Project Report Part 3

Project GitHub URL: https://github.com/yuyanwang03/IRWA

Project GitHub Tag: IRWA-2024-part-3

The steps required to execute the Python Jupyter Notebook are detailed in the README.md file. To improve the user experience and reduce the execution time, in the part of the project, we put the preprocessed data in a CSV file, which allows the user to skip the preprocess stage and execute this part of the project directly.

Difference TF-IDF and BM25

The TF-IDF (Term Frequency-Inverse Document Frequency) model and the BM25 model take different approaches to scoring relevance based on word frequency and document importance. TF-IDF assigns scores based on the importance of a term within a document relative to a corpus, focusing on words that are frequent within a document but rare across the corpus. This model is straightforward, computationally efficient and does not require parameter tuning, making it useful for identifying relevant content in smaller datasets. However, TF-IDF lacks mechanisms for document length normalization, which can lead to shorter documents being ranked higher even when longer documents are more informative. It also does not account for term frequency saturation, which may reduce its effectiveness in cases where repeated occurrences of a word add diminishing relevance.

In contrast, BM25 (Best Matching 25) is a probabilistic ranking model that builds upon TF-IDF by incorporating tunable parameters for term frequency scaling and document length normalization. It includes diminishing returns for frequent terms, allowing it to rank documents more effectively when term saturation needs to be considered, such as in social media contexts when concise texts are common. BM25 is generally more accurate for relevance ranking due to its flexibility and tunable parameters, making it a preferred choice in information retrieval. However, the model's complexity and need for parameter tuning can be drawbacks, especially in high-speed application. TF-IDF, while simpler, is often sufficient for shorter text data, and its speed makes it suitable for quick relevancy checks in datasets like social media posts.

Take, for instance, the results obtained for the query "support farmer". With TF-IDF, the top-ranked tweets are heavily skewed towards short tweets that repeat the exact terms "support farmer" multiple times. This is because TF-IDF assigns higher scores to terms based solely on their frequency in the document and their rarity across the corpus, without accounting for term saturation. Consequently, TF-IDF ranks tweets with repeated occurrences of the query terms disproportionately high, even if they add minimal additional context or insight. The BM25 model, on the other hand, applies a saturation function on term frequency, reducing the impact of each additional term occurrence. As a result, BM25's top-ranked tweets provide a more diverse set of responses that incorporate different contexts and additional information related to "support farmer", rather than focusing solely on tweets with repetitive keywords. This difference in handling term frequency saturation makes BM25 particularly effective for identifying tweets that offer a broader understanding of the topic, rather than simply matching exact keywords.

Our Score + Cosine Similarity

The custom scoring algorithm combines TF-IDF cosine similarity, social popularity, and recency to rank tweets. Using TF-IDF for textual relevance offers a balance of simplicity and relevance, particularly suitable for short texts like tweets, where term frequency saturation isn't a significant factor. By focusing on term importance within each tweet, TF-IDF effectively identifies content closely aligned with the query. In this algorithm, textual relevance is given the largest weight, as relevance to the query is essential for an accurate ranking.

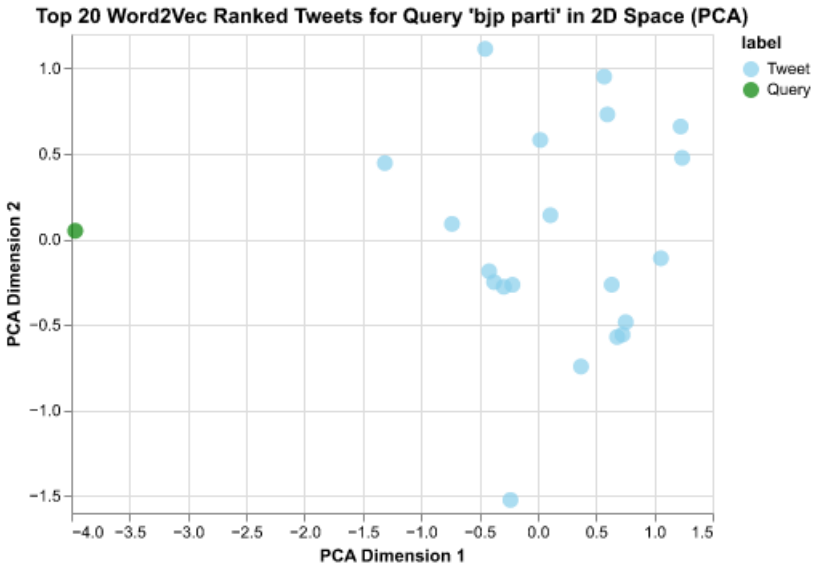
Incorporating social popularity into the score allows the algorithm to consider engagement levels, which can indicate a tweet's quality or influence within the social network. Higher engagement, measured through likes and retweets, is often associated with valuable or popular content, enhancing the algorithm's effectiveness in highlighting impactful tweets. This component is weighted to reflect that engagement is a strong but not sole indicator of relevance; a highly popular tweet may not be directly relevant to the query without further filtering by textual relevance.

