Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

TC3002B

Desarrollo de Aplicaciones avanzadas de ciencias Computacionales

Escuela de ingeniería y ciencias

Grupo 301

Evidencia 2

Funcionalidad y Documentación de Detección de plagio y ciencias computacionales

Etapa 2

Eduardo Acosta Hernández	A01375206
Renata Montserrat De Luna Flores	A01750484
Roberto Valdez Jasso	A01746863

Profesores

Raúl Monroy

Jorge Adolfo Ramírez Uresti

Ariel Ortiz Ramírez

Miguel González Mendoza

Fecha de entrega

9 de junio de 2023

Tabla de Contenido

1. Resumen

La detección de plagio y reutilización de texto es de suma importancia en la actualidad, ya que caer en esto es cometer un delito. Es por esto que se han desarrollado diversas técnicas para detectarlo. En este documento, desarrollaremos y describiremos la implementación del software desarrollado para lograr la detección tanto de plagio como de reutilización de texto.
Para lograr esto, fue necesario hacer la diferencia entre plagio y reutilización de texto, encontrando textos completos idénticos al texto original o únicamente pasajes utilizados sin haber sido citados. Conociendo esto, decidimos que el software desarrollado fuera capaz tanto de reconocer plagio como reutilización de texto.
Para esto, realizamos el preprocesamiento de los textos, incluyendo la separación y limpieza mediante lematización, stemming, inteligencia artificial y la eliminación de signos de puntuación de estos, para posteriormente realizar una comparación del texto sospechoso y el texto original, agregando el uso de IA para esta nueva solución. Una vez realizada dicha comparación, obtendremos como resultado el porcentaje de similitud y la ocurrencia entre ambos textos, con lo que podremos determinar si el texto es sospechoso o si el texto es genuino.

2. Abstract

The detection of plagiarism and reuse of text is of the utmost importance today, since falling into this is committing a crime. This is why various techniques have been developed to detect it. In this paper, we will develop and describe the implementation of software developed to achieve both plagiarism and text reuse detection.
To achieve this, it was necessary to make the difference between plagiarism and text reuse, finding complete texts identical to the original text or only passages used without having been cited. Knowing this, we decided that the developed software would be capable of both recognizing plagiarism and text reuse.

For this, we carry out the pre-processing of the texts, including the separation and cleaning by means of lemmatization, stemming, artificial intelligence and the subtraction of punctuation marks of these, to later carry out a comparison of the suspicious text and the original text, adding the use of AI for this new solution. Once this comparison is made, we will obtain as a result the percentage of similarity and the occurrence between both texts, with which we will be able to determine if the suspicious text is text reuse or not.

3. Introducción

3.1 Descripción de la problemática

En la actualidad, las leyes de derechos de autor protegen muchos tipos de diferentes obras como son pinturas, fotografías, ilustraciones, composiciones musicales, grabación de sonido, programas de computadora, libros, poemas, publicaciones de blogs, películas, obras arquitectónicas, obras de teatro y mucho más. Es por esto que la protección de los derechos de autor es de suma importancia.

La detección de plagio es el proceso de localizar instancias de plagio o infracción de derechos de autor dentro de un trabajo o documento. Sin embargo, cuando se da crédito al trabajo o se emplea un entrecomillado, se puede convertir en una referencia. Es por ello que en también se conoce como detección de similitud de contenido o reutilización de texto. Esta situación escala a diversos ámbitos como el desarrollo de software. Existen herramientas de detección de similitud de código permiten determinar qué tan similar es el código fuente de diferentes desarrolladores. Desarrollar herramientas computacionales que abarquen todos los tipos es muy complejo y demandante en tiempo. Por eso, nos enfocaremos en el reto al desarrollo de herramientas para la detección de infracciones en texto.

En el desarrollo de nuestra solución, implementaremos una herramienta para poder identificar la reutilización de texto, que se basará en encontrar si el texto analizado tiene un porcentaje alto de similitud con el texto original, identificando así si el texto sospechoso incurre en la reutilización de texto.

