NewsTweet - Information Retrieval

Ranking tweets based on News articles

Tan Chia Yik (1003391), Tan Dao Rong, Eugene (1003442)

August 2021

1 Abstract

Twitter is a popular social media platform, otherwise known as "micro-blogging" platform that allows users to send and receive short posts called tweets. Each tweet is allowed to be only 140 characters long. In this 140 characters, the user can choose to insert a link or even tag other users using the same platform.

With an estimated 199 million users worldwide, there are an average of 500 million tweets posted every day. This gives rise to the need of a search engine to accurately query tweets related to an event. Given the shortness of each tweet, users find creative ways to fit more information into the limited amount of characters. This may come in the form of acronyms, slangs, and broken forms of languages. The semantic nature of tweets presents a major hurdle in accurately retrieving relevant tweets given a query.

This paper serves to investigate various information retrieval methods in improving search results of tweets to provide accurate results in the shortest time possible. It begins by describing the dataset involved, and various pre-processing methods utilised to clean the data. Following which, the implementation of various retrieval methods are described within the context of tweets. Their accuracies and time performances are then evaluated, with Normalized Discounted Cumulative Gain (NDCG) as an evaluation metric. The paper ends by describing shortcomings for the various search methods, possible future works, and concluding the best information retrieval method for this project.

2 Introduction

Twitter is largely used as a means of finding timely information. This information includes, but not limited to, news articles and updates. Tweets that are related to news articles can be useful in analysing an events context. Therefore, finding, ranking, and aggregating tweets about a particular news article is helpful to understand popularity and virality of an article. This platform is not limited to users, and various large organisations often use it as a means of providing real time updates to their followers. As social media evolves, it has become more important than ever for large companies to establish a social media presence, and accurately gather feedback and response towards a subject matter.

Tweets appear largely different from common English text and phrases and often contain acronyms and other broken forms of languages. Traditional search and ranking methods have been proven to not work well with texts that contains contextual meanings which are best interpreted by humans. Traditional ranking methods such as tf-idf usually rely on exact matches with words itself and do not accurately capture the context which the word was used in. As such, traditional search engines are not optimised to understand the semantics of tweets.

With the introduction of Neural Information Retrieval methods, search engines will be able to capture contextual information and return more accurate and precise results. Models such as Bidirectional Encoder Representations from Transformers (BERT) have shown promising results in sentiment analysis for semantic texts. Since BERT model is able to capture the semantics of texts, it would likely be useful in searching for tweets, using it as an retrieval method that retains the semantics of tweets. This paper shall compare the efficacy of different retrieval methods including both traditional and the newer neural retrieval methods. In this paper, documents refer to tweets, while queries refer to article titles.

3 Dataset Details and Preprocessing techniques

3.1 Dataset: Signal-1M Related Tweets Dataset

The dataset used in this paper is the Signal-1m Related Tweets Dataset. This dataset comes as a Comma-Separated Values file containing 4 columns, article id, iteration, tweet id, and relevancy score. The iteration column here is not used, hence dropped. The relevancy score is divided into 3 possible scores 0, 1, and 2, with 0 denoting not relevant, 1 denoting somewhat relevant, and 2 denoting highly relevant.

From the dataset CSV file, we have access to the articles and its titles that we filtered from the Signal-1M dataset directly. Although the dataset contains different meta information for each article, we only utilised the article titles as queries for the search engine, as search queries are often short and dense. Using Twitter Developer API and Python Tweepy library[1], we used the Tweet ids provided by the Signal-1m dataset to scrape the tweets text. Due to privacy concerns and some tweets being deleted, not all Tweets could be scrapped, and on average, for every 10 tweet ids provided, 8 original tweets were retrieved. All these information was than stored into 2 new CSV files. The first CSV contains all the article titles with its respective IDs, whilst the second CSV contained all the tweets, tweets IDs and its texts alongside its related article ID and relevance score. The latter CSV was then used for text pre-processing on tweets.

