Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México

Escuela de Ingeniería y Ciencias

Desarrollo de Aplicaciones Avanzadas de Ciencias Computacionales (Gpo 301)

Evidencia 2: Modelo Mejorado

Alumnos:

Jorge Chávez Badillo	A01749448
Amy Murakami Tsutsumi	A01750185
Ariadna Jocelyn Guzmán Jiménez	A01749373

Profesores:

Raúl Monroy Borja
Ariel Ortíz Ramírez
Jorge Adolfo Ramírez Uresti
Miguel González Mendoza

Fecha: 9 de junio de 2022

Índice

Problemática

El plagio, es un fenómeno que se está volviendo cada vez más común y generalizado en la era de las nuevas tecnologías e información digital. Gracias al acceso rápido y fácil a una amplia variedad de contenidos en Internet, es aún más fácil para algunas personas copiar y plagiar el trabajo de otros sin consecuencias.
El impacto del plagio es significativo a nivel individual, social y académico. Enfocándonos en este último, el plagio amenaza la credibilidad y calidad de los productos intelectuales.
Las nuevas tecnologías como el Internet han facilitado la lucha contra el plagio al proporcionar herramientas para copiar, pegar y modificar fácilmente el trabajo de otras personas. La disponibilidad de información en Internet dificulta rastrear el origen de un texto o idea y aumenta la probabilidad de plagio.
Para abordar este problema, se han tomado medidas contra el plagio, como el uso de software de detección de similitudes y la concientización sobre la importancia de la integridad académica y los derechos de autor, donde mediante el presente documento se presentará una solución y análisis para debatir contra este problema.

Solución Propuesta

Repositorio de Github: E3-Deteccion-de-plagio-TC3002B.301 (github.com)
Notebook con el código implementado: Evidencia 2 (Google Colab)

Mediante la función propuesta, se busca poder realizar una detección de plagio comparando un archivo sospechoso con cada uno de los archivos genuinos existentes en la base de datos. Por lo tanto, se mejoró el sistema implementado en la evidencia 1 por medio de encoders utilizando el transformer BERT (Bidirectional Encoder Representations for Transformers). De esta manera, los documentos se convierten en secuencias de tokens, se codifican y se realiza una comparación de similitud a través de la distancia coseno para obtener los documentos que sí son plagio y los que no.

Para ello, es importante mencionar los siguientes conceptos importantes:

- **BERT:** (Bidirectional Encoder Representations Transformers) es un modelo de lenguaje que utiliza la arquitectura de redes transformer para obtener las relaciones contextuales entre palabras de un documento.
- **Token:** Unidad de texto que utiliza el modelo BERT para poder procesar el lenguaje. Puede ser un carácter, una subpalabra o una palabra dependiendo del tokenizador seleccionado.
- **Cosine similarity:** Medida de similitud entre dos vectores en un espacio multidimensional.
- **AUC:** Área Bajo la Curva. Métrica de rendimiento utilizada en problemas de clasificación.
- **ROC:** Curva ROC. Representación gráfica del rendimiento de un modelo de clasificación binaria a medida que se varía el umbral de clasificación.

Como primer paso de implementación, se realizó un procesamiento de datos, donde a través de algunas librerías de python como nltk para lematización y re para expresiones regulares, se transforma cada palabra del texto en su forma raíz y se eliminan los caracteres especiales existentes para lograr así un mejor funcionamiento del modelo de detección de plagio.
Con ayuda del algoritmo BERT mencionado anteriormente, se realiza una comparación entre textos a través de vectores, donde por medio de un umbral de entrada, tenemos los resultados de detección de plagio basados en los siguientes valores:

- Valor de 1: Si el score es igual a 1, significa que los dos vectores son idénticos y están en la misma dirección. Esto indica una similitud máxima entre los vectores, ya que el ángulo entre ellos es cero grados.

- Valores cercanos a 1: Cuando el score se acerca a 1, indica una alta similitud entre los vectores. A medida que el valor se acerca a 1, el ángulo entre los vectores se acerca a cero grados y la similitud aumenta.

- Valor de 0: Si el score es igual a 0, significa que los vectores son ortogonales entre sí, es decir, están en ángulo recto. Esto indica que no hay similitud entre los vectores.