Con este desarrollo, generamos la siguiente hipótesis:

El software desarrollado podrá detectar al menos 90% de precisión la presencia y diferenciación de un texto reutilizado de una fuente original en otro texto plano generado con ayuda de diferentes técnicas como lematización, stemming, inteligencia artificial y similitud coseno, que en un inicio será probado y entrenado con textos totalmente desconocidos al modelo.

3.2 Enfoque de la solución

Tomando en cuenta lo anterior, se desarrollará una herramienta de detección de reutilización de texto, que utiliza diversas técnicas de Aprendizaje Automático, Métodos Cuantitativos y Compiladores, para producir una solución efectiva y eficiente. Para determinar lo anterior, compararemos la solución propuesta con una lectura de textos y documentos desconocidos al modelo.

En la solución del problema, se deberá aplicar técnicas vistas en clase, u otras que sean apropiadas, tales como la subsecuencia común más larga, entre otras. Inicialmente, utilizaremos las lista de palabras claves generadas por los métodos de stemming y lematización en base al resultado dados por la limpieza de los textos presentes en el dataset y por medio de inteligencia artificial generar la tokenización y vectorización de las palabras clave, como también la similitud de coseno para poder dar a conocer que texto es genuino y cual es plagio o sospechoso dependiendo de la misma similitud de un texto con uno o más documentos presentes en un conjunto de documentos desconocidos para el modelo.

4. Trabajo relacionado

4.1 Funciones principales

4.1.1 Resumen de la Arquitectura

La solución se divide en tres etapas principales:

- Etapa de preprocesamiento

  En esta etapa, nos encargamos de la lectura, la limpieza y separación de los textos para que estén listos para ser analizados.

- Etapa de preparación

  En esta etapa, nos encargamos de la generación de palabras clave por medio de los métodos de Stemming y Lematización por medio de reglas y procesos que nos regresan las listas de dicha listas con la finalidad de poder utilizarlas para la

detección de tipo de documento y su comparaciones más adelante.

- Etapa de toma de decisiones

  En esta etapa , nos encargamos de generar la tokenización y vectorización de las listas de palabras clave, previamente generadas por documento con la inteligencia artificial basada en la arquitectura LaBSE con soporte multi-lenguaje y usar el resultado para la toma de decisiones.

  Por otro lado nos encargamos de comparar los resultados de las tapas anteriores para comparar la similitud de ambos textos y, con base en las métricas establecidas con la similitud de coseno, denotando cuales de los textos son genuinos y cuales tiene texto o textos reutilizados de los genuinos, destacando así la diferencia esperada entre los tipo de documentos presentes en los documentos desconocidos.

4.1.2 Especificación de la etapa de preprocesamiento

4.1.2.1 Lectura de textos

En la etapa de preprocesamiento es donde nos encargamos de la lectura de los textos sospechosos y genuinos incluidos en el dataset desde una carpeta local.

4.1.2.2 Limpieza de textos

Posteriormente, realizamos la limpieza de los textos, con el fin de no aumentar el porcentaje de similitud entre dos textos por elementos que se encuentran de manera recurrente en todos o la mayoría de los textos. En nuestro caso, eliminamos los signos de puntuación e ignoramos

tanto las expresiones unidas a un símbolo de arroba como enlaces que comienzan por http o https.

4.1.2.3 Resultado

Después de este procedimiento, obtenemos como resultado en forma lista los textos pre-procesados limpios de acuerdo con las reglas de eliminación de signos de puntuación y enlaces establecidos.

4.1.3 Especificación de la etapa de preparación

4.1.3.1 Textos preprocesados

En esta etapa, tomamos los documentos pre procesados con los paso realizados en la etapa previa

4.1.3.2 Stemming

Con el método de Stemming realizamos una tokenización de las palabras para separarlas por medio de los espacios en blanco. Llevamos a cabo el stemming del texto, con ayuda del LancasterStemmer siendo el más agresivo para encontrar las palabras raíces y realizando de manera automática la limpieza de los textos con funciones anteriores que apoyan a obtener dichas palabras en gracias a este método.

