COVID-19 Twitter Misinformation Lookup: An Investigation of Relevance Ranking Methods

Swati Adhikari
Department of Computer Science
Virginia Tech
swatiadhikari@vt.edu

Provakar Mondal
Department of Computer Science
Virginia Tech
provakar@vt.edu

Kiet Nguyen
Department of Computer Science
Virginia Tech
kiet@vt.edu

Ismini Lourentzou
Department of Computer Science
Virginia Tech
ilourentzou@vt.edu

Abstract

The emergence of the COVID-19 pandemic has permanently altered the course of people's lives, and is thus cause for prolonged discussion. Along with discourse on the issue arise various sources of online misinformation, especially on social media platforms such as Twitter. As the culmination of our CS 5604 course on Information Storage and Retrieval in the fall of 2021, we present a search engine that retrieves Tweets corresponding to a COVID-19-related topic indicated by a user's query. In particular, we implemented and evaluated 3 different relevance ranking models: Okapi BM25, RankNet, and TF-Ranking. An analysis section discusses the factors affecting the performances of our models, and we conclude this paper by opening up our work to possible future directions. The source code is available at `https://github.com/kanguyen-vn/covidtweet_rr`.

1 Introduction

1.1 COVID-19 misinformation

On March 11, 2020, the WHO officially declared COVID-19 a global pandemic [14]. Since then, it has been the center of discussion on almost every social media platform. Big platforms like Twitter and Facebook started getting loaded with a wide range of conspiracy theories, false narratives, and unsubstantiated rumors, which have left the public with a lot of misinformation and hindered them from making informed decisions regarding their health. This content can possibly mislead people regarding the nature of the COVID-19 virus, the safety of preventative measures and treatments, or even official regulations and restrictions pertaining to health advisories. There seems to be more misinformation doing rounds on the social media platforms than ever: Tweets containing falsehoods are 70 percent more likely to be retweeted than truthful ones, and false news also propagated faster and wider across social media platforms[18]. Therefore, for our project, we aim to study the misinformation in Twitter data on which we investigate different relevance ranking methods.

1.2 Twitter search

Millions of people use Twitter to obtain updates on what is happening around the world. During any major event, Twitter observes traffic spikes as people use it to find timely information and perspectives, and the overall traffic volume keeps increasing. However, the Twitter search challenge is unique compared to traditional information retrieval applications because of its real-time nature, corpus

size, document format, and multiple result types. Tweets have unique properties: 140 characters of unstructured text with rich entities including hashtags, mentions, images, videos, and external links[15]. The search corpus is also huge, as millions of tweets are created daily in many languages. In order to return relevant, high quality search results for such a large corpus with low latency, researchers are trying to solve novel technical challenges in areas of information retrieval, natural language processing, machine learning, data science, distributed systems, etc.

1.3 Relevance ranking

Relevance ranking is a core area of Information Retrieval where given a query and a set of documents, the relevance ranking algorithms determine how relevant each text document is for the query. It is the basis of the ranking algorithm that is used in a search engine to produce the ranked list of documents. A good retrieval model will find documents that are most relevant to the query passed by the user. A search engine deployed in real time must use relevance ranking that incorporates user relevance. We investigate relevance ranking on the Twitter dataset on three relevance ranking algorithms - Okapi BM25, RankNet and TF-Ranking. We will discuss these algorithms in detail in the methodology section. Our goal in this project is to compare how these three models perform with the relevance ranking of the Twitter data.

1.4 Novelty

Our project is novel because it performs relevance ranking in a very specific domain: social media response surrounding the COVID-19 pandemic. Similarly to the Covid-HeRA dataset [7]—which this project utilizes and which will be elaborated upon in Section 3—our project aims to help users explore public sentiment and detect potential online misinformation around the topic. However, the nature of our model is inherently different: to generate the Covid-HeRA dataset, its authors solved a classification problem, whereas we are solving a ranking problem, the solution to which would be useful for search engines like Twitter's. By showing the most relevant Tweets to a user-provided query, our search engine would assist the user both in exploring opinions and ideas on COVID-19 Twitter and in making informed decisions about a particular topic.

2 Background and related works

2.1 Background and motivation

As Twitter has become a home ground for misinformation during the COVID-19 pandemic, we decided to explore the misinformation in regards to COVID-19 in twitter and investigate tweets with three relevance ranking methods - Okapi BM25, RankNet and TF-ranking. We will only be investigating the binary relevance of the data i.e either the data is relevant or irrelevant. The Okapi BM25 Model is a popular ranking algorithm but it has certain limitations as it does not work well with very long documents and it is very difficult to fine tune the free parameters with the model. Hence, we also explore the RankNet model which combines all documents under a query into document pairs and each pair is used as a sample. And for our final model, we use a learning-to-rank (LTR) algorithm TF-ranking which is an open-source TensorFlow-based library for developing scalable neural LTR models, which are useful in settings where users expect to receive an ordered list of items in response to their query[2].

