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 1. Análisis de similitud empleando Inteligencia Artificial

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: 30 de mayo 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 1 - Análisis de Similitud

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, para así poder identificar la existencia de cierto grado de similitud entre los archivos para finalmente poder clasificar el archivo como Plagio (0) o Genuino (1).

Para ello, es importante mencionar los siguientes conceptos importantes:

- **stemming:** Técnica utilizada en procesamiento de lenguaje natural que consiste en reducir las palabras a su forma base o raíz.
- **n-gram:** Secuencia continua de n elementos tomados de una muestra de texto.

- **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 inicio de implementación, se realizó un preprocesamiento de datos, donde a través de diferentes funciones globales y locales, se obtienen las palabras sin repeticiones que existen en el texto así como su reducción a su raíz utilizando la herramienta.
Después de ello, se hace una comparación con unigramas y bigramas los cuales a través de las secuencias que arrojen se realiza una comparación de detección de plagio a través de la similitud de coseno, donde es importante considerar lo siguiente para su interpretación:

- Valor de 1: Si la distancia de coseno 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 la distancia de coseno 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 la distancia de coseno 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 la distancia de coseno 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 la distancia de coseno 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 considerando un *target* de 0.7 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:

- 0: Plagio (Alta similitud)
- 1: 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.

```
files = []
distance = []

# Función para detectar el plagio usando la similitud de coseno
def plag_detection(sos_text):
   with open(sospechosos + sos_text, 'r') as file:
      datasos = file.read().rstrip()
      datasos_str = stemm_parrafo(datasos)
      unisos = crear_ngram(datasos_str, 1)
   for pathgen in os.listdir(genuinos):
      with open(genuinos + pathgen, 'r') as file:
        datagen = file.read().rstrip()
        datagen_str = stemm_parrafo(datagen)
        unigen = crear_ngram(datagen_str, 1)
      files.append(pathgen + ' ' + sos_text)
      distance.append(distancia(unigen, unisos, 1)[0][1])
   bothfiles = files[distance.index(max(distance))]
   bothfiles = bothfiles.split(' ')
   genfile = bothfiles[0]
   sosfile = bothfiles[1]
   if max(distance) > 0.7: #target
    print("El archivo {} tiene alta similitud con el archivo genuino {}. Similitud: {}".format(sosfile,
genfile, max(distance)))
```

```
true_labels = []

for i in name_sos:
  if int(i[4:6]) % 2 == 0:
    true_labels.append(1)
  else:
    true_labels.append(0)
# Función para detectar el plagio usando la similitud de coseno para todos los archivos sospechosos
def all_plag_detection(n):
  pred_labels = []
  files = []
  distance = []
  for pathsos in name_sos:
    with open(sospechosos + pathsos, 'r') as file:
      datasos = file.read().rstrip()
      datasos_str = stemm_parrafo(datasos)
      unisos = crear_ngram(datasos_str, n)

    for pathgen in name_gen:
      with open(genuinos + pathgen, 'r') as file:
        datagen = file.read().rstrip()
        datagen_str = stemm_parrafo(datagen)
        unigen = crear_ngram(datagen_str, n)
      distance.append(distancia(unigen, unisos, n)[0][1])
      if max(distance) > 0.7: # Target
        files.append('P' + ' ' + pathgen + ' ' + pathsos)
      else:
        files.append('NP' + ' ' + pathgen + ' ' + pathsos)
    bothfiles = files[distance.index(max(distance))]
    bothfiles = bothfiles.split(' ')
    label = bothfiles[0]
    genfile = bothfiles[1]
    sosfile = bothfiles[2]
    #print(files)
    if max(distance) > 0.7: # Target
      pred_labels.append(0)
      print("El archivo {} tiene alta similitud con el archivo genuino {}, Similitud: {:.4f}, Status:
{}".format(sosfile, genfile, max(distance), '0'))
    else:
      pred_labels.append(1)
      print("El archivo {} tiene baja similitud con el archivo genuino {}, Similitud: {:.4f}, Status:
{}".format(sosfile, genfile, max(distance), '1'))

    files = []
    distance = []
  print("Prediction:\n", pred_labels)
  return pred_labels
```

Funciones

Librerías Implementadas:

- **nltk:** Se utiliza para el procesamiento del lenguaje natural, en caso de esta evidencia para la implementación de *stemming* y la creación de *eneagramas.*
- **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.