[1]Tweepy library can be found at https://docs.tweepy.org/en/stable/

3.2 Pre-processing of Tweets

The text of tweets scrapped were then pre-processed before being added into the dataset as another column. The pre-processing steps included tokenizing, removing white spaces, converting to lower cases, and stemming to its root word. Removing of stopwords is another commonly used text pre-processing technique. However, considering that each tweet is only a maximum of 140 characters long, removing stopwords may not be ideal as it may remove too many words, compromising on the semantics of the tweet. The below tables show sample rows from the post-processed dataset.

id	**published**	**source**	**title**
5a07213b-7e81-4e0c-9dfa-758ae4c1ae3c	2015-09-30T02:26:54Z	Dominion Post	Mars sparks Antarctic interest
68047331-25e1-4af5-974b-766f1338bef0	2015-09-06T13:31:36Z	The Independent	Italian Grand Prix 2015 report: Lewis Hamilton feels little pressure despite tyre probe

Table 1: Sample dataset containing article ids and titles

Article ID	**Tweet ID**	**Relevance Score**	**Tweet**	**Clean Text**
00ecb565-8dcb-443e-875c-71babdce2269	638637000000000000	0	Video: Shippers look to grab a piece of Northwest Passage http://t.co/oXfsajtBtl http://t.co/RwvGKG3rue	video shipper look to grab a piec of northwest passag http http
00ecb565-8dcb-443e-875c-71babdce2269	638728000000000000	1	Police say woman was driving drunk with kids in the car, and drugs in the car. Oh, and she was pregnant. http://t.co/LlzxxQOuY2	polic say woman wa drive drunk with kid in the car and drug in the car oh and she wa pregnant http

Table 2: Sample dataset containing tweetids, articleids, original tweet text and post-processed text

4 Retrieval Methods

4.1 Term Frequency-Inverse Document Frequency, with Cosine Similarity and Euclidean Distance

4.1.1 Tf-Idf

Term Frequency-Inverse Document Frequency (Tf-Idf) is one of the methods used to index the dataset. Tf-Idf is a technique to quantify a word in documents. The weight given to the word signifies the importance of the word relative to the document and the the entire corpus.

Tf measures the frequency of the given word in a document then divided by the total number of words in the document. This assumes that a greater number of occurrences of the word means that the word has greater importance. Idf measures the informativeness of a term and reduces the overall weights for stop words. Smoothing is used in idf calculation to prevent a division by zero error.

Tf-idf is calculated for each word in a document, following which, a document is vectorized based on the various Tf-Idf values for its vocabulary. With vectorized documents, common comparison methods such as Cosine Similarity and Euclidean Distance can be applied to obtain the most similar vectors.

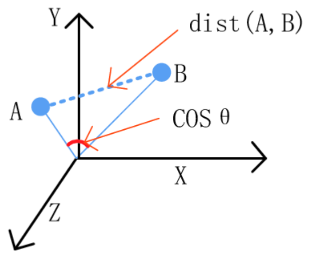


Figure 1: Vector Space Showing Cosine Similarity and Euclidean Distance

4.1.2 Cosine Similarity

Cosine Similarity is one metric used to compare the similarity between the query and the tweets. The query is first vectorized in a similar fashion with the documents mentioned in the corpus, by calculating the Tf-Idf values for each word within the query. The query vector is then compared to all other document vectors. Each comparison returns a Cosine Similarity score, which is the angle between the query vector and the document vector as seen in Figure 1. The closer the score is to 1, the more similar the 2 vectors are. This means that the higher the cosine similarity score for each document vector, the more similar it is to the query, therefore the document is likely more relevant.

4.1.3 Euclidean Distance

Similar to Cosine Similarity, Euclidean Distance is another metric to score the similarity between 2 vectors. Euclidean Distance is the shortest distance between 2 vector points relative to an origin. As seen in Figure 1, the Euclidean Distance between point A and B is shown by dist(A, B). Similarly, the Euclidean Distance between 2 point returns a score, where the smaller the score, the more similar the 2 vectors are, thus more relevant.

4.2 Okapi-BM25

Apart from traditional ranking methods, we also made use of Okapi BM25, which is a probabilistic retrieval method. BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms in each document, regardless of their proximity within the document.

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Figure 2: BM25 formula

The formula in Figure 2 describes the usage of BM25, where f(q, D) represents the term frequency of a term within a document, —D— is the length of the document D, avgdl is the average document length. IDF is the inverse document frequency, to reduce the weight of words that appear frequently. k and b are hyper parameters, and were set to 1.2 and 0.75 in our implementation.

4.3 Word2Vec Embedding Representations

Word2Vec was the first neural information retrieval method that we explored, and is an algorithm that uses a neural network to learn word associations in a large corpus of texts. As the name implies, the Word2Vec algorithm transforms each word into a vector representation. Word2Vec takes in a large corpus of text as input and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus assigned a corresponding vector. Unlike neural models such as BERT, Word2Vec models are usually shallow, 2 layer neural networks.

