Jaume Camps Romaguera - NIA 252021
Arol Garcia Rodríguez - NIA 252718
Albert Garrell Golobardes - NIA 254635

Group 101_5

IRWA 2024 Part 3: Indexing and Evaluation

Introduction

The goal of Part 3 in the IRWA project is to evaluate and compare different ranking methods to effectively retrieve tweets related to a query. In the context of social media, it's essential to balance traditional content relevance with contextual factors such as social impact (likes, retweets, and comments) and semantic similarity.

This analysis tests four ranking methods:

1. **TF-IDF + Cosine Similarity**: A classic relevance model that prioritizes term matching within tweets.
2. **Our Score + Cosine Similarity**: A custom ranking that combines TF-IDF with tweet popularity and recency factors.
3. **BM25**: A probabilistic model that incorporates term frequency saturation and document length normalization.
4. **Word2Vec + Cosine Similarity**: A semantic approach using Word2Vec embeddings to capture broader contextual relationships.

The results across these methods will help determine the most effective way to rank social media content, particularly when balancing relevance with popularity in a rapidly changing information landscape.

In our repository GitHub Repository contains both code and documentation for this segment, tagged for tagged for IRWA-2024-part-3.

Ranking 1: TF-IDF + cosine similarity

Objective: To rank tweets based on a combination of TF-IDF similarity between query terms and tweet content, supplemented by BERT-based similarity to capture semantic relevance of tweet hashtags (done in the second part):

Methodology:

1. **TF-IDF Calculation**:
    - **Full Inverted Index**: The tweets are first indexed in an inverted index structure to allow efficient retrieval of documents containing the query terms.
    - **Term Frequency (TF)**: For each term in a document, TF is calculated based on the normalized frequency of the term within the tweet.

- **Inverse Document Frequency (IDF)**: The rarity of each term is captured by IDF, which is higher for unique terms and lower for frequent terms across the corpus.
- **TF-IDF Vectors**: Each tweet is represented as a TF-IDF vector, and the query is similarly transformed into a vector based on its terms.

2. **Cosine Similarity**:
   - Cosine similarity is calculated between the TF-IDF vector of the query and each tweet, using the formula:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

   - Tweets are ranked based on their cosine similarity scores with the query, ensuring higher-ranked tweets have a stronger match with the query terms.
3. **BERT Hashtag Similarity**:
   - To capture semantic relevance, BERT embeddings are used to compare the query with tweet hashtags. Specifically, the query is tokenized into pairs of words, and BERT embeddings for these pairs are compared against tweet hashtag embeddings.
   - **Best Match Calculation**: For each query-hashtag pair, the BERT similarity score is computed. The maximum similarity score is recorded and added as an additional relevance weight to the tweet's TF-IDF score.
4. **Final Score Calculation**:
   - The final score is a combination of the cosine similarity score and the BERT-based hashtag similarity score, with an adjustable weight factor for BERT to balance content relevance with semantic hashtag alignment.

Implementation:

- The *search_tf_idf* function preprocesses the query and retrieves tweets containing all query terms.
- In *rank_documents*, we compute TF-IDF scores and cosine similarity between the query and tweet content. If BERT is enabled (*flag=True*), a secondary similarity score is added based on the hashtag-query pairs.

Example Execution and Observations

- For the query "Support Farmers of India", applying TF-IDF and BERT similarity retrieves tweets containing the query terms, while boosting the scores of tweets with hashtags like #SupportFarmers or #IndiaFarmers.

- BERT similarity helps identify tweets with hashtags that are contextually related but not direct matches, which pure TF-IDF would not capture.

Pros and Cons

- **Pros**:
  - Combines exact term matching with semantic hashtag relevance, enhancing the retrieval of contextually relevant tweets.
  - BERT's embeddings allow for broader contextual interpretation, capturing related hashtags even if terms don't exactly match.
- **Cons**:
  - Adds computational complexity due to BERT embedding generation.
  - Pure TF-IDF + Cosine Similarity lacks depth in meaning; BERT mitigates this but can still miss broader social context aspects like tweet popularity.

The TF-IDF + Cosine Similarity (with or without BERT method) balances term-based relevance with contextual hashtag relevance. It offers strong performance for queries where both exact matches and semantic alignment with hashtags are desirable, making it effective for retrieving relevant tweets with a mix of term specificity and semantic context.

Ranking 2: Our score + cosine similarity

Objective: To improve relevance ranking by integrating TF-IDF-based cosine similarity with additional factors that consider tweet popularity (likes, retweets, comments) and recency, allowing us to prioritize both contextually relevant and socially impactful tweets.

