IRWA Project: Part 3 - Ranking

Git URL: https://github.com/raquel-sb/IRWA_2023.git

1. Introducción

Este laboratorio constituye la continuación del proyecto en el que nos enfocaremos durante todo el trimestre. Partimos de los avances de la primera y segunda práctica, es decir, el pre-procesamiento de nuestros tweets por un lado, y la indexación-evaluación de los documentos y el sistema de búsqueda resultante por el otro. En esta tercera parte nos centraremos en la creación de un modelo de ranking.

Así pues, nuestra meta es, dada una query, encontrar todos los documentos donde aparezcan todas las palabras de la consulta y posteriormente ordenarlos por relevancia respecto a la query, lo que se define como *Ranking*.

2. Funciones Desarrolladas

Describiremos únicamente las nuevas funciones creadas para esta tercera parte del proyecto.

2.1. Función create_mapping()

La función create_maping() nos permite crear un diccionario a partir de un mapeo entre IDs de documentos y IDs de tweets. Como parámetro recibe una lista de tweets.

Inicialmente la función crea un diccionario vacío llamado mapping para almacenar los valores ID del documento y ID del tweet. Divide la cadena de carácteres en dos partes, para extraer el ID del tweet de la primera parte y el ID del documento de la segunda parte. Agrega una entrada al diccionario mapping utilizando el ID del documento como clave y el ID del tweet como valor. Finalmente, la función imprime la longitud del mapeo (número de entradas en el diccionario) y un ejemplo de la estructura (el primer par de IDs).

2.2. Función rank_our_score()

Para esta función hemos utilizado los pesos tf-idf para clasificar los documentos que coinciden con una consulta dada. Teniendo en cuenta los siguientes parámetros de los tweets con sus respectivos pesos:

- El número de likes: 20%
- El número de retweets: 20%
- Los tags del tweet: 60%

Este código implementa una función llamada `rank_our_score` que realiza la clasificación de los resultados de una búsqueda basada en pesos tf-idf. A continuación, se proporciona una descripción detallada de las operaciones realizadas por la función:

Se inicializa un diccionario llamado doc_vectors que almacena los vectores de documentos correspondientes a los términos de la consulta. Los elementos no existentes se inicializan en cero. Además se crea un vector de consulta con valores inicializados en cero. y se calcula la norma del vector de consulta utilizando las frecuencias de los términos en la consulta.

Por cada término de la consulta se calcula el producto tf-idf normalizado y se actualiza el vector de consulta. La puntuación de cada documenteo se calcula mediante la similitud del coseno entre el vector de consulta y los vectores de documentos. Almacenando esta puntuación juntamente son el identificador.

Posteriormente se recorren las puntuaciones de los documentos y se ajustan mediante la cantidad de "Likes" y "Retweets" de los tweets asociados a los documentos, así como la presencia de hashtags relacionados con los términos de la consulta. Finalmente se ordenan estos resultados.

2.3. Función search_our_score()

Esta función realiza una búsqueda de documentos en función de una consulta y luego clasifica los resultados utilizando la función rank_our_score. Recibe como parametros de entrada una query, un indice (índice previamente creado que utilizaremos), una colección de tweets y un diccionario.

Integrando así una búsqueda de documentos basada en términos de consulta y la clasificación de resultados utilizando la función rank_our_score. El resultado final es una lista de documentos ordenados según su relevancia con respecto a la consulta dada.

2.4. Función embedding_w2v()

Esta función genera la representación de un tweet como un vector único en la misma dimensión que las palabras.

La función calcula un vector de representación para un tweet utilizando embeddings de palabras (word2vec). Tomando como argumentos la lista de términos en el tweet (terms) y el diccionario de vectores de palabras (wv). Devolviendo como resultado el promedio de los vectores de palabras como la representación del tweet.

