Tweelink : Linking Twitter Hashtags to News Articles

Amisha Aggarwal
amisha19016@iiitd.ac.in

Aryan Behal
aryan19026@iiitd.ac.in

Harman Singh
harman19042@iiitd.ac.in

Mayank Gupta
mayank19059@iiitd.ac.in

Yash Bhargava
yash19289@iiitd.ac.in

Yash Tanwar
yash19130@iiitd.ac.in

1. INTRODUCTION

1.1 Motivation and Problem Statement

Twitter is an extremely popular microblogging and social networking site launched in 2006. Today it boasts of hundreds of millions of monthly active users. Users can post, like, tweet and retweet 280 characters at a time. No other social platform comes close to capturing the pulse of the world in real time. Twitter users comprise people from all walks of life. Equally rich is the diversity of topics tweeted by users every second of the day. While a tweet is representative of a person's thought process, a hashtag is the culmination of collective thought of the populace. While the tweets shape the trend of a hashtag, the hashtag shapes the conversation as well, making a self feeding loop.

For better or for worse, Twitter plays a key role in shaping the conversation on topics that affect everyone's lives. From coronavirus to anti-vaccination protests to the Russia-Ukraine crisis, Twitter makes its presence felt everywhere. **Our project aims to capture the sentiment and context of a hashtag and link it to relevant news articles.** The context behind a hashtag is clouded by its informal structure. A hashtag is generally an amalgamation or portmanteau of several words making it difficult to decipher. This will help the uninformed wade through countless tweets to understand the context of the trend while combating misinformation at the same time.

1.2 Novelty

The novelty of our project lies in its uniqueness and ability to retrieve information out of a hashtag.To the best of our knowledge, there isn't any existing research or technology attempting to link hashtags to news articles. Some existing papers have tried to use the tweets containing URLs, but we are not dependent only on URLs within the tweet. Our novelty lies in the fact that we consider external news articles explaining the trending news/controversy. We face the challenge of working with a corpus of extremely short texts on which regular NLP techniques may yield poor results. The text itself may be informal, multi-lingual and full of grammatical errors.

1.3 Dataset Description

We built a dataset of tweets and news articles. We used Twitter API v2 to track a hashtag across the most relevant tweets every hour for 72 hours. Around 20 hashtags were collected everyday. News articles were collected by web scraping. For a particular hashtag, after understanding its context through tweets, we collected around 10 news articles within the time frame of its occurrence which consist of both relevant and less-relevant content.

Our dataset consists of 224213 tweets. The dataframe for tweets has the following fields - author id, date, geolocation, language, tweet text, hashtags, urls, sensitive tweet or not. The dataframe for articles has the following fields - link of article, corresponding hashtag, geolocation, article heading, article text.

Exploratory Data Analysis

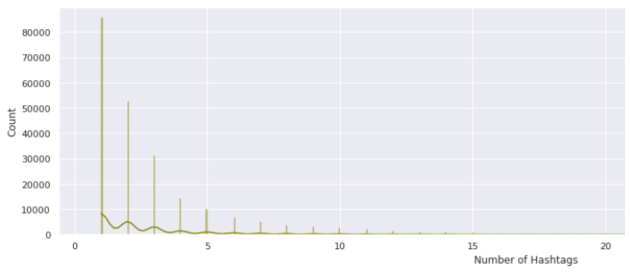


Figure 1: Number of hashtags per tweet

Figure 1 shows most tweets use lesser number of hashtags.

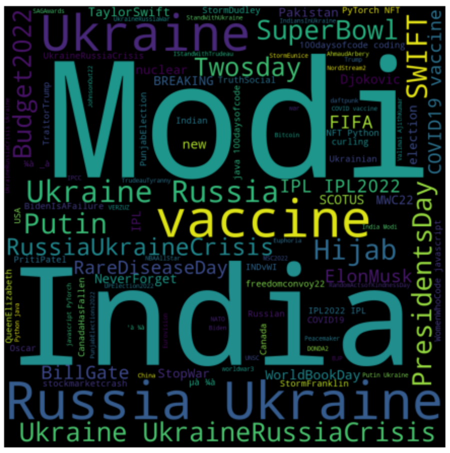


Figure 2: Word Cloud of hashtags

Figure 2 shows that hashtags like Modi, RussiaUkraineWar trend for a long time, and hence have been used in a large number of tweets.