4.1.3.3 Lematización

Por otro lado, además de realizar stemming, también en esta solución, generamos la tokenización de las palabras por medio de los espacios y llevamos la búsqueda de palabras raíces pero a diferencia del anterior método, la lematización la realizamos por medio de reglas, las cuales

la llevamos a cabo para poder encontrar la raíz únicamente de las palabras que son verbos, sustantivos y adjetivos que fueron previamente identificados. Con la lematización, se eliminan los afijos morfológicos, identificando así las raíces de cada una de las palabras. Una vez los textos lematizados, guardamos en una lista las palabras raíces por oración y las palabras raíces por párrafo. Además, generamos una lista que contiene una única raíz, sin repeticiones.

4.1.3.3 Resultados

En esta sección, obtenemos como resultado el texto en minúsculas, listas del texto separado por oraciones, la lista de los textos modificados por Stemming y finalmente una lista de los textos procesados con lematización y limpios de acuerdo con las reglas de eliminación de signos de puntuación y enlaces establecidas, además de una lista con las palabras únicas por frase y por párrafo. Este procedimiento lo llevamos a cabo tanto en los textos genuinos como en los textos sospechosos.

4.1.4 Especificación de la etapa de toma de decisiones

4.1.4.1 Textos procesados

Para esta etapa, utilizamos las listas de textos de las etapas precedentes, tanto las listas de lematización y de stemming para su uso en esta etapa.

4.1.4.2 Inteligencia Artificial LaBSE

En esta sección, tomamos los textos preprocesados y los pasamos a una Inteligencia Artificial basada en la arquitectura LaBSE, la cual es un modelo pre entrenado con 109 lenguajes, entre ellos el español y el inglés, usando el modelo de transformación por sentencia presente , como se apoya por Word Embedding, el cual codifica la palabras, lo

cual apoya al modelo aprender la representación interna del modelo a analizar en el conjunto de entrenamiento que luego se pueda usar para extraer funciones útiles, como el resultado de cada texto tokenizados y vectorizado por el modelo.

4.1.4.3 Similitud de coseno

Para calcular la similitud entre ambos textos, calculamos la similitud de coseno entre dos vectores pertenecientes a un texto genuino y un texto sospechoso. Utilizamos el vector resultante de los textos con el fin denotar por medio de porcentajes cuales de los textos es realmente un texto genuino como también cual es un texto con texto reutilizado dentro del dataset y así destacar cada documento sospechoso con la comparación de cada uno de los textos genuinos pertenecientes al dataset, obteniendo como resultado una matriz con números de 0 a 1, siendo el más alto aquel con el que tiene mayor similitud, el cual nos permite determinar si el texto incurre en reutilización de texto.

4.1.4.4 Curva ROC- AUC

Ya teniendo detectados de los documentos genuinos como los documentos con texto reutilizado, generamos el gráfico de curva ROC-AUC que nos apoya a denotar que tan bueno fue el modelo en distinguir entre los documentos genuinos y sospechosos tomando en cuenta el umbral de detección el cual es del 81% de aceptación para así detectar y clasificar aquellos que realmente son genuinos como también los que son plagio o tienen texto reutilizado, y a la vez, destacar que también funciona el modelo, siendo perfectamente capaz de distinguir entre la clase positiva y la clase negativa, dando asi lo verdaderos positivos del modelo como también sus falso positivos dentro del mismo.

4.1.4.5 Resultados

Posteriormente al cálculo de la similitud de coseno y su curva ROC, obtenemos el valor de la similitud entre el texto genuino y sospechoso, y de qué texto original se hizo plagio en el texto sospechoso.

4.2 Dependencias de la solución

Para la implementación de esta solución, utilizamos las siguientes dependencias:

