Mireia Pou Oliveras 251725
Iria Quintero García 254373
Javier González Otero 243078

Part 3: Ranking

Github link: https://github.com/javiergonzalez10upf/IRWA/tree/main/IRWA-2024-part-3
TAG: IRWA-2024-part-3

Introduction

The objective of this project is to develop a search engine.

After preprocessing the data, making an exploratory data analysis, creating an index and evaluating the performance, in this third part we are dealing with ranking. We will divide this ranking into three parts:

- Part 1: Ranking

In Part 1, we create different ranking metrics.

- Part 2: Top-20 documents using Word2Vec

In Part 2, we retrieve the top-20 relevant documents using Word2Vec and cosine similarity, and then we evaluate the results for our queries.

- Part 3: Better representation

In Part 3, we find a representation that works better than Word2Vec.

1. Ranking

This part is structured in the following way:

- 1.1 Ranking with TF-IDF and Cosine Similarity.
- 1.2 Ranking with “Our Score” and Cosine Similarity
- 1.3 Ranking with BM25

1.1. Ranking with TF-IDF and Cosine Similarity

As stated, this first part implements a search and ranking system that uses TF-IDF weighting combined with cosine similarity, allowing retrieval and ranking of tweets based on their relevance. The process includes three main phases.

First of all, before making the ranking, we need to construct an **inverted index**. For this, we reused the one created in the last practice, which returns the index and the following three dictionaries: tf, df, and idf.

Secondly, we proceed to perform the **ranking**. To do this, we adapted the *rank_documents* function used in the practice. The created function uses TF-IDF weights to create the vectors and cosine similarity to predict relevance. Then it returns the ranking and each document's predicted relevance score.

Finally, we created a **search function** that searches for each term of the query within the inverted index to find documents that include it. Then, we take the intersection of the term lists (AND operation) to get only the documents that include ALL the terms in the query.

Results

We tried the previous procedure for a specific query: Disha Ravi. The results obtained were the following:

```
======================
Top 10 results out of 628 ranked documents for the searched query:

----------------------

doc_id = doc_35225
doc_content = Free Disha Ravi

#FarmersProtest  #IndiaBeingSilenced

----------------------

doc_id = doc_32205
doc_content = #IndiaBeingSilenced
#farmersprotest
Disha ravi https://t.co/OVmvJy3d8V

----------------------

doc_id = doc_28152
doc_content = I Stand With Disha Ravi
#ReleaseDishaRavi
#FarmersProtest
```

As we can see, the results are good because all the documents (tweets) retrieved include ALL query terms, but we believe that they are not significant. In other words, the content of the documents does not provide any useful meaning.

1.2. Ranking with "Our Score" and Cosine Similarity

In this second part, we implement a search and ranking system that uses a **custom-created score** combined with cosine similarity. The procedure to create the ranking is the same as before, without creating the inverted index (since we already have the previous one). So, in essence, we produce a ranking function and a search function.

For the **ranking function**, we use several criterias to sort the terms by relevance. The criterias used are as follows:

1. **Cosine similarity:** Cosine similarity is used as before, computing the similarity between the TF-IDF weights of tweets and query terms.

2. **Popularity score:** We add a score that reflects the "popularity" of each tweet. This popularity score is a weighted sum of:
    - The number of retweets the tweet has. We gave this a weight of 0.4 as the number of retweets is a good indicator of the popularity and quality of the tweet.
    - The number of comments the tweet receives. We also gave this a weight of 0.4 as a tweet with lots of comments is a more popular tweet.
    - The number of likes the tweet has. This gets a weight of 0.2, slightly smaller than the previous two, because we believe that comments and retweets are better indicators than likes.
3. **Hashtag boost:** This boost adds weight if a tweet contains hashtags that match any of the original query terms. The weight added is 0.2, as we think it's a similar metric to the like score.
4. **Contextual similarity:** Beyond just using data, we also wanted to check how similar the context of each tweet is to the entire query. For this, we used a BERT model to encode the tweet and applied cosine similarity to measure the similarity with the query terms. Remark: This makes our function much slower, but we believe it improves the results.

Then, we calculate the final score as follows:

```python
combined_score = (cosine_sim + popularity_score) * hashtag_boost +
contextual_similarity
```

And the ranking is simply this score sorted!.

The **search function,** that uses the previously created ranking function to return the ranking and the document scores, is really similar to the one used in TF-IDF. The only difference is that we don't limit the search to documents containing all query terms. Instead, we consider documents that contain ONE OR MORE terms because popularity metrics (like retweets and likes) or hashtag relevance can make a document highly relevant, even if it only includes some of the terms. Also, by considering terms which may only have one part of the query, we are focusing more on capturing relevant content and not only matching between terms.

Results

We also tried the search function for a specific query: *Disha Ravi*. The results obtained were the following:

```
======================
Top 10 results out of 829 ranked documents for the searched query:

----------------------

Rank: 1 | Score: 3672.0499
doc_id = doc_38410
doc_content = disha ravi, a 21-year-old climate activist, has been arrested by

disha's arrest is alarming and the world needs to pay attention. #freedisharavi

https://t.co/IYGsLpNjwZ
```

```
----------------------

Rank: 2 | Score: 3583.4351
doc_id = doc_38012
doc_content = Disha Ravi broke down in court room and told judge that she had m

----------------------

Rank: 3 | Score: 1183.9029
doc_id = doc_37099
doc_content = Disha Ravi is 21 yrs

A climate activist from India she campaigns for clean air, clean water and a li

She is now facing state sanctioned violence for peacefully supporting farmers

Silence is not an option we must all condemn this act of suppression

#FarmersProtest

----------------------
```

We can see that the results are much better than the ones of TF-IDF. The retrieved documents not only include all the query terms, but also are more informative. The phrases are longer and have more sense than simply three words.

(Although our search function is not defined to search documents with all the query terms, it makes sense that the ones that have a higher score are those that contain all query terms.)

1.3. Ranking with BM25

In this third and final part, we implement a search and ranking system that uses BM25 combined with cosine similarity. The procedure to create the ranking is the same as the previous: a ranking function and a search function.

The ranking function, in this case, is easier, as it only applies the BM25 formula to the query terms, without using any cosine similarity or weighted vectors. Then it returns the scores sorted by the BM25 formula.

The search function is exactly the same as the one used in 1.1, but using the ranking function of BM25.

Results

We also tried the search function for the query *Disha Ravi* and the results obtained were the following:

```
======================
Top 10 results out of 628 ranked documents for the searched query:

----------------------

Rank: 1 | Score: 9.6486
doc_id = doc_35225
doc_content = Free Disha Ravi

#FarmersProtest  #IndiaBeingSilenced

----------------------

Rank: 2 | Score: 9.6486
doc_id = doc_11995
doc_content = #FarmersProtest. Disha Ravi. https://t.co/VbW9rYLUIa

----------------------

Rank: 3 | Score: 9.6486
doc_id = doc_32205
doc_content = #IndiaBeingSilenced
#farmersprotest
Disha ravi https://t.co/OVmvJy3d8V

----------------------
```

The results obtained are very similar to the ones obtained on TF-IDF scoring. They successfully include all the query terms, but don't give any information about the context. For this reason, we think our score outperforms this result.

2. Top-20 documents using Word2Vec

This part is structured in the following way:

- 2.1 Ranking with Word2Vec and Cosine Similarity
- 2.2 Testing our queries

2.1. Ranking with Word2Vec and Cosine Similarity

In this first part, we implement a search and ranking system using Word2Vec representation combined with cosine similarity. To create this ranking system, we follow two preprocessing steps:

1. **Training the Word2Vec model**: We use a Word2Vec model from the gensim.models library and train it on the tokenized tweets.
2. **Representing the tweets:** After training the model, we represent each tweet as a vector. Before representing a tweet, we check whether each word exists in the model's vocabulary, as Word2Vec cannot represent words it has not seen during training.

Once all the tweets are represented as Word2Vec vectors, we can build the **ranking function**. This function calculates the cosine similarity between the query vector and each tweet vector, and then returns the tweets sorted by similarity.

Finally, we create a **search function** that computes the query vector, considers only the documents that contain all query terms, retrieves the ranking of tweets using the ranking function, and displays the top 20 results.

2.2. Testing our queries

To test the function, we have used the set of queries defined in the previous practice, and evaluated each result. That is:

1. **India government farmers protest:** The top results talk about the government's actions, the farmer protests, the international support, etc. This matches the query well! However, the phrases in rank 9 to 16 are repeated, so the model could be improved.

   #results: 55

```
======================
Top 20 results out of 55 ranked documents for query: India government farmers p

----------------------

Rank: 1 | Score: 0.9472
doc_id = doc_39097
doc_content = To hear more about what led to widespread farmer protests in Indi

#FarmersProtest #शहीद_जवान_शहीद_किसान https://t.co/GxYjDjChtp

----------------------

Rank: 2 | Score: 0.9449
doc_id = doc_45211
doc_content = @YourAnonCentral Please also highlight about the ongoing farmers

----------------------

Rank: 3 | Score: 0.9430
doc_id = doc_8933
doc_content = 87 US farmers' unions have extended solidarity to the ongoing pro

----------------------

Rank: 4 | Score: 0.9345
doc_id = doc_25562
doc_content = "Farmers across India have peacefully organized and protested for
```

2. **Increasing prices:** The top results show a good match for the query, with tweets discussing the costs of fuel. Several tweets contain similar content, but phrased in different ways. This repetition is good, as it reflects a shared view of the topic.

   #results: 109

```
Top 20 results out of 109 ranked documents for query: Increasing prices

----------------------

Rank: 1 | Score: 0.9826
doc_id = doc_10847
doc_content = Khalisathan is responsible to increase price of diesel and petro

----------------------

Rank: 2 | Score: 0.9589
doc_id = doc_33490
doc_content = Earlier,when the prices on petrol were increased, the concerns o

----------------------

Rank: 3 | Score: 0.9369
doc_id = doc_7119
doc_content = Why the outcry of petrol price increases,   If There is a higher
#PetrolPriceHike
#FarmersProtest https://t.co/eXJPmlw29m

----------------------

Rank: 4 | Score: 0.9263
doc_id = doc_38875
doc_content = #fuel prices revised again . In 11 months #petrol have increased

----------------------
```

3. **Narendra Modi** (India's prime minister): The top results include many tweets criticizing the prime minister policies regarding the farmers' protest. We think that the results are relevant, as it captured perfectly the information needed for the query.

   #results: 69

```
Top 20 results out of 69 ranked documents for query: Narendra Modi

----------------------

Rank: 1 | Score: 0.8924
doc_id = doc_9065
doc_content = If prime minister narendra modi knows better whey'not he is teac

----------------------

Rank: 2 | Score: 0.8798
doc_id = doc_13190
doc_content = Narendra Modi should answer why did Vijay Mallya, Nirav Modi , M
B4 preaching privatisation 😡

#modi #bjp #bjplies #bjpfails #FarmersProtest #AtmaNirbharBharat

https://t.co/HMS5N1gWZz

----------------------

Rank: 3 | Score: 0.8694
doc_id = doc_26001
doc_content = I dare you, amit shah and narendra modi to contest from punjab @
#FarmersMakeIndia
#FarmersProtest https://t.co/CpTZPjTpst
```

4. **Disha Ravi** (Principal activist for farmer's rights): The top results primarily focus on Disha Ravi and the activist involvement in the farmers' protest. In this case, we have a similar situation to the one with TF-IDF, where the results contain all the query terms but offer little information about the query. They mostly consist of simple phrases that want the release of the activist.

   #results: 628

```
======================
Top 20 results out of 628 ranked documents for query: Disha Ravi

----------------------

Rank: 1 | Score: 1.0000
doc_id = doc_11995
doc_content = #FarmersProtest. Disha Ravi. https://t.co/VbW9rYLUIa

----------------------

Rank: 2 | Score: 1.0000
doc_id = doc_32205
doc_content = #IndiaBeingSilenced
#farmersprotest
Disha ravi https://t.co/OVmvJy3d8V

----------------------

Rank: 3 | Score: 0.9996
doc_id = doc_35462
doc_content = #FreeDishaRavi
#FarmersProtest
Freezer Disha Ravi nowwww https://t.co/PP9KwfiVrD

----------------------

Rank: 4 | Score: 0.9988
doc_id = doc_10044
doc_content = Hats off to Disha Ravi..

#FarmersProtest https://t.co/5QjLp5BVuQ
```

5. **Anime:** The top results contain tweets completely unrelated to anime. This is due to stemming, where the word anime is set to 'anim', leading to results related to 'animals'. As anime has nothing to do with farmers protests, few results should be gathered, which is exactly what happened..

   #results: 25

```
======================
Top 20 results out of 25 ranked documents for query: Anime

----------------------

Rank: 1 | Score: 0.9431
doc_id = doc_1565
doc_content = The guy wearing blue turban is my spirit animal.

#FarmersProtest https://t.co/0iX9YaLNmW

----------------------

Rank: 2 | Score: 0.8632
doc_id = doc_33877
doc_content = Modi stray dogs can bark I m sorry throwing stone on a rabid anim
U can't affect an ounce of my mental peace now keep on commenting
#FarmersProtest #IndiaBeingSilenced

----------------------

Rank: 3 | Score: 0.8499
doc_id = doc_38825
doc_content = Unacceptable!!!   The fact that @DelhiPolice assumed beating an e

----------------------

Rank: 4 | Score: 0.8413
doc_id = doc_23921
doc_content = 'He will win whose army is animated by the same spirit throughout
```

3. Better Representation

This part does not follow a predefined structure, but simply states the models we have tried:

- 3.1 Doc2Vec
- 3.2 Sentence2Vec
- 3.3 Glove
- 3.4 Other models

3.1. Doc2Vec representation

Doc2Vec generates an unique embedding for each document (for each tweet in this case), so it works on a document level, capturing semantic nuances and taking into account the word order. Its main benefit is that it captures context and document semantics, which may not be that relevant in our work case for the poor complexity and short length nature of tweets. More complex contents, such as

research papers, news articles or legal documents could benefit more from this representation. A noticeable weakness is its slow training, as it is quite resource-intensive.

It may seem like an improvement from the previous Word2Vec representation for its ability to capture document-level semantics, but we encountered a memory error when executing due to the computational cost and the size of the dataset. Working with Doc2Vec on a large dataset of short texts involves a significant trade-off, as the complexity benefits may not justify the computational cost, leading to memory issues.

3.2. Sentence2Vec representation

Secondly, we have tried to use **Sentence2Vec representations**, which capture the overall semantic meaning of a sentence (or tweet), unlike Word2Vec, which focuses on individual word meanings, but is still more lightweight than Doc2Vec. Specifically for this task, we believe that Sentence2Vec would perform better, as understanding the full context of the tweets is important.

To implement this, we developed a code similar to the one used for Word2Vec, but using a predefined sentence transformer. Although we expected it to provide better results, we found that it is extremely **computationally expensive**. We spent **30 minutes trying to train the model**, but the training did NOT finish, so we were unable to see the actual results.

Because of this, we believe it is better to use Word2Vec embeddings, as they allow us to obtain results within a matter of seconds, even though they may not be the best or most accurate ones.

3.3. GloVe representation

Additionally to the proposed functions, we have tried to use a **GloVe representation**. GloVe is an algorithm that captures the semantic meaning of words by representing them as vectors in a high-dimensional space. Unlike models like Word2Vec, which focus on local context within a sliding window, GloVe considers **global context**.

The reason for trying GloVe is that, by considering semantic relationships between words, it may provide better word representations.

To implement this approach, we adapted the code used for Word2Vec, but instead of training a model, we used a pretrained GloVe model provided by gensim. Specifically, we used the *glove.6B.300d.txt* model, which contains 300-dimensional word vectors trained on a large corpus of text.

We generate tweet vectors by averaging the vectors of individual words in a tweet, and then compute the similarity between the chosen tweet vectors and the query vectors in the same way as with Word2Vec. The chosen tweet vectors are those that contain ALL the query terms.

A good thing to keep in mind is that GloVe is really fast at creating the embedding vectors and searching the queries.

Results

For GloVe representation, we have tried the same query as in all the previous representations: Disha Ravi. The results were the following ones:

```
Insert your query (e.g., 'Disha Ravi'):

Disha Ravi

======================
Top 20 results out of 628 ranked documents for query: Disha Ravi

----------------------

Rank: 1 | Score: 1.0000
doc_id = doc_28156
doc_content = #FarmersProtest                          Release Disha Ravi..

----------------------

Rank: 2 | Score: 1.0000
doc_id = doc_30467
doc_content = #farmersprotest
Release Disha Ravi https://t.co/QsVdv1V6dH

----------------------

Rank: 3 | Score: 1.0000
doc_id = doc_11995
doc_content = #FarmersProtest. Disha Ravi. https://t.co/VbW9rYLUIa

----------------------

Rank: 4 | Score: 1.0000
doc_id = doc_32205
doc_content = #IndiaBeingSilenced
#farmersprotest
Disha ravi https://t.co/OVmvJy3d8V
```

We can see that the results are really similar to the ones from Word2vec, providing small queries without any meaningful content.

Overall, although it was a good idea, we don't think that GloVe representation outperforms Word2Vec.

3.4. Other representations

To further analyse better embeddings, we could try deep learning models such as BERT, RoBERTa, etc. Due to time constraints, we have not done it, but it could be interesting to try it.