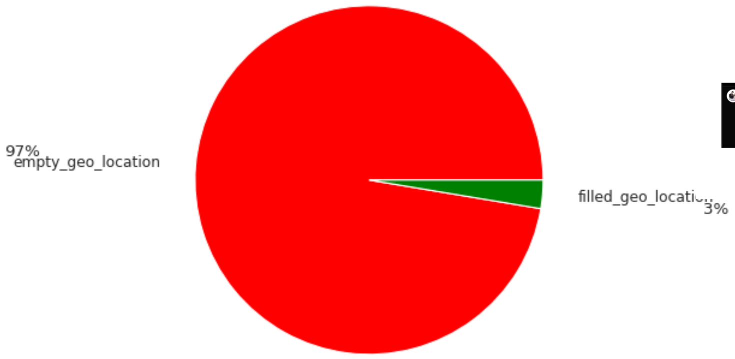


Figure 3: Twitter API geolocation data

Figure 3 shows that only 3% of the tweets extracted from the Twitter API have geolocation field in them.

1.4 Proposed Solution

We ask the user to input a hashtag, time and location. Based on the input, we retrieve relevant tweets from our dataset and create a feature vector which is used to retrieve relevant articles in a ranked order. The workflow of our system can be seen in Figure 1. The ranking is determined as a function of the hashtag, text in relevant tweets, location of tweets, location of articles and difference in time input by the user, time of tweets and date of publication of articles. Time is an important factor in the retrieval of relevant results, as illustrated in Figure 2. We will provide a web interface to facilitate the process or provide a browser extension, which will be decided as the project proceeds.

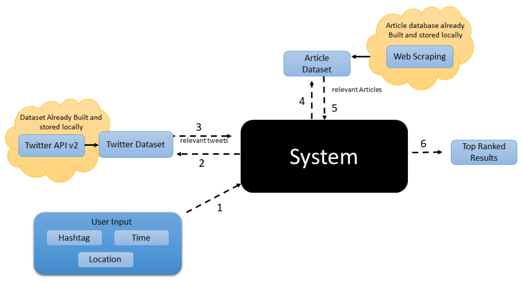
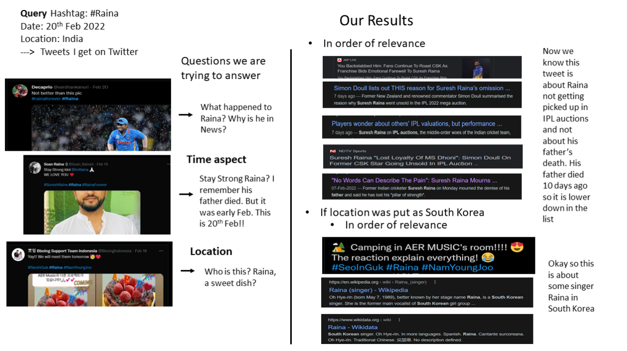


Figure 4: Workflow of our system

2. LITERATURE REVIEW

2.1 Keyword Extraction and Semantic Similarity


Query Hashtag: #Raina
Date: 20th Feb 2022
Location: India
---> Tweets I get on Twitter

Questions we are trying to answer

Decaprio @vardhankanuri · Feb 20
Not better than this pic
#rainaforever #Raina

→ What happened to Raina? Why is he in News?

Time aspect

Soan Raina :) @Soan_Raina3 · Feb 19
Stay Strong Idol @ImRaina 🙏
WE LOVE YOU ❤️
#SureshRaina #Raina #RainaForever

→ Stay Strong Raina? I remember his father died. But it was early Feb. This is 20th Feb!!