Funciones Principales:

- **stemm_parrafo:** Función que realiza el stemming de un documento utilizando el algoritmo de Porter que reduce todas las palabras a su forma raíz. Esta regresa el documento con todas las palabras en su forma más básica en un string.
- **vectorizar:** Función que recibe como parámetros dos n-gramas y una variable n que indica el tipo de n-grama. Esta regresa un array que contiene dos listas (una para cada n-grama) con las veces que se repite cada n elementos en los textos.
- **distancia:** Calcula la distancia entre los párrafos utilizando la distancia de coseno que es una medida de similitud entre dos vectores distintos de cero. Esta medida devuelve un valor entre -1 y 1, donde el valor igual a 1 significa una similitud total entre los dos vectores; un valor igual a cero indica que no hay similitud entre los vectores y un valor igual a -1 significa que los vectores son opuestos entre sí.
- **crear_ngram:** Función que recibe como parámetros un texto y una variable n que indica el tipo de n-grama a realizar. Devuelve una lista con las palabras que conforman el n-grama dividido por comas entre cada n elementos.
- **plag_detection:** Función que recibe un archivo sospechoso como entrada y evalúa con cuál de los archivos genuinos tiene más coincidencia.
- **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).
- **all_plag_detection:** Función que compara todos los archivos genuinos con todos los archivos sospechosos para finalmente encontrar la relación de menor cantidad de ángulo entre cada uno de ellos para definir su similitud.

Dependencia

- La función de detección de similitud entre textos (all_plag_detection y plag_detection) dependen de todos los procesos nlk y sklearn para poder realizar un cálculo preciso y correcto.
- 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

Los resultados contienen dos procedimientos, el primero que contiene el análisis de similitud utilizando unigramas y el segundo que contiene el análisis de similitud realizado utilizando bigramas. Dentro de estos procedimientos se encuentran las pruebas unitarias, las métricas del modelo y una gráfica con la curva AUC-ROC.

