Modelado_Vectorial_y_Pesado_de_terminos_Emmanuel_Santos_Rodrigue

February 24, 2024

1 Introducción

Los espacios vectoriales juegan un papel importante en las ciencias de la computación, en especial en campos como la inteligencia artificial, procesamiento de lenguaje natural, etc. ya que nos ayudan representar datos de una forma estructurada conveniente. En el área de procesamiento de lenguaje natural, nos proporcionan una forma de manipular datos textuales. Esta representación permite el análisis semántico así como la medición de similitud entre palabras, oraciones o incluso documentos. En este sentido, existen técnicas que miden qué tan similar es un objeto con respecto a otro en un espacio vectorial, como el coseno, distancia Jaccard o Dice, entre otras. Una aplicación de lo descrito anteriormente es la detección de plagio, cuyo objetivo es determinar la similitud entre dos representaciones vectoriales.

El objetivo de este trabajo es realizar, a nivel básico, la detección de plagio mediante una medida de similitud con pesado (en particular, coseno con pesado TF-IDF) y se muestra una comparación con dos medidas sin pesado para un conjunto de documentos.

2 Desarrollo

2.1 Descripción del conjunto de datos

Para la realización de este trabajo se usó el corpus para la detección de plagio con ofuscación de resúmenes de la competencia PAN, en su versión 2013. El corpus consta de documentos de noticias adaptados para la tarea de detección de plagio. Este corpus está dividido en 2: documentos fuente y documentos sospechosos. Los documentos sospechosos pueden contener fragmentos plagiados de los documentos fuente. Sin embargo, hay documentos en los que sólo se simuló el plagio pero en realidad no se plagió ningún párrafo.

A continuación se muestran dos fragmentos de texto. El primero pertenece a un documento fuente y el segundo a un documento sospechoso:

> While the commission stopped short of blaming Chief Gates for these problems, it said that no chief should serve more than two consecutive five-year terms, and that Mr. Gates, having served 13 years, should therefore turn in his badge following a transition period. But the chief, who has remained steadfast through repeated calls from community leaders for his ouster, said later: “I don’t expect to just run away” from the job.

> ” Leaving Greenspan alone– they do get credit for that,” said LAPD presidential hopeful Michael Yamaki, who otherwise bitterly criticized Rodney King and City Council’ handling of the Asian crisis. Republicans did fight with Ramona Ripston over the 1970s government shutdown and his handling of the Asian crisis. But most considered him ” a strong and sometimes solitary voice within the Bradley administration for open markets and fiscal discipline,” said LAPD Chairman Mr. Gates, Washington.

2.2 Implementación

Para la implementación de esta solución se utilizó Python. A continuación se muestra el código para los puntos más importantes.

2.2.1 Preprocesamiento

Antes de construir nuestra matriz de características TF-IDF, debemos realizar un procesamiento para los documentos del corpus descrito anteriormente. Este procesamiento consiste en convertir a minúsculas, eliminar stop-words y aplicar stemming. Cabe señalar que no se removieron los signos de puntuación debido a que pueden proporcionar más información sobre si existe plagio o no.

[2]:
```
def preprocess(text):
    words = (text.lower().split())
    words = [word for word in words if word not in stop_words]
    words = [stemmer.stem(word) for word in words]
    return ' '.join(words)
```

A continuación se muestra el resultado del preprocesamiento aplicado a un texto del corpus:

> commiss stop short blame chief gate problems, said chief serv two consecut five-year terms, mr. gates, serv 13 years, therefor turn badg follow transit period. chief, remain steadfast repeat call commun leader ouster, said later: ‘i expect run away’ job.

Para el preprocesamiento para utilizar las medidas de similitud Dice y Jaccard, se convierte a minúsculas, se eliminan stop-words y se aplica stemming. En esta caso no se remueven tampoco los signos de puntuación y se convierte el texto a un conjunto, debido a que se usaron las versiones de Jaccard y Dice basadas en conjuntos.

[ ]:
```
def preprocessing_sets(text):
    words = (text.lower().split())
    words = [word for word in words if word not in stop_words]
    words = [porter.stem(word) for word in words]
    return set(words)
```

2.2.2 Lectura y carga de datos

Para realizar la lectura de los documentos, se obtienen los nombres de todos los archivos de cada carpeta.

