TweetPress - News Recommendation System

Manas Agarwal
IIIT Delhi
manas20443@iiitd.ac.in

Kartik Jain
IIIT Delhi
kartik20440@iiitd.ac.in

Uttkarsh Singh
IIIT Delhi
uttkarsh20479@iiitd.ac.in

Rishabh Oberoi
IIIT Delhi
rishabh20459@iiitd.ac.in

Darsh Parikh
IIIT Delhi
darsh20560@iiitd.ac.in

21 March 2023

1 Introduction

1.1 Motivation and Problem Statement

Reading the news online has exploded as the web provides access to millions of news sources worldwide. The sheer volume of articles can be overwhelming to readers. A key challenge of a news service website is to help users find news articles that match their interests. Personalised News Recommendation is advantageous to both users and the news service, as the user can quickly find what he or she needs, and this helps the news service retain and increase the customer base.
Twitter shows us trending news based on demographics and trends because of this it has become the main news source for many people. The problem with this type of recommendation is that the recommendations are not personalised to the user.
Another problem with current news platforms is that they tend to show polarised news articles which prevents people to know about the other side of the story as well before coming to any conclusion about something.

1.2 Novelty

The novelty of our project lies in creating a personalized news recommendation system that takes into account the user's interests based on their tweets and recommends relevant articles. For, the model implementation we use different information retrieval techniques. Most users use Twitter to get daily updates on what's going around and our model helps them to get personalised news articles that they might like or might be looking for based on their tweets. We also use keyword extraction, similarity matching and sentiment analysis to shortlist the list of relevant news articles for a user and show him the top 'k' relevant articles of both parities to produce non-polarised results.

1.3 Dataset Description

We scraped out a user's tweet using snscrape python library. For further use, the retrieved tweets were preprocessed, and hashtags and keywords were separated. For the news articles dataset, we used Newsapi.org and collected articles on different topics and for the mid-project, about 1800 news articles were collected. We have collected tweets from various famous personalities and tried recommending the news articles based on them.

1.4 Proposed Solution

Creating a personalised news recommendation system that considers users' interests based on their tweets and recommends relevant articles. Most users use Twitter to get daily updates on what is happening, and our model helps them get personalised news articles that they might like or might be looking for. For the model, we consider different factors like keyword extraction, similarity matching and sentiment analysis. Based on all the above mentioned, we shortlist the list of relevant news

articles for a user and show him the top 'k' relevant articles of both parities to produce a non-polarised result.

2 Literature Review

The paper titled "Discovering significant news sources in Twitter"[1]proposes a method to discover significant news sources on Twitter. The authors highlight the importance of identifying reliable and trustworthy news sources on Twitter, given the abundance of information available on the platform. They have made a framework which downloads a list of news-related relevant tweets from twitter, extracts URLs associated with those tweets and infers the significance of those URLs in Twitter.They collected tweets using the REST and Streaming API of Twitter, and then extracted URLs from these tweets. Further, the tweets have been ranked based on relevancy. They used a threshold of 15% similarity between the headline and the tweet, thus finding news-related relevant tweets and the corresponding news topics.

Comprehending and summarizing extensive texts can be challenging due to the ever-increasing volume of information. To address this issue, keyword extraction systems are necessary to automate the process of generating keywords because large amounts of text often lack descriptive terms. The paper titled "YAKE! Keyword extraction from single documents using multiple local features"[2] presents an unsupervised automated keyword extraction approach known as YAKE that is not reliant on the dataset or its description. By utilizing statistical data and local text features like term frequencies, the authors can identify significant keywords without requiring a corpus.

In their paper, Bordoloi et al.[3] presented a modified graph-based approach called KWC for automatic keyword extraction from a Twitter corpus. Their approach used various graph measures, including frequency, centrality, position, strength of the neighbors, and others, to assign weights to nodes in the graph. The control flow of their method included data preprocessing, textual graph representation, collective node-weight assignment, and keyword extraction. The nodes in the graph were tokens, and an edge was created between two tokens if they co-occurred within the same tweet. The graph was weighted according to co-occurrence frequency, and the nodes were weighted based on important graph measures such as term frequency and selectivity centrality. Keyword extraction was performed using NE-Rank and degree centrality, and the best n keywords were outputted. Bordoloi et al.'s method outperformed several existing graph-based methods and could be useful for our project, which involves extracting keywords for a given set of tweets for a hashtag.

Krestel et al.[4] conducted a study where they developed models to suggest relevant tweets for news articles, and evaluated their effectiveness through user feedback. To identify tweets that contained similar content to the articles, they used language and topic models to compare word and concept overlap. They then combined these similarity scores with other tweet features such as recency and popularity, using logistic regression and boosting. For language modeling, they used a document likelihood model to determine the likelihood that a tweet was generated from a news article, rather than a query likelihood model. Additionally, to prevent duplicate tweets in the final ranking, they excluded tweets that had a high word overlap with tweets that were already included in the set of recommended tweets.

