxfilesize', PARAM_INT);
        $mform->disabledIf('maxfilesize', 'mossenabled');

        $mform->addElement('text', 'antiwordpath', get_string('antiwordpath', 'plagiarism_moss'));
        $mform->addHelpButton('antiwordpath', 'antiwordpath', 'plagiarism_moss');
        $mform->setDefault('antiwordpath', '');
        $mform->setType('antiwordpath', PARAM_RAW);
        $mform->disabledIf('antiwordpath', 'mossenabled');

        if ($CFG->ostype == 'WINDOWS') {
            $mform->addElement('text', 'cygwinpath', get_string('cygwinpath', 'plagiarism_moss'));
            $mform->setDefault('cygwinpath', 'C:\cygwin');
            $mform->setType('cygwinpath', PARAM_RAW);
            $mform->disabledIf('cygwinpath', 'mossenabled');
        }

        $this->add_action_buttons(false);
    }

    function validation($data, $files) {
        global $CFG;
        $errors = parent::validation($data, $files);

        if (!empty($data['mossenabled'])) {
            if (!is_numeric($data['mossuserid'])) {
                $errors['mossuserid'] = get_string('err_numeric', 'form');
            }
            if ($CFG->ostype == 'WINDOWS') {
                if (!is_executable($data['cygwinpath'].'\\bin\\perl.exe')) {
                    $errors['cygwinpath'] = get_string('err_cygwinpath', 'plagiarism_moss');
                }
            }
        }

        return $errors;
    }
}
```

File: /home/gabri/moodle/settings.php

```php

require_login();
admin_externalpage_setup('plagiarismmoss');
$context = get_context_instance(CONTEXT_SYSTEM);
require_capability('moodle/site:config', $context, $USER->id);

$mform = new moss_global_settings_form();

echo $OUTPUT->header();

if (($data = $mform->get_data()) && confirm_sesskey()) {

    if (!empty($data->mossenabled)) {
        set_config('moss_use', $data->mossenabled, 'plagiarism');
        set_config('mossuserid', $data->mossuserid, 'plagiarism_moss');
        set_config('defaultlanguage', $data->defaultlanguage, 'plagiarism_moss');
        set_config('showidnumber', $data->showidnumber, 'plagiarism_moss');
        set_config('maxfilesize', $data->maxfilesize, 'plagiarism_moss');
        set_config('antiwordpath', $data->antiwordpath, 'plagiarism_moss');
        if ($CFG->ostype == 'WINDOWS') {
            set_config('cygwinpath', $data->cygwinpath, 'plagiarism_moss');
        }
    } else {
        set_config('moss_use', 0, 'plagiarism');
    }

    echo $OUTPUT->notification(get_string('savedconfigsuccess', 'plagiarism_moss'),
'notifysuccess');
}

$settings = array();
if ($moss_use = get_config('plagiarism', 'moss_use')) {
    $settings['mossenabled'] = $moss_use;
}
$moss_configs = (array)get_config('plagiarism_moss');
$settings = array_merge($settings, $moss_configs);
$mform->set_data($settings);

echo $OUTPUT->box_start();
$mform->display();
echo $OUTPUT->box_end();
echo $OUTPUT->footer();
```

File: /home/gabri/moodle/renderer.php

```php
<?php
defined('MOODLE_INTERNAL') || die();
require_once($CFG->libdir.'/gradelib.php');

/**
 * moss result page renderer class
 */
class plagiarism_moss_renderer extends plugin_renderer_base {
    protected $can_viewdiff;
    protected $can_confirm;
    protected $can_viewunconfirmed;
    protected $context;
    protected $confirm_htmls;
    protected $showidnumber;
    var $moss = null;
    var $cm = null;

    public function __construct(moodle_page $page, $target) {
        parent::__construct($page, $target);

        $this->context = $page->context;
        $this->can_viewdiff = has_capability('plagiarism/moss:viewdiff', $this->context);
        $this->can_viewunconfirmed = has_capability('plagiarism/moss:viewunconfirmed', $this->context);
        $this->can_confirm = has_capability('plagiarism/moss:confirm', $this->context);
        $this->can_grade  = has_capability('mod/assignment:grade', $this->context);
        $this->showidnumber = get_config('plagiarism_moss', 'showidnumber');

        $this->confirm_htmls = array (
            $this->pix_icon('i/completion-manual-n', get_string('unconfirmed', 'plagiarism_moss')),
            $this->pix_icon('i/completion-manual-y', get_string('confirmed', 'plagiarism_moss'))
        );
    }

    function user_stats($user) {
        global $DB;

        $sql = 'SELECT COUNT(DISTINCT moss)
                FROM {plagiarism_moss_results}
                WHERE userid = ? AND confirmed = 1';
        $a->total = $DB->count_records_sql($sql, array($user->id));
        $a->fullname = fullname($user);

        return $this->container(get_string('confirmedresults', 'plagiarism_moss', $a));
    }

    function cm_stats() {
        global $DB;

        $sql = 'SELECT COUNT(DISTINCT userid)
                FROM {plagiarism_moss_results}
                WHERE moss = ? AND confirmed = 1';
        $total = $DB->count_records_sql($sql, array($this->moss->id));

        return $this->container(get_string('activityconfirmedresults', 'plagiarism_moss', $total));
    }

    function user_result($user) {
        global $DB;

        $output = $this->user_stats($user);
        $user_timesubmitted = moss_get_submit_time($this->moss->cmid, $user->id);

        $table = new html_table();

        $table->head = array(
            get_string('user'),
            get_string('filepatterns', 'plagiarism_moss'),
            get_string('confirm', 'plagiarism_moss').$this->help_icon('confirm', 'plagiarism_moss'),
            get_string('deltatime', 'plagiarism_moss'),
            get_string('percentage', 'plagiarism_moss'),
            get_string('matchedlines', 'plagiarism_moss'),
            get_string('matchedusers', 'plagiarism_moss')
        );
        if ($this->showidnumber) {
```

File: /home/gabri/moodle/renderer.php

```
75              $table->head[] = get_string('idnumber');
76          }
77          $table->head[] = get_string('confirm', 'plagiarism_moss').$this->help_icon('confirm',
    'plagiarism_moss');
78
79          $configs = $DB->get_records('plagiarism_moss_configs', array('moss' => $this->moss->id));
80          foreach ($configs as $config) {
81              if (empty($config->filepatterns)) {
82                  continue;
83              }
84
85              $sql = 'SELECT r1.*, r2.userid AS other
86                  FROM {plagiarism_moss_results} r1
87                  LEFT JOIN {plagiarism_moss_results} r2 ON r1.pair = r2.id
88                  WHERE r1.userid = ? AND r1.moss = ? AND r1.config = ? ';
89              if (!$this->can_viewunconfirmed) {
90                  $sql .= 'AND r1.confirmed = 1 ';
91              }
92              $sql .= 'ORDER BY rank ASC';
93              $params = array($user->id, $this->moss->id, $config->id);
94              $matches = $DB->get_records_sql($sql, $params);
95
96              $first_match = true;
97              foreach ($matches as $match) {
98                  $cells = array();
99
100                 // time submitted
101                 $delta = $user_timesubmitted - moss_get_submit_time($this->moss->cmid, $match-
    >other);
102                 if ($delta > 0) {
103                     $delta_text = get_string('late', 'assignment', format_time($delta));
104                 } else if ($delta <0) {
105                     $delta_text = get_string('early', 'assignment', format_time($delta));
106                 } else {
107                     $delta_text = get_string('numseconds', '', 0);
108                 }
109                 $cells[] = new html_table_cell($delta_text);
110
111                 // percentage and linesmatched
112                 $percentage = $match->percentage.'%';
113                 $linesmatched = $match->linesmatched;
114                 if ($this->can_viewdiff) {
115                     $url = new moodle_url($match->link);
116                     $attributes = array('target' => '_blank');
117                     $percentage = html_writer::link($url, $percentage, $attributes);
118                     $linesmatched = html_writer::link($url, $linesmatched, $attributes);
119                 }
120                 $cells[] = new html_table_cell($percentage);
121                 $cells[] = new html_table_cell($linesmatched);
122
123                 // other user
124                 $other = $DB->get_record('user', array('id' => $match->other));
125                 $cells[] = new html_table_cell($this->user($other));
126                 if ($this->showidnumber) {
127                     $cells[] = new html_table_cell($other->idnumber);
128                 }
129                 $cells[] = new html_table_cell($this->confirm_button($match->pair));
130
131                 if ($first_match) { // first row of the filepatterns
132                     // confirm button
133                     $confirmcell = new html_table_cell($this->confirm_button($match));
134                     $confirmcell->rowspan = count($matches);
135
136                     $pattern_cell = new html_table_cell($config->filepatterns);
137                     $pattern_cell->rowspan = count($matches);
138
139                     $cells = array_merge(array($pattern_cell, $confirmcell), $cells);
140
141                     $first_match = false;
142                 }
143
144                 if (empty($table->data)) { //first row
145                     $user_text = $this-
    >user_picture($user).html_writer::empty_tag('br').fullname($user).html_writer::empty_tag('br').
    $user->idnumber;
146                     $cell = new html_table_cell($user_text);
```

File: /home/gabri/moodle/renderer.php

```
147                     $rowcount = &$cell->rowspan; // assign it later
148                     $rowcount = 0;
149                     $cells = array_merge(array($cell), $cells);
150                 }
151 
152                 $table->data[] = new html_table_row($cells);
153                 $rowcount++;
154             }
155         }
156 
157         if (empty($table->data)) {
158             $output .= $this->notification(get_string('nouserresults', 'plagiarism_moss',
    fullname($user)));
159         } else {
160             $output .= html_writer::table($table);
161         }
162 
163         return $output;
164     }
165 
166     /**
167      * List all results in a course module
168      */
169     function cm_result($tabid=0, $from=0, $num=30) {
170         global $DB, $CFG, $PAGE;
171 
172         $output = $this->cm_stats();
173 
174         /// find out current groups mode
175         $groupmode = groups_get_activity_groupmode($this->cm);
176         $currentgroup = groups_get_activity_group($this->cm, true);
177         $output .= groups_print_activity_menu($this->cm, clone($PAGE->url), true);
178 
179         /// Tabs
180         $select = 'moss = ? AND filepatterns != \'\'';
181         $configs = $DB->get_records_select('plagiarism_moss_configs', $select, array($this->moss-
    >id));
182         $current_config = $tabid == -1 ? $tabid : reset($configs)->id;  // -1 means confirmed only
183         $row = array();
184         foreach ($configs as $conf) {
185             $link = new moodle_url('/plagiarism/moss/view.php', array('id' => $this->cm->id, 'tab'
    => $conf->id));
186             $row[] = new tabobject($conf->id, $link, $conf->filepatterns, get_string('filepatterns',
    'plagiarism_moss').' '.$conf->filepatterns);
187             if ($tabid == $conf->id) {
188                 $current_config = $conf->id;
189             }
190         }
191         $link = new moodle_url('/plagiarism/moss/view.php', array('id' => $this->cm->id, 'tab' =>
    -1));
192         $row[] = new tabobject(-1, $link, get_string('confirmed', 'plagiarism_moss'));
193         $output .= print_tabs(array($row), $current_config, NULL, NULL, true);
194 
195         /// Table header
196         $head = array();
197         $head[] = get_string('user').'1';
198         if ($this->showidnumber) {
199             $head[] = get_string('idnumber').'1';
200         }
201         $head[] = get_string('confirm', 'plagiarism_moss').$this->help_icon('confirm',
    'plagiarism_moss');
202         $head[] = get_string('percentage', 'plagiarism_moss');
203         $head[] = get_string('matchedlines', 'plagiarism_moss');
204         $head[] = get_string('user').'2';
205         if ($this->showidnumber) {
206             $head[] = get_string('idnumber').'2';
207         }
208         $head[] = get_string('confirm', 'plagiarism_moss').$this->help_icon('confirm',
    'plagiarism_moss');
209         $head[] = get_string('percentage', 'plagiarism_moss');
210         $head[] = get_string('matchedlines', 'plagiarism_moss');
211         $head[] = get_string('deltatime', 'plagiarism_moss').$this->help_icon('deltatime',
    'plagiarism_moss');
212         $table = new html_table();
213         $table->head = $head;
214 
215         $select = 'SELECT r1.*,
```