Location

보잉 Bboing Support Team Indonesia @BboingIndonesia · Feb 18
Yay!! We will meet them tomorrow 🥳❤️
#SeoInGuk #Raina #NamYoungJoo

→ Who is this? Raina, a sweet dish?

Our Results

- In order of relevance

ABP LIVE
You Backstabbed Him: Fans Continue To Roast CSK As Franchise Bids Emotional Farewell To Suresh Raina

Simon Doull lists out THIS reason for Suresh Raina's omission ...
7 days ago — Former New Zealand and renowned commentator Simon Doull summarised the reason why Suresh Raina went unsold in the IPL 2022 mega auction.

Players wonder about others' IPL valuations, but performance ...
7 days ago — Suresh Raina on IPL auctions, the middle-order woes of the Indian cricket team,

NDTV Sports
Suresh Raina "Lost Loyalty Of MS Dhoni": Simon Doull On Former CSK Star Going Unsold In IPL Auction ..

"No Words Can Describe The Pain": Suresh Raina Mourns ...
07-Feb-2022 — Former Indian cricketer Suresh Raina on Monday mourned the demise of his father and said he has lost his "pillar of strength".

Now we know this tweet is about Raina not getting picked up in IPL auctions and not about his father's death. His father died 10 days ago so it is lower down in the list

- If location was put as South Korea
  - In order of relevance

🏕️ Camping in AER MUSIC's room!!!! 😍
The reaction explain everything! 😂
#SeoInGuk #Raina #NamYoungJoo

https://en.wikipedia.org › wiki › Raina_(singer)
Raina (singer) - Wikipedia
Oh Hye-rin (born May 7, 1989), better known by her stage name Raina, is a South Korean singer. She is the former main vocalist of South Korean girl group ...

https://www.wikidata.org › wiki
Raina - Wikidata
South Korean singer. Oh Hye-rin. In more languages. Spanish. Raina. Cantante surcoreana. Oh Hye-rin. Traditional Chinese. 吳慧琳. No description defined.

Okay so this is about some singer Raina in South Korea

Figure 5: An example scenario of our problem statement

Bordoloi and Biswas [1] proposed a modified graph based approach KWC (Keywords from Collective Weights) for automatic keyword extraction from a twitter corpus using frequency, centrality, position, strength of the neighbours and other influencing graph measures. Their method was based on node-edge rank centrality with node weight depending on various features. The control flow of their approach followed data preprocessing, textual graph representation, collective node-weight assignment, and keyword extraction, resulting in top keywords as the output. For the textual graph representation part, they represented each token as a vertex and if two tokens co-occurred within the same window (i.e., a tweet), then an edge was created between them. The generated graph revolved around a single topic and thus held many common tokens that were associated with more than one tweet. The edges were weighted according to co-occurrence frequency. The nodes were weighted based on important graph measures like term frequency, selectivity centrality, etc., which made their algorithm unique and more powerful. Finally, keyword extraction was performed using NE-Rank and degree centrality and best n keywords were given as output. Their novel approach outperformed several existing graph based methods. Since our project revolves around similar lines of extracting keywords for a given set of tweets for a hashtag, their superior method might come handy in our methodology.

Mihalcea et al. [2] suggested a method for measuring the semantic similarity of texts by using the information that can be drawn from the similarity of the words. Their main focus was to find the semantic similarity instead of lexical similarity. In addition to the similarity of words, they also took into account the specificity of words, which was determined using the inverse document frequency (idf). Starting with each of the two text segments, and for each word, they determined the most similar word in the other text segment, according to the PMI-IR similarity measure. Next, they combined the word similarities and their corresponding specificity, and determined the semantic similarity of the two texts. They achieved an accuracy of 70.3%, representing a significant 13.8% error rate reduction with respect to the traditional vector-based cosine similarity baseline. Our project also aims at finding the semantic similarity between the tweets and the articles. So, this technique of measuring the similarity using the PMI-IR method will be useful for our work too.