- Valores cercanos a 0: A medida que el score se acerca a 0, indica que los vectores tienen una similitud menor. Cuanto más cercano a cero sea el valor, mayor será el ángulo entre los vectores y menor será la similitud.

- Valor de -1: Si el score es igual a -1, significa que los vectores son opuestos entre sí. Están en direcciones completamente opuestas, lo que indica una similitud negativa máxima.

Finalmente, se realiza una evaluación de los resultados para poder hacer la clasificación de los archivos, considerando que los valores más cercanos a 1 son los más probables de plagio. Después de ello, mediante esta evaluación se almacena una lista con las predicciones arrojadas por el modelo considerando etiquetas 0 y 1, donde:

- 1: Plagio (Alta similitud)
- 0: Genuino (Baja similitud)

Con ello, se realiza una comparación de las predicciones con los valores reales (proporcionados por el profesor) para así tener las métricas del modelo verificando su efectividad.

```python
def plagiarism_analysis(sus_name, sus_text, database_vector, threshold):
  '''
  Función que regresa los resultados de la detección de plagio, señalando
  los archivos comparados y el nivel de similitud obtenido.
  '''

  sus_vect = vectorize_cosine(sus_text)

  # Similitud Coseno
  database_vector['similarity'] = database_vector['vectors'].apply(
      lambda x: cosine_similarity(sus_vect, x))
  database_vector['similarity'] = database_vector['similarity'].apply(
      lambda x: x[0][0])

 # Top 5 Archivos similares
  similar_files = database_vector.sort_values(by = 'similarity',
                                              ascending = False)[0:4]
  df_result = similar_files[['name',
                             'content',
                             'similarity']].reset_index(drop = True)

  similarity_score = df_result.iloc[0]['similarity']
  most_similar_f_name = df_result.iloc[0]['name']
  most_similar_f_content = df_result.iloc[0]['content']
  status = detect_plagiarism(similarity_score, threshold)


  plagiarism_analysis = [sus_name,
                         most_similar_f_name,
                         status,
                         similarity_score,
                         most_similar_f_content,
                         sus_text]

  return plagiarism_analysis
```

```python
def sus_analysis(path_sus):
  '''
  Función que compara todos los archivos genuinos con todos los archivos
  sospechosos. Muestra el archivo sospechoso, el estado, la similitud y
  el archivo que tiene mayor similitud.
  '''
  df_sus = create_frame(path_sus)

  for row in range(len(df_sus)):
    name_sus = df_sus.iloc[row]['name']
    content_sus = df_sus.iloc[row]['content']
    analysis = plagiarism_analysis(name_sus, content_sus, vector_genuinos, 0.93)
    print("="*30)
    print('Nombre del archivo sospechoso:', analysis[0])
    print('Nombre del archivo genuino similar:', analysis[1])
    print('Status:', analysis[2])
    print('Score de similitud:', analysis[3])
    prediction_values.append(analysis[2])
    prediction_score.append(analysis[3])
  return
```

Funciones

Librerías Implementadas:

- **sklearn:** Se utiliza para realizar tareas relacionadas con el aprendizaje automático, en este caso la extracción de palabras y el cálculo de la distancia de coseno.
- **os:** Sirve para realizar la gestión de archivos y el directorio donde se encuentra la base de datos con textos genuinos y sospechosos.
- matplotlib: Para la visualización de las métricas a través de gráficos.
- **pandas:** Se utiliza para la manipulación de datos a través de DataFrames.
- **tqdm:** Permite visualizar el progreso de las tareas en tiempo real.
- **numpy:** Ayuda a la creación de matrices y arreglos para su manipulación y operaciones matemáticas.
- **keras_preprocessing:** Permite la normalización de datos, codificación de variables categóricas y la división de conjuntos de datos en entrenamiento y prueba.
- **transformers:** Se centra en el procesamiento de lenguaje natural, en este caso, es la biblioteca más importante nos brinda el modelo BERT para su implementación.
- **re:** Para la implementación de expresiones regulares en la limpieza de texto.
- **nltk:** Se utiliza para implementar la lematización en los textos.
- **torch:** Utilizado para construir y entrenar deep neural networks, principalmente utilizado para el proceso de lenguaje natural.