2.5. Función rank_Word2Vec()

La función rank_Word2Vec() genera un ranking de tweets según una consulta utilizando el algoritmo word2vec. Por lo tanto, esta función calcula un ranking de documentos (tweets) en función de su similitud con una consulta dada. Tomando como argumentos: query (términos de la consulta), docs (tweets que se deben clasificar), doc_to_tweet (diccionario que mapea identificadores de documentos a identificadores de tweets) y tweets (colección de tweets).

Para ello almacenamos todos los tweets en una lista llamada sentences. Para posteriormente entrenar un modelo word2vec en los tweets para aprender los embeddings y el vocabulario. Crea representaciones "tweet2vec" para cada tweet utilizando los embeddings de palabras. Además, calcula la representación de la consulta y la similitud del coseno entre todos los documentos y la consulta. Finalmente ordena los documentos por similitud descendente y nos devuelve los documentos clasificados y sus puntuaciones correspondientes.

2.6. Función search_word2vec()

La función search_word2vec realiza el ranking de los resultados de una búsqueda basándose en word2vec y aplicando la similitud del coseno. Donde tiene como argumentos entre otros una query y un índice invertido. Primero realiza un preprocesamiento de la consulta. Luego, busca los documentos relevantes (tweets) en función de los términos de la consulta en el índice invertido. Utilizando la función rank_Word2Vec() (definida en el punto anterior) para calcular la similitud entre los documentos y la consulta. Finalmente devuelve los documentos clasificados y sus puntuaciones correspondientes.

Por lo tanto, esta función utiliza embeddings de palabras (word2vec) para clasificar los tweets según su relevancia con respecto a una consulta específica, aplicando la similitud del coseno.

3. Scoring

En esta sección tenemos que calcular el top 20 tweets para cada una de nuestras 5 queries utilizando tf-idf + cosine similarity y utilizando un scoring de nuestra propia creación que incluya cosine similarity. Y, posteriormente, comparar ambos resultados.

Como recordatorio, nuestras queries, ya explicadas en la parte 2 del proyecto son:

- Query 1: “Eastern separatists groups”
- Query 2: “Humanitarian impact”
- Query 3: “Media coverage of war”
- Query 4: “Negotiations i war”
- Query 5: “Russian propaganda and disinformation”

3.1. TF-IDF + Cosine Similarity

Respecto a las funciones que realizan un indexing en base a tf-idf y, posteriormente, una ordenación de ranking en base a cosine similarity, ya lo tenemos de la parte 2, con las siguientes funciones:

- create_index_tfidf: crea un índice en base a tf-idf
- rank_documents: se realiza el cálculo en base a cosine similarity para presentar el ranking de documentos en base a unos términos (o words) introducidos

Los resultados que obtenemos para cada query los podemos ver en el fichero adjuntado IRWA-2023-u186651-u172942-u172957-part-3-appendix.pdf (Apartado: 1. Top 20-list of documents TF-IDF + Cosine Similarity).

Así pues, haciendo un análisis podemos decir que:

- Query 1

Los principales resultados se centran sistemáticamente en el cerco de grupos rusos en las regiones orientales de Ucrania, particularmente en Lyman, proporcionando información sobre operaciones militares, informes oficiales y comentarios de diversas fuentes. Esto sugiere que el enfoque TF-IDF + Cosine Similarity ha identificado efectivamente los documentos relacionados con la consulta y los ha clasificado según su relevancia.

- Query 2

Los resultados muestran una combinación de tuits que analizan incidentes específicos (por ejemplo, ataques con misiles a convoyes) e impactos más amplios (por ejemplo, en el comercio y la inversión mundiales). Con esto podemos llegar a la conclusión de que el enfoque TF-IDF + Cosine Similarity no identifica del todo bien los documentos relacionados con la consulta, aunque los detecte como los más relevantes.

- Query 3

Los resultados indican una combinación de perspectivas, incluidas críticas a los medios occidentales, actualizaciones de cobertura en vivo, publicaciones en las redes sociales desde la zona de conflicto y opiniones sobre la guerra. Las puntuaciones altas sugieren relevancia para la consulta, de esta manera, podemos finalizar que el enfoque TF-IDF + Cosine Similarity ha identificado efectivamente los documentos relacionados con la consulta.