Prakoso et. al.[3] conducted a systematic literature review on the various short text similarity(STS) measurements available. They grouped the different STS methods into 4 different categories depending on the attributes, procedures and resources that the techniques used: string-based, knowledge-based, corpus-based and hybrid-based similarity. The string-based methods reviews compared word and character sequences. The knowledge-based methods used word-level affinity by making use of the semantic information present and scale it up to sentences. The corpus-based methods used techniques such as Latent semantic analysis(LSA), Latent Dirichlet Allocation(LDA) and word2vec to find the relations between words in a corpus, which were then utilized at the sentence level to measure similarity. Hybrid-based methods were a combination of several different methods from the above categories and were employed to improve the accuracy. The authors analyzed the benefits and limitations of these similarity measures on 7 different features, namely (i) domain independence, (ii) language independence, (iii) semantic knowledge requirement, (iv) corpus and training data, (v) capability to identify semantic meaning, (vi) word order similarity and (vii) polysemy. For the string-based STS methods, they concluded that although it's relatively easier to implement and is advantageous in cases where we need the similarity between sequences of words, sentences can be structured in different ways to convey the same meaning and string-based methods fail to justify their similarity. For the knowledge-based category, the semantic meaning of a sentence was captured and then used to determine the similarity. The authors concluded that knowledge-based methods in conjunction with semantic information on a specific domain is more helpful in a scenario where we have a specialized use case. For the corpus-based category, the authors concluded that conventional techniques such as LSA and LDA, although very reliable, are computationally expensive and the bag-of-words model employed by these methods disables word-similarity measurement. Lastly, hybrid-based methods were concluded to increase the accuracy as well as the processing time required for measuring the similarity as a combination of semantic and corpus based methods was used. We can use the comparison between the different short text similarity measures analyzed by the authors and accordingly employ a select few of these methods and compare the results in order to find out the most suitable method of similarity for our project.

Pradhan et. al.[4] also take on the problem of obtaining the most relevant documents from a web search by analyzing the different similarity methods used for document retrieval. They've categorized the similarity measures into 2 categories: lexical similarity and semantic similarity. Lexical Similarity assigns similarity based on character and sentence matching[19]. It works on different sets of strings and assigns a score depending on the degree of overlap between them. The authors analysed 4 character based similarity measures namely: Longest Common Subsequence, N-Gram similarity, Levenshtein distance Similarity and Jaro Distance, which were followed up with 5 statement based similarity measures namely: Cosine similarity, Centroid based similarity, Web Jaccard Similarity, Web Simpson Similarity and PMI similarity. Semantic similarity on the other hand distinguishes itself from lexical similarity by calculating the similarity between texts based on their meaning rather than sequence of characters. The authors subdivided the methods under semantic similarity into 2 categories, Corpus-based and knowledge-based. Corpus-based similarity measures determine the similarity between words that have the same meaning based on the information derived from a corpus and included Normalized Google Distance(NGD), Normalized Information Distance(NID), Normalized Compression Distance(NCD) and Latent Semantic Similarity (LSA), which is the most popular vectorial semantics method. Knowledge-based similarity measures on the other hand use semantic networks such as "WordNet" and Natural Language ToolKit (NLTK) to measure the contextual similarity between words. These measures include Resnik Similarity and Vector similarity. Lastly, a hybrid method that used kernel based similarity and cosine similarity was discussed and it was concluded that the kernel based similarity measure gives a better accuracy compared to cosine similarity. A subset of the various similarity measures discussed by the authors will be used to solve the problem of obtaining the most relevant articles that describe the context of a tweet.