3 Methodology

We take TwitterID, Number of News articles (X), the Weighting Scheme and the Keyword Extraction Method from the user. After applying the models based on the weighting scheme mentioned by the user we give the output as top X articles. If the user has chosen a keyword extraction method then we use that specific method and sentiment analysis to retrieve X different news articles of both similar opinion as the user (according to the user's tweets) and X news articles of the opposite opinion.

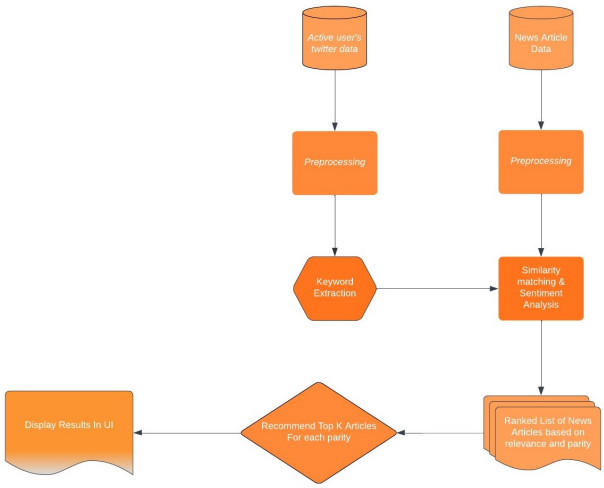

Figure 1: Workflow of our Model

3.1 Working of the Model

1. Initially we extract the tweets of the user for the last 6 months. Then we make a corpus for the user's tweets.
2. Due to limited resources, we could not make a dynamic model, so we made different categories or tags for news articles like AI, Tech, Business, Climate etc. We used newsapi.org to extract the news articles and their categories.
3. We treat a news article and its corresponding tag as a separate document and associate it with a link.
4. After creating these documents, we make TF-IDF of all these documents based on the evaluation metric chosen by the user.

5. Made a query vector using the corpus of the user's tweet made above. The size of the query vector will be the same as the TF-IDF matrix and it will contain the words that will be present in the intersection of the vocabulary of TF-IDF and the tweet corpus.

6. We have used the Cosine Similarity to find the similarity between the query vector and the news articles.

7. Then we used different weighting schemes and keyword extraction to recommend the news articles to the user.

3.2 PreProcessing

To achieve better results we have performed preprocessing steps on our data. The following steps were performed on both, user tweets and news articles data :

1. Removed punctuation marks
2. Converted the text to lowercase
3. Performed tokenization
4. Removed stopwords from the tokens
5. Removed single alphabets
6. Performed lemmatization

3.3 Keyword Extraction Method

We used the following Keyword Extraction methods along with sentiment analysis:

1. **RAKE(Rapid Automatic Keyword Extraction):** This algorithm is not specific to any particular domain and is based on the observation that natural language has many unnecessary words, such as stopwords and punctuation, that do not contribute much to the meaning and context of the text. To extract the candidate keywords, these extra words are removed and the remaining words are assigned a score based on their degree and frequency in a co-occurrence graph.
2. **YAKE(Yet Another Keyword Extractor):** This method is a simple and automated way to extract keywords from text, using statistical features. It is independent of the language and the corpus of the text being analyzed. Furthermore, it can identify the important keywords in a single document.
3. **KeyBERT:** BERT is a type of transformer model that is capable of transforming phrases and documents into vectors that represent their meaning. KeyBERT is a straightforward and user-friendly technique for extracting keywords from text, which relies on BERT-embeddings and cosine similarity to identify the most relevant sub-documents within a larger document. By leveraging BERT, KeyBERT can generate document-level representations that facilitate the extraction of meaningful keywords.
4. **Textacy:** It provides several algorithms for extracting important words or phrases from text, such as the TextRank and SingleRank algorithms, and supports multiple languages. It also offers tools for preprocessing text and filtering out unwanted words. Overall, Textacy's keyword extraction module can help simplify the task of identifying important topics in large volumes of text data.

3.4 Evaluation Metric

We have used the following evaluation metrics in our model:

1. **TF-IDF similarity matching:** TF-IDF is a statistical measure that helps determine the significance of a term in a document or corpus. It increases in proportion to the number of times a term appears in a document. To calculate it, we need to compute the Term Frequency, which is the number of times a word appears in each document, as well as the Inverse Document Frequency, which is the reciprocal of the number of documents in which a term appears. TF-IDF is calculated by multiplying these two statistics together. Five different TF-IDF matrices are created using five different weight calculation methods.

    - The first method is **binary weighting** where term frequency is either 1 or 0, depending on whether the term is present in the document or not.
    - The second method is **raw count**, which uses the number of times the term appears in the document.
    - The third method is **term frequency**, which normalizes the raw count of a term by the sum of the raw count of all terms in the document.
    - The fourth method is **log normalization**, which takes the logarithm of (1+raw count of the term).
    - Finally, the fifth method is **double normalization**, which uses the formula 0.5+0.5(raw count of the given term/ maximum raw count in the document).