```
216                         r2.id AS id2, r2.userid AS userid2, r2.percentage AS percentage2,
    r2.linesmatched AS linesmatched2,
217                         r2.confirmed AS confirmed2, r2.confirmer AS confirmer2, r2.timeconfirmed
    AS timeconfirmed2
218                   FROM {plagiarism_moss_results} r1 LEFT JOIN {plagiarism_moss_results} r2 ON
    r1.pair = r2.id ';
219         if ($tabid == -1) {
220             $where = 'WHERE r1.moss = ? AND r1.userid < r2.userid AND ( r1.confirmed = 1 or
    r2.confirmed = 1 )';
221         } else {
222             $where = 'WHERE r1.moss = ? AND r1.userid < r2.userid AND r1.config = ?';
223         }
224         $orderby = 'ORDER BY r1.rank ASC';
225         if ($currentgroup) {
226             if ($users = groups_get_members($currentgroup, 'u.id', 'u.id')) {
227                 $users = array_keys($users);
228                 $userids = implode(',',$users);
229             } else {
230                 $userids = '0';
231             }
232             $where .= "AND (r1.userid IN ($userids) OR r2.userid IN ($userids)) ";
233         }
234
235         $results = $DB->get_records_sql($select.$where.$orderby, array($this->moss->id,
    $current_config), $from, $num);
236
237         foreach ($results as $result) {
238             $user2result = clone($result);
239             $user2result->id = $result->id2;
240             $user2result->userid = $result->userid2;
241             $user2result->percentage = $result->percentage2;
242             $user2result->linesmatched = $result->linesmatched2;
243             $user2result->confirmed = $result->confirmed2;
244             $user2result->confirmer = $result->confirmer2;
245             $user2result->timeconfirmed = $result->timeconfirmed2;
246
247             // The persons who submitted later are displayed in the left column
248             // since they are more likely to be copiers.
249             $delta = moss_get_submit_time($this->moss->cmid, $result->userid) -
    moss_get_submit_time($this->moss->cmid, $user2result->userid);
250             if ($delta >= 0) {
251                 $user1cells = $this->fill_result_in_cells($result);
252                 $user2cells = $this->fill_result_in_cells($user2result);
253             } else {
254                 $user2cells = $this->fill_result_in_cells($result);
255                 $user1cells = $this->fill_result_in_cells($user2result);
256             }
257
258             $cells = array_merge($user1cells, $user2cells);
259             $cells[] = $delta != 0 ? format_time($delta) : get_string('numseconds', '', 0);
260             $table->data[] = new html_table_row($cells);
261         }
262
263         if (empty($table->data)) {
264             $output .= $this->notification(get_string('nocmresults', 'plagiarism_moss'));
265         } else {
266             // append pager line
267             $prevlink = '';
268             $nextlink = '';
269             if ($from != 0) {
270                 $url = clone($PAGE->url);
271                 $newfrom = $from-$num >= 0 ? $from-$num : 0;
272                 $url->param('from', $newfrom);
273                 $prevlink = html_writer::link($url, get_string('previous'));
274             }
275             if ($num <= count($results)) {
276                 $url = clone($PAGE->url);
277                 $newfrom = $from+$num;
278                 $url->param('from', $newfrom);
279                 $nextlink = html_writer::link($url, get_string('next'));
280             }
281             $pager = new html_table_cell($prevlink.' '.$nextlink);
282             $pager->attributes['class'] = 'pager';
283             $pager->colspan = count($table->head);
284             $table->data[] = new html_table_row(array($pager));
285             $output .= html_writer::table($table);
```

File: /home/gabri/moodle/renderer.php

```
286         }
287
288         return $output;
289     }
290
291     protected function user($user) {
292         global $COURSE, $USER;
293
294         if (is_enrolled($this->context, $user)) {
295             if ($user->id == $USER->id or has_capability('plagiarism/moss:viewallresults', $this->context)) {
296                 $url = new moodle_url('/plagiarism/moss/view.php', array('id' => $this->cm->id, 'user' => $user->id));
297             } else {
298                 $url = new moodle_url('/user/view.php', array('id' => $user->id, 'course' => $COURSE->id));
299             }
300         } else {
301             $url = new moodle_url('/user/view.php', array('id' => $user->id));
302         }
303         return html_writer::link($url, fullname($user));
304     }
305
306     /**
307      * return html of confirm/unconfirm button and grade button
308      */
309     function confirm_button($result, $ajax = false) {
310         global $DB, $PAGE;
311
312         if (!is_object($result)) { // $result is id
313             $result = $DB->get_record('plagiarism_moss_results', array('id' => $result), '*', MUST_EXIST);
314         }
315
316         if (!is_enrolled($this->context, $result->userid)) { // show nothing for unenrolled users
317             return '';
318         }
319
320         // Confirm/unconfirm button
321         $output = $this->confirm_htmls[$result->confirmed];
322         if ($this->can_confirm) {
323             $url = $PAGE->url;
324             $url->param('sesskey', sesskey());
325             $url->param('result', $result->id);
326             $url->param('confirm', !$result->confirmed);
327             $output = html_writer::link($url, $output, array('class' => 'confirmbutton'));
328         }
329
330         // Grade button
331         if ($this->can_grade) {
332             $url = new moodle_url('/mod/assignment/submissions.php',
333                 array(
334                     'id' => $this->cm->id,
335                     'userid' => $result->userid,
336                     'mode' => 'single',
337                     'filter' => 0,
338                     'offset' => 99999999  // HACK: Use a big offset to hide the next buttons
339                 )
340             );
341
342             // get grade
343             $grading_info = grade_get_grades($this->cm->course, 'mod', 'assignment', $this->cm->instance, $result->userid);
344             if (!empty($grading_info->items[0]->grades)) {
345                 $grade = reset($grading_info->items[0]->grades)->grade;
346                 if (empty($grade)) {
347                     $grade = '-';
348                 } else {
349                     $grade = round($grade);
350                 }
351             } else {
352                 $grade = '-';
353             }
354
355             $output .= html_writer::link($url, ' ('.$grade.')', array('target' => '_blank', 'title' => get_string('grade')));
```

File: /home/gabri/moodle/renderer.php

```
356             }
357
358             if ($ajax) {
359                 return $output;
360             } else {
361                 return html_writer::tag('span', $output, array('id' => "user$result->userid-
    config$result->config"));
362             }
363         }
364
365         public function get_confirm_htmls() {
366             return $this->confirm_htmls;
367         }
368
369         protected function fill_result_in_cells($result) {
370             global $DB;
371             $cells = array();
372
373             // user name
374             $user = $DB->get_record('user', array('id' => $result->userid), 'id, firstname, lastname,
    idnumber');
375             $cells[] = new html_table_cell($this->user($user));
376             if ($this->showidnumber) {
377                 $cells[] = new html_table_cell($user->idnumber);
378             }
379             $cells[] = new html_table_cell($this->confirm_button($result));
380
381             // percentage and linesmatched
382             $percentage = $result->percentage.'%';
383             $linesmatched = $result->linesmatched;
384             if ($this->can_viewdiff) {
385                 $url = new moodle_url($result->link);
386                 $attributes = array('target' => '_blank');
387                 $percentage = html_writer::link($url, $percentage, $attributes);
388                 $linesmatched = html_writer::link($url, $linesmatched, $attributes);
389             }
390             $cells[] = new html_table_cell($percentage);
391             $cells[] = new html_table_cell($linesmatched);
392
393             return $cells;
394         }
395     }
```

File: /home/gabri/moodle/moss.php

```php
<?php
defined('MOODLE_INTERNAL') || die('Direct access to this script is forbidden.');

require_once($CFG->dirroot.'/plagiarism/moss/locallib.php');
require_once($CFG->dirroot.'/plagiarism/moss/textlib.php');

/**
 * moss script interface
 */
class moss {
    protected $moss;
    protected $tempdir;

    public function __construct($cmid) {
        global $CFG, $DB, $UNITTEST;

        $this->moss = $DB->get_record('plagiarism_moss', array('cmid' => $cmid));
        if (!isset($UNITTEST)) { // testcase can not construct course structure
            $this->moss->course = $DB->get_field('course_modules', 'course', array('id' => $this->moss->cmid));
        }

        $this->tempdir = $CFG->dataroot.'/temp/moss/'.$this->moss->id;
        remove_dir($this->tempdir); // Perhaps it is not cleaned in previous run
        if (!check_dir_exists($this->tempdir)) {
            throw new moodle_exception('errorcreatingdirectory', '', '', $this->tempdir);
        }

        if ($CFG->ostype == 'WINDOWS') {
            // the tempdir will be passed to cygwin which require '/' path spliter
            $this->tempdir = str_replace('\\', '/', $this->tempdir);
        }
    }

    public function __destruct() {
        if (!debugging('', DEBUG_DEVELOPER)) {
            remove_dir($this->tempdir);
        }
    }

    protected function get_config($name) {
        $value = get_config('plagiarism_moss', $name);
        if ($value === false) {
            mtrace("\t!!! WARNING !!! - global settings of moss may not proper.");
        }
        return $value;
    }

    /**
     * Measure the current course module
     *
     * @return bool success or not
     */
    public function measure() {
        global $DB;

        if (!moss_enabled($this->moss->cmid)) {
            return false;
        }

        $mosses = $DB->get_records_select('plagiarism_moss', 'tag = ? AND tag != 0', array($this->moss->tag));
        foreach ($mosses as $moss) {
            if ($moss->cmid == $this->moss->cmid) {
                // current moss must be extracted lastly
                // to overwrite other files belong to the same person
                continue;
            }
            $this->extract_files($moss);
        }

        $this->extract_files();

        $result = $this->call_moss();

```

File: /home/gabri/moodle/moss.php

```
74          $this->moss->timemeasured = time();
75          $DB->update_record('plagiarism_moss', $this->moss);
76
77          return $result;
78      }
79
80      protected function extract_files($moss = null) {
81          if ($moss == null) {
82              $moss = $this->moss;
83          }
84
85          $sizelimit = $this->get_config('maxfilesize');
86
87          $fs = get_file_storage();
88          $files = $fs->get_area_files(get_system_context()->id, 'plagiarism_moss', 'files', $moss-
    >cmid, 'sortorder', false);
89          foreach ($files as $file) {
90              if ($file->get_filesize() > $sizelimit) {
91                  continue;
92              }
93
94              $content = $this->get_clear_utf8_content($file);
95              if (empty($content)) {
96                  continue;
97              }
98
99              $path = $this->tempdir.$file->get_filepath();
100             $fullpath = $path.$file->get_filename();
101             if (!check_dir_exists($path)) {
102                 throw new moodle_exception('errorcreatingdirectory', '', '', $path);
103             }
104             file_put_contents($fullpath, $content);
105         }
106     }
107
108     /**
109      * Convert binary files to text and ensure the charset is UTF8
110      *
111      * @param object $file moodle storedfile
112      * @return content or false
113      */
114     protected function get_clear_utf8_content($file) {
115         $localewincharset = get_string('localewincharset', 'langconfig');
116
117         $filen = $file->get_filename();
118         $file_type = strtolower(substr($filen, strlen($filen)-4, 4));
119
120         if (array_search($file_type, array('.pdf', '.rtf', '.odt', '.doc', 'docx'))) {
121             $temp_file = $this->tempdir."/$filen.tmp";
122             $file->copy_content_to($temp_file);
123             switch ($file_type) {
124             case '.pdf':
125                 $content = pdf2text($temp_file);
126                 break;
127             case '.rtf':
128                 $content = textlib_get_instance()->entities_to_utf8(rtf2text($temp_file));
129                 break;
130             case '.odt':
131                 $content =  getTextFromZippedXML($temp_file,'content.xml');
132                 break;
133             case '.doc':
134                 $antiwordpath = $this->get_config('antiwordpath');
135                 $magic = file_get_contents($temp_file, NULL, NULL, -1, 2);
136                 if ($magic === 'PK') {
137                     // It is really a docx
138                     $content = getTextFromZippedXML($temp_file,'word/document.xml');
139                 } else if (empty($antiwordpath) || !is_executable($antiwordpath)) {
140                     $content = textlib_get_instance()->entities_to_utf8(doc2text($temp_file));
141                 } else {
142                     $content = shell_exec($antiwordpath.' -f -w 0 '.escapeshellarg($temp_file));
143                     if (empty($content)) { // antiword can not recognize this file
144                         $content = textlib_get_instance()->entities_to_utf8(doc2text($temp_file));
145                     }
146                 }
147                 break;
```

File: /home/gabri/moodle/moss.php