Methodology:

1. **Popularity Score**:
   - Purpose: Tweets with high social engagement (likes, retweets, comments) indicate popularity and impact, making them likely candidates for higher relevance in a social media context.
   - Calculation:

$$\text{popularity\_score} = (\gamma \cdot \text{likes} + \delta \cdot \text{retweets} + \epsilon \cdot \text{comments}) * \text{date}$$

Here, γ, δ and ϵ are weights assigned to likes, retweets, and comments, respectively. The *date* factor (calculated using exponential decay based on tweet recency) emphasizes newer tweets, as they tend to be more relevant in a rapidly changing social media environment.

2. **Final Score Computation**:
    - The final ranking score integrates three components:
        - TF-IDF Score: Reflects term relevance in the tweet content.
        - Hashtag Similarity Score: Measures contextual alignment between the query and tweet hashtags using a BERT-based similarity metric.
        - Popularity Score: Indicates the social impact of the tweet based on engagement metrics and recency.
    - The formula for *final_score* is:

$$\text{final\_score} = \alpha \cdot \text{tf\_idf\_score} + \beta \cdot \text{hashtag\_similarity\_score} + \Gamma \cdot \text{popularity\_score}$$

where α, β, and Γ are weights that adjust the influence of each component based on the desired balance between content relevance, contextual alignment, and social engagement.

Implementation

- The *rank_documents* function processes the query terms, computes the TF-IDF and hashtag similarity scores (if *flag=True*), and then calculates the popularity score based on the engagement metrics and recency factor.
- The ranking step ensures that tweets containing all query terms are included, and scores are computed by applying the respective weights to each component.

Example Execution and Observations

- For the query "Support Farmers of India", applying this ranking function yields tweets sorted by a combined measure of relevance and popularity.
- When the hashtag similarity flag is active (*flag=True*), the model assigns higher relevance to tweets with hashtags aligned to query terms (e.g., #supportfarmers), increasing their final score in the ranking.

Pros and Cons:

- **Pros**:
    - Captures social significance by favoring tweets with high engagement and recent posting dates, which may be more contextually relevant on social media.
    - Increases relevance by focusing on query-hashtag similarity.
- **Cons**:
    - May overemphasize popular tweets even when they are not strictly relevant to the query.

- Requires tuning of multiple weight parameters (α, β, γ, δ, ϵ, and Γ), which can be challenging and data-dependent.

Ranking 3: BM25 + cosine similarity

Objective: To rank tweets based on term frequency, inverse document frequency, and document length normalization using the BM25 formula. BM25 is a probabilistic ranking function widely used in information retrieval because it captures term importance effectively without needing additional similarity metrics like cosine similarity.

Methodology

1. **BM25 Formula**:
   - BM25 scores each document by considering term frequency, inverse document frequency, and document length normalization:

$$\text{BM25\_score}(Q, D) = \sum_{t \in Q} \text{idf}(t) \cdot \frac{f(t, D) \cdot (k_1 + 1)}{f(t, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

   - **Parameters**:
     - k1 adjusts the saturation point of term frequency, controlling the impact of term repetition.
     - b is the document length normalization factor. b=0.75 is commonly used, providing a balanced approach between short and long documents.
     - idf(t): Inverse Document Frequency of term t, calculated similarly to TF-IDF but applied probabilistically.
2. **BM25 in Social Media Context**:
   - Since tweets often vary in length and structure, BM25's normalization feature helps to balance the impact of shorter tweets versus longer ones. Unlike TF-IDF, which assumes a linear relationship between term frequency and relevance, BM25 adjusts for the diminishing returns of term repetition, yielding more stable scores across varying tweet lengths.

Implementation:

- The *bm25_score* function computes the BM25 score for each document against the query, requiring that each document contains all query terms.
- **Document Length Normalization**: *avgdl* (average document length) ensures that the lengthier tweets are not unduly favored, while k1 and b parameters allow us to tune the model's sensitivity to term frequency and length variations.

Example Execution and Observations:

- For the query "Support Farmers of India", BM25 produces a ranked list of tweets that prioritize both query relevance and document length.
- This approach yields highly relevant top results, especially in cases where shorter tweets contain dense, relevant information.

Comparison to TF-IDF:

- **Advantages of BM25**:
    - Better suited for handling tweets with repetitive terms due to frequency saturation.
    - Length normalization ensures that shorter tweets can compete with longer ones, avoiding TF-IDF's potential bias toward longer documents.
- **Disadvantages of BM25**:
    - BM25 can be sensitive to its hyperparameters, k1 and b, which require tuning based on the dataset.
    - It assumes relevance based primarily on content without incorporating contextual factors like popularity or recency, which may be less effective for social media.