4.3.1 Implementation

We used the gensim library to help us with our implementation[2]. When the server first starts, the word2vec model processes the entire corpus of text, creates a model and trains it. Considering that we aimed to predict the relevance of

[2]More details about gensim and its word2vec usage can be found at- https://radimrehurek.com/gensim/models/word2vec.html.

a tweet given a word sentence, we used the continuous bag of words (CBOW) model architecture. The table below describes our various parameter inputs.

Parameter Name	**Chosen Value**	**Description**
vectorSize	100	Dimensionality of word vectors
window	3	max window size of words around it
minCount	0	minimum count of words to be considered as an input
sg	0	continuous bag of words

Table 3: Description of Parameter Inputs

We choose a large vector-size of 100 as we wanted to create an accurate a model as possible. The minimum count of words and window size was set to 0 and 3 respectively, as the word count of many tweets is often short. Once the query was vectorized, documents were then assigned a Cosine Similarity Score to the query vector, following which, a ranked list of documents was returned.

4.4 Query Likelihood Model

The Query Likelihood Model is another probabilistic retrieval method that ranks the documents based on the probability of how similar they are to the query. It scores each document according to the probability given by the language model and then ranks them in descending order. In this case, the article title is query and the tweets are the documents. For every term in the article, it is checked against the term's probability in each tweet. The probabilities of every term is then combined for each tweet, and the tweet with the highest probability is the most relevant one. As terms in the article title may not necessarily appear in the tweet, add-one smoothing is applied to the terms. This is to prevent zero probability when combining the probabilities of each document

4.5 Pairwise Learning to Rank

The Pairwise Learning to Rank model is a Neural Information Retrieval method where it uses a Recurrent Neural Network (RNN) to learn the difference between a relevant and irrelevant document. In this method, labelled data is used to learn the differences. To learn the weights, the dataset is split into train, test, and validation sets. The train set is used to learn the model's weights, validation set to validate the weights and minimise loss, and the test set to test the model's accuracy.

During learning, each relevant tweet to an article is paired with an irrelevant document. As every article has multiple relevant and irrelevant articles, the same article title will be tagged to each pair of relevant and irrelevant documents and then used to learn the differences between the two. Figure 3 shows the ranking model.