```
148             case 'docx':
149                 $content = getTextFromZippedXML($temp_file,'word/document.xml');
150                 break;
151             }
152             unlink($temp_file);
153             return $this->wordwrap($content, 80);
154         }
155 
156         // Files no need to covert format go here
157         $content = $file->get_content();
158 
159         if (!mb_check_encoding($content, 'UTF-8')) {
160             if (mb_check_encoding($content, $localewincharset)) {
161                 // Convert content charset to UTF-8
162                 $content = textlib_get_instance()->convert($content, $localewincharset);
163             } else {
164                 // Unknown charset, possible binary file. Skip it
165                 mtrace("\tSkip unknown charset/binary file ".$file->get_filepath().$file-
    >get_filename());
166                 return false;
167             }
168         }
169 
170         return $content;
171     }
172 
173     /**
174      * Auto wordwarp text
175      */
176     protected function wordwrap($text, $width) {
177         $i = 0;
178         $return = '';
179         $linelen = 0;
180         $prev_ch = '';
181         while (($ch = mb_substr($text, $i, 1, 'UTF-8')) !== '') {
182             if ($linelen >= $width and (!ctype_alnum($prev_ch) or ctype_space($ch))) { // Do not
    break latin words
183                 $return .= "\n";
184                 $linelen = 0;
185             }
186             if ($linelen != 0 or !ctype_space($ch)) {   // trim heading whitespaces
187                 $return .= $ch;
188                 $linelen += mb_strwidth($ch, 'UTF-8');   // Multy-byte chars may twice the width
189             }
190             if ($ch === "\n") {
191                 $linelen = 0;
192             }
193             $i++;
194             $prev_ch = $ch;
195         }
196         return $return;
197     }
198 
199         /**
200          * this function will call moss script and save anti-plagiarism results
201      *
202      * @return sucessful true or failed false
203          */
204     protected function call_moss() {
205         global $CFG, $DB;
206 
207         $commands = $this->get_commands();
208         if(empty($commands)) {
209             mtrace("\tNo valid config to run");
210             return false;
211         }
212 
213         //delete previous results
214         $this->clean_results();
215 
216         //connect moss server and save results
217         foreach($commands as $configid => $cmd) {
218             $descriptorspec = array(
219                 0 => array('pipe', 'r'),  // stdin
220                 1 => array('pipe', 'w'),  // stdout
```

```
221                 2 => array('pipe', 'w')   // stderr
222             );
223             $proc = proc_open($cmd, $descriptorspec, $pipes);
224             if (!is_resource($proc)) {
225                 mtrace("\tCall moss failed.");
226                 return false;
227             }
228 
229             //get standard output and standard error output
230             $out = stream_get_contents($pipes[1]);
231             $err = stream_get_contents($pipes[2]);
232             $rval = proc_close($proc);
233             if ($rval != 0) {
234                 mtrace($out);
235                 mtrace($err);
236                 return false;
237             }
238 
239             mtrace(substr_count($out, 'done').' files are uploaded.');
240 
241             $url_p = '/http:\/\/moss\.stanford\.edu\/results\/\d+/';
242             if (!preg_match($url_p, $out, $match)) {
243                 mtrace($out);
244                 mtrace($err);
245                 return false;
246             }
247 
248             $url = $match[0].'/'; // Some curl do not process 301. Add a tailing / to prevent 301
    from moss
249 
250             if (!$this->save_results($url, $configid)) {
251                 return false;
252             }
253         }
254         return true;
255     }
256 
257     /**
258      * Return commands of moss for all configs
259      *
260      * @return array
261      */
262     protected function get_commands() {
263         global $CFG, $DB;
264 
265         $settings = $DB->get_records('plagiarism_moss_configs', array('moss' => $this->moss->id));
266         $fs = get_file_storage();
267         $context = get_system_context();
268         $cmds = array();
269 
270         foreach($settings as $setting) {
271             if (empty($setting->filepatterns)) { // no filepatterns means invalid
272                 continue;
273             }
274 
275             if ($CFG->ostype == 'WINDOWS') {
276                 $cygwin = $this->get_config('cygwinpath');
277                 $perl = $cygwin.'\\bin\\perl.exe';
278                 $mossscrpit = $CFG->dirroot.'\\plagiarism\\moss\\moss';
279                 $cmd = str_replace(' ', '\\ ', $CFG->dirroot.'\\plagiarism\\moss\\moss.bat');
280                 $cmd .= ' "'.$perl.'" "'.$mossscrpit.'"';
281             } else {
282                 $cmd = '"'.$CFG->dirroot.'/plagiarism/moss/moss'.'"';
283             }
284             $cmd .= ' -d';
285             $cmd .= ' -u '.escapeshellarg($this->get_config('mossuserid'));
286             $cmd .= ' -l '.escapeshellarg($setting->language);
287             if (!empty($this->moss->sensitivity)) {
288                 $cmd .= ' -m '.escapeshellarg($this->moss->sensitivity);
289             }
290 
291             $basefiles = $fs->get_area_files($context->id, 'plagiarism_moss', 'basefiles', $setting-
    >id, 'filename', false);
292             foreach ($basefiles as $basefile) {
293                 $realpath = $this->tempdir.'/'.$setting->id.'_'.$basefile->get_filename();
```

File: /home/gabri/moodle/moss.php

```
294                 $basefile->copy_content_to($realpath);
295                 $cmd .= ' -b '.escapeshellarg($realpath);
296             }
297             $filepatterns = explode(' ', $setting->filepatterns);
298             foreach ($filepatterns as $filepattern) {
299                 $cmd .= ' '.str_replace(' ', '\\ ', $this->tempdir.'/*/'.$filepattern);
300             }
301             if (debugging('', DEBUG_DEVELOPER)) {
302                 mtrace("\t".$cmd);
303             }
304             $cmds[$setting->id] = $cmd;
305         }
306 
307         return $cmds;
308     }
309 
310     /**
311      * Parse moss result page and store to DB
312      *
313      * @param string $url moss result page url
314      * @param int $configid
315      * @return true or false
316      */
317     protected function save_results($url, $configid) {
318         global $DB, $UNITTEST;
319 
320         mtrace("\tProcessing $url");
321 
322         if (!$result_page = download_file_content($url)) {
323             mtrace("\tcan not read $url");
324             return false;
325         }
326 
327         preg_match_all(
328             '/(?P<link>http:\/\/moss\.stanford\.edu\/results\/\d+\/match\d+\.html)">.+\/(?
    P<user1>\d+)\/ \((?P<percentage1>\d+)%\).+\/(?P<user2>\d+)\/ \((?P<percentage2>\d+)%\).+right>(?
    P<linesmatched>\d+)\\n/Us',
329             $result_page,
330             $matches,
331             PREG_SET_ORDER
332         );
333 
334         if (!isset($UNITTEST)) { // testcase can not construct course structure
335             $context = get_context_instance(CONTEXT_COURSE, $this->moss->course);
336         }
337 
338         $filepatterns = $DB->get_field('plagiarism_moss_configs', 'filepatterns', array('id' =>
    $configid));
339         $filepatterns = explode(' ', $filepatterns);
340         $fs = get_file_storage();
341         $rank = 0;
342         foreach ($matches as $result) {
343             $rank++;
344 
345             $result['moss'] = $this->moss->id;
346             $result['config'] = $configid;
347             $result['rank'] = $rank + 1;
348             $result1 = (object)$result;
349             $result1->userid = $result['user1'];
350             $result1->percentage = $result['percentage1'];
351             $result2 = (object)$result;
352             $result2->userid = $result['user2'];
353             $result2->percentage = $result['percentage2'];
354 
355             if (!isset($UNITTEST)) { // testcase can not construct course structure
356                 // skip unenrolled users
357                 if (!is_enrolled($context, $result1->userid) and !is_enrolled($context, $result2-
    >userid)) {
358                     continue;
359                 }
360             }
361 
362             $result1->id = $DB->insert_record('plagiarism_moss_results', $result1);
363             $result2->pair = $result1->id;
364             $result2->id = $DB->insert_record('plagiarism_moss_results', $result2);
365             $result1->pair = $result2->id;
```

File: /home/gabri/moodle/moss.php

```
366             $DB->update_record('plagiarism_moss_results', $result1);
367
368             // update moss_matchedfiles db
369             for ($i=1; $i<=2; $i++) {
370                 $userid = eval('return $result'.$i.'->userid;');
371                 $resultid = eval('return $result'.$i.'->id;');
372
373                 $files = $fs->get_directory_files(get_system_context()->id, 'plagiarism_moss',
    'files', $this->moss->cmid, "/$userid/");
374                 foreach ($files as $file) {
375                     foreach ($filepatterns as $pattern) {
376                         if (fnmatch($pattern, $file->get_filename())) {
377                             $obj = new stdClass();
378                             $obj->result = $resultid;
379                             $obj->contenthash = $file->get_contenthash();
380                             $DB->insert_record('plagiarism_moss_matchedfiles', $obj);
381                         }
382                     }
383                 }
384             }
385         }
386
387         mtrace("\tGot $rank pairs");
388
389         return true;
390     }
391
392     /**
393      * Clean current stored results
394      */
395     protected function clean_results() {
396         global $DB;
397
398         $sql = 'DELETE FROM {plagiarism_moss_matchedfiles}
399                 WHERE result in (
400                     SELECT id FROM {plagiarism_moss_results}
401                     WHERE moss = ?
402                 )';
403         $DB->execute($sql, array($this->moss->id));
404         $DB->delete_records('plagiarism_moss_results', array('moss' => $this->moss->id));
405         //TODO: remove cached pages
406     }
407 }
```

File: /home/gabri/moodle/moss.bat

```
1   @echo off
2   rem Accept arguments as a line of command and run it
3   %*
```

File: /home/gabri/moodle/moss

```
#!/usr/bin/perl
use IO::Socket;

#
# As of the date this script was written, the following languages were supported.  This script will work with
# languages added later however.  Check the moss website for the full list of supported languages.
#
@languages = ("c", "cc", "java", "ml", "pascal", "ada", "lisp", "scheme", "haskell", "fortran", "ascii", "vhdl", "perl", "matlab", "python", "mips", "prolog", "spice", "vb", "csharp", "modula2", "a8086", "javascript", "plsql", "verilog");

$server = 'moss.stanford.edu';
$port = '7690';
$noreq = "Request not sent.";
$usage = "usage: moss [-u userid] [-x] [-l language] [-d] [-b basefile1] ... [-b basefilen] [-m #] [-c \"string\"] file1 file2 file3 ...";

#
# The userid is used to authenticate your queries to the server; don't change it!
#
$userid=0;

#
# Process the command line options.  This is done in a non-standard
# way to allow multiple -b's.
#
$opt_l = "c";   # default language is c
$opt_m = 10;
$opt_d = 0;
$opt_x = 0;
$opt_c = "";
$opt_n = 250;
$bindex = 0;    # this becomes non-zero if we have any base files

while (@ARGV && ($_ = $ARGV[0]) =~ /^-(.)(.*)/) {
   ($first,$rest) = ($1,$2);

   shift(@ARGV);
   if ($first eq "d") {
       $opt_d = 1;
       next;
   }
   if ($first eq "b") {
       if($rest eq '') {
           die "No argument for option -b.\n" unless @ARGV;
           $rest = shift(@ARGV);
       }
       $opt_b[$bindex++] = $rest;
       next;
   }
   if ($first eq "l") {
       if ($rest eq '') {
           die "No argument for option -l.\n" unless @ARGV;
           $rest = shift(@ARGV);
       }
       $opt_l = $rest;
       next;
   }
   if ($first eq "m") {
       if($rest eq '') {
           die "No argument for option -m.\n" unless @ARGV;
           $rest = shift(@ARGV);
       }
       $opt_m = $rest;
       next;
   }
   if ($first eq "c") {
       if($rest eq '') {
           die "No argument for option -c.\n" unless @ARGV;
           $rest = shift(@ARGV);
       }
       $opt_c = $rest;
       next;
   }
   if ($first eq "n") {
```