Funciones Principales:

- **clean_data:** Función que se utiliza para pre-procesar los datos a través de la eliminación de los caracteres especiales del texto.
- **lemm_parrafo:** Función que aplica la técnica de lematización en los párrafos para poder reducir las palabras a su forma diccionario.
- **create_frame:** Función que crea el data frame que contiene el nombre del archivo y contenido para la carpeta de archivos (path) seleccionada.
- **vectorize_text:** Función que genera la representación vectorial de un documento. Recibe como parámetros el tokenizador, el modelo, el documento y el número máximo de tokens.

- **vectorize_database:** Función que genera una base de datos que contiene los vectores de todos los documentos.
- **vectorize_cosine:** Función que crea un vector de acuerdo al texto de entrada para ajustarlo y poder calcular la similitud coseno.
- **detect_plagiarism:** Función que determina si el texto es plagio o no de acuerdo a un cierto umbral.
- **plagiarism_analysis:** Función que regresa los resultados de la detección de plagio. Regresa el archivo sospechoso, el archivo que tiene mayor similitud, el estado (es plagio o no) y el nivel de similitud obtenido.
- **sus_analysis:** Función que compara todos los archivos genuinos con todos los archivos sospechosos. Muestra el archivo sospechoso, el estado, la similitud y el archivo que tiene mayor similitud.
- **metricas:** Función en la que se calculan las métricas de la matriz de confusión, se despliega la proporción de la predicción correcta de positivos (precision), la proporción de la predicción correcta de positivos reales (recall) y la proporción de la predicción correcta de los negativos reales (specificity).
- **plot_roc_curve:** Función que genera la gráfica de la curva AUC-ROC (área bajo la curva - característica operativa del receptor).

Dependencia

- La función para realizar la lematización depende de la librería nltk para poder reducir las palabras a su forma diccionario.
- La función vectorize_text depende de la librería torch y transformers para poder realizar la representación vectorial de un documento e implementar BERT.
- La función plagiarism_detection depende de todas las funciones implementadas previas a ella para poder realizar correctamente su funcionamiento.
- El funcionamiento del código se basa en la carpeta autorizada para almacenamiento de archivos genuinos y sospechosos ubicada en Google Drive.
- El entorno de ejecución dónde se realizó el código (Google Colab) depende de internet para realizar correctamente su función.

Consideraciones

- La base de datos donde se encuentran todos los archivos evaluados está en la siguiente carpeta: Datasets

Resultados

En esta evidencia, los resultados contienen principalmente un algoritmo para la detección de plagio, donde con ayuda de la herramienta de BERT, es posible obtener mejores resultados en comparación con la evidencia 1, pues ya con este se puede desarrollar un modelo más eficaz y complejo; de igual manera, es importante mencionar que la fase anterior a la implementación de BERT es de suma importancia ya que esta se encarga del preprocesamiento de los textos, donde con ayuda de expresiones regulares es posible eliminar caracteres especiales y también, al utilizar lematización es posible vincular las palabras de los textos a palabras similares en una sola, aportando así contexto a las palabras para finalmente obtener un mejor resultado al utilizar el modelo de detección de plagio, además de ello, los resultados contienen una serie de métricas con las cuales es posible evaluar el modelo y se muestran también las pruebas unitarias de todos los archivos sospechosos al ser analizados.