- re: Nos permite realizar operaciones de coincidencia de expresiones regulares.
- os: Nos permite abrir y leer los documentos de texto plano.
- nltk: Esta librería nos permite realizar análisis de texto por medio del procesamiento de lenguaje natural.
    - tokenize -> word_tokenize: Este módulo permite dividir un string en una lista de substrings de acuerdo con los espacios en blanco y los signos de puntuación.
    - stem -> WordNetLemmatizer: Este módulo nos permite eliminar los afijos de las palabras, conservando únicamente la raíz de la palabra.
    - tag -> pos_tag: Este módulo permite identificar y clasificar el tipo de palabra del que se trata (adjetivo, verbo, sustantivo, entre otros).
- sklearn: Nos permite implementar modelos de aprendizaje máquina y modelos estadísticos.
    - CountVectorizer: Este módulo permite convertir un texto a una matriz de tokens.
    - cosine_similarity: Este módulo permite calcular la similitud de coseno entre dos entradas, X y Y.
    - metrics -> roc_curve: Este módulo permite calcular la curva característica operativa del receptor (ROC), restringida a una clasificación binaria.
    - metrics -> auc: Este módulo permite calcular el área bajo la curva (AUC), usando la regla trapezoidal

- sentence_Transformers
    - SentenceTrasformer: Este módulo permite generar un framework para texto de un “state of the art” y prepararlos para modelos BERT y LaBSE.
- Matplotlib Pyplot: Este módulo nos permite generar gráficos implícitamente generados como un gráfico de MATLAB y representar el gráfico resultado en pantalla.

4.3 Marco experimental

En esta sección describiremos nuestro marco experimental. Para abordar la problemática, decidimos realizar la limpieza de los textos para posteriormente llevar a cabo tanto la lematización como el stemming de los textos, y vectorización de los textos por medio de un modelo pre entrenado de inteligencia artificial (LaBSE).

La limpieza de los textos la realizamos con ayuda de expresiones regulares y reemplazamiento de caracteres. Además, llevamos a cabo la lematización y el Stemming con ayuda del Lancaster Stemmer, con el fin de obtener las palabras raíces de cada texto (este proceso está disponible en https://github.com/Lalcosta/PlagiarismDetectorTeam4.git , accesado el 08 de junio de 2023).

Finalmente, para la toma de decisiones y cálculo de la similitud coseno, utilizamos los textos lematizados y vectorizados con ayuda del modelo de inteligencia artificial LaBse. Además, obtuvimos el área bajo la curva para el análisis de los resultados obtenidos.

4.3.1 Análisis de datasets

Para la elaboración de nuestra solución, utilizamos un dataset conteniendo 15 textos sospechosos y diversos textos genuinos, sin saber cuáles están clasificados como textos genuinos y textos sospechosos, por lo que los textos dentro de nuestro dataset no estarán pre clasificados.

4.3.2 Inteligencia Artificial y Vectorización

Para la implementación de nuestra solución, utilizamos el modelo de inteligencia artificial LaBse. Este modelo nos permite realizar de manera automática la vectorización de los textos previamente lematizados. Logrando un *word embedding* que nos permite una representación de palabras por medio de vectores de números, logrando así capturar el contexto en el que se encuentran las palabras, su similitud semántica y sintáctica. De esta forma, las palabras se pueden relacionar o diferenciar de otras. Esto nos permite, dentro de nuestra solución, lograr identificar si alguna palabra fue reemplazada por un sinónimo o palabra similar, y lograr calcular de una manera más precisa la similitud coseno.

4.3.2 Protocolo de evaluación

Nuestro protocolo de evaluación fue de la siguiente manera. Para iniciar, realizamos la limpieza de los textos. Posteriormente, utilizamos Lancaster Stemmer para encontrar las palabras raíz del texto. A la par, realizamos la lematización de los textos limpios para, de igual manera, encontrar la raíz de las palabras clave del texto. Además, llevamos a cabo tanto la lematización como el stemming pero de las palabras únicas.

Con ayuda de la IA y del modelo LaBse, que es un modelo pre entrenado, realizamos la vectorización de los textos previamente pre procesados con lematización y stemming, y lematización y stemming con palabras únicas.

Después de este proceso, calculamos la similitud de coseno, que mide la similitud entre dos vectores distintos, generados por el modelo LaBse. Esto nos dará un valor entre 0 y 1. Dentro de este rango, definimos que si los resultados eran mayores a 0.81, el texto incurre en reutilización de texto. Un valor menor a este rango, es un texto genuino.

4.4 Análisis de Resultados

Como se mencionó en apartados anteriores, se utilizaron cuatro métodos de preprocesamiento los cuales son: lematización, lematización de palabras únicas, stemming y stemming de palabras únicas.