```
        if($rest eq '') {
            die "No argument for option -n.\n" unless @ARGV;
            $rest = shift(@ARGV);
        }
        $opt_n = $rest;
        next;
    }
    if ($first eq "x") {
        $opt_x = 1;
        next;
    }
    #
    # Set userid. A moodle moss plugin hack
    #
    if ($first eq "u") {
        if ($rest eq '') {
            die "No argument for option -u.\n" unless @ARGV;
            $rest = shift(@ARGV);
        }
        $userid = $rest;
        next;
    }
    #
    # Override the name of the server.  This is used for testing this script.
    #
    if ($first eq "s") {
        $server = shift(@ARGV);
        next;
    }
    #
    # Override the port.  This is used for testing this script.
    #
    if ($first eq "p") {
        $port = shift(@ARGV);
        next;
    }
    die "Unrecognized option -$first.  $usage\n";
}

if ("$userid" eq '0') {
   die "No valid userid provided.\n $usage";
}

#
# Check a bunch of things first to ensure that the
# script will be able to run to completion.
#

#
# Make sure all the argument files exist and are readable.
#
print "Checking files . . . \n";
$i = 0;
while($i < $bindex)
{
   die "Base file $opt_b[$i] does not exist. $noreq\n" unless -e "$opt_b[$i]";
   die "Base file $opt_b[$i] is not readable. $noreq\n" unless -r "$opt_b[$i]";
# comment out by sunner to allow non-ascii words
#   die "Base file $opt_b is not a text file. $noreq\n" unless -T "$opt_b[$i]";
   $i++;
}
$uploadable_file_count = 0;
foreach $file (@ARGV)
{
# change 'die' to 'print' by sunner to skip some glob format filenames
   print "File $file does not exist. $noreq\n" unless -e "$file";
   print "File $file is not readable. $noreq\n" unless -r "$file";
# comment out by sunner to allow non-ascii words
#   die "File $file is not a text file. $noreq\n" unless -T "$file";
   $uploadable_file_count++ if -r "$file";
}

if ("@ARGV" eq '') {
   die "No files submitted.\n $usage";
}
```

File: /home/gabri/moodle/moss

```
148   if ($uploadable_file_count eq 0) {
149      die "No files can be submitted.";
150   }
151   print "OK\n";
152
153   #
154   # Now the real processing begins.
155   #
156
157
158   $sock = new IO::Socket::INET (
159                                   PeerAddr => $server,
160                                   PeerPort => $port,
161                                   Proto => 'tcp',
162                                  );
163   die "Could not connect to server $server: $!\n" unless $sock;
164   $sock->autoflush(1);
165
166   sub read_from_server {
167      $msg = <$sock>;
168      print $msg;
169   }
170
171   sub upload_file {
172      local ($file, $id, $lang) = @_;
173      return unless -r "$file"; # skip unreadable files
174   #
175   # The stat function does not seem to give correct filesizes on windows, so
176   # we compute the size here via brute force.
177   #
178      open(F,$file);
179      $size = 0;
180      while (<F>) {
181          $size += length($_);
182      }
183      close(F);
184
185      print "Uploading $file ...";
186      print $sock "file $id $lang $size $file\n";
187      open(F,$file);
188      while (<F>) {
189          print $sock $_;
190      }
191      close(F);
192      print "done.\n";
193   }
194
195
196   print $sock "moss $userid\n";      # authenticate user
197   print $sock "directory $opt_d\n";
198   print $sock "X $opt_x\n";
199   print $sock "maxmatches $opt_m\n";
200   print $sock "show $opt_n\n";
201
202   #
203   # confirm that we have a supported languages
204   #
205   print $sock "language $opt_l\n";
206   $msg = <$sock>;
207   chop($msg);
208   if ($msg eq "no") {
209      print $sock "end\n";
210      die "Unrecognized language $opt_l.";
211   }
212
213
214   # upload any base files
215   $i = 0;
216   while($i < $bindex) {
217      &upload_file($opt_b[$i++],0,$opt_l);
218   }
219
220   $setid = 1;
221   foreach $file (@ARGV) {
222      &upload_file($file,$setid++,$opt_l);
```

File: /home/gabri/moodle/moss

```
223   }
224
225   print $sock "query 0 $opt_c\n";
226   print "Query submitted.  Waiting for the server's response.\n";
227   &read_from_server();
228   print $sock "end\n";
229   close($sock);
230
231
232
```

File: /home/gabri/moodle/module.js

```
1
2   M.plagiarism_moss = {};
3
4   M.plagiarism_moss.Y = {};
5
6   M.plagiarism_moss.confirm_button_clicked = function(e) {
7       e.preventDefault();
8       var Y = M.plagiarism_moss.Y;
9
10      var button = e.currentTarget;
11
12      var link = button.get('href');
13      var id = button.get('parentNode').get('id');
14
15      var sp = link.split('?', 2);
16      var uri = sp[0];
17      var data = sp[1];
18
19      if (data.charAt(data.length-1) == '1') { // unconfirmed now
20          if (!confirm(M.util.get_string('confirmmessage', 'plagiarism_moss'))) {
21              return;
22          }
23      }
24
25      // bind io events
26      Y.once('io:success', M.plagiarism_moss.iocomplete, Y, [id]);
27      Y.once('io:failure', M.plagiarism_moss.iofailure, Y, [id,
    button.get('parentNode').get('innerHTML')]);
28
29      data += '&ajax=1';
30      var cfg = {
31          method : 'GET',
32          data : data
33      }
34      Y.io(uri, cfg);
35
36      var buttons = Y.all('span#'+id);
37      buttons.setContent(M.plagiarism_moss.updating_html);
38  }
39
40  M.plagiarism_moss.iocomplete = function(transactionid, response, arguments) {
41      var Y = M.plagiarism_moss.Y;
42      var id = arguments[0];
43
44      var buttons = Y.all('span#'+id);
45      buttons.setContent(response.responseText);
46
47      M.plagiarism_moss.bind_buttons(id);
48  }
49
50  M.plagiarism_moss.iofailure = function(transactionid, response, arguments) {
51      alert('Network error');
52
53      var Y = M.plagiarism_moss.Y;
54      var id = arguments[0];
55      var old_html = arguments[1];
56
57      var buttons = Y.all('span#'+id);
58      buttons.setContent(old_html);
59
60      M.plagiarism_moss.bind_buttons(id);
61  }
62
63  M.plagiarism_moss.init = function(Y, updating_html) {
64      M.plagiarism_moss.Y = Y;
65      M.plagiarism_moss.updating_html = updating_html;
66
67      M.plagiarism_moss.Y.on("click", M.plagiarism_moss.confirm_button_clicked, "a.confirmbutton");
68  };
69
70  M.plagiarism_moss.bind_buttons = function(id) {
71      M.plagiarism_moss.Y.on("click", M.plagiarism_moss.confirm_button_clicked, "span#"+id+'
    a.confirmbutton');
72  }
```

File: /home/gabri/moodle/locallib.php

```php
<?php
defined('MOODLE_INTERNAL') || die('Direct access to this script is forbidden.');

require_once($CFG->dirroot.'/plagiarism/moss/moss.php');

define('MOSS_DEFAULT_MAXFILESIZE', 64 * 1024 * 1024);   // default max size of files to be submitted. I think 64K is big enough

/**
 * Whether moss is enabled
 *
 * @param int cmid
 * @return bool
 */
function moss_enabled($cmid = 0) {
    global $DB;

    if (!get_config('plagiarism', 'moss_use')) {
        return false;
    } else if ($cmid == 0) {
        return true;
    } else {
        return $DB->get_field('plagiarism_moss', 'enabled', array('cmid' => $cmid));
    }
}

/**
 * Save files in $eventdata to moss file area
 *
 * @param object $eventdata
 */
function moss_save_files_from_event($eventdata) {
    global $DB;
    $result = true;

    if (!moss_enabled($eventdata->cmid)) {
        return $result;
    }

    if (!empty($eventdata->file) && empty($eventdata->files)) { //single assignment type passes a single file
        $eventdata->files[] = $eventdata->file;
    }

    if (isset($eventdata->files)) {
        moss_save_storedfiles($eventdata->files, $eventdata->cmid, $eventdata->userid);
    }

    return $result;
}

/**
 * Save storedfiles into moss
 *
 * The function will clean all previous stored files
 * @param array $storedfiles
 * @param int $cmid
 * @param int $userid
 */
function moss_save_storedfiles($storedfiles, $cmid, $userid) {
    $context = get_system_context();
    $fs = get_file_storage();

    // remove all old files
    $old_files = $fs->get_directory_files($context->id, 'plagiarism_moss', 'files', $cmid, "/$userid/", true, true);
    foreach($old_files as $oldfile) {
        $oldfile->delete();
    }

    // store files
    foreach($storedfiles as $file) {
        if ($file->get_filename() === '.') {
            continue;
        }
```

File: /home/gabri/moodle/locallib.php

```
73          //hacky way to check file still exists
74          $fileid = $fs->get_file_by_id($file->get_id());
75          if (empty($fileid)) {
76              mtrace("nofilefound!");
77              continue;
78          }
79
80          $fileinfo = array(
81              'contextid' => $context->id,
82              'component' => 'plagiarism_moss',
83              'filearea'  => 'files',
84              'itemid'    => $cmid,
85              'filepath'  => "/$userid/",
86              // save /abc/def/ghi.c as abc_def_ghi.c
87              'filename'  => str_replace('/', '_', ltrim($file->get_filepath(), '/')).$file-
    >get_filename());
88          $fs->create_file_from_storedfile($fileinfo, $file);
89      }
90  }
91
92  /**
93   * Trigger assessable_file_uploaded and assessable_files_done events of specified assignment.
94   *
95   * @param int $cmid
96   * @param bool $trigger_done whether trigger assessable_files_done event. Advanced upload
    assignment only.
97   * @return number of submissions found, or false on failed
98   */
99  function moss_trigger_assignment_events($cmid, $trigger_done = true) {
100     global $DB, $CFG;
101     require_once($CFG->dirroot.'/mod/assignment/lib.php');
102
103     $cm = get_coursemodule_from_id('assignment', $cmid);
104     if (empty($cm)) {
105         return false;
106     }
107
108     $context = get_context_instance(CONTEXT_MODULE, $cmid);
109     $fs = get_file_storage();
110     $assignment = $DB->get_record('assignment', array('id' => $cm->instance), '*', MUST_EXIST);
111     $submissions = assignment_get_all_submissions($assignment);
112
113     foreach ($submissions as $submission) {
114         $files = $fs->get_area_files($context->id, 'mod_assignment', 'submission', $submission->id,
    "timemodified", false);
115         $eventdata = new stdClass();
116         $eventdata->modulename   = 'assignment';
117         $eventdata->cmid         = $cmid;
118         $eventdata->itemid       = $submission->id;
119         $eventdata->courseid     = $cm->course;
120         $eventdata->userid       = $submission->userid;
121         if ($files) {
122             if ($assignment->assignmenttype == 'upload') {
123                 $eventdata->files = $files;
124             } else { // uploadsingle
125                 $eventdata->file = current($files);
126             }
127         }
128         events_trigger('assessable_file_uploaded', $eventdata);
129         if ($trigger_done and $assignment->assignmenttype == 'upload') {
130             unset($eventdata->files);
131             events_trigger('assessable_files_done', $eventdata);
132         }
133     }
134
135     return count($submissions);
136 }
137
138 /**
139  * Sent notification
140  *
141  * @param object $result
142  */
143 function moss_message_send($result) {
144     global $DB, $CFG;
```

```php

    $teacher = $DB->get_record('user', array('id' => $result->confirmer));
    $user = $DB->get_record('user', array('id' => $result->userid));

    $subject = get_string('messagesubject', 'plagiarism_moss');

    $moss = $DB->get_record('plagiarism_moss', array('id' => $result->moss));
    $moss->link = $CFG->wwwroot."/plagiarism/moss/view.php?id=$moss->cmid&user=$result->userid";
    $html = '';
    if ($result->confirmed) {
        $text = get_string('messageconfirmedtext', 'plagiarism_moss', $moss);
        if ($user->mailformat == 1) {  // HTML
            $html = get_string('messageconfirmedhtml', 'plagiarism_moss', $moss);
        }
    } else {
        $text = get_string('messageunconfirmedtext', 'plagiarism_moss', $moss);
        if ($user->mailformat == 1) {  // HTML
            $html = get_string('messageunconfirmedhtml', 'plagiarism_moss', $moss);
        }
    }

    $eventdata = new stdClass();
    $eventdata->modulename       = 'moss';
    $eventdata->userfrom         = $teacher;
    $eventdata->userto           = $user;
    $eventdata->subject          = $subject;
    $eventdata->fullmessage      = $text;
    $eventdata->fullmessageformat = FORMAT_PLAIN;
    $eventdata->fullmessagehtml  = $html;
    $eventdata->smallmessage     = $subject;