2.2 Related works

Twitter, one of the most popular microblogging services, provides large quantities of real-time information including but not limited to news, comments, conversations, advertisements, etc. When it comes to Twitter search it is beneficial to examine relevance ranking. A lot of works have been done and already is undergoing to rank relevance among the numerous tweets posted every day. Twitter itself maintains a specialized search engine that ranks tweets based on posting time and popularity. Twazzup and Google rank real-time tweets by time. Less trivially, Tweefind and Twitority rank search results based on the popularity and/or activities of a Tweet's author. Additionally, Bing and CrowdEye rank tweets by time or the relevance of their contents[8]. Deep learning model such as RankNet employs a probabilistic cost function that uses a pair of sample items and learns how to

rank them[3] or DeepRank which utilizes a convolutional neural network (CNN) or two-dimensional gated recurrent units (2D-GRU) and then an aggregation network with sequential integration and term gating mechanism to produce a global relevance score[11], rank the relevance of tweets. Another related work is the LambdaLoss framewrok[19] which is a probabilistic framework to model ranking loss functions and show that existing ranking losses are instances of LambdaLoss.

3 Dataset

We introduce the source of our data and how we processed and transformed it to fit the purpose of our work.

3.1 Data source

For this project, we make use of a subset of the CoAID misinformation data collection [6] which originates from the Covid-HeRA dataset [7]. Covid-HeRA is a misinformation multi-class classification dataset, in which microblogging posts—Tweets, in particular—are ranked on a misinformation scale. They are categorized as either (1) **Real News/Claims**, (2) **Not severe**, (3) **Possibly severe**, (4) **Highly severe** misinformation, or (5) **Refutes/Rebuts** against an incorrect statement.

In addition, Covid-HeRA contains URLs to true and fake news articles and claims collected as part of the CoAID dataset, as well as their corresponding body texts. These news articles and claims were the original queries passed to the Twitter API to fetch the aforementioned Tweets [6]; thus, as described below in section 3.4, we utilize their body texts as the training queries for our learning-to-rank models.

3.2 Data collection

We had initially planned on working with a different COVID-19-related Twitter dataset [10] and already performed data collection on it; however, due to concerns surrounding annotation time, we decided to switch to the Covid-HeRA dataset, which was provided directly by Dr. Ismini Lourentzou. Had this not been the case, the data collection process would have been almost identical. Thus, we now describe our data collection process for the initial dataset.

According to Twitter's Developer Agreement and Policy,

> If you provide Twitter Content to third parties, including downloadable datasets or via an API, you may only distribute Tweet IDs, Direct Message IDs, and/or User IDs (except as described below). We also grant special permissions to academic researchers sharing Tweet IDs and User IDs for non-commercial research purposes.[1]

Therefore, only Tweet IDs were available in the initial dataset; we then had to implement code to fetch Tweet information using the Twitter API[2]. The biggest challenge we encountered during this phase was Twitter's rate limit of 900 requests per 15 minutes[3], which significantly prolonged the collection time. To avoid losing substantial progress, we collected the data in chunks of smaller size, which were then concatenated into one big file.

3.3 Data preprocessing

Because our data is unstructured text, it needs preprocessing before we can use it to train our models. Preprocessing for Tweet data is mostly similar to regular text preprocessing; however, one unique aspect is that we needed to take hashtags into account. Since hashtags are sequences of concatenated words, we employed the *ekphrasis* library to segment them [1].

[1] `https://developer.twitter.com/en/developer-terms/agreement-and-policy`.

[2] `https://developer.twitter.com/en/docs/twitter-api`.

[3] `https://developer.twitter.com/en/docs/twitter-api/v1/rate-limits`.

3.4 Data construction

Because Covid-HeRA is a classification dataset whereas we aimed to build a relevance ranking system, we made slight changes to the dataset to fit the purpose of our project. As mentioned, we take advantage of the body texts within the news articles and claims within the dataset, using them as queries to train our models; the Tweets then become our documents. For now, we are not using the classification labels in the dataset, but we hope to utilize them as part of our future work.