The BM25 model provides a robust relevance ranking by balancing term frequency and document length considerations. It is particularly effective in handling varied document lengths and term repetitions, making it suitable for social media data where tweet length and term usage fluctuate widely.

Ranking 4: Word2Vec + cosine similarity

In this section, we use Word2Vec embeddings and cosine similarity to rank tweets based on their relevance to a given query. We use the same library (gensim) we used in the visualization part of the previous part of this project. First, we train a Word2Vec model on the tweet corpus, which learns vector representations for each word based on its context within the tweets.

To rank the tweets, we generate an "average vector" for both the query and each tweet, representing their overall meaning. Cosine similarity is then calculated between the query vector and each tweet vector to determine their similarity. We filter tweets to include only those containing all query terms and sort them by their cosine similarity score, selecting the top 20 most similar tweets for the final output. This method captures semantic similarities, ranking tweets based on both exact words and related context.

To test this model, we use the same queries as in the previous part of the project

4.1 Query 1: Are farmers being respected in India?

- Top 1 similarity score: 0.9656
- Top 5 similarity score: 0.9146
- Top 10 similarity score:0.9039

4.2 Query 2: Are people supporting farmers?

- Top 1 similarity score: 1.0
- Top 5 similarity score: 0.9623
- Top 10 similarity score: 0.9514

4.3 Query 3: Are farmers being respected in India?

- Top 1 similarity score: 0.9980
- Top 5 similarity score: 0.9697
- Top 10 similarity score: 0.9481

4.4 Query 4: Impact of the Protests in India

- Top 1 similarity score: 0.9299
- Top 5 similarity score: 0.9135
- Top 10 similarity score: 0.9023

4.5 Query 5: Protest against Indian Government?

- Top 1 similarity score: 0.9774
- Top 5 similarity score: 0.9514
- Top 10 similarity score: 0.9476

The results from this method show relatively high similarity scores across all five queries, indicating that this approach effectively identifies tweets closely related to the queries in terms of semantic content. For each query, the similarity scores for the top-1, top-5, and top-10 tweets remain above 0.9, suggesting that the model consistently retrieves relevant tweets with high contextual alignment to the given topics.

Comparing results across queries, we observe slight variations in the highest similarity scores, with Query 2 ("Are people supporting farmers?") achieving the perfect similarity score of 1.0 for its top match. Meanwhile, queries like Query 4 ("Impact of the Protests in India") and Query 1 ("Are farmers being respected in India?") have lower top-10 similarity scores, which may suggest a broader or more complex range of tweet contexts and vocabulary associated with these topics, resulting in more moderate similarity matches.

Overall, these high similarity scores demonstrate the method's ability to capture semantic meaning effectively, though the variations also hint that query phrasing and content specificity influence similarity results.

5. Other tests

In order to explore alternative representation methods for ranking tweets based on semantic similarity, we tested several models: Universal Sentence Encoder (USE), Sentence-BERT (SBERT), and Doc2Vec. The following discusses each model's performance, highlighting the challenges faced and overall feasibility.

1. **Universal Sentence Encoder (USE)**[1]: USE is a strong candidate for sentence-level embeddings, often performing well in capturing semantic meaning across varied contexts. However, in practice, USE was computationally intensive, taking an impractically long time to process the full set of tweets. While USE could potentially yield high-quality results, the time required for embedding each tweet was not feasible within the constraints of this task.

2. **Sentence-BERT (SBERT)**[2]: SBERT, which builds upon BERT's transformer-based architecture to generate sentence embeddings, is typically highly effective for tasks requiring sentence-level similarity calculations. We already used BERT models in our project, that is why we wanted to test another model more specific for this task. Unfortunately, similar to USE, SBERT also took an excessive amount of time to generate results across all tweets. This limitation, likely due to the resource-intensive nature of transformers, rendered SBERT impractical for this project's time-sensitive needs.

3. **Doc2Vec**[3]: Doc2Vec is designed to generate vector representations for entire documents or sentences, making it suitable for embedding tweets. Compared to word2vec, Doc2Vec was computationally efficient and considerably faster to run. However, the results obtained from Doc2Vec proved to be of lower quality, with embeddings that did not capture tweet relevance as effectively as expected. This might be due to the brevity and specific nature of tweet content, which Doc2Vec may struggle to represent accurately in comparison to other models designed for sentence-level or short-text data.

Note: Information has been extracted from the resources below.

[1] https://github.com/tensorflow/tfjs-models/blob/master/universal-sentence-encoder/README.md
[2] https://sbert.net/
[3] https://radimrehurek.com/gensim/models/doc2vec.html