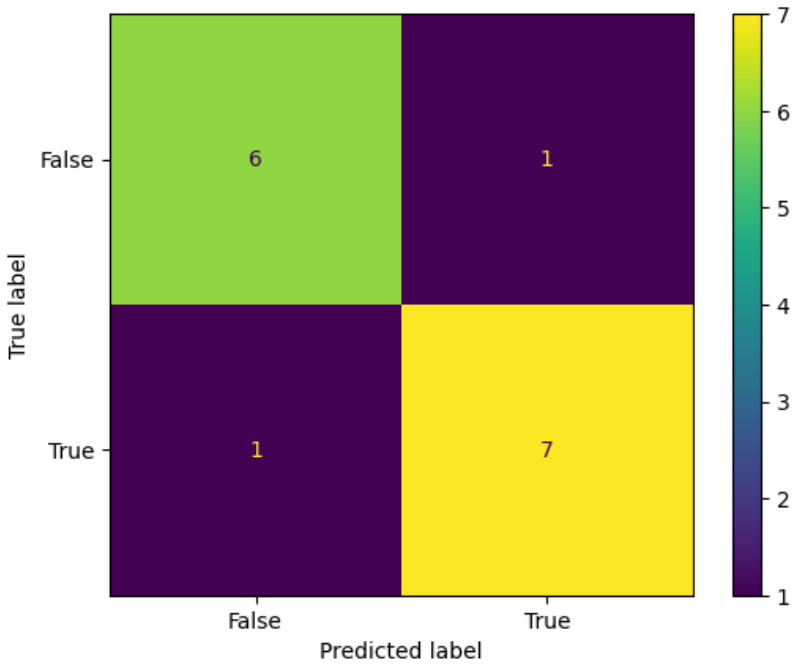
Análisis de Similitud con el Algoritmo BERT

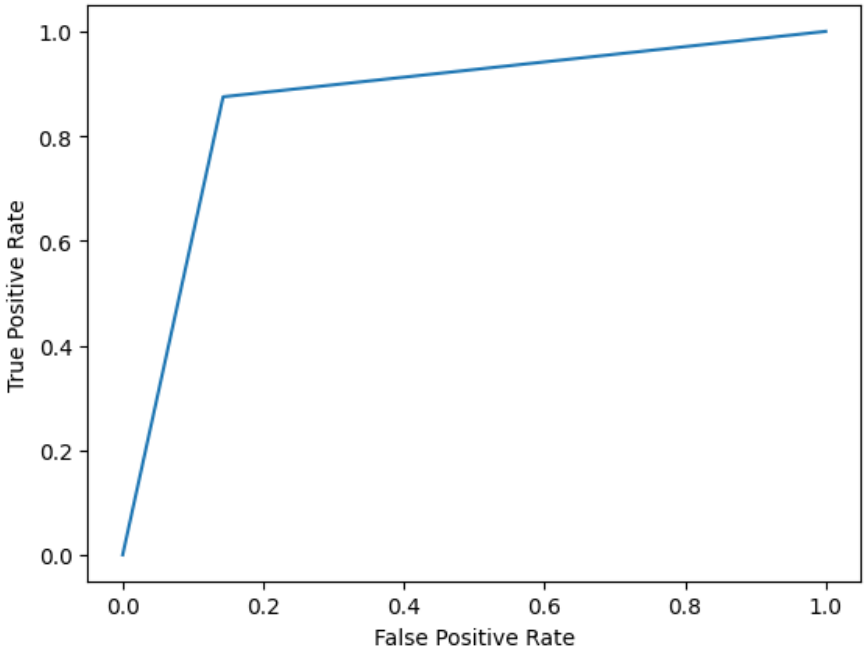
```
sus_analysis(sospechosos)
```

```
32 s
==============================
Nombre del archivo sospechoso: FID-01.txt
Nombre del archivo genuino similar: org-325.txt
Status: 1
Score de similitud: 0.9951241
==============================
Nombre del archivo sospechoso: FID-02.txt
Nombre del archivo genuino similar: org-402.txt
Status: 0
Score de similitud: 0.92035645
==============================
Nombre del archivo sospechoso: FID-03.txt
Nombre del archivo genuino similar: org-306.txt
Status: 0
Score de similitud: 0.91379523
==============================
Nombre del archivo sospechoso: FID-04.txt
Nombre del archivo genuino similar: org-339.txt
Status: 0
Score de similitud: 0.9133811
==============================
Nombre del archivo sospechoso: FID-05.txt
Nombre del archivo genuino similar: org-340.txt
Status: 1
Score de similitud: 0.9728546
==============================
Nombre del archivo sospechoso: FID-06.txt
Nombre del archivo genuino similar: org-339.txt
Status: 1
Score de similitud: 0.9411098
==============================
Nombre del archivo sospechoso: FID-07.txt
Nombre del archivo genuino similar: org-418.txt
Status: 1
Score de similitud: 0.93230796
==============================
Nombre del archivo sospechoso: FID-08.txt
Nombre del archivo genuino similar: org-351.txt
Status: 0
Score de similitud: 0.929445
```

```
32 s
==============================
Nombre del archivo sospechoso: FID-09.txt
Nombre del archivo genuino similar: org-366.txt
Status: 1
Score de similitud: 0.9310549
==============================
Nombre del archivo sospechoso: FID-10.txt
Nombre del archivo genuino similar: org-354.txt
Status: 0
Score de similitud: 0.9262341
==============================
Nombre del archivo sospechoso: FID-11.txt
Nombre del archivo genuino similar: org-372.txt
Status: 1
Score de similitud: 0.94530517
==============================
Nombre del archivo sospechoso: FID-12.txt
Nombre del archivo genuino similar: org-339.txt
Status: 0
Score de similitud: 0.9088948
==============================
Nombre del archivo sospechoso: FID-13.txt
Nombre del archivo genuino similar: org-396.txt
Status: 1
Score de similitud: 0.95588624
==============================
Nombre del archivo sospechoso: FID-14.txt
Nombre del archivo genuino similar: org-380.txt
Status: 0
Score de similitud: 0.9196389
==============================
Nombre del archivo sospechoso: FID-15.txt
Nombre del archivo genuino similar: org-400.txt
Status: 1
Score de similitud: 0.94353676
```

```
Precision: 0.8571
Recall: 0.8571
Specificity: 0.8750
Accuracy Score: 0.8667
F1 Score : 0.8750
Confusion Matrix:
[[6 1]
 [1 7]]
```