    $eventdata->name            = 'moss_updates';
    $eventdata->component       = 'plagiarism_moss';
    $eventdata->notification    = 1;
    $eventdata->contexturl      = $moss->link;
    $eventdata->contexturlname  = $moss->modulename;

    message_send($eventdata);
}

function moss_get_supported_languages() {
    $langs = array(
        'ada'     => 'Ada',
        'a8086'   => get_string('langa8086', 'plagiarism_moss'),
        'c'       => 'C',
        'cc'      => 'C++',
        'csharp'  => 'C#',
        'fortran' => 'FORTRAN',
        'haskell' => 'Haskell',
        'java'    => 'Java',
        'javascript' => 'Javascript',
        'lisp'    => 'Lisp',
        'matlab'  => 'Matlab',
        'mips'    => get_string('langmips', 'plagiarism_moss'),
        'ml'      => 'ML',
        'modula2' => 'Modula2',
        'pascal'  => 'Pascal',
        'perl'    => 'Perl',
        'ascii'   => get_string('langascii', 'plagiarism_moss'),
        'plsql'   => 'PLSQL',
        'prolog'  => 'Prolog',
        'python'  => 'Python',
        'scheme'  => 'Scheme',
        'spice'   => 'Spice',
        'vhdl'    => 'VHDL',
        'vb'      => 'Visual Basic');

    textlib_get_instance()->asort($langs);
    return $langs;
}

/**
 * Clean up all data related with cmid
 *
 * @param int $cmid
```

File: /home/gabri/moodle/locallib.php

```
220     * @return true or false
221     */
222    function moss_clean_cm($cmid) {
223        global $DB;
224
225        if ($moss = $DB->get_record('plagiarism_moss', array('cmid' => $cmid))) {
226            // clean up configs
227            $DB->delete_records('plagiarism_moss_configs', array('moss' => $moss->id));
228
229            // clean up results
230            $results = $DB->get_records('plagiarism_moss_results', array('moss' => $moss->id));
231            foreach ($results as $result) {
232                $DB->delete_records('plagiarism_moss_matchedfiles', array('result' => $result->id));
233            }
234
235            // clean up files and results
236            if ($moss->tag == 0) {
237                // if no tag setted, no need to keep the files and moss record for further detection
238                $fs = get_file_storage();
239                $fs->delete_area_files(get_system_context()->id, 'plagiarism_moss', 'files', $cmid);
240                $DB->delete_records('plagiarism_moss', array('cmid' => $cmid));
241
242                // Clean results
243                $DB->delete_records('plagiarism_moss_results', array('moss' => $moss->id));
244            } else {
245                moss_clean_noise($moss);
246
247                // Disable moss record related with a deleted cm
248                // The record will be reserved for further reference by other mosses
249                // with the same tags
250                $moss->enabled = 0;
251                $DB->update_record('plagiarism_moss', $moss);
252
253                // Clean results
254                $DB->delete_records('plagiarism_moss_results', array('moss' => $moss->id));
255            }
256        }
257
258        return true;
259    }
260
261    /**
262     * Remove noise files.
263     *
264     * Noise files were uploaded by unenrolled users and confirmed as plagiarism.
265     * They may be noise in further detection
266     *
267     * @param object $moss  null means all moss instance
268     */
269    function moss_clean_noise($moss = null) {
270        global $DB;
271
272        $mosses = array();
273        if (empty($moss)) {
274            $mosses = $DB->get_records('plagiarism_moss');
275        } else {
276            $mosses[] = $moss;
277        }
278
279        foreach ($mosses as $moss) {
280            $sql = 'SELECT DISTINCT userid
281                from {plagiarism_moss_results}
282                where moss = ? and confirmed = 1';
283            if (! $confirmed_results = $DB->get_records_sql($sql, array($moss->id))) {
284                continue;
285            }
286
287            $fs = get_file_storage();
288            $systemcontext = get_system_context();
289            $cmcontext = get_context_instance(CONTEXT_MODULE, $moss->cmid);
290
291            foreach ($confirmed_results as $result) {
292                if (!is_enrolled($cmcontext, $result->userid)) {
293                    $old_files = $fs->get_directory_files($systemcontext->id, 'plagiarism_moss',
    'files', $moss->cmid, "/$result->userid/", true, true);
```

File: /home/gabri/moodle/locallib.php

```
294                 foreach($old_files as $oldfile) {
295                     $oldfile->delete();
296                 }
297                 if ($userdir = $fs->get_file($systemcontext->id, 'plagiarism_moss', 'files', $moss->cmid, "/$result->userid/", '.')) {
298                     $userdir->delete();
299                 }
300             }
301         }
302     }
303 }
304
305 /**
306  * Return the submit time
307  */
308 function moss_get_submit_time($cmid, $userid) {
309     global $DB;
310
311     $fs = get_file_storage();
312     $context = get_system_context();
313     $time = 0;
314     if ($files = $fs->get_directory_files($context->id, 'plagiarism_moss', 'files', $cmid, "/$userid/")) {
315         // search for the latest submitted file
316         foreach ($files as $file) {
317             if ($file->get_timemodified() > $time) {
318                 $time = $file->get_timemodified();
319             }
320         }
321     } else {
322         // lookup in cm with the same tag
323         $moss = $DB->get_record('plagiarism_moss', array('cmid' => $cmid), '*', MUST_EXIST);
324         $mosses = $DB->get_records_select('plagiarism_moss', 'tag = ? AND tag != 0', array($moss->tag));
325         foreach ($mosses as $moss) {
326             if ($files = $fs->get_directory_files($context->id, 'plagiarism_moss', 'files', $moss->cmid, "/$userid/")) {
327                 // search for the latest submitted file
328                 foreach ($files as $file) {
329                     if ($file->get_timemodified() > $time) {
330                         $time = $file->get_timemodified();
331                     }
332                 }
333             }
334             // Do not lookup other mosses any more if got one matched
335             if ($time > 0) {
336                 break;
337             }
338         }
339     }
340
341     return $time;
342 }
343
344 /**
345  * Measure all moss instances that should be measured
346  */
347 function moss_measure_all() {
348     global $DB;
349
350     // get mosses measure on specified time
351     $select  = 'timetomeasure < ? AND timetomeasure > timemeasured AND enabled = 1 AND timetomeasure != 0';
352     $mosses = $DB->get_records_select('plagiarism_moss', $select, array(time()));
353
354     // get mosses measure on activity due date
355     $duemosses = $DB->get_records('plagiarism_moss', array('enabled' => 1, 'timetomeasure' => 0));
356     foreach ($duemosses as $moss) {
357         if ($cm = get_coursemodule_from_id('', $moss->cmid)) {
358             switch ($cm->modname) {
359             case 'assignment':
360                 $duedate = $DB->get_field('assignment', 'timedue', array('id' => $cm->instance));
361             }
362             if (!empty($duedate)) {
363                 if ($moss->timemeasured < $duedate and $duedate < time()) {
364                     // it should be measured
```

File: /home/gabri/moodle/locallib.php

```
365                     $mosses[] = $moss;
366                 }
367             }
368         }
369     }
370
371     mtrace("\tFound ".count($mosses)." moss instances to measure");
372
373     foreach ($mosses as $moss) {
374         mtrace("\tMeasure $moss->modulename ($moss->cmid) in $moss->coursename");
375         $moss_obj = new moss($moss->cmid);
376         $moss_obj->measure();
377     }
378 }
```

File: /home/gabri/moodle/lib.php

```php
<?php
defined('MOODLE_INTERNAL') || die('Direct access to this script is forbidden.');

require_once($CFG->dirroot.'/plagiarism/lib.php');
require_once($CFG->dirroot.'/plagiarism/moss/locallib.php');

define('MOSS_MAX_PATTERNS', 3);

/**
 * plagiarism_plugin_moss inherit from plagiarism_plugin class, this is the most important class in plagiarism plugin,
 * Moodle platform will automatically call the function of this class.
 * @author Sun Zhigang
 *
 */
class plagiarism_plugin_moss extends plagiarism_plugin {
        /**
         * (non-PHPdoc)
         * @see plagiarism_plugin::print_disclosure()
         */
    public function print_disclosure($cmid) {
        global $OUTPUT, $DB;
        if (moss_enabled($cmid)) {
            echo $OUTPUT->box_start('generalbox boxaligncenter', 'intro');

            $moss = $DB->get_record('plagiarism_moss', array('cmid' => $cmid), 'timetomeasure, timemeasured');
            $a->timemeasured = userdate($moss->timemeasured);
            if ($moss->timemeasured == 0) {
                $disclosure = get_string('disclosurenevermeasured', 'plagiarism_moss', $a);
            } else {
                $disclosure = get_string('disclosurehasmeasured', 'plagiarism_moss', $a);
            }

            if ($moss->timemeasured != 0 and has_capability('plagiarism/moss:viewallresults', get_context_instance(CONTEXT_MODULE, $cmid))) {
                $url = new moodle_url('/plagiarism/moss/view.php', array('id' => $cmid));
                $disclosure .= ' '.html_writer::link($url, get_string('clicktoviewresults', 'plagiarism_moss'));
            }

            echo format_text($disclosure, FORMAT_MOODLE);
            echo $OUTPUT->box_end();
        }
    }

    /**
     * Hook to save plagiarism specific settings on a module settings page
     * @param object $data - data from an mform submission.
     */
    public function save_form_elements($data) {
        global $DB;

        if (!moss_enabled()) {
            return;
        }

        $moss = new stdClass();
        $moss->enabled = empty($data->enabled) ? 0 : 1;
        if (!$moss->enabled) {
            if (isset($data->mossid)) { // disable it
                $DB->set_field('plagiarism_moss', 'enabled', 0, array('id' => $data->mossid));
            }
            // disabled mosses keep old configs
            return;
        }

        $moss->timetomeasure = $data->timetomeasure;
        $moss->cmid = $data->coursemodule;
        $moss->sensitivity = $data->sensitivity;
        $moss->modulename = $data->name;
        $moss->coursename = $DB->get_field('course', 'shortname', array('id' => $data->course));

        // process tag
        if (empty($data->tag)) {
            $moss->tag = 0;
```

File: /home/gabri/moodle/lib.php

```
73          } else {
74              if ($tagid = $DB->get_field('plagiarism_moss_tags', 'id', array('name' => $data->tag)))
    {
75                  $moss->tag = $tagid;
76              } else {
77                  $tag = new stdClass();
78                  $tag->name = $data->tag;
79                  $moss->tag = $DB->insert_record('plagiarism_moss_tags', $tag);
80              }
81          }
82
83          if (isset($data->mossid)) {
84              $moss->id = $data->mossid;
85              $DB->update_record('plagiarism_moss', $moss);
86          } else {
87              $data->mossid = $DB->insert_record('plagiarism_moss', $moss);
88          }
89
90          // sub configs
91          for($index = 0; $index < MOSS_MAX_PATTERNS; $index++) {
92              $config = new stdClass();
93              $config->moss = $data->mossid;
94              $member = 'language'.$index;
95              $config->language = isset($data->$member) ? $data->$member :
    get_config('plagiarism_moss', 'defaultlanguage');
96
97              $member = 'filepatterns'.$index;
98              $config->filepatterns = str_replace('\\', '_', str_replace('/', '_', $data-
    >$member)); // filter out path chars
99              if ($index == 0 and empty($config->filepatterns)) {
100                 $config->filepatterns = '*';
101             }
102
103             $member= 'configid'.$index;
104             if (isset($data->$member)) {
105                 $config->id = $data->$member;
106                 $DB->update_record('plagiarism_moss_configs', $config);
107             } else {
108                 $config->id = $DB->insert_record('plagiarism_moss_configs', $config);
109             }
110
111             $context = get_system_context();
112             $member= 'basefile'.$index;
113             file_save_draft_area_files($data->$member, $context->id, 'plagiarism_moss', 'basefiles',
    $config->id);
114         }
115     }
116
117     /**
118      * (non-PHPdoc)
119      * @see plagiarism_plugin::get_form_elements_module()
120      */
121     public function get_form_elements_module($mform, $context) {
122         global $DB;
123
124         if (!moss_enabled()) {
125             return;
126         }
127
128         // Construct the form
129         $mform->addElement('header', 'mossdesc', get_string('moss', 'plagiarism_moss'));
130         $mform->addHelpButton('mossdesc', 'moss', 'plagiarism_moss');
131
132         $mform->addElement('checkbox', 'enabled', get_string('mossenabled', 'plagiarism_moss'));
133
134         $mform->addElement('date_time_selector', 'timetomeasure', get_string('timetomeasure',
    'plagiarism_moss'), array('optional' => true));
135         $mform->addHelpButton('timetomeasure', 'timetomeasure', 'plagiarism_moss');
136         $mform->disabledIf('timetomeasure', 'enabled');
137
138         $mform->addElement('text', 'tag', get_string('tag', 'plagiarism_moss'));
139         $mform->addHelpButton('tag', 'tag', 'plagiarism_moss');
140         $mform->setType('tag', PARAM_TEXT);
141         $mform->disabledIf('tag', 'enabled');
142
143         $mform->addElement('text', 'sensitivity', get_string('sensitivity', 'plagiarism_moss'),
```