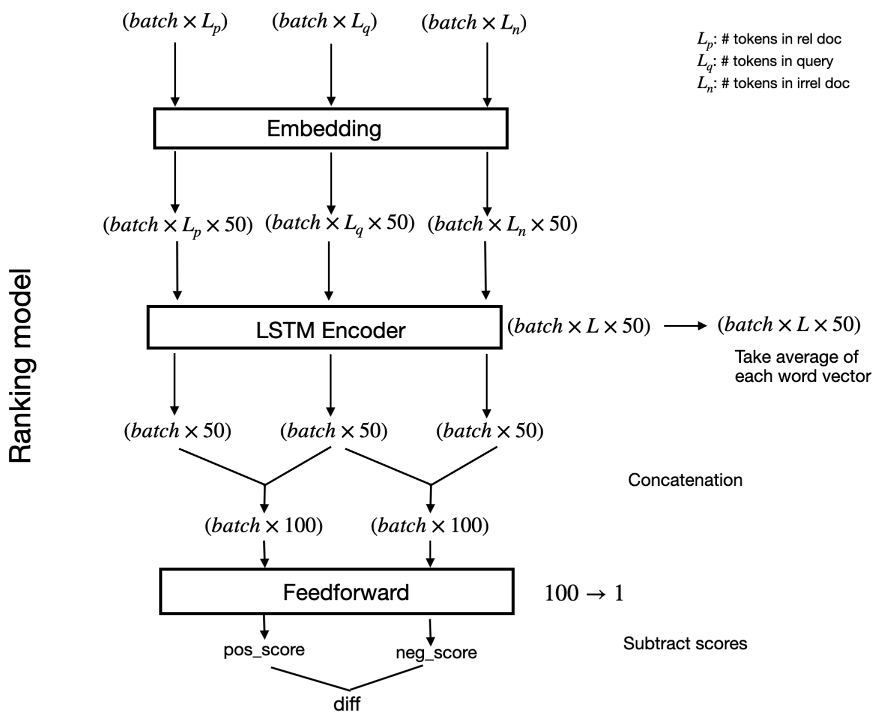


Figure 3: Vector Space Showing Cosine Similarity and Euclidean Distance

For each 3-tuple of article title, relevant tweet, and irrelevant tweet, they are passed through an embedding layer to convert the words into numerical form such that the features could be learnt, as seen in Figure 3. From the output of the embedding, they are passed through a Long Short-Term Memory (LSTM) module. The LSTM module is used to learn the contextual information of the inputs, this allows the model to learn the semantic features of the article titles as well as the tweets. After passing through the LSTM Module, the output for the article title is concatenated to both the relevant and irrelevant tweets output, before putting into a feedforward layer to calculate a positve score (relevant tweet) and a negative score (irrelevant tweet). The difference between the scores if then used as the loss function and as well as differentiating between the tweets.

Once the weights of the model has been learnt, the weights can be used to predict the relevance of a tweet to the article title. With a ranking function, the relevance scores of all the tweets can be retrieved and sorted where higher scores represent relevance and lower score represent irrelevance. The scores are ranked from high to low, then returned as a list of scores relative to the tweets.

4.5.1 Implementation

Although Pairwise learning to rank produced relatively good results, the speed of each query was slow. Therefore, to cut down on query time whilst maintaining the accuracy of the model, we decided to first retrieve 100 tweets with the help of Tf-Idf and Cosine Similarity, following which Pairwise learning to rank was used to re rank the top 100 tweets. The speed of retrieval was cut down significantly, by about 50 times. This implementation was also implemented in our BERT & Tf-Idf search method, and will be presented in greater details in section 4.6.2

4.6 BERT & tf-idf with Cosine Similarity

In an another attempt to evaluate the accuracy of Neural methods in retrieving relevant tweets, we included a Bidirectional Encoder Representations from Transformers (BERT). BERT was recently published by researchers at Google AI Language. It has shown promise in a wide variety of NLP tasks, including Question Answering), Natural Language Inference. BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. Various research results have shown that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.

4.6.1 Model Background

To achieve our objective, we made used of the python sentence-transformers library[3]. To aid us in the model implementation, we made use of the paraphrase-multilingual-mpnet-base-v2 pre-trained model, which was trained on a wide variety of training data, such as stack-overflow-duplicate-questions and simple-wiki, and was optimised to identify paraphrases. We chose this model as it was identified as the model that best recognized paraphrases within tweets.

4.6.2 Implementation

While neural information retrieval methods are often praised for the accuracy of their results, they are also often criticised for the time trade-off. In our initial model implementation, the query and the documents after encoded with the aforementioned BERT model, following which, the output vectors were reshaped and a Cosine Similarity score was assigned. Although this naive implementation produced relatively accurate results, it took upwards of 5 minutes for a single run, roughly 100 times the amount of time compared to traditional ranking methods such as Tf-Idf and VSM. Therefore, to cut down on query time whilst maintaining the accuracy of the model, we decided to first retrieve 100 tweets with the help of Tf-Idf and Cosine Similarity, following which BERT was used

[3]More information about the library can be found here https://www.sbert.net/

to re rank the top 100 tweets. With our new implementation, the run time of our model was cut down significantly to roughly 16.4 seconds, as will be further described in section 6 of the paper.

4.6.3 Shortcomings of our implementation

Perhaps the greatest shortcoming of our model was that we were unable to accurately fine tune our model. Due to the time constraints and the small training size of our dataset, it was unlikely that we would produce good results with finetuning. Furthermore, we could not find a pre-trained BERT model that was specifically trained on a twitter dataset, and we had to resort to using a more generic model. With that said, we are still pleased with the outcome of our BERT implementation, and we are convinced that it is a model that could potentially produce accurate results should it be further developed.

5 Query Evaluation Metrics - Normalized Discounted Cumulative Gain

To evalute the effectiveness of the various search methods, we decided to use Normalized Discounted Cumulative Gain (NDCG). We decided to use NDCG@20 as a ranking metric as the dataset has a relevance scale for the relevance of tweets to a article, and NDCG is a useful metric when there is a relevance scale for the output labels. Therefore, when a query returns a ranked list of outputs, it is passed through a function that calculates the NDCG value for the top 20 results.

6 Ranking of different evaluation methods

In an attempt to discover the best query method for retrieving tweets, we took the NDCG score over 50 different queries. Our approach was as such, within our dataset, we first shuffled the dataset, following which, we randomly sampled 50 different article titles and article ids. For each article title within the test dataset, a retrieval score was calculated for each query method, an added to an array. The mean values of the array calculated, and can be found in the table below. It is important to note that the average query time is merely an estimate of the speed of a search model and not the true reflection of its speed. This is because certain search methods such as VSM were able to leverage on existing python libraries such as gensim, and the authors of these libraries have painstakingly optimised the speed of their libraries. In contrast, other search methods such as Okapi BM-25 were manually constructed, and do not benefit from similar time optimisations.

Query Method	**Average NDCG Score (3sf)**	**Average Query Time, in seconds (3sf)**
tf-idf with Cosine Similarity	0.589	9.6
tf-idf with Euclidean Distance	0.150	6.85
Okapi BM25	0.0418	5.42
Word2Vec with Cosine Similarity	0.534	1.67
Word2Vec with Euclidean Distance	0.00578	1.35
Query Likelihood Model	0.447	5.01
BERT with tf-idf	0.663	16.4
Okapi BM25	0.0418	5.42
Pairwise Learning to Rank *tested on 15 data samples instead of 50	0.371	12.5

Table 4: Average NDCG Scores and Query Time Across Methods

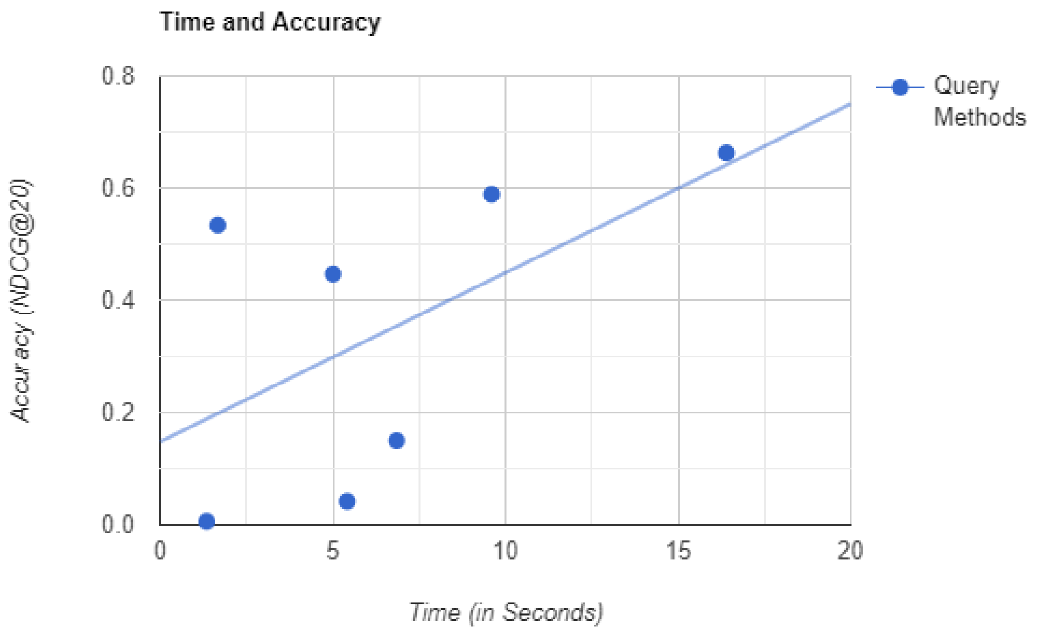
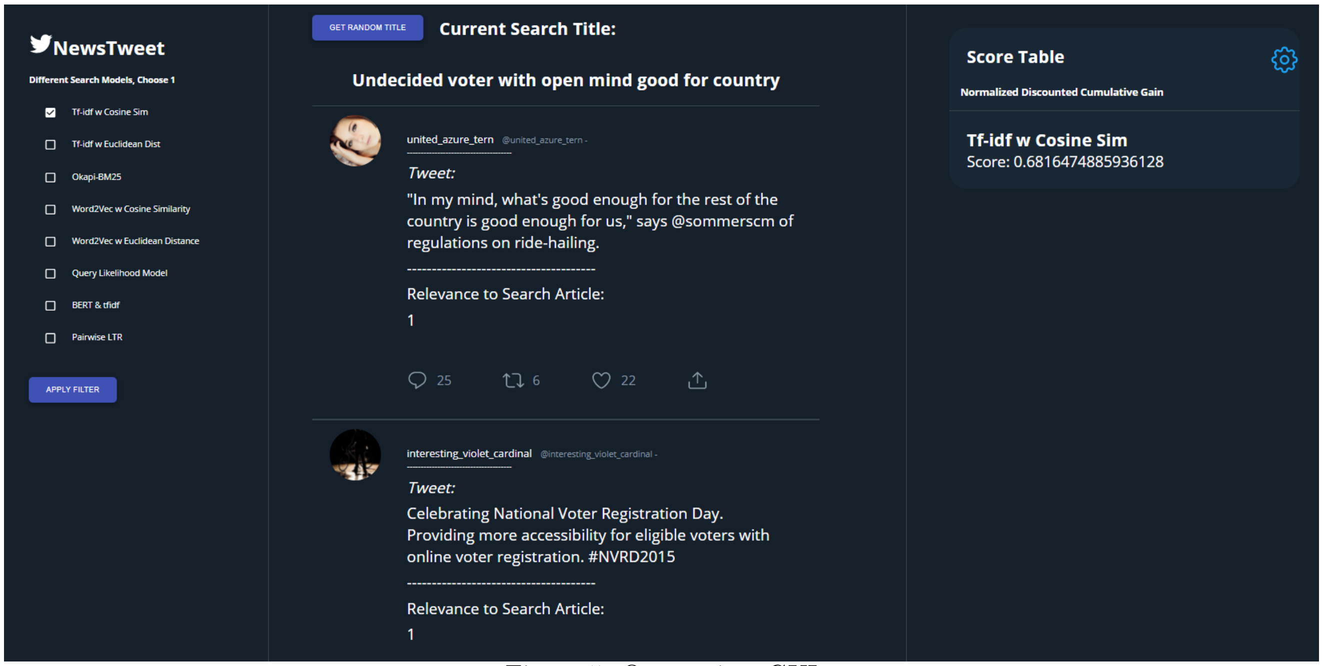


Figure 4: Scatter plot showing NDCG accuracy@20 vs retrieval time in Seconds

The figure above roughly depicts the time and accuracy trade-off of the various search methods (Pairwise LTR was left out). It is likely that with a larger dataset and the line plot eventually resembles a decreasing exponential.

7 Graphical User Interface


NewsTweet

Different Search Models, Choose 1

- [✓ Tf-idf w Cosine Sim]
- Tf-idf w Euclidean Dist
- Okapi-BM25
- Word2Vec w Cosine Similarity
- Word2Vec w Euclidean Distance
- Query Likelihood Model
- BERT & tfidf
- Pairwise LTR

APPLY FILTER

GET RANDOM TITLE **Current Search Title:**

Undecided voter with open mind good for country

united_azure_tern @united_azure_tern -

Tweet:
"In my mind, what's good enough for the rest of the country is good enough for us," says @sommerscm of regulations on ride-hailing.

Relevance to Search Article:
1

25 6 22

interesting_violet_cardinal @interesting_violet_cardinal -

Tweet:
Celebrating National Voter Registration Day. Providing more accessibility for eligible voters with online voter registration. #NVRD2015

Relevance to Search Article:
1

Score Table

Normalized Discounted Cumulative Gain

Tf-idf w Cosine Sim
Score: 0.6816474885936128

Figure 5: Our project GUI

7.1 Description

The image above shows our GUI implementation. The column on the left shows the various search methods. The top-middle section shows the current search query. Users can click on the "Get Random Title" Button to get another random article title. The column on the right shows the calculated NDCG score for a query.

The middle column shows all the tweets retrieved for a given query. The names and profile pictures are artificially generated for privacy reasons. The important information displayed is the retrieved Tweet, and its relevence to the query.

To use the GUI, ensure a search query is present, following which select a search method and click on the "Apply Filter" button, following which, the queries will be retrieved in a few seconds.

8 Conclusion

We began our project with the intention on finding the best search method to retrieve relevant Tweets with a article title as a query. Using the average NDCG score as an accuracy metric, we have discovered that the BERT with

Tf-Idf search model produces the most accurate results. This is likely due to the highly contextual nature of Tweets, which is often littered with slang, acronyms and trending words. We surmise that BERT is able to semi-accurately capture the semantics of Tweets, and therefore produce accurate results, unlike traditional search methods. Furthermore, as mentioned previously the BERT model that we used was optimised to detect paraphrases within Tweets. For future works, perhaps parameter optimisation techniques such as conducting grid searches to discover optimal parameters of each model could be carried out to further improve the various search methods.