2. **Cosine Similarity:** Cosine similarity is a widely used method in natural language processing to determine the similarity between two texts, regardless of their length and size. The documents are represented in n-dimensional vector space, where the cosine of the angle between the vectors is calculated to measure the similarity. The resulting value ranges from 0 to 1, with values closer to 1 indicating high similarity and values closer to 0 indicating low similarity. To calculate the word occurrence for each document, either Count Vectorizer or TF-IDF Vectorizer is used. The vocabulary includes all the terms in the query and the document being compared. Using binary value vectors, the dot product is calculated to report the similarity score. The top k documents with the highest similarity scores are then identified. We used two metrics - TF-IDF and Count Vectorizer - where each query and document is converted into vectors and compared to calculate similarity scores.

4 Evaluation and Results

4.1 Baseline Results

4.1.1 Greta Thunberg

Weighting Scheme	Precision
Binary TF	*0.24*
Raw Count	*0.3*
Term Frequency TF	*0.3*
Log Normalization TF	*0.26*
Double Normalization TF	*0.28*

Precision @20

4.1.2 Virat Kohli

Weighting Scheme	Precision
Binary TF	*0.2*
Raw Count	*0.3*
Term Frequency TF	*0.3*
Log Normalization TF	*0.3*
Double Normalization TF	*0.325*

Precision @20

4.2 Final Results

4.2.1 Narendra Modi

Weighting Scheme	Precision
Binary TF	*0.8*
Raw Count	*0.5*
Term Frequency TF	*0.5*
Log Normalization TF	*0.7*
Double Normalization TF	*0.5*

Precision @20

4.2.2 Sundar Pichai

Weighting Scheme	Precision
Binary TF	*0.6*
Raw Count	*0.6*
Term Frequency TF	*0.7*
Log Normalization TF	*0.6*
Double Normalization TF	*0.6*

Precision @20

4.2.3 Samay Raina

Weighting Scheme	Precision
Binary TF	*0.5*
Raw Count	*0.7*
Term Frequency TF	*0.7*
Log Normalization TF	*0.7*
Double Normalization TF	*0.6*

Precision @20

4.2.4 Greta Thunberg

Weighting Scheme	Precision
Binary TF	*0.5*
Raw Count	*0.5*
Term Frequency TF	*0.6*
Log Normalization TF	*0.5*
Double Normalization TF	*0.4*

Precision @20

4.2.5 Virat Kohli

Weighting Scheme	Precision
Binary TF	*0.6*
Raw Count	*0.8*
Term Frequency TF	*0.8*
Log Normalization TF	*0.7*
Double Normalization TF	*0.8*

Precision @20

4.3 Keyword Extraction Result

4.3.1 Similar Motion

4.3.2 Narendra Modi

Keyword Extraction	Precision
YAKE	*0.3*
RAKE	*0.2*
KeyBERT	*0.5*
Textacy	*0.7*

Precision @10

4.3.3 Greta Thunberg

Keyword Extraction	Precision
YAKE	*0.7*
RAKE	*0.2*
KeyBERT	*0.4*
Textacy	*0.3*

Precision @10

4.3.4 Sundar Pichai

Keyword Extraction	Precision
YAKE	*0.3*
RAKE	*0.5*
KeyBERT	*0.3*
Textacy	*0.3*

Precision @10

4.3.5 Samay Raina

Keyword Extraction	Precision
YAKE	*0.5*
RAKE	*0.3*
KeyBERT	*0.4*
Textacy	*0.3*

Precision @10

4.3.6 Opposite Motion

4.3.7 Narendra Modi

Keyword Extraction	Precision
YAKE	*0.4*
RAKE	*0.3*
KeyBERT	*0.6*
Textacy	*0.2*

Precision @10

4.3.8 Greta Thunberg

Keyword Extraction	Precision
YAKE	*0.5*
RAKE	*0.1*
KeyBERT	*0.4*
Textacy	*0.2*

Precision @10

4.3.9 Sundar Pichai

Keyword Extraction	Precision
YAKE	*0.4*
RAKE	*0.4*
KeyBERT	*0.4*
Textacy	*0.3*

Precision @10

4.3.10 Samay Raina

Keyword Extraction	Precision
YAKE	*0.4*
RAKE	*0.3*
KeyBERT	*0.3*
Textacy	*0.2*

Precision @10

References

[1] S. Ahuja, "Discovering significant news sources in twitter," in *2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pp. 434–438, 2015.

[2] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "Yake! keyword extraction from single documents using multiple local features," *Information Sciences*, vol. 509, pp. 257–289, 2020.

[3] M. Bordoloi and S. Biswas, "Keyword extraction from micro-blogs using collective weight," *Social Network Analysis and Mining*, vol. 8, p. 58, 09 2018.

[4] R. Krestel, T. Werkmeister, T. P. Wiradarma, and G. Kasneci, "Tweet-recommender: Finding relevant tweets for news articles," WWW '15 Companion, (New York, NY, USA), p. 53–54, Association for Computing Machinery, 2015.