```
    'size = "10"');
144         $mform->addHelpButton('sensitivity', 'sensitivity', 'plagiarism_moss');
145         $mform->setType('sensitivity', PARAM_NUMBER);
146         $mform->addRule('sensitivity', null, 'numeric', null, 'client');
147         $mform->disabledIf('sensitivity', 'enabled');
148 
149         // multi configs
150         for($index = 0; $index < MOSS_MAX_PATTERNS; $index++) {
151             if ($index == 0) {
152                 $subheader = get_string('configrequired', 'plagiarism_moss', $index+1);
153             } else {
154                 $subheader = get_string('configoptional', 'plagiarism_moss', $index+1);
155             }
156             $subheader = html_writer::tag('strong', $subheader);
157             $mform->addElement('static', 'subheader'.$index, $subheader);
158 
159             $mform->addElement('text', 'filepatterns'.$index, get_string('filepatterns',
    'plagiarism_moss'));
160             $mform->addHelpButton('filepatterns'.$index, 'filepatterns', 'plagiarism_moss');
161             $mform->setType('filepatterns'.$index, PARAM_TEXT);
162             $mform->disabledIf('filepatterns'.$index, 'enabled');
163 
164             $choices = moss_get_supported_languages();
165             $mform->addElement('select', 'language'.$index, get_string('language',
    'plagiarism_moss'), $choices);
166             $mform->disabledIf('language'.$index, 'enabled');
167             $mform->setDefault('language'.$index, get_config('plagiarism_moss', 'defaultlanguage'));
168 
169             $mform->addElement('filemanager', 'basefile'.$index, get_string('basefile',
    'plagiarism_moss'), null, array('subdirs' => 0));
170             $mform->addHelpButton('basefile'.$index, 'basefile', 'plagiarism_moss');
171             $mform->disabledIf('basefile'.$index, 'enabled');
172 
173             if ($index != 0) {
174                 $mform->setAdvanced('subheader'.$index);
175                 $mform->setAdvanced('filepatterns'.$index);
176                 $mform->setAdvanced('language'.$index);
177                 $mform->setAdvanced('basefile'.$index);
178             }
179         }
180 
181         // set config values
182         $cmid = optional_param('update', 0, PARAM_INT); //there doesn't seem to be a way to obtain
    the current cm a better way - $this->_cm is not available here.
183         if ($cmid != 0 and $moss = $DB->get_record('plagiarism_moss', array('cmid'=>$cmid))) { //
    configed
184             $mform->setDefault('enabled', $moss->enabled);
185             $mform->setDefault('timetomeasure', $moss->timetomeasure);
186             $mform->setDefault('tag', $DB->get_field('plagiarism_moss_tags', 'name', array('id' =>
    $moss->tag')));
187             if (!empty($moss->sensitivity)) {
188                 $mform->setDefault('sensitivity', $moss->sensitivity);
189             }
190             $mform->addElement('hidden', 'mossid', $moss->id);
191 
192             $subconfigs = $DB->get_records('plagiarism_moss_configs', array('moss'=>$moss->id));
193             $index = 0;
194             foreach ($subconfigs as $subconfig) {
195                 $mform->setDefault('filepatterns'.$index, $subconfig->filepatterns);
196                 $mform->setDefault('language'.$index, $subconfig->language);
197                 $mform->addElement('hidden', 'configid'.$index, $subconfig->id);
198 
199                 $context = get_system_context();
200                 $draftitemid = 0;
201                 file_prepare_draft_area($draftitemid, $context->id, 'plagiarism_moss', 'basefiles',
    $subconfig->id);
202                 $mform->setDefault('basefile'.$index, $draftitemid);
203 
204                 $index++;
205             }
206         } else { // new config
207             $mform->setDefault('enabled', 0);
208             $mform->setDefault('tag', '');
209             $mform->setDefault('filepatterns0', '*');
210             // leave other subconfig empty
211         }
212     }
```

File: /home/gabri/moodle/lib.php

```
213
214         /**
215          * hook to allow plagiarism specific information to be displayed beside a submission
216          *
217          * @param array  $linkarraycontains all relevant information for the plugin to generate a link
218          * @return string
219          */
220         public function get_links($linkarray) {
221             global $DB, $OUTPUT;
222
223             $link = '';
224             if (!moss_enabled($linkarray['cmid'])) {
225                 return $link;
226             }
227
228             $sql = 'SELECT r.*
229                     FROM {plagiarism_moss_results} r
230                     LEFT JOIN {plagiarism_moss_matchedfiles} f ON r.id = f.result
231                     WHERE f.contenthash = :contenthash AND r.userid = :userid AND r.moss = :mossid ';
232             if (!has_capability('plagiarism/moss:viewunconfirmed', get_context_instance(CONTEXT_MODULE,
    $linkarray['cmid']))) {
233                 $sql .= 'AND r.confirmed = 1 ';
234             }
235             $sql .= 'ORDER BY r.rank ASC';
236             $params = array(
237                 'userid'      => $linkarray['userid'],
238                 'contenthash' => $linkarray['file']->get_contenthash(),
239                 'mossid'      => $DB->get_field('plagiarism_moss', 'id', array('cmid' =>
    $linkarray['cmid']))
240             );
241
242             $results = $DB->get_records_sql($sql, $params);
243             if (!empty($results)){
244                 $result = current($results);
245
246                 $text = $result->percentage.'%'."($result->linesmatched)";
247                 $icon = $OUTPUT->pix_icon('i/completion-manual-n', get_string('unconfirmed',
    'plagiarism_moss'));
248                 foreach ($results as $r) {
249                     if ($r->confirmed) {
250                         $icon = $OUTPUT->pix_icon('i/completion-manual-y', get_string('confirmed',
    'plagiarism_moss'));
251                         break;
252                     }
253                 }
254                 $text .= $icon;
255
256                 $result->count = count($results);
257                 $title = get_string('resultlinktitle', 'plagiarism_moss', $result);
258
259                 $params = array('id' => $linkarray['cmid'], 'user' => $linkarray['userid']);
260                 $url = new moodle_url('/plagiarism/moss/view.php', $params);
261
262                 $link = html_writer::link($url, $text, array('title' => $title));
263             }
264
265             return $link;
266         }
267
268         /**
269          * Hook for cron
270          */
271         public function cron() {
272             mtrace('---Moss begins---');
273
274             moss_clean_noise();
275             moss_measure_all();
276
277             mtrace('---Moss done---');
278         }
279
280     }
281
282     /**
283      *
284      * Enter description here ...
```

File: /home/gabri/moodle/lib.php

```php
     * @param unknown_type $eventdata
     */
    function moss_event_file_uploaded($eventdata) {
        return moss_save_files_from_event($eventdata);
    }

    /**
     * A module has been deleted
     */
    function moss_event_mod_deleted($eventdata) {
        return moss_clean_cm($eventdata->cmid);
    }
```

File: /home/gabri/moodle/simpletest/testmoss.php

```php
<?php
if (!defined('MOODLE_INTERNAL')) {
    die('Direct access to this script is forbidden.'); //  It must be included from a Moodle page
}

// access to use global variables.
require_once(dirname(dirname(dirname(dirname(__FILE__)))) . '/config.php');

// Make sure the code being tested is accessible.
require_once($CFG->dirroot . '/plagiarism/moss/lib.php');

/** This class contains the test cases for the functions in judegelib.php. */
class plagiarism_moss_test extends UnitTestCase {
        function setUp() {
        global $DB, $CFG;

        $this->realDB = $DB;
        $dbclass = get_class($this->realDB);
        $DB = new $dbclass();
        $DB->connect($CFG->dbhost, $CFG->dbuser, $CFG->dbpass, $CFG->dbname, $CFG->unittestprefix);

        if ($DB->get_manager()->table_exists('plagiarism_moss')) {
            $DB->get_manager()->delete_tables_from_xmldb_file($CFG->dirroot . '/plagiarism/moss/db/install.xml');
        }
        $DB->get_manager()->install_from_xmldb_file($CFG->dirroot . '/plagiarism/moss/db/install.xml');

        if ($DB->get_manager()->table_exists('files')) {
            $table = new xmldb_table('files');
            $DB->get_manager()->drop_table($table);
            $table = new xmldb_table('config_plugins');
            $DB->get_manager()->drop_table($table);
        }
        $DB->get_manager()->install_one_table_from_xmldb_file($CFG->dirroot . '/lib/db/install.xml', 'files');
        $DB->get_manager()->install_one_table_from_xmldb_file($CFG->dirroot . '/lib/db/install.xml', 'config_plugins');

        set_config('mossuserid', 580031178, 'plagiarism_moss');
        set_config('maxfilesize', MOSS_DEFAULT_MAXFILESIZE, 'plagiarism_moss');
        set_config('cygwinpath', 'C:\\cygwin', 'plagiarism_moss');
        set_config('moss_use', 1, 'plagiarism');

        $mosses = array(
            array('cmid' => 1, 'tag' => 1, 'enabled' => 0, 'timetomeasure' => time(), 'sensitivity' => 10),
            array('cmid' => 2, 'tag' => 1, 'enabled' => 0, 'timetomeasure' => time(), 'sensitivity' => 10),
            array('cmid' => 3, 'tag' => 1, 'enabled' => 0, 'timetomeasure' => time(), 'sensitivity' => 10),
            array('cmid' => 4, 'tag' => 2, 'enabled' => 0, 'timetomeasure' => time(), 'sensitivity' => 10),
        );
        $tags = array(
            array('name' => 'test1'),
            array('name' => 'test2'),
            array('name' => 'test3'),
            array('name' => 'test4')
        );
        foreach ($tags as $tag) {
            $DB->insert_record('plagiarism_moss_tags', $tag);
        }
        foreach ($mosses as $moss) {
            $DB->insert_record('plagiarism_moss', $moss);
        }
    }

        function tearDown() {
                global $DB, $CFG;
        $DB = $this->realDB;
        }

        function test_onefile() {
        $events = array(
            array('userid' => 1, 'cmid' => 1),
            array('userid' => 2, 'cmid' => 1),
```

File: /home/gabri/moodle/simpletest/testmoss.php