Reading and summing up large texts is very difficult as the amount of information generated keeps on increasing. Keyword extraction systems are required to automate the process of keyword generation owing to the lack of descriptive terms present in large volumes of text. Campos et. al. [5] resort to an unsupervised automated keyword extraction method YAKE! that is independent of the dataset and it's description. They make use of statistical information and the local specific text features such as term frequencies to identify the important keywords, without the need of a corpus. They adopt the following two steps to extract the keywords: computing the importance of the terms individually using statistical features and applying a n-gram model that forms multi-word terms and evaluating them heuristically. The proposed algorithm comprised of 5 main steps: (1) preprocessing the text and identifying the candidate terms; (2) representing the input by a set of statistical features; (3) calculating term scores by evaluating them heuristically; (4) n-gram generation and candidate keyword score computation based on their importance; and lastly (5) data deduplication and ranking. The algorithm first divides the text into sentences, followed by splitting, annotation, tokenization and stopwords identification. They then statistically analyze the sentence structure, term frequencies and co-occurrence for the purpose of feature extraction. They devise a set of 5 features that depict the nature of a candidate term: (1) Casing of the terms; (2) Term position; (3) Term frequency Normalization; (4) Term context relatedness; and (5) Number of appearances of the same term in different sentences. The term score is computed according the formula: S(t) = (TRel * TPosition)/(TCase + TFNorm/TRel + TSentence/TRel). After n-gram generation, the candidate keyword scores are computed using S(kw) = t kw S(t) / KF(kw)*(1+t kw S(t)) where kw represents the candidate keyword terms and S(kw) represents the final score. Due to high chances of similarity among candidate keywords, the next step is to perform data deduplication where potentially similar candidate keywords are eliminated. The authors evaluate the effectiveness of their system on 20 different datasets having a diversity of language, domains, text size and types of documents. By observing the results obtained, YAKE! outperformed not only the statistical methods such as TF-IDF, KP-Miner and Rake, but also the graph-based methods such as TextRank, TopicRank, PositionRank across multiple datasets independent of document languages or a trained model. The keyword extraction mechanism followed by YAKE! can help us in devising a model to extract the important keywords from the tweets and articles present in our dataset.

2.2 Similar Work

Uta Losch et al[6] tried to map microblogs to Encyclopedia articles. The idea of their method was to input a search query (a hashtag) and output an RDF document containing the most recent messages that matched the query, the authors of these messages and the most relevant wikipedia entities for this result. They extracted the 100 most recently published tweets using Twitter Search API along with author data, geo-location, publishing date and the URLs posted in the messages. All text data of the tweets and articles hyperlinked by the URLs was put in Wikifier for obtaining the content annotations. This tool took a text as input to return a set of DBpedia entities which were relevant for the analysed text in RDF format. Our project also runs along similar lines except for the fact that we are suggesting articles and also ranking them based on relevance. For making our tweet database from hashtags, we have followed a similar approach as they have used by using Twitter Search API and combining text to search for relevant articles. In contrast to their work, we plan to use our own methods rather than wikifier api for searching relevant articles.

Krestel et al.[7] trained various models to suggest relevant tweets for articles and evaluated their results on the basis of user study. To identify similar content tweets they employed language models (to find word overlap) and topic models (to find concept overlap). To combine the resulting content similarity scores with other features, such as recency or popularity of a tweet, they used logistic regression as well as boosting. For language modeling, they used a document likelihood model to compute the probability that a tweet is generated from a news article instead of a query likelihood model (which assumes that queries are short and documents are long). They used Dirichlet smoothing to smoothen their language model and geometric mean for normalizing tweet lengths. For relevance, they used the document likelihood and latent Dirichlet allocation to find the most relevant tweets. They used logistic regression apart from above techniques to mark a tweet as relevant or irrelevant which was trained, considering 16 additional features, such as publication time, length, follower count, etc and Adaboost and decision stumps to identify less similar tweets. To ensure unique results in the final ranking, they ignored tweets that have a high word overlap with tweets that were already in the set of recommended tweets. In their set-up, the topic model outperformed language model. Adding more features increased the accuracy of the model. In our model, we employ similar techniques of language model and topic model to check for word and concept similarity. For normalizing tweets lengths, we plan to use geometric mean. We also plan to incorporate Adaboost to identify less similar articles (along with latent Dirichlet allocation).