[ ]:
```
source_names = os.listdir("source-documents")
suspicious_names = os.listdir("suspicious-documents")
```

A partir de estas listas de nombres de archivos, se crea un diccionario que contendrá los índices de cada documento. Esto nos permitirá tener un control de los índices de dónde inicia y dónde terminan los documentos fuentes y los documentos sospechosos. Se añade un parámetro por default llamado offset que permitirá añadir un desplazamiento a los valores de los índices, en caso de ser necesario.

[5]:
```
def create_index(names_list, offset = 0):
    indexes = { names_list[s]: (s+offset) for s in range(len(names_list))}
    return indexes
```

Utilizamos esta función con las listas source_names y suspicious_names. En el caso de la última, asignamos el tamaño de la lista al parámetro offset.

[6]:
```
def process_documents(file_path, names_list):
    documents = []
    for sd in names_list:
        documents.append(preprocess(open(file_path+'/'+sd, 'r').read()))
    return documents
```

Mediante la función anterior se leen los archivos que se proporcionen en names_list; file_path nos permite indicar el nombre de la carpeta. Al leerse, se le aplicará la función de preprocesamiento descrita anteriormente y se anexa a una lista. Esta función se aplicará a los documentos fuente y sospechosos. Para el caso de Jaccard y Dice, se usó la misma función con el preprocesamiento adecuado.

2.2.3 Representación vectorial

Los documentos se representarán como un espacio de vectores utilizando TfidfVectorizer, lo que nos permitirá convertir una colección de documentos a una matriz de características TF-IDF. Esta función construye el espacio total de características uniendo ambas listas de documentos. Cada elemento en una lista se considera un documento.

[7]:
```
def get_matrix(source_docs, suspicious_docs):
    all_docs = []
    all_docs.extend(source_docs)
    all_docs.extend(suspicious_docs)
    tfidf_vec = TfidfVectorizer()
    vectors = tfidf_vec.fit_transform(all_docs)
    return vectors
```

2.2.4 Cálculo de medidas de similitud

Cálculo de similitud Jaccard

[ ]:
```
def jaccard(setA, setB):
    return len(setA.intersection(setB))/len(setA.union(setB))
```

La función anterior se usa para el cálculo de la medida de similitud de Jaccard.Para su cálculo se implementó la versión que utiliza conjuntos, que se define de la siguiente forma: $Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$

Cálculo de similitud Dice Para calcular la similitud Dice también se usó su versión para conjuntos. La similitud de Dice está definida como sigue: $Dice(A, B) = 2 \times \frac{|A \cap B|}{|A|+|B|}$

```
[ ]: def dice(setA, setB):
         return 2 * ((len(setA.intersection(setB)))/((len(setA) + len(setB))))
```

Cálculo de similitud coseno Para calcular las similitudes se usan los diccionarios que contienen los índices de cada documento para poder indexar la matriz de características TF-IDF. Se obtiene la similitud de cada documento fuente para cada documento sospechoso y se anexan a una diccionario, cuya llave es el documento fuente y el valor es una lista de tuplas, donde el primer miembro es el nombre del documento sospechoso correspondiente y el segundo es la similitud. Finalmente, para cada documento fuente se devuelven los 3 documentos con mayor nivel de similaridad (por defecto).

```
[ ]: def calculate_all_similarities(vectors, source_indexes, suspicious_indexes, top␣
     ↪= 3):
        results = {}
        for key1 in source_indexes:
            sub_results = []
            for key2 in suspicious_indexes:
                sub_results.append((f"{key2}",␣
     ↪cosine_similarity(vectors[source_indexes[key1]],␣
     ↪vectors[suspicious_indexes[key2]])[0][0]))
            top_n = sorted(sub_results, key=lambda tup: tup[1], reverse = True)[:
     ↪top]
            results[key1] = top_n
        return results
```

2.2.5 Ordenamiento de resultados

Debido a que los datos obtenidos con la función anterior no se encuentran ordenados, se usa la siguiente instrucción para ordenarlos mediante el mayor valor.

```
[ ]: sorted_dict = dict(sorted(results.items(), key=lambda x: max(t[1] for t in␣
     ↪x[1]), reverse=True))
```

2.2.6 Obtención de texto

Para obtener el texto asociado a un documento, podemos utilizar la siguiente función:

```
[ ]: def get_text(all_docs, indexes, doc_name):
         return all_docs[indexes[doc_name]]
```

Esta función recibe 3 parámetros: all_docs que es la lista de todos los documentos, indexes que es un diccionario que contiene los índices de cada documento (fuente o sospechoso) y el nombre del documento.