```
70              array('userid' => 3, 'cmid' => 1),
71              array('userid' => 4, 'cmid' => 1)
72          );
73          $contents = array(
74              array(
75                  '/file1.txt' => 'What is Moss?
76
77              Moss (for a Measure Of Software Similarity) is an automatic system for determining the
    similarity of programs. To date, the main application of Moss has been in detecting plagiarism in
    programming classes. Since its development in 1994, Moss has been very effective in this role. The
    algorithm behind moss is a significant improvement over other cheating detection algorithms (at
    least, over those known to us).',
78                  '/source1.c' => 'What is Moss?
79
80                  Moss (for a Measure Of Software Similarity) is an automatic system for determining
    the similarity of programs. To date, the main application of Moss has been in detecting plagiarism
    in programming classes. Since its development in 1994, Moss has been very effective in this role.
    The algorithm behind moss is a significant improvement over other cheating detection algorithms (at
    least, over those known to us).'
81              ),
82              array(
83                  '/file2.txt' => 'What is Moss?
84
85              Moss (for a Measure Of Software Similarity) is an automatic system for determining the
    similarity of programs. To date, the main application of Moss has been in detecting plagiarism in
    programming classes. Since its development in 1994, Moss has been very effective in this role. The
    algorithm behind moss is a significant improvement over other cheating detection algorithms (at
    least, over those known to us).',
86                  '/source2.c' => 'What is Moss?
87
88                  Moss (for a Measure Of Software Similarity) is an automatic system for determining
    the similarity of programs. To date, the main application of Moss has been in detecting plagiarism
    in programming classes. Since its development in 1994, Moss has been very effective in this role.
    The algorithm behind moss is a significant improvement over other cheating detection algorithms (at
    least, over those known to us).'
89              ),
90              array('/file3.txt' => 'What is Moss?
91
92              Moss (for a Measure Of Software Similarity) is an automatic system for determining the
    similarity of programs. To date, the main application of Moss has been in detecting plagiarism in
    programming classes. Since its development in 1994, Moss has been very effective in this role. The
    algorithm behind moss is a significant improvement over other cheating detection algorithms (at
    least, over those known to us).
93              Languages
94
95                  Moss can currently analyze code written in the following languages:
96                  C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp,
    Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086
    assembly, a8086 assembly, MIPS assembly, HCL2.'),
97              array('/file4.txt' => 'Languages
98
99                  Moss can currently analyze code written in the following languages:
100                 C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp,
    Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086
    assembly, a8086 assembly, MIPS assembly, HCL2.')
101         );
102
103         $fs = get_file_storage();
104         $i = 0;
105         $files = array();
106         foreach ($contents as $oneuser) {
107             foreach ($oneuser as $key => $content) {
108                 $file_record = new stdClass();
109                 $file_record->contextid = 1;
110                 $file_record->component = 'test';
111                 $file_record->filearea = 'test';
112                 $file_record->filepath = dirname($key).'/';
113                 $file_record->filename = basename($key);
114                 $file_record->itemid = $i;
115                 $fs->create_file_from_string($file_record, $content);
116             }
117             $files[] = $fs->get_area_files(1, 'test', 'test', $i, 'sortorder, filename', false);
118             $i++;
119         }
120
121         foreach($events as $i => & $event) {
122             $event = (object)$event;
123             $event->files = current($files);
124             next($files);
```

File: /home/gabri/moodle/simpletest/testmoss.php

```
125             }
126
127             $this->add_config(1, '*.txt');
128             $this->trigger(1, $events);
129             }
130
131         function add_config($cmid, $filepatterns = '*', $language = 'ascii') {
132             global $DB;
133             $moss = $cmid; // Yes, they should be same
134             $DB->insert_record(
135                 'plagiarism_moss_configs',
136                 array(
137                     'moss' => $moss,
138                     'filepatterns' => $filepatterns,
139                     'language' => $language,
140                 )
141             );
142         }
143
144         function trigger($cmid, $events) {
145             global $DB;
146
147             $DB->set_field('plagiarism_moss', 'enabled', 1, array('cmid' => $cmid));
148             foreach ($events as $event) {
149                 moss_event_file_uploaded($event);
150             }
151
152             $plagiarism = new moss($cmid);
153             $this->assertTrue($plagiarism->measure());
154         }
155     }
```

File: /home/gabri/moodle/lang/en/plagiarism_moss.php

```php
<?php
$string['activityconfirmedresults'] = 'There are {$a} users are confirmed as plagiarim.';
$string['allresults'] = 'All results';
$string['antiwordpath'] = 'Path of antiword';
$string['antiwordpath_help'] = 'Path of antiword executable binary file

If set (recommended), moss will use antiword to extract text from .doc files. If leave blank, moss will do it through a less stable internal function.

You can download antiword from <a href="http://www.winfield.demon.nl/">http://www.winfield.demon.nl/</a>.';
$string['basefile'] = 'Base file';
$string['basefile_help'] = 'Moss normally reports all code that matches in pairs of files. When a base file is supplied, program code that also appears in the base file is not counted in matches. A typical base file will include, for example, the instructor-supplied code for an assignment. You can provide multiple base files here. Base files improve results, but are not usually necessary for obtaining useful information.';
$string['clicktoviewresults'] = 'Click here to view results.';
$string['configrequired'] = 'Config {$a} (required):';
$string['configoptional'] = 'Config {$a} (optional):';
$string['confirm'] = 'Confirm';
$string['confirm_help'] = 'This user is confirmed as a cheater or not.

* Confirmed plagiarism - yes, he/she is a cheater
* Unconfirmed - no, he/she is not a cheater
* No icon - the user is not enrolled to this course

The icon is clickable for teachers. Every click will send a notification to corresponding user.';
$string['confirmed'] = 'Confirmed plagiarism';
$string['confirmedresults'] = '{$a->fullname} has confirmed plagiarism records in <strong>{$a->total}</strong> activities.';
$string['confirmmessage'] = 'Are you sure this is plagiarism?';
$string['cygwinpath'] = 'Path to Cygwin installation';
$string['deltatime'] = 'Delta time';
$string['deltatime_help'] = 'How long the user 1 submitted later than the user 2.

The user 1 is always the later one. In common, the person who submitted later is the copier and another person is the copiee. However, it is not always true.';
$string['defaultlanguage'] = 'Default content type';
$string['disclosurehasmeasured'] ='All files submitted here has been measured by a plagiarism detection service at {$a->timemeasured}.';
$string['disclosurenevermeasured'] ='All files submitted here will be measured by a plagiarism detection service.';
$string['err_cygwinpath'] ='Bad Cygwin path or perl for Cygwin is not installed';
$string['filepatterns'] = 'Filename patterns';
$string['filepatterns_help'] = 'Glob format. E.g. \*.c, hello.\*, a?c.java. Use blank space to seperate multi patterns. Leave blank to disable the config.';
$string['language'] = 'Submission content type';
$string['langa8086'] = 'a8086 assembly';
$string['langascii'] = 'Text';
$string['langmips'] = 'MIPS assembly';
$string['matchedlines'] = 'Matched lines';
$string['matchedusers'] = 'Matched users';
$string['maxfilesize'] = 'File size limit in bytes';
$string['maxfilesize_help'] = 'All files bigger than the limit will be skipped. Please keep it reasonable to protect the free moss service.';
$string['messageprovider:moss_updates'] = 'Moss anti-plagiarism notifications';
$string['messagesubject'] = 'Moss anti-plagiarism notification';
$string['messageconfirmedhtml'] = '<p>Your submissions of {$a->modulename} in {$a->coursename} have been confirmed as <em>plagiarism</em>. </p><p>Visit <a href="{$a->link}">{$a->link}</a> for details.</p>';
$string['messageconfirmedtext'] = 'Your submissions of {$a->modulename} in {$a->coursename} have been confirmed as PLAGIARISM.
Visit {$a->link} for details.';
$string['messageunconfirmedhtml'] = '<p>Your submissions of {$a->modulename} in {$a->coursename} have been confirmed as <em>not</em> plagiarism. </p><p>Visit <a href="{$a->link}">{$a->link}</a> for details.</p>';
$string['messageunconfirmedtext'] = 'Your submissions of {$a->modulename} in {$a->coursename} have been confirmed as NOT plagiarism.
Visit {$a->link} for details.';
$string['moss'] = 'Moss anti-plagiarism';
$string['moss_help'] = '<a href="http://theory.stanford.edu/~aiken/moss/">Moss</a> (for a Measure Of Software Similarity) is an automatic system for determining the similarity of source code and plain text. It supports all kinds of source code files as well as pdf, doc, docx, rtf and odt documents.

Notice: only the files submitted when moss is enabled will be checked.';
$string['moss:confirm'] = 'Confirm plagiarism';
$string['moss:viewallresults'] = 'View results of everyone';
$string['moss:viewdiff'] = 'View pair compare';
```

File: /home/gabri/moodle/lang/en/plagiarism_moss.php

```
60  $string['moss:viewunconfirmed'] = 'View unconfirmed results';
61  $string['mossexplain'] = '<a href="https://github.com/hit-moodle/moodle-plagiarism_moss">Moss Anti-
    Plagiarism Plugin</a> is developped by <a href="http://www.hit.edu.cn/">Harbin Institute of
    Technology</a>. The plagiarism engine is <a href="http://theory.stanford.edu/~aiken/moss/">Moss</
    a>.';
62  $string['mossenabled'] ='Enable moss';
63  $string['mossuserid'] ='Moss account';
64  $string['mossuserid_help'] ='To obtain a Moss account, send a mail message to <a
    href="mailto:moss@moss.stanford.edu">moss@moss.stanford.edu</a>. The body of the message should be
    in <strong>PLAIN TEXT</strong>(without any HTML tags) format and appear exactly as follows:
65
66      registeruser
67      mail username@domain
68
69  After registration, you will get a reply mail which contains a perl script with one line of code
    just likes:
70
71      $userid=1234567890;
72
73  The number is exactly your moss account.';
74  $string['nocmresults'] = 'No plagiarism records';
75  $string['nouserresults'] = 'No plagiarism records related with {$a}';
76  $string['percentage'] = 'Similarity';
77  $string['personalresults'] = 'Personal results';
78  $string['pluginname'] = 'Moss anti-plagiarism';
79  $string['resultlinktitle'] = 'Up to {$a->percentage}% ({$a->linesmatched} lines) is similar with
    other {$a->count} user(s)';
80  $string['savedconfigsuccess'] = 'Moss anti-plagiarism settings saved';
81  $string['sensitivity'] = 'Sensitivity';
82  $string['sensitivity_help'] = 'The sensitivity option sets the maximum number of times a given
    passage may appear before it is ignored. A passage of code that appears in many programs is probably
    legitimate sharing and not the result of plagiarism. With sensitivity N, any passage appearing in
    more than N programs is treated as if it appeared in a base file (i.e., it is never reported). With
    sensitivity 2, moss reports only passages that appear in exactly two programs. If one expects many
    very similar solutions (e.g., the short first assignments typical of introductory programming
    courses) then using 3 or 4 is a good way to eliminate all but truly unusual matches between programs
    while still being able to detect 3-way or 4-way plagiarism. With 1000000 (or any very large number),
    moss reports all matches, no matter how often they appear. The setting is most useful for large
    assignments where one also a base file expected to hold all legitimately shared code.';
83  $string['showidnumber'] = 'Show idnumbers in results';
84  $string['tag'] = 'Tag';
85  $string['tag_help'] = 'Different activities using the same tag will be measured together. Tag is
    helpful to prevent plagiarism among courses.';
86  $string['timesubmitted'] ='Time submitted';
87  $string['timetomeasure'] ='Time to measure';
88  $string['timetomeasure_help'] ='Set the time to measure all submissions to detect plagiarism. If not
    set, the measure will occur after the activity\'s due time.
89
90  The measure will be executed only once against all existing submissions. If you want to measure
    again, reset the time.';
91
92  $string['unconfirmed'] = 'Unconfirmed';
93  $string['unsupportedmodule'] ='Moss does not support this module.';
94  $string['updating'] = 'Updating...';
```

File: /home/gabri/moodle/db/access.php

```php
<?php
$capabilities = array(

    'plagiarism/moss:viewunconfirmed'    => array(
        'captype'        => 'read',
        'contextlevel'   => CONTEXT_MODULE,
        'archetypes' => array(
            'student'        => CAP_ALLOW,
            'teacher'        => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager'        => CAP_ALLOW
        )
    ),

    'plagiarism/moss:viewallresults'    => array(
        'riskbitmask' => RISK_PERSONAL,

        'captype'        => 'read',
        'contextlevel'   => CONTEXT_MODULE,
        'archetypes' => array(
            'teacher'        => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager'        => CAP_ALLOW
        )
    ),

    'plagiarism/moss:viewdiff'    => array(
        'captype'        => 'read',
        'contextlevel'   => CONTEXT_MODULE,
        'archetypes' => array(
            'teacher'        => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager'        => CAP_ALLOW
        )
    ),

    'plagiarism/moss:confirm'    => array(
        'riskbitmask' => RISK_XSS,

        'captype'        => 'write',
        'contextlevel'   => CONTEXT_MODULE,
        'archetypes' => array(
            'teacher'        => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager'        => CAP_ALLOW
        )
    )
);
```

File: /home/gabri/moodle/db/events.php