Ahuja et al.[8] 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 headline and the tweet, thus finding news-related relevant tweets and the corresponding news topics. We can use this work for extracting URLs from tweets and then using the corresponding news articles. But we are not dependent only on the URLs. We are also using the tweets that don't have URLs, and are mapping them to relevant news articles from the articles database.

2.3 Ranking Evaluation

In an ideal scenario, our system would link the hashtag directly to the most relevant article present on that particular topic. However, there could be multiple articles that are closely related with only slight variations amongst them. Corso et. al.[9] proposed a ranking algorithm that takes into account a number of desirable properties of each piece of news and subsequently assigns them a rank based on these properties. They evaluated the consistency of their algorithm based on its behaviour for two important limiting cases, the mean rank of independent news articles should be independent of the time and size of observation window and two sources, where one acts as a "mirror" source, should have a similar rank. The authors introduced two classes of algorithms: Non-Time-aware ranking algorithms and Time-Aware ranking algorithms. Non-Time-aware ranking algorithms dealt with a static dataset of news articles rather than a continuous stream of news flow. Both algorithms under this class, however, violated the limiting cases and coupled with the loss of temporal information associated with the fixed time-window scheme, Time-aware ranking algorithms were made necessary. According to Corso et. al.[6],the importance of a piece of news and the time of its emission were strictly related. To account for this, the author introduced a "freshness" decay parameter "α" that is obtained from the half-life decay time. Moreover, these algorithms took into account another parameter "β" which could be varied to tune the change in the rank of a news source based on the arrival of a fresh piece of news. Although an improvement from the Non-Time-aware algorithms, each of the proposed Time-aware algorithms also failed to satisfy one desirable property and failed to pass the second limiting case. However, for the purpose of ranking articles, Time-aware algorithms sufficed.

Yilmaz et. al.[10] studied the shortcomings of traditional rank correlation coefficients such as Kendall's τ and Spearman rank correlation coefficient for two ranked lists and proposed a new coefficient that penalized errors at high rankings more than the errors at low rankings and had an underlying probabilistic interpretation that was easy to understand. The AP rank correlation coefficient introduced by the authors possessed two important properties when compared to Kendall's : its equality to Kendall's τ when the errors are uniformly distributed over the entire ranking and the difference in value when the error in ranking is concentrated more at the top/bottom of the list.

3. BASELINE RESULTS

We take the hashtag, date and location as input from the user. After applying our models in the backend, we give as output the top k relevant news articles corresponding to that hashtag, date and location. The workflow of our models goes in this order: Preprocessed tweets corresponding to the hashtag, date (including the previous and the next date), and location are extracted from our tweets dataset.

1. The tokens corresponding to all the tweets are combined to make the query vector.

2. Preprocessed articles corrsponding to that date, the previous date and the next date are extracted from the articles database.

3. Different similarity metrics are applied between the given query and the news articles. The ranking is then evaluated using methods like Precison@K, Recall@K, nDCG etc and the results (top k articles) corresponding to the best evaluation score are reported to the user.

4. The following evaluation metrics have been used: Jaccard Similarity, TF-IDF matrix, Cosine Similarity and Binary Indpendence Model.

3.1 Preprocessing

The following preprocessig steps were performed on the tweets and the articles dataset -

1. Converted the text to lower case
2. Performed tokenization
3. Removed stopwords from tokens
4. Removed Punctuations from tokens
5. Removed blank spaces from tokens
6. Performed lemmatization

3.2 Evaluation Metrics

The following evaluation metrics has been used: Jaccard Similarity, TF-IDF matrix, Cosine Similarity and Binary Indpendence Model.

1. **Jaccard Coefficient**: Jaccard Coefficient measures the similarity between finite sample sets. It is computed as the size of the intersection of the sets divided by their union.
Jaccard(A,B) = $\frac{|A \cap B|}{|A \cup B|}$
We iterate over our set of preprocessed documents one at a time and calculate the Jaccard similarity between the query and the document using the above formula. We sort the relevant documents in descending order and return a list of the most relevant documents along with their Jaccard similarity scores. Jaccard similarity failed to capture term frequency and ordering of terms into account. This is evident from the table below.