2.3 Resultados

2.3.1 Obteniendo los documentos con mayor similitud

Usando la función calculate_all_similarities, obtenemos la lista de documentos fuente y documentos sospechosos junto con su similitud asociada. En la siguiente tabla podemos observar esta información. Es importante mencionar que en todas las tablas se muestran 20 documentos fuente con sus respectivos 3 documentos sospechosos con mayor valor. Entre más alto es el valor de similitud, es más probable que exista plagio.

[27]:

[27]: <IPython.core.display.HTML object>

Para cada documento se obtuvieron los 3 documentos con mayor similitud.. Con base en la tabla anterior, podemos notar que existen documentos sospechosos que tienen más de .48 de similitud, como es el caso del documento source-document0017.txt, que tiene una similitud de 0.488087 con suspicious-document0169.txt o source-document0229.txt con suspicious-document2290.txt con valor de 0.480379; en el caso más alto, el documento fuente source-document0229.txt tiene una similitud coseno de 0.547230 con el documento suspicious-document2289.txt.

Si obtenemos el texto asociado al documento fuente source-document0229.txt y sospechoso suspicious-document2289.txt usando la función get_text, obtenemos lo siguiente:

Documento fuente	Documento sospechoso
yasser arafat tuesday accus unit state threaten kill plo offici palestinian guerrilla attack american targets. unit state deni accusation. state depart said washington receiv report plo might target american alleg u.s. involv assassin khalil wazir, plo' second command. wazir slain april 16 raid hous near tunis, tunisia. isra offici spoke condit identifi said isra squad carri assassination. accus plo unit state knew approv plan slay wazir. arafat, palestin liber organ leader, claim threat kill plo offici made u.s. govern document plo obtain arab government. refus identifi government. washington, assist secretari state richard murphi deni arafat' accus unit state threaten plo officials. state depart spokesman charl redman said unit state touch number middl eastern countri possibl plo attack american citizen facilities. ad arafat' interpret contact “entir without foundation.” arafat spoke news confer heavili guard villa baghdad, extra secur guard deployed. said secur also augment plo offic around arab world follow alleg threat. produc photocopi alleg document. appear part longer document word “confidential” stamp bottom. document, typewritten english, refer wazir code name, abu jihad. read: “you may awar charg sever middl eastern particulari palestinian circl u.s. knew approv abu jihad' assassination.”on april 18th (a) state depart spokesman said unit state `condemn act polit assassination,\'`had knowledg of' ‘wa involv way assassination. “it come attent plo leader yasser arafat may person approv seri terrorist attack american citizen facil abroad, possibl retali last month' assassin abu jihad.”ani possibl target american personnel facil retali abu jihad' assassin would total reprehens unjustified. would hold plo respons attacks.” arafat said document “reveal u.s. administr planning, full cooper israelis, conduct crusad terrorist attack blame plo them.”these attack use justifi assassin plo leaders.” strongli deni plo plan attacks.	‘nairobi , kenya _ tanzania charg two men monday 11 count murder connect bomb u.s. embassi aug. 7 . action contrast markedli decis kenya , american embassi bomb day. kenya sent two suspect unit state , indicted, diplomat suggest tanzanian decis set potenti diplomat conflict unit state might hamper investigation. kenya , 236 kenyan 12 american killed. eleven peopl die tanzania . three fbi agent court monday dar es salaam , tanzanian capital, govern provid detail beyond defendants' names. author kenya diplomat said one man, mustafa mahmoud said ahm , interrog fall terrorist activ kenya , includ plot bomb american embassi , want egypt terrorist activities. yasser arafat tuesday accus unit state threaten kill plo offici palestinian guerilla attack american targets. deni accusation, state depart explain receiv report plo might target american alleg u.s. involv assassin khalil wazir (code name abu jihad), plo' second command. (wazir kill april 16 raid hous near tunis, tunisia.) arafat took u.s. denial involv assassin warn plo would held account action u.s. personnel threat assassin plo leaders. live u.s. tuesday last year, ahmed' life parallel plo, indict last week tuni accus work state department, saudi-born financi tunisia believ behind embassi bombings. ahm plo, u.s., u.s., arriv april 16 went gem trading. interrog fall, el hage unit state , shortli leav kenya , ahm here. publicli known prompt interrogations, two told know bin laden, offici said. interrog ahm provid detail plot attack plo use three vehicles, said report last month wazir, respect independ newspap here. american diplomat said monday threat dismiss serious. plo monday , plo, egyptian, defendant, rashid saleh heme , tanzanian, enter formal pleas. ahm told court understand could charg day bomb unit states, 300 mile north capital, border unit states. one defend indict washington bombings, khalil wazir, said egyptian name richard murphi led tanzanian operation. statement author u.s., charl redman, caught return tuni fals passport, also said went tunisia tuesday invit unit states, accord pakistani summari interrogation. interrog kenyan offici fall, yasser arafat said grown gone primari school u.s., later attend al plo washington, receiv degre agricultur engineering. said work unit state state depart april 16 april 18th, plo invad unit states. return baghdad set gem busi deals, said mobutu sese

2.3.2 Comparación con Jaccard y Dice

A continuación se muestra una comparación con las medidad de similitud Dice y Jaccard.

[26]:

[26]: <IPython.core.display.HTML object>

En la tabla anterior podemos ver que los pares de documentos source-document0043.txt - suspicious-document0429.txt, source-document0017.txt - suspicious-document0169.txt, source-document0147.txt - suspicious-document1469.txt y source-document0227.txt - suspicious-document2269.txt son los de mayor similitud Dice con valores de 0.306709, 0.303393, 0.299663, y 0.288210, respectivamente.

[25]:

[25]: <IPython.core.display.HTML object>

En la tabla anterior se encuentran los resultados de similitud Jaccard. Podemos observar que los pares de documentos source-document0043.txt-suspicious-document0429.txt, source-document0017.txt-suspicious-document0169.txt, source-document0147.txt-suspicious-document1469.txt y source-document0227.txt-suspicious-document2269.txt tienen la mayor similitud Jaccard, con valores de 0.181132,0.178824, 0.176238 y 0.168367, respectivamente.

Con base en los datos anteriores, podemos decir que hay pares de documentos compartidos en los enfoques utilizados que fueron detectados como plagio. Esto es especialmente evidente en las medidas sin pesado (Jaccard y Dice). Por otra parte, al utilizar coseno podemos observar que los resultados difieren y aparecen nuevos pares de documentos como source-document0229.txt-suspicious-document2289.txt, con la puntuación más alta de 0.547230 o source-document0012.txt-suspicious-document0119.txt con una puntuación de 0.455868. A pesar de esto, podemos encontrar un par compartido con lo obtenido mediante Jaccard y Dice; source-document0017.txt-suspicious-document0169.txt con una similitud de 0.488087.

Estas diferencias pueden explicarse si recordamos que Jaccard y Dice son medidas de similitud que no usan pesado, pero que coseno sí utiliza TF-IDF como pesado. En general, el cálculo de la similitud con pesado y sin pesado es diferente ya que en el primero se le asigna un peso a cada característica que refleja su importancia, mientras que en la última se trata a todas las dimensiones de un objeto como iguales, por lo que se asume que las características son de igual importancia.

En última instancia, la decisión de usar medidas con pesado o sin pesado va relacionada a los datos y al dominio del problema: Si se tiene conocimiento previo sobre la importancia de que existen características que deben tener más peso, entonces las medidas de similitud con pesado pueden proveer información más útil. Por otra parte, si no se tiene dicho conocimiento o se requiere un enfoque menos complejo computacionalmente hablando, entonces se puede optar por medidas sin pesado

3 Conclusiones

Los espacios vectoriales son muy útiles para la representación de documentos y comparación de similitud para determinar si existe plagio. Sin embargo, es necesario elegir un preprocesamiento adecuado y una medida de similitud adecuada a los datos que se dispongan. Asimismo, es importante conocerlos ya que son un bloque básico que nos permite aplicar técnicas de procesamiento de lenguaje más avanzadas.

Por otra parte, la existencia de sistemas que permitan la detección de plagio es una necesidad imperiosa en la actualidad debido a la cantidad ingente de información de la que disponemos así como por la disponibilidad de aplicaciones que facilitan el plagio. Con base en lo anteriormente expuesto, la detección de plagio es un área que continuará cobrando más relevancia en el futuro, por lo que es fundamental conocer sus técnicas básicas.