```php
<?php

$handlers = array (

/*
 * Event Handlers
 */
    'assessable_file_uploaded' => array (
        'handlerfile'      => '/plagiarism/moss/lib.php',
        'handlerfunction'  => 'moss_event_file_uploaded',
        'schedule'         => 'instant'
    ),
    'mod_deleted' => array (
        'handlerfile'      => '/plagiarism/moss/lib.php',
        'handlerfunction'  => 'moss_event_mod_deleted',
        'schedule'         => 'instant'
    ),
);
```

File: /home/gabri/moodle/db/install.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<XMLDB PATH="plagiarism/moss/db" VERSION="20111020" COMMENT="XMLDB file for Moodle plagiarism/moss plugin"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../../../lib/xmldb/xmldb.xsd"
>
  <TABLES>
    <TABLE NAME="plagiarism_moss" COMMENT="MOSS instances" NEXT="plagiarism_moss_configs">
      <FIELDS>
        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true" NEXT="cmid"/>
        <FIELD NAME="cmid" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="id" NEXT="timetomeasure"/>
        <FIELD NAME="timetomeasure" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="cmid" NEXT="timemeasured"/>
        <FIELD NAME="timemeasured" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" DEFAULT="0" SEQUENCE="false" PREVIOUS="timetomeasure" NEXT="tag"/>
        <FIELD NAME="tag" TYPE="int" LENGTH="10" NOTNULL="false" UNSIGNED="true" SEQUENCE="false" PREVIOUS="timemeasured" NEXT="sensitivity"/>
        <FIELD NAME="sensitivity" TYPE="int" LENGTH="10" NOTNULL="false" UNSIGNED="true" SEQUENCE="false" PREVIOUS="tag" NEXT="enabled"/>
        <FIELD NAME="enabled" TYPE="int" LENGTH="4" NOTNULL="true" UNSIGNED="true" DEFAULT="0" SEQUENCE="false" PREVIOUS="sensitivity" NEXT="coursename"/>
        <FIELD NAME="coursename" TYPE="char" LENGTH="100" NOTNULL="true" SEQUENCE="false" COMMENT="Cache course name" PREVIOUS="enabled" NEXT="modulename"/>
        <FIELD NAME="modulename" TYPE="char" LENGTH="255" NOTNULL="true" SEQUENCE="false" COMMENT="Cache activity module name" PREVIOUS="coursename"/>
      </FIELDS>
      <KEYS>
        <KEY NAME="primary" TYPE="primary" FIELDS="id" NEXT="cmid"/>
        <KEY NAME="cmid" TYPE="foreign-unique" FIELDS="cmid" REFTABLE="course_modules" REFFIELDS="id" PREVIOUS="primary" NEXT="tag"/>
        <KEY NAME="tag" TYPE="foreign" FIELDS="tag" REFTABLE="plagiarism_moss_tags" REFFIELDS="id" PREVIOUS="cmid"/>
      </KEYS>
    </TABLE>
    <TABLE NAME="plagiarism_moss_configs" COMMENT="configs for different filename patterns" PREVIOUS="plagiarism_moss" NEXT="plagiarism_moss_tags">
      <FIELDS>
        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true" NEXT="moss"/>
        <FIELD NAME="moss" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="id" NEXT="filepatterns"/>
        <FIELD NAME="filepatterns" TYPE="char" LENGTH="255" NOTNULL="false" SEQUENCE="false" PREVIOUS="moss" NEXT="language"/>
        <FIELD NAME="language" TYPE="char" LENGTH="50" NOTNULL="false" SEQUENCE="false" PREVIOUS="filepatterns"/>
      </FIELDS>
      <KEYS>
        <KEY NAME="primary" TYPE="primary" FIELDS="id" NEXT="moss"/>
        <KEY NAME="moss" TYPE="foreign" FIELDS="moss" REFTABLE="plagiarism_moss" REFFIELDS="id" PREVIOUS="primary"/>
      </KEYS>
    </TABLE>
    <TABLE NAME="plagiarism_moss_tags" COMMENT="tags to identify assignments with same task" PREVIOUS="plagiarism_moss_configs" NEXT="plagiarism_moss_results">
      <FIELDS>
        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true" NEXT="name"/>
        <FIELD NAME="name" TYPE="char" LENGTH="255" NOTNULL="true" SEQUENCE="false" PREVIOUS="id"/>
      </FIELDS>
      <KEYS>
        <KEY NAME="primary" TYPE="primary" FIELDS="id" NEXT="name"/>
        <KEY NAME="name" TYPE="unique" FIELDS="name" PREVIOUS="primary"/>
      </KEYS>
    </TABLE>
    <TABLE NAME="plagiarism_moss_results" COMMENT="place to store moss result" PREVIOUS="plagiarism_moss_tags" NEXT="plagiarism_moss_matchedfiles">
      <FIELDS>
        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true" NEXT="moss"/>
        <FIELD NAME="moss" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="id" NEXT="config"/>
        <FIELD NAME="config" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="moss" NEXT="userid"/>
        <FIELD NAME="userid" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="config" NEXT="pair"/>
        <FIELD NAME="pair" TYPE="int" LENGTH="10" NOTNULL="false" UNSIGNED="true" SEQUENCE="false" PREVIOUS="userid" NEXT="rank"/>
        <FIELD NAME="rank" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" PREVIOUS="pair" NEXT="percentage"/>
```

File: /home/gabri/moodle/db/install.xml

```
55          <FIELD NAME="percentage" TYPE="int" LENGTH="4" NOTNULL="true" UNSIGNED="true"
    SEQUENCE="false" PREVIOUS="rank" NEXT="linesmatched"/>
56          <FIELD NAME="linesmatched" TYPE="int" LENGTH="4" NOTNULL="true" UNSIGNED="true"
    SEQUENCE="false" PREVIOUS="percentage" NEXT="link"/>
57          <FIELD NAME="link" TYPE="char" LENGTH="255" NOTNULL="true" SEQUENCE="false"
    PREVIOUS="linesmatched" NEXT="confirmed"/>
58          <FIELD NAME="confirmed" TYPE="int" LENGTH="4" NOTNULL="true" UNSIGNED="true" DEFAULT="0"
    SEQUENCE="false" PREVIOUS="link" NEXT="confirmer"/>
59          <FIELD NAME="confirmer" TYPE="int" LENGTH="10" NOTNULL="false" UNSIGNED="true"
    SEQUENCE="false" PREVIOUS="confirmed" NEXT="timeconfirmed"/>
60          <FIELD NAME="timeconfirmed" TYPE="int" LENGTH="10" NOTNULL="false" UNSIGNED="true"
    SEQUENCE="false" PREVIOUS="confirmer"/>
61        </FIELDS>
62        <KEYS>
63          <KEY NAME="primary" TYPE="primary" FIELDS="id" NEXT="moss"/>
64          <KEY NAME="moss" TYPE="foreign" FIELDS="moss" REFTABLE="plagiarism_moss" REFFIELDS="id"
    PREVIOUS="primary" NEXT="config"/>
65          <KEY NAME="config" TYPE="foreign" FIELDS="config" REFTABLE="plagiarism_moss_configs"
    REFFIELDS="id" PREVIOUS="moss" NEXT="userid"/>
66          <KEY NAME="userid" TYPE="foreign" FIELDS="userid" REFTABLE="user" REFFIELDS="id"
    PREVIOUS="config" NEXT="confirmer"/>
67          <KEY NAME="confirmer" TYPE="foreign" FIELDS="confirmer" REFTABLE="user" REFFIELDS="id"
    PREVIOUS="userid" NEXT="pair"/>
68          <KEY NAME="pair" TYPE="unique" FIELDS="pair" PREVIOUS="confirmer"/>
69        </KEYS>
70        <INDEXES>
71          <INDEX NAME="rank" UNIQUE="false" FIELDS="rank"/>
72        </INDEXES>
73      </TABLE>
74      <TABLE NAME="plagiarism_moss_matchedfiles" COMMENT="Files matched"
    PREVIOUS="plagiarism_moss_results">
75        <FIELDS>
76          <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true"
    NEXT="result"/>
77          <FIELD NAME="result" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false"
    PREVIOUS="id" NEXT="contenthash"/>
78          <FIELD NAME="contenthash" TYPE="char" LENGTH="40" NOTNULL="true" SEQUENCE="false"
    PREVIOUS="result"/>
79        </FIELDS>
80        <KEYS>
81          <KEY NAME="primary" TYPE="primary" FIELDS="id" NEXT="result"/>
82          <KEY NAME="result" TYPE="foreign" FIELDS="result" REFTABLE="plagiarism_moss_results"
    REFFIELDS="id" PREVIOUS="primary"/>
83        </KEYS>
84        <INDEXES>
85          <INDEX NAME="contenthash" UNIQUE="false" FIELDS="contenthash"/>
86        </INDEXES>
87      </TABLE>
88    </TABLES>
89  </XMLDB>
```

File: /home/gabri/moodle/db/messages.php

```php
<?php
$messageproviders = array (

/// Confirmed or unconfirmed by teachers
    'moss_updates' => array (
    )

);
```

File: /home/gabri/moodle/db/upgrade.php

```php
<?php

function xmldb_plagiarism_moss_upgrade($oldversion=0) {
    global $CFG, $DB, $OUTPUT;

    $dbman = $DB->get_manager();

    if ($oldversion < 2011071500) {

        // Define field sensitivity to be dropped from moss_configs
        $table = new xmldb_table('moss_configs');
        $field = new xmldb_field('sensitivity');

        // Conditionally launch drop field sensitivity
        if ($dbman->field_exists($table, $field)) {
            $dbman->drop_field($table, $field);
        }

        // Define field sensitivity to be added to moss
        $table = new xmldb_table('moss');
        $field = new xmldb_field('sensitivity', XMLDB_TYPE_INTEGER, '10', XMLDB_UNSIGNED, null, null, null, 'tag');

        // Conditionally launch add field sensitivity
        if (!$dbman->field_exists($table, $field)) {
            $dbman->add_field($table, $field);
        }

        // moss savepoint reached
        upgrade_plugin_savepoint(true, 2011071500, 'plagiarism', 'moss');
    }

    if ($oldversion < 2011100700) {
        set_config('maxfilesize', 64 * 1024 * 1024, 'plagiarism_moss');
        // moss savepoint reached
        upgrade_plugin_savepoint(true, 2011100700, 'plagiarism', 'moss');
    }

    if ($oldversion < 2011102000) {

        // Define table moss to be renamed to plagiarism_moss
        $table = new xmldb_table('moss');
        // Launch rename table for moss
        $dbman->rename_table($table, 'plagiarism_moss');

        // Define table moss_configs to be renamed to plagiarism_moss_configs
        $table = new xmldb_table('moss_configs');
        // Launch rename table for moss_configs
        $dbman->rename_table($table, 'plagiarism_moss_configs');

        // Define table moss_tags to be renamed to plagiarism_moss_tags
        $table = new xmldb_table('moss_tags');
        // Launch rename table for moss_tags
        $dbman->rename_table($table, 'plagiarism_moss_tags');

        // Define table moss_results to be renamed to plagiarism_moss_results
        $table = new xmldb_table('moss_results');
        // Launch rename table for moss_results
        $dbman->rename_table($table, 'plagiarism_moss_results');

        // Define table plagiarism_moss_matchedfiles to be renamed to plagiarism_moss_matchedfiles
        $table = new xmldb_table('moss_matched_files');
        // Launch rename table for moss_matched_files
        $dbman->rename_table($table, 'plagiarism_moss_matchedfiles');

        // moss savepoint reached
        upgrade_plugin_savepoint(true, 2011102000, 'plagiarism', 'moss');
    }

    return true;
}
```

File: /home/gabri/moodle/cli/trigger_assignment_events.php

```php
<?php
define('CLI_SCRIPT', true);

require_once(dirname(dirname(dirname(dirname(__FILE__)))) . '/config.php');
require_once($CFG->libdir.'/adminlib.php');
require_once($CFG->libdir.'/clilib.php');      // cli only functions
require_once($CFG->dirroot.'/plagiarism/moss/lib.php');

// now get cli options
$longoptions  = array('help'=>false, 'nodone'=>false);
$shortoptions = array('h'=>'help', 'n'=>'nodone');
list($options, $unrecognized) = cli_get_params($longoptions, $shortoptions);

if ($unrecognized) {
    $cmid = (int)current($unrecognized);
    if ($cmid === 0) {
        $unrecognized = implode("\n  ", $unrecognized);
        cli_error(get_string('cliunknowoption', 'admin', $unrecognized));
    }
}

if ($options['help'] or !isset($cmid)) {
    $help =
"Trigger assessable_file_uploaded and assessable_files_done events of specified assignment.

trigger_assignment_events.php cmid

Options:
-h, --help            Print out this help
-n, --nodone          Do not trigger assessable_files_done event

Example:
\$sudo -u www-data /usr/bin/php plagiarism/moss/cli/trigger_assignment_events.php 1234
";

    echo $help;
    die;
}

$count = moss_trigger_assignment_events($cmid, !$options['nodone']);

if ($count === false) {
    cli_error('Failed!');
} else {
    mtrace("$count submission(s) are found and events are triggered.");
}
```