Query	Mean Avg Precision	Mean Avg Recall
hijab, 2022-02-19, India	0.0387	0.0325
Across queries	0.2615	0.1709

2. **TF-IDF similarity matching**: TF-IDF is another statistic that quantifies the importance of a term in a document or a corpus. It increases proportionally to the frequency of the term in the document. It involves computing the Term Frequency which is the number of times each word appears in each document as well as the Inverse Document Frequency which is computed for a term as the inverse function of the number of documents it appears in. TF-IDF is the product of the two statistics.
$\text{tf-idf}_{t,d} = (1 + \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$
It acts as a measure of how much information a word provides i.e whether it is common or rare in the corpus.For our system, all the articles are read and preprocessed. User query is taken and preprocessed in a similar way. Vocabulary of the whole dataset is found and 2 dictionaries: word2id and id2word are created to keep note of index of each word in the vocabulary. For creating TF-IDF matrix, we need both TF and IDF For TF, a word-document matrix to store the frequencies of each word in each doc. 5 different TF-IDF matrices are then formed using the 5 different weighing schemes: (1) Binary weighted; (2) Raw Count; (3) Term Frequency; (4) Log Normalization and (5) Double Normalization. The query is preprocessed in similar way as the dataset and a query vector is created of size as big as vocabulary size. Each index of this query vector corresponds to a word in dataset vocabulary. The query vector is updated with the term-frequencies of each query token at the indices which correspond to their index in vocabulary. The query vector is also weighted according to weighing scores of TF-IDF schemes and multiplied with respective IDF scores obtained from the dataset. Dot Product of each TF-IDF matrix is taken with the processed query vector to obtain the TF-IDF scores of each document for each query token. The top 5 relevant documents are reported for each weighing scheme.

Binary: Query - hijab, 2022-02-19, India

Query	Mean Avg Precision	Mean Avg Recall
Normal	0.3920	0.3749
Across queries	0.31	0.29
Top 10	0.501	0.439
Top 10 Across queries	0.44	0.35

Raw Count: Query - hijab, 2022-02-19, India

Query	Mean Avg Precision	Mean Avg Recall
Normal	0.8343	0.775
Across queries	0.75	0.68
Top 10	0.834	0.775
Top 10 Across queries	0.75	0.68

Term Frequency: Query - hijab, 2022-02-19, India

Query	Mean Avg Precision	Mean Avg Recall
Normal	0.8343	0.775
Across queries	0.75	0.68
Top 10	0.834	0.775
Top 10 Across queries	0.75	0.68

Log Normalisation: Query - hijab, 2022-02-19, India

Query	Mean Avg Precision	Mean Avg Recall
Normal	0.7652	0.71
Across queries	0.64	0.61
Top 10	0.823	0.765
Top 10 Across queries	0.70	0.625

Double Normalization: Query- hijab, 2022-02-19, India

Query	Mean Avg Precision	Mean Avg Recall
Normal	0.0502	0.065
Across queries	0.1	0.11
Top 10	0.8248	0.765
Top 10 Across queries	0.75	0.68

3. **Cosine similarity**: In NLP, cosine similarity is a standard tool to measure the text-similarity between two documents. The sizes of the 2 documents can be unequal and of any length. The documents are represented in n-dimensional vector space. The n-D vectors are projected to a multidimensional space and the cosine of angle between them is measured. The output will vary between 0 and 1, values closer to 1 representing high matching between the documents and closer to 0 representing poor matching. The word occurence for both the documents is calculated using Count Vectorizer or TfIdf Vectorizer. The vocabulary considered is the union of the terms in the query and the document being compared. Dot product is taken between the vectors consisting of binary values and the similarity score is reported. Top k documents with the highest similarity score are reported.