Ventajas y Desventajas con la Evidencia 1

Evidencia 1			Evidencia 2		
Técnica	**Ventajas**	**Desventajas**	**Técnica**	**Ventajas**	**Desventajas**
Stemming	-Reduce las palabras a su raíz común -Mejora la eficiencia para la comparación de textos	-No considera los contextos de las palabras -Puede generar errores en la eliminación de sufijos o realizar prefijos incorrectamente	**Lematización**	-Conserva el significado de las palabras al reducirlas en su forma base -Reduce el ruido de comparación de textos	-Puede perder información gramatical como tiempos verbales
N-gramas	-Tiene mejor conocimiento de la estructura y coherencia del texto -Puede detectar el plagio incluso con reordenamientos o adiciones en el texto	-Tiene mayor complejidad computacional -No considera el contexto semántico de las palabras	**Expresiones regulares**	-Permite la búsqueda de patrones específicos de plagio -Adaptable a diferentes contextos	-No considera el significado semántico de las palabras
Distancia de coseno	-No depende del tamaño del texto -Da una comparación rápida y eficiente entre vectores	-No considera el significado semántico de las palabras, únicamente las características vectoriales.	**BERT con distancia coseno**	-Proporciona una detección más precisa y contextual del plagio -Captura el contexto y significado de las palabras en	-Puede requerir una gran cantidad de datos

				el texto	

Conclusiones

En conclusión, el análisis de detección de plagio utilizando la lematización y BERT en Python es una estrategia más efectiva para identificar similitudes y posibles casos de plagio en documentos o textos. La lematización permite reducir las palabras a su forma base, lo que ayuda a agrupar términos similares y minimizar las variaciones léxicas. Por otro lado, BERT es un modelo de lenguaje pre entrenado que captura el contexto y las relaciones semánticas entre las palabras.

A comparación de la evidencia 1, el combinar la lematización con BERT este entregable marcó una diferencia significativa, ya que se logró un enfoque más preciso para detectar el plagio gracias a una comprensión más profunda del significado de las palabras y las estructuras gramaticales. Esto permite identificar similitudes incluso si las palabras han sido modificadas o reorganizadas en cierta medida.

Finalmente, dada nuestra implementación, se considera como plan a futuro poder agregar más funciones que aporten a nuestro modelo mayor valor agregado. Por ejemplo, considerar la librería de Python *langdetect* para poder detectar el plagio en textos traducidos, considerando que este también es un tipo de plagio que se encuentra comúnmente.