El programa se ejecutó para cada uno de los textos preprocesados tanto con los documentos genuinos como con los documentos sospechosos.

Para la evaluación de resultados se utilizó la curva ROC y la medida AUC, las cuales son herramientas que nos permiten tener una evaluación del desempeño de nuestro clasificador.

La curva ROC es una herramienta visual donde la posición de perfección se encuentra en el punto (0,1) de la tabla, mientras que la posición de aleatoriedad se encuentra en la diagonal. El AUC proporciona una medida de rendimiento cuantitativa donde el valor 1 es perfección, mientras que el valor 0.5 es aleatoriedad.

Estas medidas toman en cuenta la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos.

Los resultados obtenidos son los siguientes:

1. Lematización - AUC = 0.98
2. Stemming de palabras únicas - AUC = 0.96
3. Stemming - AUC - 0.91
4. Lematización con palabras únicas AUC - 0.91

La evidencia de dichos resultados se muestra en las siguientes imágenes

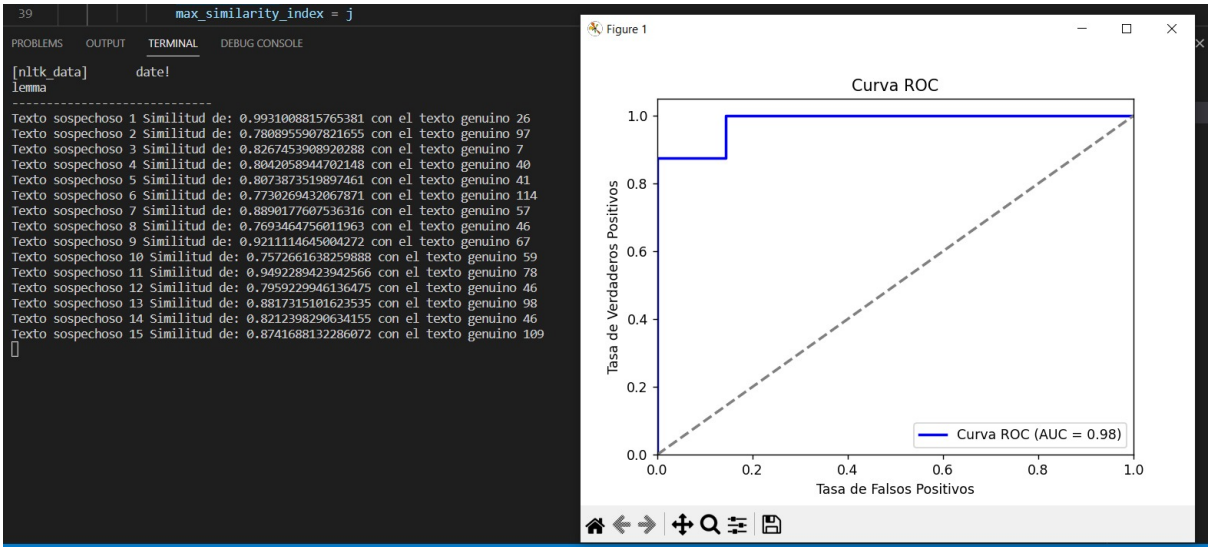

Imagen 1: Resultado de similitud de coseno, curva ROC y AUC de lematización

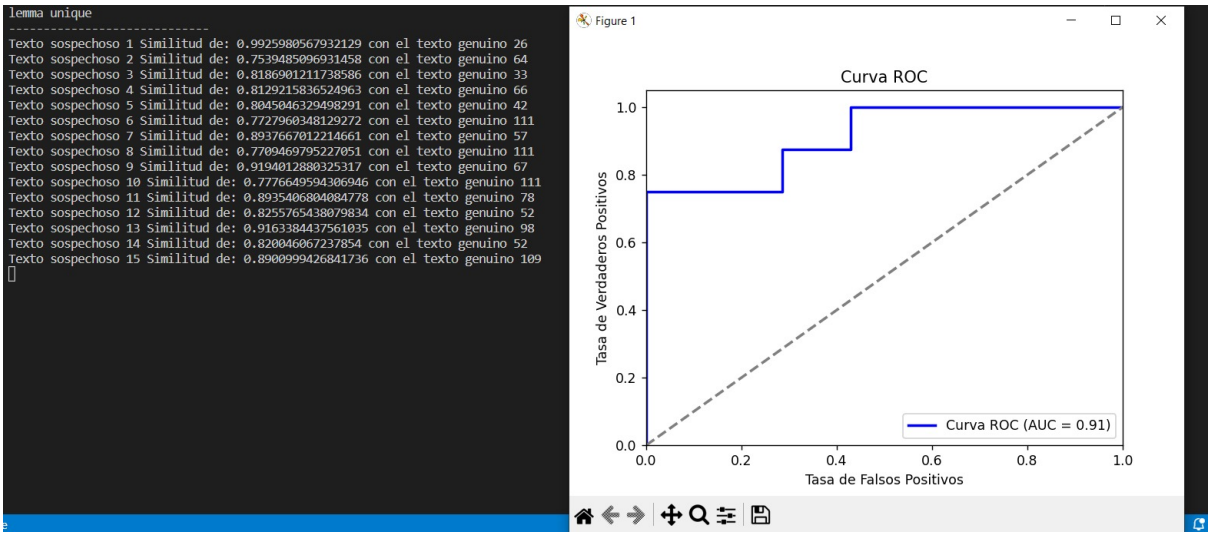

Imagen 2 : Resultado de similitud de coseno, curva ROC y AUC de Lematización de palabras únicas

