**Abstract**

El plagio es el uso del trabajo o ideas de otras personas como si fueran de uno mismo, apropiándose el crédito de la creación (Chen et al., 2010).

La detección de plagio en los trabajos académicos es un problema que se hace cada vez más grande, a causa de la gran y creciente cantidad de información que existe en Internet, a la facilidad de encontrarla utilizando navegadores y a la entrega electrónica de los trabajos (Rico Juan et al., 2016).

En el presente trabajo se propondrá un algoritmo de detección de plagios de trabajos prácticos de alumnos en base a un dataset, el cual contendrá trabajos de años anteriores.

Palabras clave

Detección, plagio, algoritmo, documentos

1. **Introducción**

Actualmente, existen diferentes tipos de plagio: plagiar textos solo traduciéndolos, reutilizar ideas sin citar, o incluso comprar trabajos con certificados de garantía de no ser detectados por programas anti plagio (Tripathi, 2009). Esto representa un grave problema para el ambiente académico, ya que se dificulta cada vez más la detección de copias.

*“La mayoría de los tipos de plagio son difíciles de*

*detectar incluso para expertos debido a su sofisticación para enmascarar las ideas o párrafos originales de los que provienen.”*(Rico Juan et al., 2016).

Ante este problema, es necesario un desarrollo constante y sostenido de aplicaciones de detección de plagio cada vez más sofisticadas. El uso de herramientas de detección de plagio proveen un fuerte apoyo al ámbito académico, y es por este motivo que en países desarrollados como, por ejemplo, Estados Unidos, su uso se encuentra ampliamente extendido y aceptado en la comunidad académica (Rico Juan et al., 2016).

Existen dos tipos principales de análisis de plagio: el intrínseco y el extrínseco. El análisis de plagio extrínseco compara un documento candidato con un corpus de documentos referencia, y trata de encontrar similitudes con respecto a los mismos. Por otro lado, el análisis de plagio intrínseco no utiliza ninguna colección de referencia, y trata de determinar si existe plagio a través del análisis en los cambios de estilo dentro del documento (Seaward & Matwin, 2009).

El enfoque de este trabajo es el desarrollo de un algoritmo de detección de plagios que utiliza un análisis extrínseco, utilizando el lenguaje Python y bibliotecas que provee la comunidad. De esta manera, se prosigue a estructurar el artículo de la siguiente manera: en la sección 2 se hablará sobre la base de documentos utilizada para el entrenamiento del algoritmo; en la sección 3 se hablará sobre el entrenamiento del algoritmo para el reconocimiento de plagio en un nuevo documento; en la sección 4 se discutirá sobre las técnicas utilizadas para la detección de plagio; y en la sección 5 se proseguirá con las conclusiones y futuras líneas de trabajo.

1. **Dataset de documentos**

El dataset utilizado consiste en una serie de trabajos realizados por alumnos para una materia universitaria. Los mismos fueron compilados a través de años y las sucesivas entregas de los estudiantes sobre diferentes temas. De esta manera, este dataset nos permitirá detectar plagio entre los mismos trabajos del dataset, o incluso entre nuevos trabajos de la misma materia con respecto a los de años anteriores.

1. **Entrenamiento del algoritmo**

Los documentos del dataset se encuentran en distintos formatos, entre los cuales se destacan .doc, .docx, .ppt, .pptx y .pdf. Los mismos son transformados a texto plano a través de las bibliotecas python–docx, python–pptx, pdfplumber y tika.

Una vez que se obtiene el texto plano de un documento, se lo pasa a un string, y el mismo pasa a ser dividido en párrafos.

Cada párrafo de cada documento será preprocesado, lo que significa que pasará por una tokenización, remoción de signos de puntuación, conversión a minúscula, clasificación POS, remoción de stopwords y stemming . De esta manera, cada párrafo contendrá una matadata, la cual es la lista de tokens resultantes del preprocesamiento sobre el texto real.

El preprocesamiento mencionado me brindará las siguientes ventajas:

* La remoción de stopwords y signos de puntuación me permitirán evadir falsos positivos, ya que ni las stopwords ni los signos de puntuación sumarán en el puntaje final de similitud entre dos documentos.
* El stemming me permitirá llevar todas las palabras hacia su raíz común, lo que me permitirá compararlas con mayor facilidad. Por ejemplo: “camina” y “caminó” son convertidas a “camin” por el Snowball Stemmer de nltk, que es la biblioteca que se utilizara para tal fin.
* La clasificación POS me permitirá verificar que dos palabras iguales hayan sido usadas con el mismo sentido dentro de la oración, evitando así falsos positivos.
* La conversión a minúscula es necesaria, ya que evita falsos negativos en la comparación entre dos palabras cuya única diferencia es el uso de letras mayúscula. Por ejemplo, “Enero” y “enero”.

Las herramientas utilizadas para el procesamiento fueron las siguientes:

* Biblioteca ***nltk***: la misma se utilizó para la tokenización y reconocimiento de stopwords y signos de puntuación .
* Biblioteca ***spacy***: la misma se utilizó parta la clasificación POS de cada uno de los tokens del texto.

De esta forma, se genera un objeto Doc por cada uno de los documentos del dataset. Dicho objeto contendrá el texto plano, así como una lista de objetos Parragraph. Un objeto Parragraph contendrá el texto plano vinculado al mismo y, además, la metadata vinculada a ese texto (recordar que se le llama metadata al texto ya procesado).

En las imágenes 1 y 2 se pueden observar la conformación de los objetos Doc y Parragraph respectivamente:

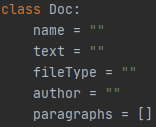


Imagen 1

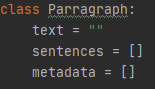


Imagen 2

Realizar dicho preprocesamiento sobre cada uno de los documentos del dataset conlleva un tiempo extenso. Por este motivo, se decidió cachear todo el dataset preprocesado en pequeños archivos, para que así el algoritmo pueda valerse direcmente de estos, y no tener que preprocesar todo el dataset cada vez que va a analizar un nuevo documento.

Para realizar esto, se utilizó la biblioteca ***pickle***, la cual permite crear un dump de un objeto dentro de un archivo. Como se explicó, cada documento del dataset se corresponde con un objeto Doc. De esta manera, se crea un dump de todos los objetos Doc generados luego del preprocesamiento, obteniendo como resultado un pequeño archivo caché por documento del dataset.

1. **El algoritmo de detección de plagio**

El primer paso a realizar es la obtención de un objeto Doc a partir del documento candidato. Luego de esto, se cargan los documentos del dataset a partir del caché. A continuación se itera sobre cada uno de los documentos del dataset (documentos referencia), para así compararlos con el documento candidato.