**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; en la sección 5 se expondrán los testeos realizados al algoritmo para verificar su correcto funcionamiento; y en la sección 6 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:

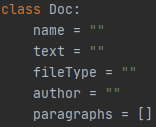


Imagen 1

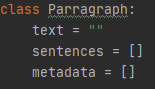


Imagen 2

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 será la obtención de un objeto Doc a partir del documento candidato. Luego de esto, se cargan los documentos(Chen et al., 2010) 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.

De esta forma, el documento candidato se comparará con cada uno de los documentos referencia. Cada párrafo del documento candidato será comparado con todos los párrafos del documento referencia. Esta comparación se realizará mediante una versión modificada del Jaccard Similarity Score, en la cual también se tiene en cuenta el POS tagging (Adam & Suharjito, 2014).

El Jaccard Similarity Score se define, tal como se muestra en la imagen 3, como el cociente entre la cardinalidad de la unión de dos conjuntos y la cardinalidad de la intersección de esos conjuntos(Chen et al., 2010).

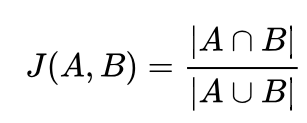


Imagen 3

Para el caso de este trabajo, los conjuntos a comparar a través del Jaccard Similarity Score serán las metadata de los párrafos de los documentos. Se definirá, de esta manera, como la intersección de una metadata A y una metadata B a aquellos tokens que se encuentren en ambos conjuntos y a la vez posee el mismo POS tag. Esto significa, que las palabras no solo serán comparadas por su similitud textual, sino que también se tendrá en cuenta su POS, buscando asegurar que se hayan usado de la misma forma en los dos párrafos. Es importante aclarar que aquellos párrafos que se encuentren citados poseerán un Jaccar Similarity Score de 0.

Este proceso se repite con cada uno de los documentos del dataset. Una vez que se terminó la comparación de todos los párrafos del documento candidato con los párrafos del dataset, se le asignara a cada párrafo el máximo Jaccard Similarity Score que haya obtenido en un comparación, para luego realizar un promedio por párrafos en el documento y obtener el nivel de plagio general del documento candidato con respecto al dataset.

Una vez terminado el análisis, la aplicación mostrará por pantalla cada uno de los párrafos del documento candidato que son sospechosos de plagio, junto con un porcentaje de plagio y el párrafo referencia del cual puede ser una posible copia.

1. **Testing**

**Primer caso de testeo:** se crean dos instancias de objeto Doc a partir del documento “Trabajo Práctico 1 – Hernan Dalle Nogare.docx” y se los compara entre sí. Como resultado, se obtiene un puntaje del 100%, lo cual es correcto, ya que son dos documentos idénticos.

**Segundo caso de testeo:** se utilizan dos oraciones de ejemplo que son totalmente diferentes. En consecuencia, la comparación entre ambas debe dar un puntaje menor al 10%.

* Oración 1: “Rodrigo estaba paseando a su perro, cuando de repente vio un cerdo volar.”
* Oración 2: “A juan le gustan mucho las hamburguesas y las papas fritas.”

Como resultado, se obtiene un score del 0%, lo cual a simple vista se ve correcto.

**Tercer caso de testeo:** aquí se busca verificar que una cita sin referencia correspondiente penaliza el score final. Para esto, tomaremos las siguientes oraciones de ejemplo.

* s1 = “Rodrigo estaba paseando a su perro, cuando de repente vio un cerdo volar.”
* s1CopyWithCitation = “Rodrigo estaba paseando a su perro (Sentence1, 2021).”
* s1CopyWithoutCitation = “Rodrigo estaba paseando a su perro.”

Como resultado, se obtiene:

JaccardScore(s1, s1CopyWithCitation) = 0%

JaccardScore(s1, s1CopyWithoutCitation) = 42.85%

**Cuarto caso de testeo:** se le da al algoritmo una oración, y se le adjunta a ella diferentes formas de referenciar para chequear que estas sean reconocidas.

Ejemplo utilizado:

* **Oración:** “Rodrigo estaba paseando a su perro, cuando de repente vio un cerdo volar.”
* **Cita 1:** “Según [1], ” (formato ISO, se adiciona al principio de la oración).
* **Cita 2:** "(Despotovic-Zrakic et al., 2012)" (formato APA, se adiciona al final de la oración).
* **Cita 3:** "(Anónimo, 2010)" (formato APA, se adiciona al final de la oración).
* **Cita 4:** "(Sabbagh, 2010a)" (formato APA, se adiciona al final de la oración).
* **Cita 5:** "(Sabbagh, 2010b)" (formato APA, se adiciona al final de la oración).
* **Cita 6:** "("Barcelona to Ban Burqa", 2010)" (formato APA, se adiciona al final de la oración).

**Quinto caso de testeo:** en este caso se busca verificar que dos oraciones idénticas en textos distintos aumentan la similitud.

Ejemplos utilizados:

* Doc1 = “Trabajo Práctico 1 - Hernan Dalle Nogare.docx”
* Doc2 = “TP 3 The experience economy (1).docx”
* NuevaOracion = "Esta es una nueva oración para testear si aumenta la similitud entre ambos documentos."

Como resultado se obtiene que, mientras que los dos documentos originales dan un puntaje de similitud del 9.63%, los documentos con la oración agregada dan un puntaje de similitud del 10.91%.

**Sexto caso de testeo:** en este caso se busca verificar que una copia parcial de una oración penaliza menos que una copia total.

Ejemplos utilizados:

* s1 = "Spiderman salvó a la mujer que estaba atrapada bajo los escombros del edificio en llamas."
* s1CopiaParcial = "Ironman salvó al hombre que estaba atrapado dentro del edificio en llamas."

Como resultado, se obtiene:

JaccardScore(s1, s1CopiaParcial) = 36.36%

JaccardScore(s1, s1) = 100%

**Séptimo caso de testeo:** aquí se creará un caso de plagio artificial, modificando algunos detalles poco importantes, y se buscará tener un porcentaje de plagio mayor al 80%.

* Doc1 = “TP 1 - Larga Cola - Campassi Rodrigo .docx”
* Doc2 = “TP 1 - Larga Cola - Campassi Rodrigo(plagio) .docx”

Luego de borrar varios párrafos y agregar otros totalmente diferentes, se obtiene un puntaje de similitud del 82.7%.