- Query 4

Los resultados parecen relevantes para las negociaciones en el contexto de la guerra entre Ucrania y Rusia. Estos, cubren una variedad de perspectivas, incluidas declaraciones de figuras políticas, conocimientos sobre las negociaciones en curso y el sentimiento público. Además, la presencia de términos como "Putin", "Zelensky" y "negociaciones" se alinea con el enfoque de la consulta en las negociaciones de guerra.

Como en los apartados anteriores, las puntuaciones indican el grado de similitud entre los documentos y la consulta. Es importante tener en cuenta que las puntuaciones son relativas a los demás documentos de la colección. Las puntuaciones más altas generalmente sugieren una mayor relevancia.

Así pues, podemos concluir que el enfoque TF-IDF + Cosine Similarity ha identificado efectivamente los documentos relacionados con la consulta.

- Query 5

Los tweets obtenidos discuten de manera similar varios aspectos de la propaganda y la desinformación rusas, incluidas menciones a la manipulación de los medios, opiniones sobre la guerra y escepticismo sobre la cobertura de los medios occidentales.

En general, podemos decir que el enfoque TF-IDF + Cosine Similarity parece haber identificado con éxito tweets relevantes para la consulta, que cubren diferentes ángulos y perspectivas sobre el tema de la propaganda y la desinformación rusa relacionadas con la guerra entre Ucrania y Rusia.

3.2. Our-Score + Cosine Similarity

Utilizamos las nuevas funciones definidas en esta tercera parte del proyecto:

- rank_our_score
- search_our_score

Los resultados que obtenemos para cada query los podemos ver en el fichero adjuntado IRWA-2023-u186651-u172942-u172957-part-3-appendix.pdf (Apartado: 1. Top 20-list of documents Our-Score + Cosine Similarity).

Así pues, haciendo un análisis podemos decir que:

- Query 1

En resumen, los resultados sugieren una situación dinámica en la región oriental, con una combinación de actualizaciones militares, iniciativas de paz, apoyo internacional y reacciones públicas. Refleja la complejidad del conflicto en curso y los diversos aspectos involucrados. Por lo tanto, podríamos decir que la información de las recomendaciones obtenidas no son exactamente lo que estábamos buscando.

A nivel de puntuación, percibimos relativa diferencia entre el primer tweet del ranking y el número 20; pasamos de un 25 a un 4. Esto ya nos indica que el resto de tweets que no aparecen en el top tienen poca relevancia para la query y que, en realidad, en toda nuestra colección no muchos tweets están relacionados con esta cuestión.

- Query 2

Inicialmente, los tweets pintan colectivamente un panorama sombrío del impacto humanitario del conflicto, transmitiendo la necesidad de atención internacional, asistencia y posibles soluciones tecnológicas para abordar los desafíos que enfrentan los civiles en la región. Así pues, podemos decir que, en general, podemos observar cierta relación de temática con los primeros tweets del top, en consonancia con sus puntuaciones, cosa que luego decrece mucho con tweets bastante poco relevantes para la cuestión.

De hecho, observamos que el primer tweet tiene una puntuación increíblemente alta, con 192 puntos, cosa que el segundo decrece a los 14. Adicionalmente, vemos que el search solo ha encontrado 30 tweets relevantes para la query, lo que confirma nuestras sospechas.

- Query 3

Lo primero que observamos es que el top 20 empieza con un tweet cuya puntuación es de 869. En general observamos que las puntuaciones son altísimas pero no solo eso; el search ha devuelto un total de 3929 queries relevantes para la query.

Respecto al contenido de los tweets, estos indican una cobertura rica y dinámica de la guerra entre Ucrania y Rusia en las redes sociales, incorporando actualizaciones en tiempo real, diversos tipos de contenido y una variedad de perspectivas. El análisis de esta información proporciona una visión multifacética del conflicto en curso. Comprobamos que ningún tweet tiene la palabra “media” pero sí aparece el término de “coverage”.

Adicionalmente, vemos que todos los tweets del top tienen una gran cantidad de likes y retweets (hablando de miles), por lo que para nuestro scoring claramente ha supuesto un gran peso a la hora de puntuarlos.

- Query 4

El scoring para estos tweets es muy similar a los que vemos en la query 3; puntuaciones muy altas con una gran cantidad de likes y retweets, pero también vemos que una gran cantidad de tweets que hay en este top 20 también aparecen en el top de la query 3.

Por lo que hace al contenido, estos proporcionan una amplia gama de información relacionada con el conflicto, pero no parece haber una mención explícita de las negociaciones en el contexto de la consulta.

Así pues, una teoría es que tal vez las queries 3 y 4 no corresponden del todo a los tweets de la colección, por lo que al final terminan teniendo más peso aquellos tweets con mayor número de likes y retweets. Especialmente para esta query 4, donde no aparece el término “negotiations” en ningún tweet del top. Por lo tanto parece que las respuestas del top, a pesar de su alta puntuación, no son realmente relevantes para la query por la ausencia de términos en común con esta.

- Query 5

A diferencia de las queries 3 y 4, las puntuaciones no alcanzan valores tan altos como en los dos casos anteriores; el rango va desde los 90 puntos hasta 19. Con ello podemos ver que los primeros tweets del top son precisamente aquellos que cuentan con mayor número de likes y retweets, y va descendiendo progresivamente hasta el final del top, pero la magnitud de esos likes es muy inferior a la observada en las dos anteriores queries, por lo que también se justifica un scoring más modesto.

De todas formas, podemos comprobar que los tweets recomendados no parecen tener especial relevancia para la query, dado que no aparecen en ninguno de ellos los términos “disinformation” ni “propaganda”. En general, los resultados proporcionan una instantánea del panorama informativo actual en torno al conflicto Rusia-Ucrania en Twitter, mostrando una combinación de actualizaciones de noticias, opiniones y contenido multimedia.

Esto nos verifica, una vez más, que el peso que se le está dando a los likes y retweets es muy superior que el scoring por los términos de la query que puedan aparecer.

3.3. Comparación de ambos métodos

Como introducción, remarcar que, el nuevo mecanismo de puntuación también tiene en cuenta el número de likes (20%), el número de retweets (20%) y la presencia de hashtags relevantes (60%). Este cambio ha provocado algunos cambios en la clasificación de los resultados de búsqueda en comparación con el TF-IDF + Cosine Similarity.

Este nuevo enfoque de puntuación prioriza los tweets con una mayor participación (me gusta y retweets) y una presencia más fuerte de hashtags relevantes, capturando potencialmente contenido más popular y ampliamente discutido sobre la query en cuestión.

Sin embargo, es importante tener en cuenta que, como hemos podido ver en los tweets obtenidos para cada query, la naturaleza de la participación (me gusta y retweets) no siempre refleja la exactitud o credibilidad de la información.

Para llevar a cabo la comparación de ambos métodos, realizamos distintos Precision@k con 5 K's diferentes. Antes de nada, revisamos los tweets devueltos por our_score y les asignamos una ground_truth que consideramos correcta para así poder realizar esta comprobación. Respecto a la ground_truth de los tweets devueltos por tf-idf, ya teníamos una ground_truth seleccionado en la parte anterior de este proyecto, por lo que utilizamos los mismos valores en ese caso.

Obtenemos los siguientes resultados para TF-IDF + Cosine Similarity y para Our-Score + Cosine SImilarity respectivamente:

```
TF-IDF + Cosine Similarity:
======================

Query1 -  Precision@4: 1.0
Query2 -  Precision@4: 0.5
Query3 -  Precision@4: 0.5
Query4 -  Precision@4: 0.5
Query5 -  Precision@4: 0.75
Query1 -  Precision@8: 0.75
Query2 -  Precision@8: 0.625
Query3 -  Precision@8: 0.625
Query4 -  Precision@8: 0.5
Query5 -  Precision@8: 0.625
Query1 -  Precision@12: 0.5
Query2 -  Precision@12: 0.6666666666666666
Query3 -  Precision@12: 0.75
Query4 -  Precision@12: 0.5833333333333334
Query5 -  Precision@12: 0.5
Query1 -  Precision@16: 0.4375
Query2 -  Precision@16: 0.5625
Query3 -  Precision@16: 0.625
Query4 -  Precision@16: 0.5
Query5 -  Precision@16: 0.4375
Query1 -  Precision@20: 0.5
Query2 -  Precision@20: 0.5
Query3 -  Precision@20: 0.5
Query4 -  Precision@20: 0.5
Query5 -  Precision@20: 0.5
```

```
Our-Score + Cosine Similarity:
======================

Query1 -  Precision@4: 0.25
Query2 -  Precision@4: 1.0
Query3 -  Precision@4: 0.5
Query4 -  Precision@4: 0.0
Query5 -  Precision@4: 0.25
Query1 -  Precision@8: 0.375
Query2 -  Precision@8: 0.75
Query3 -  Precision@8: 0.625
Query4 -  Precision@8: 0.125
Query5 -  Precision@8: 0.25
Query1 -  Precision@12: 0.5
Query2 -  Precision@12: 0.5833333333333334
Query3 -  Precision@12: 0.5833333333333334
Query4 -  Precision@12: 0.25
Query5 -  Precision@12: 0.25
Query1 -  Precision@16: 0.5625
Query2 -  Precision@16: 0.5
Query3 -  Precision@16: 0.5
Query4 -  Precision@16: 0.4375
Query5 -  Precision@16: 0.375
Query1 -  Precision@20: 0.5
Query2 -  Precision@20: 0.5
Query3 -  Precision@20: 0.5
Query4 -  Precision@20: 0.5
Query5 -  Precision@20: 0.5
```

Analizamos caso por caso:

1. **Precision@4:** vemos que en general obtenemos levemente mejores resultados con tf-idf, aun así, hay un único caso donde tenemos mejor puntuación con nuestro score (Query 2).
2. **Precision@8:** aquí vuelve a suceder lo mismo; en general tf-idf muestra una superioridad respecto a nuestro score a excepción de la Query2.
3. **Precision@12:** en este caso vemos que claramente tf-idf performa mejor en prácticamente todas las 5 queries, a excepción de la Q1 donde empatan.

4. **Precision@16:** tf-idf vuelve a superar our_score en todas excepto en la query 2
5. **Precision@20:** aquí vemos que ambos scores lo hacen igual de mal.

Con esto podemos concluir que en general tf-idf suele funcionar mejor que nuestro score en la mayoría de casos siempre y cuando la k se mantenga relativamente baja. Una vez que buscamos precisión para mayores K's nos encontramos con que ninguno de estos dos searching methods destaca especialmente ni de forma positiva ni de forma negativa.

Adicionalmente, consideramos que no obtendremos los mismos resultados de ranking (o scoring) utilizando otras queries que no sean estas; ya hemos visto, por ejemplo, que our_score suele hacerlo mejor que tf-idf para nuestra query 2. Por lo tanto, hemos podido comprobar que la forma en la que nuestra query está escrita afecta a la performance de los distintos scorings.

4. Top 20-list of documents using Word2Vec + Cosine Similarity

En este apartado desarrollamos e implmentamos un ranking word2vec + cosine similarity para cada una de las 5 queries definidas en la parte 2 del proyecto.

Recordemos que nuestras queries son:

1. Query 1: "Eastern separatists groups"
2. Query 2: "Humanitarian impact"
3. Query 3: "Media coverage of war"
4. Query 4: "Negotiations i war"
5. Query 5: "Russian propaganda and disinformation"

Para ello hemos creado 3 funciones diferentes (definidas anteriormente):

- La función embedding_w2v que nos sirve para crear una representación vectorial del tweet.
- La función rank_Word2Vec para generar un ranking de tweets con representación vectorial utilizando cosine similarity.
- La función search_word2vec para realizar una búsqueda en base a una query para obtener los resultados con el método de word2vec.

Una vez obtenemos el top 20 tweets para cada una de nuestras 5 queries podemos observar lo siguiente:

- Las queries 1 y 2 tienen tweets con puntuación muy baja. Con esto podemos concluir que realmente estas dos queries no tienen tweets significativos o relevantes.
- Las queries 3 y 4 tienen tweets obtenemos unas puntuaciones entorno a los 20 puntos. Por lo tanto, hay tweets en nuestra colección que se ajustan muy bien con las queries presentadas.
- La query 5 tiene unos tweets con puntuaciones algo más modestas comparadas con los obtenidos en las queries 3 y 4. Con ello podemos interpretar que igualmente hay tweets significativos para esta query pero no en la misma magnitud que los tweets de las dos queries anteriores.

5. Transformer-based embeddings

Los embeddings basados en transformers, como los de BERT o RoBERTa, presentan tanto mejoras como complicaciones en el proceso de recuperación de información en comparación con embeddings tradicionales como word2vec.

Cuando analizamos la comprensión contextual, los modelos basados en transformers destacan al considerar toda la oración o el documento. Esta capacidad les permite capturar matices y contexto de manera efectiva. En el caso de los tweets, donde la concisión y el contexto son cruciales, los embeddings basados en transformers resultan especialmente beneficiosos. En contraste, los embeddings tradicionales como word2vec tratan cada palabra de forma independiente, careciendo de conciencia contextual. Esta limitación puede afectar su efectividad en textos cortos y dependientes del contexto.

En cuanto a las representaciones semánticas, los embeddings de transformers mejoran significativamente al proporcionar información semántica profunda y enriquecedora. Tienen la capacidad de capturar relaciones complejas entre palabras y frases, lo cual es ventajoso en tareas de recuperación de información que requieren una contextualización precisa y relevante. Por otro lado, los embeddings de word2vec son más simples y menos expresivos, representando palabras como vectores fijos y a menudo ignorando matices semánticos intrincados.

Si analizamos la carga computacional, los transformers presentan una complicación, ya que son computacionalmente costosos debido a los mecanismos de atención y las múltiples capas. Esto puede resultar en mayores costos computacionales al recuperar documentos relevantes. Por otro lado, los embeddings de word2vec son más ligeros computacionalmente, lo que los hace más eficientes para tareas de recuperación.

En el manejo de textos cortos como tweets, los transformers ofrecen ventajas al capturar matices, sarcasmo y patrones de lenguaje en evolución. Su comprensión contextual beneficia la recuperación de información relevante. En contraste, word2vec tiene dificultades con textos cortos debido a su falta de contexto, pudiendo pasar por alto significados sutiles o no capturar cambios de contexto.

En cuanto al fine-tuning, los modelos basados en transformers requieren ajuste fino para un rendimiento óptimo, adaptándose a tareas o dominios específicos. Aunque esto puede mejorar la precisión de la recuperación, implica esfuerzo adicional. Por el contrario, los embeddings de word2vec son fijos y no requieren ajuste fino, pero su falta de conciencia contextual puede limitar su efectividad.

Por lo tanto, los embeddings basados en **transformers** ofrecen ventajas en la comprensión contextual y la riqueza semántica. Sin embargo, su carga computacional y la posible necesidad de Fine-Tuning deben considerarse cuidadosamente al integrarlos en sistemas de Recuperación de Información. **Word2vec**, aunque más simple, puede quedarse corto al capturar las complejidades de textos cortos y dependientes del contexto, como los tweets.