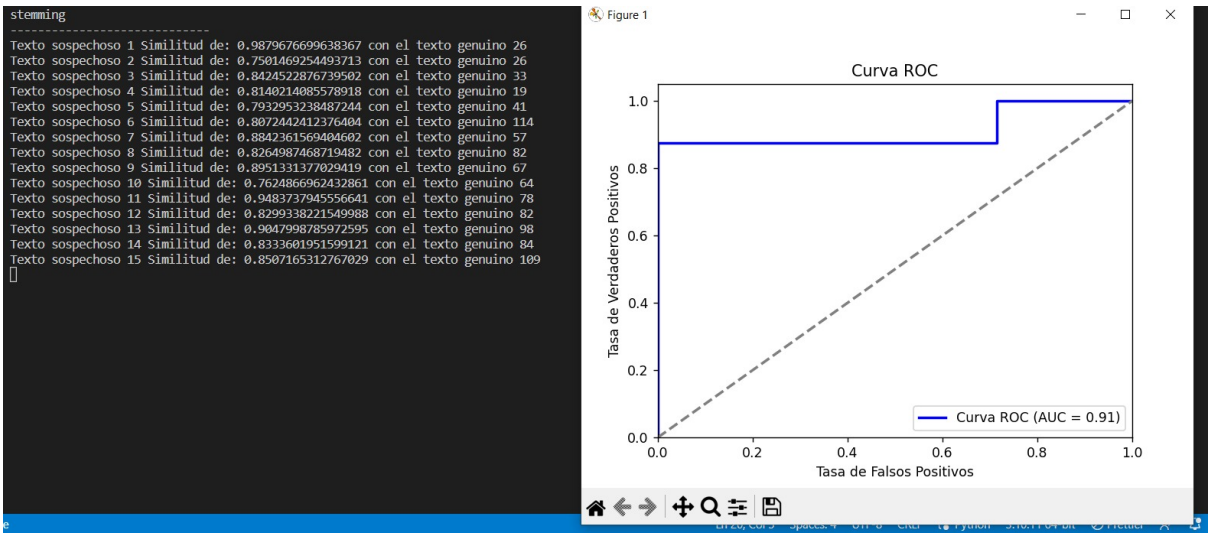

Imagen 3: Resultado de similitud de coseno, curva ROC y AUC de Stemming

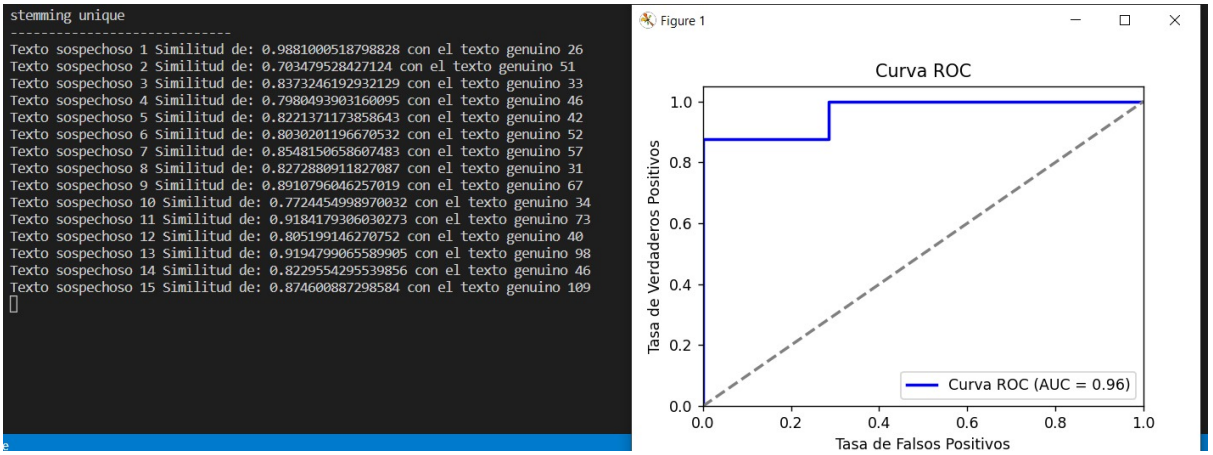

Imagen 4: Resultado de similitud de coseno, curva ROC y AUC de stemming de palabras únicas

5. Conclusiones

En esta nueva etapa, implementamos mejoras y modificaciones en nuestro modelo haciendo uso de la inteligencia artificial, específicamente de un modelo de tipo BERT, el cual funciona para vectorizar textos de una forma más exacta tomando en cuenta el contexto.

Cabe destacar que las similitudes de coseno son relativamente elevadas debido a que todos los textos genuinos y sospechosos hablan sobre plagio, sin embargo el modelo BERT es capaz de ser específico con la vectorización, permitiéndonos observar que los plagios se encuentran por encima del umbral de 0.81, tomando esa medida para realizar la toma de decisión al momento de definir un documento como plagio.

Finalmente como se pudo observar en los resultados nuestro clasificador tuvo un mejor desempeño con la implementación de la lematización, obteniendo un AUC de 0.98 lo cual nos dice que es un buen clasificador debido a su cercanía con el número 1, por lo tanto se decidió por este método de preprocesamiento de texto.

6. Referencias