Análisis de Similitud Utilizando Unigramas

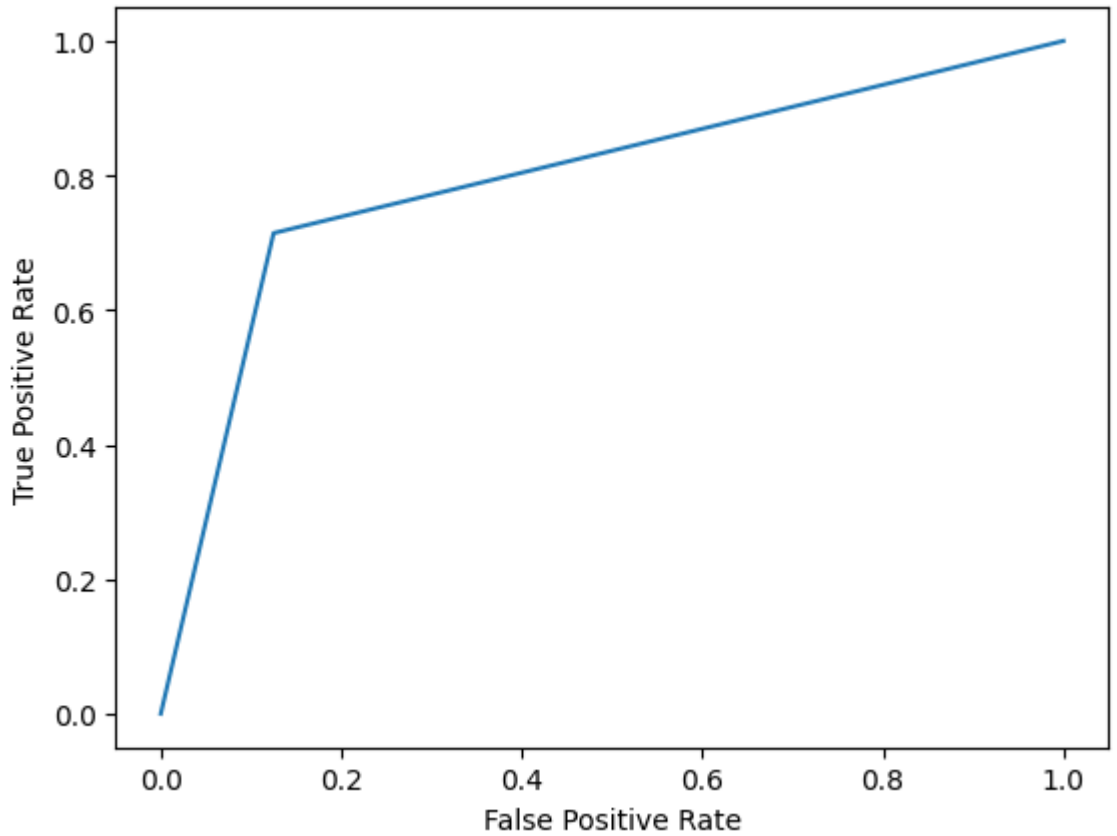
```
El archivo FID-01.txt tiene alta similitud con el archivo genuino org-325.txt, Similitud: 0.9652, Status: 0
El archivo FID-02.txt tiene alta similitud con el archivo genuino org-325.txt, Similitud: 0.7566, Status: 0
El archivo FID-03.txt tiene baja similitud con el archivo genuino org-339.txt, Similitud: 0.6794, Status: 1
El archivo FID-04.txt tiene baja similitud con el archivo genuino org-413.txt, Similitud: 0.6304, Status: 1
El archivo FID-05.txt tiene alta similitud con el archivo genuino org-340.txt, Similitud: 0.8555, Status: 0
El archivo FID-06.txt tiene baja similitud con el archivo genuino org-413.txt, Similitud: 0.6681, Status: 1
El archivo FID-07.txt tiene alta similitud con el archivo genuino org-356.txt, Similitud: 0.8366, Status: 0
El archivo FID-08.txt tiene baja similitud con el archivo genuino org-331.txt, Similitud: 0.5089, Status: 1
El archivo FID-09.txt tiene alta similitud con el archivo genuino org-388.txt, Similitud: 0.7767, Status: 0
El archivo FID-10.txt tiene baja similitud con el archivo genuino org-352.txt, Similitud: 0.5929, Status: 1
El archivo FID-11.txt tiene alta similitud con el archivo genuino org-372.txt, Similitud: 0.9056, Status: 0
El archivo FID-12.txt tiene baja similitud con el archivo genuino org-414.txt, Similitud: 0.5555, Status: 1
El archivo FID-13.txt tiene alta similitud con el archivo genuino org-396.txt, Similitud: 0.8334, Status: 0
El archivo FID-14.txt tiene alta similitud con el archivo genuino org-413.txt, Similitud: 0.7103, Status: 0
El archivo FID-15.txt tiene alta similitud con el archivo genuino org-400.txt, Similitud: 0.8234, Status: 0
Prediction:
 [0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0]
True:
 [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
```