Cosine (Count Vectorizer)

Query	Mean Avg Precision	Mean Avg Recall
hijab, 2022-02-19, India	0.834	0.3875
Across queries	0.764	0.296

Cosine (TF-IDF)

Query	Mean Avg Precision	Mean Avg Recall
hijab, 2022-02-19, India	0.834	0.3875
Across queries	0.759	0.295

4. **Binary Independence Model**: It is a probabilistic model that assumes that either term is present in document or not. A 2-D matrix is created which stores the occurrence of each word in each document. 1 mean presence and 0 is for absence and all terms are considered independent. A probability is assigned to a relevance of document depending on probability of relevance of terms vector of that document. A rank is assigned to the query which is recomputed based on weights for 10 iterations using probabilistic methods giving a final rank. Top k documents are shown.

Query- hijab, 2022-02-19, India

Query	Mean Avg Precision	Mean Avg Recall
Normal	0.0247	0.0199
Across queries	0.05	0.03
Top k	0.834	0.3875
Top k Across queries	0.54	0.30

Across all baseline models it is evident that taking top k keywords yield better results than simply considering all the keywords.

4. PROPOSED METHOD

Applying knowledge graph for keyword extraction: Some models are giving unsatisfactory results, the reason can be attributed to the fact that the enormous size of the query vector(consisting of less important tokens) leading to matching between less relevant articles. Top K keywords can be extracted by the method mentioned in [1] to limit the query space and thus giving a higher weightage o the most important terms. This would consequently lead to better matching since less relevant keywords wont contribute much to the similarity score. YAKE, the state of the art keyword extractor as explained in the literature review can also be used.

We are currently using statistic based text representstions which do not leverage co-occurrence statistics between words. They assume all words are independent of each other. They do not capture the context or semantics of the word or consider spooky scary as similar but as two independent terms with no commonality between them. Using learning based/distributed text representation can help us since word embeddings like Word2Vwc, Glove, Bert are capable of capturing relationships between different words including their syntactic semantic relationships.

Using state of the art ranking functions like BM25 might help us improve the ranking score.

5. REFERENCES

[1] Bordoloi, M. and Biswas, S.Kr. (2018). Keyword extraction from micro-blogs using collective weight. Social Network Analysis and Mining, 8(1).

[2] Mihalcea, R., Corley, C. and Strapparava, C. (2017). Corpus-based and Knowledge-based Measures of Text Semantic Similarity. [online] Available at: https://www.aaai.org/Papers/AAAI/2006/AAAI06-123.pdf

[3] Prakoso, D.W., Abdi, A. and Amrit, C. (2021). Short text similarity measurement methods: a review. Soft Computing, 25(6), pp.4699–4723.

[4] Pradhan, N., Gyanchandani, M. and Wadhvani, R. (2015). A Review on Text Similarity Technique used in IR and its Application. International Journal of Computer Applications, 120(9), pp.29–34

[5] Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C. and Jatowt, A. (2020). YAKE! Keyword extraction from single documents using multiple local features. Information Sciences, [online] 509, pp.257–289. Available at: https://www.sciencedirect.com/science/article/abs/pii [Accessed 11 Feb. 2022].

[6] Lösch, U. & Müller, D. (2011). Mapping microblog posts to encyclopedia articles. GI-Jahrestagung.

[7] Krestel, R., Werkmeister, T., Wiradarma, T.P. and Kasneci, G. (2015). Tweet-Recommender. Proceedings of the 24th International Conference on World Wide Web.

[8] Ahuja, S. (2015). Discovering significant news sources in Twitter. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7434278 [Accessed 26 Feb. 2022].

[9] Del Corso, G.M., Gullí, A. and Romani, F. (2005). Ranking a stream of news. Proceedings of the 14th international conference on World Wide Web - WWW '05.

[10] Yilmaz, E., Aslam, J.A. and Robertson, S. (2008). A new rank correlation coefficient for information retrieval. Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08.