Joshi, S., & Banerjee, I. (2023). *Ngram-LSTM Open Rate Prediction Model (NLORP) and Error_accuracy@C metric: Simple effective, and easy to implement approach to predict open rates for marketing email.*

Toporkov, O., & Agerri, R. (2023). *On the Role of Morphological Information for Contextual Lemmatization.*

Srivastava, R. P. (2023). A New Measure of Similarity in Textual Analysis: Vector Similarity Metric versus Cosine Similarity Metric. *Journal of Emerging Technologies in Accounting, 20(1), 77–90.* *https://0-doi-org.biblioteca-ils.tec.mx/10.2308/JETA-2021-043*

Shah, J. N., Shah, J., Baral, G., Baral, R., & Shah, J. (2021). Types of plagiarism and how to avoid misconduct: Pros and cons of plagiarism detection tools in research writing and publication. *Nepal Journal of Obstetrics & Gynaecology, 16(2), 3–18.* *https://0-doi-org.biblioteca-ils.tec.mx/10.3126/njog.v16i2.42085*

Tipos-de-Plagio. (s. f.). https://diegoliveros.github.io/PactoHonor/tipos-de-plagio.html

Turkel, W. J. (2012, 17 julio). *Palabras clave en contexto (usando n-grams) con Python*. Programming Historian. https://programminghistorian.org/es/lecciones/palabras-clave-en-contexto-n-grams

Stemming and lemmatization. (s. f.). https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html