Los resultados de la función utilizando unigramas muestra la comparación de los 15 archivos que se encuentran en la carpeta de sospechosos, su posible similitud con alguno de los textos genuinos y su estado indica si tiene alta (status = 0) o baja similitud (status = 1).

```
Accuracy Acore: 0.8000
Confusion Matrix
[[7 1]
 [2 5]]
```

```
Precision: 0.8750
Recall: 0.7778
Specificity: 0.8333
```

Más adelante se calcula la precisión de la predicción (80%); la matriz de confusión y los valores de la proporción de la predicción correcta de positivos(87%), proporción correcta de positivos reales(77%) y proporción correcta de negativos reales (83%).



Análisis de Similitud Utilizando Bigramas

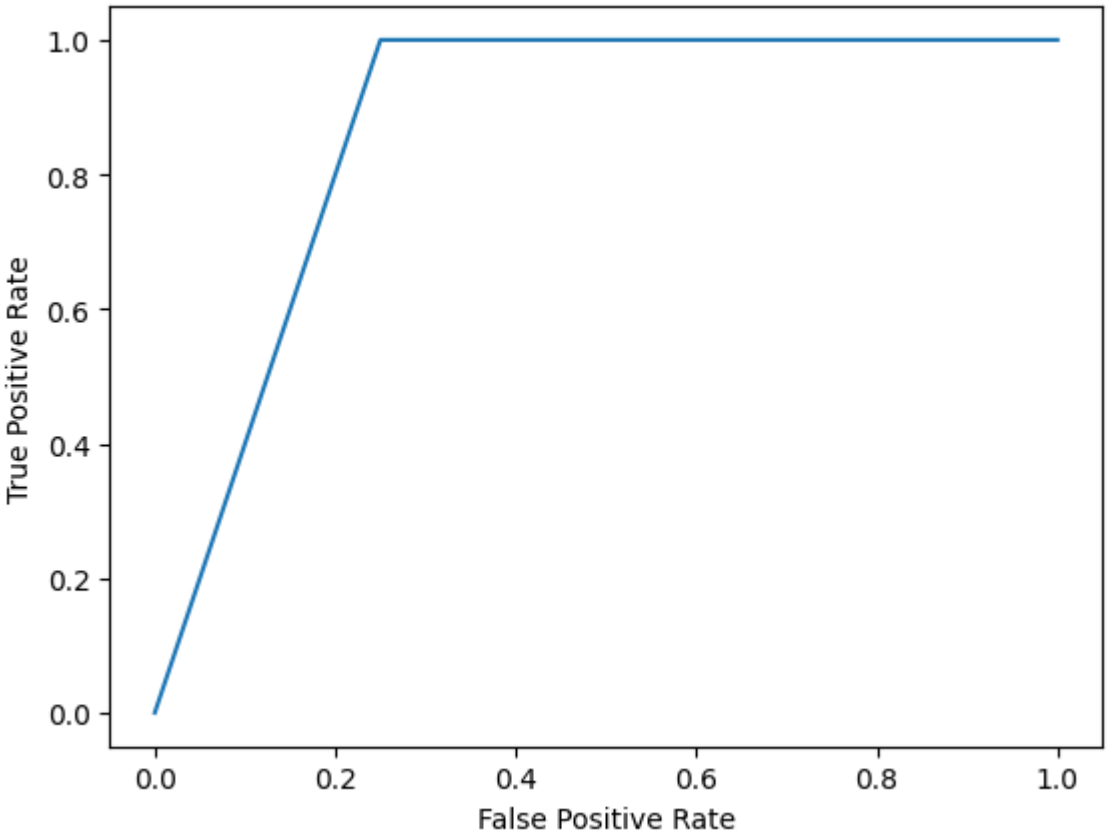
```
El archivo FID-01.txt tiene alta similitud con el archivo genuino org-325.txt, Similitud: 0.9196, Status: 0
El archivo FID-02.txt tiene baja similitud con el archivo genuino org-413.txt, Similitud: 0.6108, Status: 1
El archivo FID-03.txt tiene baja similitud con el archivo genuino org-339.txt, Similitud: 0.5263, Status: 1
El archivo FID-04.txt tiene baja similitud con el archivo genuino org-413.txt, Similitud: 0.4838, Status: 1
El archivo FID-05.txt tiene alta similitud con el archivo genuino org-340.txt, Similitud: 0.7788, Status: 0
El archivo FID-06.txt tiene baja similitud con el archivo genuino org-413.txt, Similitud: 0.4767, Status: 1
El archivo FID-07.txt tiene alta similitud con el archivo genuino org-356.txt, Similitud: 0.7374, Status: 0
El archivo FID-08.txt tiene baja similitud con el archivo genuino org-331.txt, Similitud: 0.3285, Status: 1
El archivo FID-09.txt tiene baja similitud con el archivo genuino org-388.txt, Similitud: 0.6186, Status: 1
El archivo FID-10.txt tiene baja similitud con el archivo genuino org-352.txt, Similitud: 0.4385, Status: 1
El archivo FID-11.txt tiene alta similitud con el archivo genuino org-372.txt, Similitud: 0.8703, Status: 0
El archivo FID-12.txt tiene baja similitud con el archivo genuino org-414.txt, Similitud: 0.3982, Status: 1
El archivo FID-13.txt tiene alta similitud con el archivo genuino org-396.txt, Similitud: 0.7515, Status: 0
El archivo FID-14.txt tiene baja similitud con el archivo genuino org-388.txt, Similitud: 0.5464, Status: 1
El archivo FID-15.txt tiene alta similitud con el archivo genuino org-400.txt, Similitud: 0.7235, Status: 0
Prediction:
 [0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0]
True:
 [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
```

Los resultados de la función utilizando bigramas muestra la comparación de los 15 archivos que se encuentran en la carpeta de sospechosos, su posible similitud con alguno de los textos genuinos y su estado indica si tiene alta (status = 0) o baja similitud (status = 1).

```
Accuracy Acore: 0.8667
Confusion Matrix
[[6 2]
 [0 7]]
```

```
Precision: 0.7500
Recall: 1.0000
Specificity: 0.7778
```

Más adelante se calcula la precisión de la predicción (86.6%); la matriz de confusión y los valores de proporción de la predicción correcta de positivos (75%), proporción correcta de positivos reales (100%) y proporción correcta de negativos reales (77%). Se puede observar que al utilizar bigramas se obtuvo una mayor precisión en la predicción.


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate