Guillem Gauchia Torres - u186410 - 240215
Àlex Herrero Díaz - u186402 - 240799
Adrià Julià Parada - u188319 - 242195
Date: 11/11/2023

Part 3: Ranking

This project has been carried out by implementing in a notebook the necessary functions and code to process a database of tweets about the current war between Ukraine and Russia. And for the third part it is intended to test different systems of ranking such as Word2Vec and our implementation, evaluate the results and a brief summary on how transformer-based embeddings could enhance or complicate the retrieval process compared with word2vec.

IRWA_2023_u186402_u186410_u188319_part_3.ipynb is the Notebook used to carry out the objectives of the practice and to obtain results on the analysis of the queries and evaluations obtained from the results. The code files are located at the following Github Repository:
https://github.com/AlexHerreroDiaz/IRWA-2023/tree/main/IRWA-2023-part-3

Improvements over the previous part

As we were asked to improve in this practice with respect to the previous one, we have made changes in the search system so that now the searches are performed in such a way that all the terms of the query are taken into account (conjunctive queries).
And for the queries we were asked to manually create a document that determines the ground-truth of a query to indicate whether a document in a particular collection is relevant or not, and thus use the search method implemented in part 2, which previously we used only the ***evaluation_gt.csv*** document provided in the project material and not one created for new queries determined by us.

For this practice we have created a file called ***our_evaluation_gt.csv*** on the following queries:

- Q1 – "Sanctions on Russia"
- Q2 – "War crimes on Russa"
- Q3 – "Russia occupies territories"

On the following screenshot it is shown a snippet on the csv to show how it is formatted:

doc	query_id	label
doc_2402	Q1	1
doc_427	Q1	1
doc_3650	Q1	1
doc_88	Q1	1
doc_2037	Q1	1
doc_184	Q1	1
doc_2976	Q1	1
doc_3318	Q1	1
doc_2915	Q1	1
doc_1203	Q1	1
doc_423	Q1	0
doc_2151	Q1	0
doc_1866	Q1	0
doc_1345	Q1	0
doc_154	Q1	0
doc_2029	Q1	0
doc_3114	Q1	0

Ranking
1. Your-Score + Cosine Similarity

For this section, in which we were asked to create our own method for a ranking system based on the social parameters present in each tweet from the social network twitter, we created the ***social_score*** function, which takes as parameters the documents, in this case the collection of tweets, and a query represented by a dictionary, in which are stored the values of ***likes***, ***retweets*** and ***followers*** that we want to enter in the search to obtain tweets with similar social parameters.

To compute the social score we need to import the following functions from *sklearn*:

from *sklearn.metrics.pairwise* ***import*** *cosine_similarity*
from *sklearn.metrics* ***import*** *euclidean_distances*

The formula used is as follows:

$$cosine\_score = cosine\_similarity(social\_query\_vector, tweet\_vector)$$

$$eucliedean\_score = eucliedean\_distance(social\_query\_vector, tweet\_vector)$$

$$social\_score = (cosine\_score \times 0.2) + 0.8/(1 + euclidean\_score * distance\_weight)$$

where the ***distance_weight*** is a parameter set as default at 0.01 to tune how important is the euclidean distance factor on the formula.

Once the formula has been applied to each tweet, the list of scores obtained is collected and sorted decrementally, so that in the next cell of code we simply call the ***social_score*** function with the appropriate parameters, and the resulting tweets are returned with a vector of social parameters similar to the query provided.

In the following screenshot we can see the results of the following ***social_query***:

*{'Likes': **50**, 'Retweets': **5**, 'Followers': **100**}*

```
Top 10 ranked documents using our method 'Social-Score'
Social-query {'Likes': 50, 'Retweets': 5, 'Followers': 100}:

Rank 1: Doc_id: doc_913 has Social-Score: 0.856729671392403
Tweet:
Alleged footage from the outskirts of #Lyman. According to many sources, about 5 thousand Russian troops are surrounded by the Armed Forces of Ukraine🔥UA
#war #UkraineRussiaWar #UkraineWar #afu #russia #UkrainianArmy #Donbass #eu #us #usa #nato #Kharkiv #odessa #kyiv https://t.co/sfYbekA2BN

User: Maksym Nychyporuk✚ -- Likes: 34 -- Retweets: 5 -- Followers: 114

--------------------------
Rank 2: Doc_id: doc_3463 has Social-Score: 0.7800832038230776
Tweet:
#Russians and BTR-82A destroyed in an ambush of the Armed Forces of #Ukraine in the #Kharkiv region.#KharkivOffensive #UkraineRussiaWar #UkraineWar #UkrainianCounteroffensive #StandWithUkrai

User: Feher_Junior -- Likes: 32 -- Retweets: 5 -- Followers: 132

--------------------------
Rank 3: Doc_id: doc_3534 has Social-Score: 0.7798868504773203
Tweet:
Ukraine update: Counteroffensive continues. All roads to Lyman now under Ukrainian control

#Ukraine #Russia #UkraineWillWin #UkraineRussiaWar #UkraineNews #UkraineFrontLines #RussianMobilization #RussianUkrainianWar #Lyman #Kherson #Donetsk #Kharkiv
https://t.co/tTkepUS322

User: 24Talker -- Likes: 26 -- Retweets: 4 -- Followers: 71

--------------------------
Rank 4: Doc_id: doc_3013 has Social-Score: 0.7652494862447591
Tweet:
Russia: MVP arms aid to #Ukraine.  Ukraine may form one or two "Russian MVP BTR-82A regiments"😊😎😎🤣
#UkraineRussiaWar #UkraineUnderAttack #RussiaIsATerroristState #RussianWarCrimes #HIMARS #StandWithUkraine #ukraine #HIMARSOCLOCK #crimea #donbas #SlavaUkraini #KharkivOffensive https://t.co

User: 🏃People Live One Life.🍀Plants Live An Autumn🍒🍁 -- Likes: 42 -- Retweets: 5 -- Followers: 140

--------------------------
Rank 5: Doc_id: doc_3999 has Social-Score: 0.7620446318936378
Tweet:
After the staged fake referendum as of September 2022, Russian forces continued to occupy parts of Donetsk, Luhansk, Kherson and Zaporizhzhia which make up about 15% of Ukraine's territory.
#UkraineRussiaWar https://t.co/XQOJq8b3M1

User: Yusuf Adam -- Likes: 13 -- Retweets: 2 -- Followers: 113
```

As we can see from the screenshot above, the results obtained are tweets that have similar social parameters to the query. The use of cosine similarity ensures that the results follow a similar ratio between the values, and as the vectors being compared always have the same size of elements, we can use Euclidean distance to get those that not only have the same ratio, but also look for those where the values are similar.

Something to keep in mind is that to choose the values of the *social_query* is that these values correspond to the tweets in the database, because although in this social network there may be tweets with very high numbers, in this context on the subject of the war in Ukraine and Russia, the numbers of likes, retweets and followers are not very high because it is difficult to get virality dealing with topics of such seriousness.

Considering our ranking system created in comparison to **TF-IDF**, we can determine that the two systems have quite different objectives, as the previous provides a search in relation to the context of the text, thus making it easier to find tweets in relation to a topic specified in the query entered, whereas our system provides a method for finding tweets based on social parameters, the results do not necessarily have to be meaningful to each other, but it can provide a tool to search for tweets that for example have had a higher proportion of Likes than Retweets, or to search for accounts with few followers that have posted a tweet that has had a lot of impact, etc.

In short, our method does not offer results where the documents have similar semantics, but allows the search for similar accounts in terms of social parameters.

2. Word2Vec + Cosine Similarity

For this section, we have started by training the Word2Vec model, using the **Gensim** library to train a Word2Vec model with the processed terms stored in **dictionary['Tokens']**. Once created, the trained model is saved in the path specified in ***model_path***.

The next thing we do is load the pre-trained Word2Vec model from the file stored in the ***word2vec_model*** variable.

In order to compute scores using Word2Vec + Cosine similarity we have defined the function ***calculate_tweet2vec(tweet, model)*** which takes a tweet and the Word2Vec model and returns its average vector representation and this function is called inside the following function defined as ***rank_tweets_word2vec(query, tweets, model)*** which takes a query, a list of tweets and the Word2Vec model, and returns a ranking of tweets based on cosine similarity to the query.

Finally, we have defined the function ***tweet2vec_rank(query)*** which ranks tweets based on cosine similarity to a specific query using the Word2Vec model. and calling the above functions returns the results to end up printing the top 20 ranked tweets along with their similarity score.

First Query

```
tweet2vec_rank("sanctions on russia")
```

```
Top 20 Ranked Tweets (Word2Vec + Cosine Similarity) for query 'sanctions on russia':

#1 Rank -- Tweet:
The Ukrainian army
#Ukraine-#UkraineRussianWar -#UkraineWar -#UkraineRussiaWar-#Russian https://t.co/SZbBK2cMV0
Cosine Similarity Score: 0.9735499024391174
-------------
#2 Rank -- Tweet:
This man! #UkraineRussiaWar 🙏 https://t.co/pFur2iMnNa
Cosine Similarity Score: 0.9735463857650757
-------------
#3 Rank -- Tweet:
lol...+1 POW

#Ukraine #UkraineRussiaWar #UkraineWar #ruSSia https://t.co/QUFeY6pYX6
Cosine Similarity Score: 0.9733633399009705
-------------
#4 Rank -- Tweet:
@DonaldJTrumpJr Trump stands with Russia

RU 🐂

USA stands with Ukraine

UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA UA

#UkraineRussiaWar #UkraineWar #UkrainianArmy #Ukraine #UkraineWillWin
https://t.co/HXpzHN0XCC
Cosine Similarity Score: 0.9732458591461182
-------------
#5 Rank -- Tweet:
Russian tank T-90 can not swim!

#UkraineRussiaWar #Ukraine #Russia #RussiaUkraineWar https://t.co/S4kkn969we
Cosine Similarity Score: 0.973231852054596
```

Second Query

```
tweet2vec_rank("war crimes in russia")
```

```
Top 20 Ranked Tweets (Word2Vec + Cosine Similarity) for query 'war crimes in russia':

#1 Rank -- Tweet:
Today it's Lyman day.. 👁️ #Lyman  #UkraineWillWin #UkraineRussiaWar #PutinWarCriminal #SlavaUkraine ✌️ UA
Cosine Similarity Score: 0.9332593679428101
-------------
#2 Rank -- Tweet:
Watch out! Warming up WW3 in #UkraineRussiaWar
#poundcrash https://t.co/ez8MOh7gDq https://t.co/5fdHxUfp1s
Cosine Similarity Score: 0.9330117106437683
-------------
#3 Rank -- Tweet:
@Euan_MacDonald Joining NATO means Finland is now safe from invasion by Russia.
Non NATO Nation Ukraine is being invaded by Russia.
#UkraineRussiaWar
Cosine Similarity Score: 0.9329205751419067
-------------
#4 Rank -- Tweet:
If #Putin is that strong and powerful he had to announce #annexation in the #frontline!

#putinswas #UkraineRussiaWar
Cosine Similarity Score: 0.9324485063552856
-------------
#5 Rank -- Tweet:
#UkraineRussiaWar #RussianUkrainianWar
Tankers vs Anti Tank infantry. https://t.co/dUGSxFSHym
Cosine Similarity Score: 0.9323928356170654
-------------
```

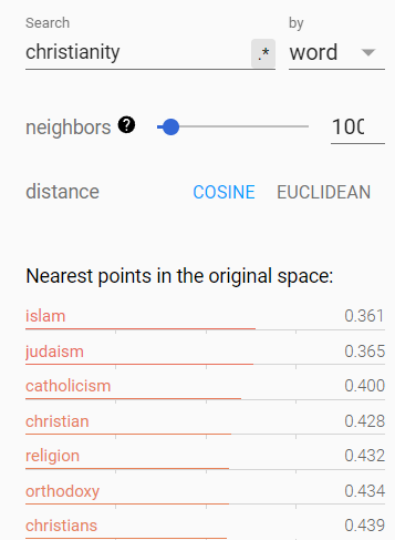
Third Query

```
tweet2vec_rank("russia occupies territories")
```

```
Top 20 Ranked Tweets (Word2Vec + Cosine Similarity) for query 'russia occupies territories':

#1 Rank -- Tweet:
lol...+1 POW

#Ukraine #UkraineRussiaWar #UkraineWar #ruSSia https://t.co/QUFeY6pYX6
Cosine Similarity Score: 0.9709658026695251
-------------
#2 Rank -- Tweet:
The Ukrainian army
#Ukraine-#UkraineRussianWar -#UkraineWar -#UkraineRussiaWar-#Russian https://t.co/SZbBK2cMV0
Cosine Similarity Score: 0.9707585573196411
-------------
#3 Rank -- Tweet:
Towards sunrise .... till all #ruSSian invaders kicked out from #Ukraine !

#UkraineRussiaWar #UkraineWar #ruSSia https://t.co/yvguLw7KUG
Cosine Similarity Score: 0.9706763029098511
-------------
#4 Rank -- Tweet:
This man! #UkraineRussiaWar 🙏 https://t.co/pFur2iMnNa
Cosine Similarity Score: 0.970584511756897
-------------
#5 Rank -- Tweet:
When will this war stop? 😞 #Ukraine #UkraineRussiaWar https://t.co/KEeh4JCNPi
Cosine Similarity Score: 0.9704949259757996
-------------
```

The results obtained are not entirely good, since in comparison with the method used previously, in this case much priority is given to the length of the document, where we can observe that the best valued results are not those in which the query is related to the best tweet, but those that after being processed have a similar length to each other. If we look for example at the first query, the first result is a short tweet that refers to the Ukrainian army while the query talks about the sanctions on Russia, therefore, what we can understand is that for the trained **Word2Vec** model, it understands that the words between query and tweet have a high relationship, and also the length of the document is similar and therefore these documents end up rated with high similarity score. This is the same for the rest of the queries.

The problem with Word2Vec is that in order to measure similar words in a word, it gives a high score to words that are related to each other. To give an example, the following link shows a graph with 10,000 English words where you can search for a word and it returns the words with the highest Word2Vec score.



We have searched for the word *christianity* and found that the closest words are other religions that could be interpreted as antagonistic religions. The fourth closest value is the denomination of its believers or practitioners.
Therefore, we can say that Word2Vec suggests words that may be related, even though they may not be interesting for users because it recommends other religions instead of recommending biblical characters or important elements of Christianity.
Therefore, it makes mistakes when it recommends Ukrainian facts in a query where we are interested in specific information from Russia.

In conclusion we can say that this method is not performing better than TF-IDF as the results are not relevant to the queries in the same way that TF-IDF manages to do better.

3. Transformer-based Embeddings vs. Word2Vec in Information Retrieval

BERT and **RoBERTa** are text-based prediction models. These models are trained to analyze the context of the document, either through **Masked Language Model** (MLM) or **Next Sentence Prediction** (NSP). Thanks to this, a robust language understanding model is achieved, which can be fine-tuned for more specific tasks using tuning variables, thereby improving results.

So, in comparison with Word2Vec, Transformer-based methods have the following positive characteristics that contribute to improvement and other negative aspects in systems like BERT and RoBERTa:

Enhancements:

These models offer a greater contextual understanding by considering an entire document. In the case of tweets, where context is crucial due to their short length, Transformer-based Embeddings can more effectively capture specific meanings, sarcasm, or sentiment by taking adjacent words into account.

Moreover, these models can discern between polysemous words or phrases with a similar meaning. This implies that two sentences with similar meanings but formulated differently can still be effectively matched.

Additionally, transformers generate detailed representations, enabling them to capture complex relationships and nuances present in short texts. This feature is particularly beneficial in tweets, where brevity and ambiguity are common.

Another interesting point is the use of pre-trained models. Models like BERT are trained on large-scale datasets, providing them with a general understanding of language. This means that these models can handle a wide variety of topics and contexts without the need for extensive and domain-specific training.

Complexities:

First, there is a higher computational cost associated with transformers compared to methods similar to Word2Vec. Another factor to consider is the size of the model because tTransformers tend to be larger compared to word2vec models.

In addition, even though transformer models are pre-trained on large data sets, parameter-tuning may need to be performed much more exhaustively to achieve optimal performance on IR tasks. This need for fine-tuning introduces an additional step and potential challenges in implementing and maintaining these models in information retrieval applications.

In conclusion, Transformer-based models except for being systems that with respect to Word2Vec are computationally more expensive and in certain cases slower to implement, they are a remarkable improvement for performing a search engine on documents representing tweets, since these models perform much better at capturing the multiple

semantics of a sentence and better understanding the similarity between different documents.