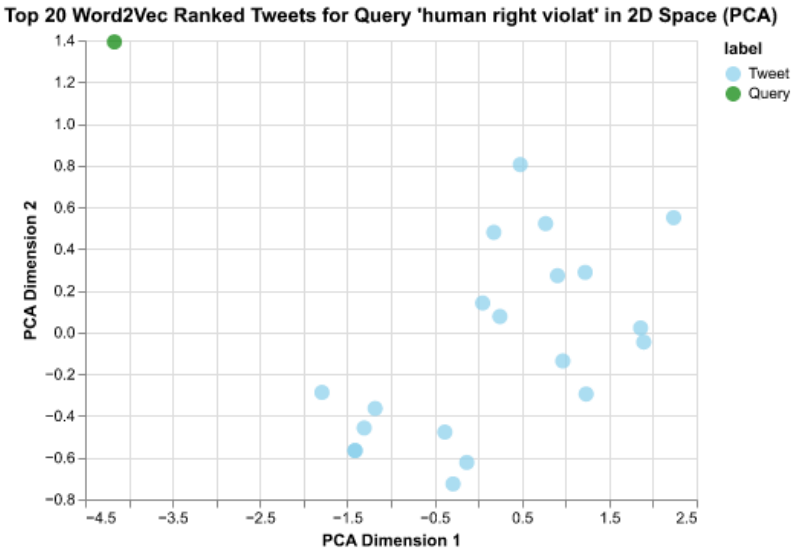
Recency is also integrated into the scoring model, giving more recent tweets a boost in score. This is crucial in dynamic contexts like social media, where timeliness often impacts the relevance of information. Assigning a smaller weight to recency ensures that while recent tweets receive a boost, older, more popular tweets are not overshadowed solely because of their age. Recency is essential for trending topics, but it is balanced here to avoid ranking less relevant new tweets higher simply due to their recency.

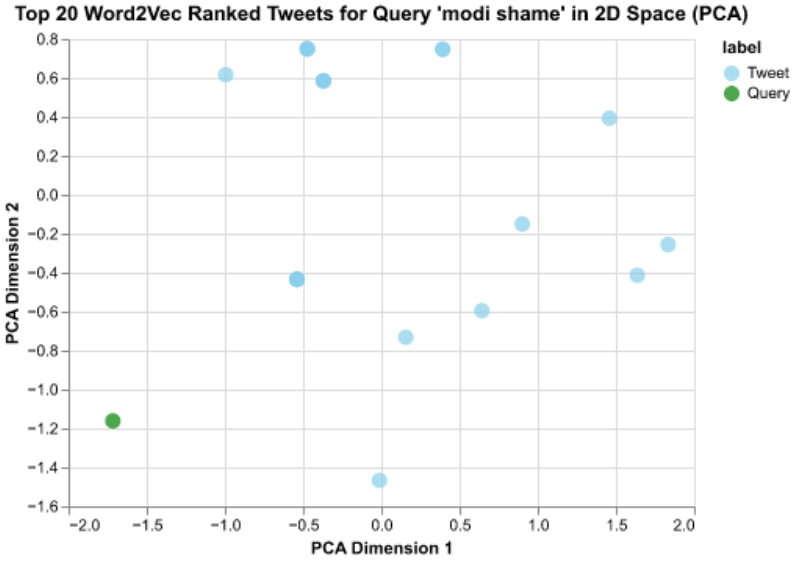
The most obvious limitation about our algorithm is that the weights are manually set (0.55/0.30/0.15), without having fine-tuned them or assessed if that distribution is the optimal one. Besides that, any adjustment to these weights requires testing and tuning to find an optimal balance, and this might not generalize across all queries or dataset; in other words, it is difficult to efficiently adapt it to different situations. Additionally, this algorithm might introduce unintended bias; e.g. new tweets might be ranked over old tweets.

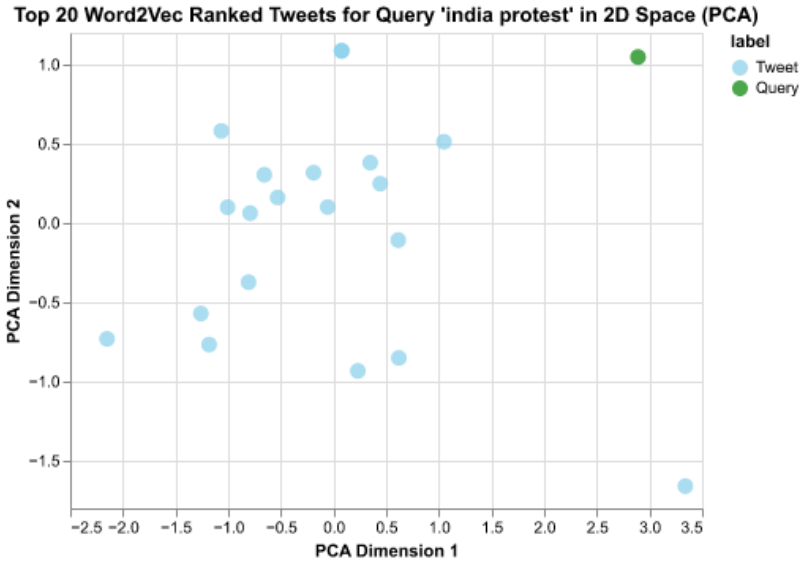
Word2Vec + Cosine Similarity

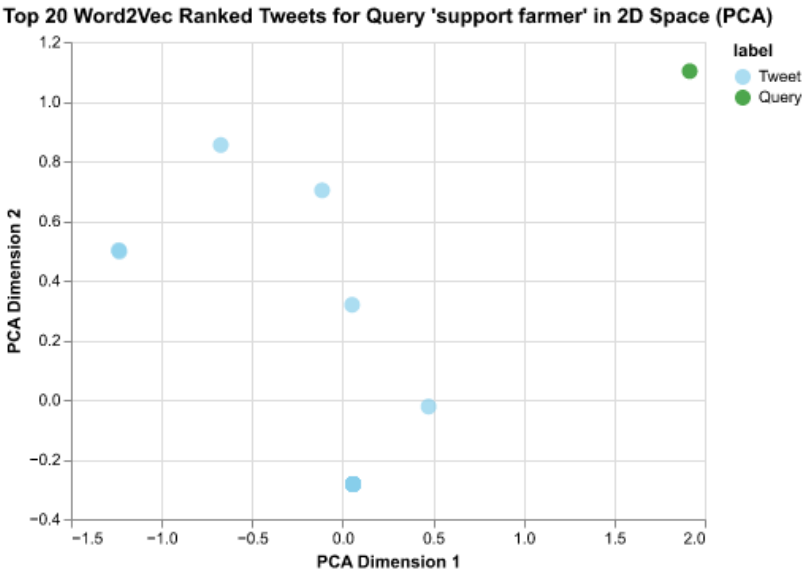
We used Word2Vec together with cosine similarity to rank the results for the five queries we selected in the previous part. For each query a list of the top 20 documents is returned ranked by the relevance based on their vector representations. Word2Vec allows us to transform each tweet into a vector representation by averaging the vectors of the words it contains. These are the visualizations of the model for each of the queries and the obtained result documents.




Top 20 Word2Vec Ranked Tweets for Query 'human right violat' in 2D Space (PCA)
label
Tweet
Query
PCA Dimension 2
PCA Dimension 1
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−4.5
−3.5
−2.5
−1.5
−0.5
0.5
1.5
2.5


Top 20 Word2Vec Ranked Tweets for Query 'modi shame' in 2D Space (PCA)
label
Tweet
Query
PCA Dimension 2
PCA Dimension 1
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
−1.6
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Top 20 Word2Vec Ranked Tweets for Query 'india protest' in 2D Space (PCA)
label
Tweet
Query
PCA Dimension 2
PCA Dimension 1
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5



Better Representation than Word2Vec

Word2Vec is really effective at generating word-level embeddings and in understanding local contexts within the text. However, as the hint suggests, Doc2Vec and Sentence2Vec are the best choice when the task requires to understand the semantic meaning at a larger scale, that is if the focus of the task involves sentence-to-sentence or document-to-document comparisons, either Doc2Vec or Sentence2Vec can offer better representations by capturing additional contextual information.

On one hand, Doc2Vec is an extension of Word2Vec and it is designed to create embeddings at the document level rather than individual words, which allows for the representation of longer texts with a fixed-length vector. The pros of this model are that it captures semantic meaning over entire documents and it preserves word order within documents, which leads to more substantial representations for longer texts. Still, using this model can have some disadvantages, such as the training can be more computationally intensive due to the added complexity of learning paragraph-level representations, and it performs better with larger datasets, so it may not be useful for smaller corpora.

On the other hand, Sentence2Vec is more centered towards capturing the meaning of individual sentences, as it represents sentences as vectors, and aims to keep the semantic context and syntax within each sentence. The advantages that the model offers are a high level of granularity, useful in applications where sentence-level distinctions are important (chatbot responses, summarization, …), and it is faster for tasks where sentences are the main focus. Some cons are that it is less effective for tasks that require document-level understanding as it may miss the larger context and coherentness that is in multiple sentences, and it can miss the hierarchical structure of ideas as it is focused on individual sentence context.

Therefore, comparing the pros and cons of each model, the better representation than Word2Vec depends on the task requirements. On one hand, Doc2Vec is useful for tasks that rely on understanding the overarching themes or sentiments in full documents. On the other hand, Sentence2Vec is better for applications that need a detailed understanding of subtle meanings within individuals sentences.