For our learning-to-rank models where vectorization is needed, we use 50-dimensional GloVe word embeddings [13] to convert these documents into vectors, where each document vector is equal to the average of all of its term vectors. Each query-document pair in our data is then simply a 100-dimensional concatenation of a query and a document, each containing 50 features. We then make use of the LIBSVM format [5] for our input data, which renders each line of the following format:

```
{relevance label} qid:{query ID} 1:{feature 1} 2:{feature 2} ...
                        100:{feature 100}
```

In our implementation, a Tweet in the Covid-HeRA dataset is deemed **relevant** to a query if it refers to that query in the original dataset. Since there is no irrelevance data in the dataset, for each query, we generated the data by randomly sampling "irrelevant Tweets" from the set of Tweets that do not refer to that query. For now, the number of irrelevant Tweets is twice that of relevant Tweets for each query. The full LIBSVM data is finally split into training, validation, and test sets, which are used by our learning-to-rank models.

4 Methodology

We describe the implementation of the three different models we employed as part of this work.

4.1 Okapi BM25

Okapi BM25 is a bag-of-words retrieval algorithm which is used by search engines to estimate the relevance of documents to a given search query. It is based on the probabilistic retrieval framework. It can also be used as a better replacement of TF-IDF and for term-weighting for each document. BM25 has been used as a content relevance feature in many studies of relevance ranking of tweets [8]. We use the rank-bm25 libary of the Python Package Index (PyPI) to implement the BM25 ranking function.[4] However there are certain limitations of the ranking function such as difficulty in optimizing the function parameters for a given information retrieval measure [16], and inefficiency of the ranking function in very long documents.

4.2 RankNet

RankNet is a pairwise method that combines all documents under a query into document pairs and each document pair is used as a sample. RankNet was originally developed using neural nets, but the underlying model can also be implemented using boosted trees. As a reference to build up the logic, we have taken help from the paper *From RankNet to LambdaRank to LambdaMART: An Overview*[4]For a model, the model learns probability for a document pair and then the model predicts which document is more relevant to that query. Later the model calculates cross-entropy to fit the true probability and the predicted probability. The closer the distributions are, the smaller the cross-entropy. While implementing our RankNet model based on this logic we faced two problems. Problem 1: Some evaluation indicators in sorting are not used directly as the cost function. The reason is that these indicator functions are not continuous, it is not easy to derive, and it is not easy to use gradient descent. Cross entropy is suitable for gradient descent. Problem 2: In normal training, the parameters are updated once for each sample pair i,j. When backpropagation is used the update requires the previous prediction, and the backpropagation after the error will be very slow. We used neural networks with hidden layers-based logic to implement our RankNet model. Our learning process uses the error backpropagation method to train. To train the model we had the following two ideas to apply.

[4] `https://pypi.org/project/rank-bm25/`.

1. We chose a sample pair (X_i, X_j), first brought X_i and then X_j into the neural network for forwarding feedback. Then we calculated the difference result and carry out error backpropagation, and then took the next sample pair. This method was very intuitive, the disadvantage was that the convergence speed is slow.

2. Batch training. We could bring all document pairs under the same order into the neural network for X_j feedback, Then we would calculate the total difference and would carry out the error backpropagation, which would greatly reduce the number of error back propagation.

As the first idea holds the disadvantage of slow convergence, we finally picked the second idea to train our model.

4.3 TF-Ranking

Introduced by Google in 2018, TF-Ranking is one of the first open-source libraries for solving ranking problems using learning-to-rank techniques. It contains commonly used loss functions and evaluation metrics in learning-to-rank problems. Built on top of the popular TensorFlow library, TF-Ranking inherits the former's well-known capability for large-scale training. The library was launched to great success for wide usage in different Google systems[12].

For our project, we implemented a learning-to-rank model using TF-Ranking, taking in training input as a series of query-document pairs in LIBSVM format, as described in Section 3.4. We discuss the model's performance, including hyperparameter tuning and its evaluation, in Section 5.3.

5 Evaluation

5.1 Okapi BM25

We used Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) to evaluate the BM25 model. For both the evaluation metrics, we used the Scikit-learn library to calculate the evaluation of the BM25 Model.[5]

For MAP, we needed to pass the relevance scores produced by the model and the actual label of the tweets with regards to the query. The label contained 0 or 1 meaning either the tweet is relevant to the query or not. In our dataset, we had every tweet labeled to which query it was relevant to. According to the calculations, the MRR of the BM25 Model was 0.5286 and the MAP of the BM25 Model was 0.5418. These scores are poor as compared to the other models and we will explore the evaluation for the other models below.

5.2 RankNet

We evaluated our RankNet model running with a different number of parameters and changing the values of the parameters. At the first stage, the learning standard was not very good and the loss values were not very good. At that stage, we used **Adadelta**, a Keras implemented optimizer. The learning rate was used 0.2 which was too high for the model. So, we received the output, not a satisfactory one.

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html`.

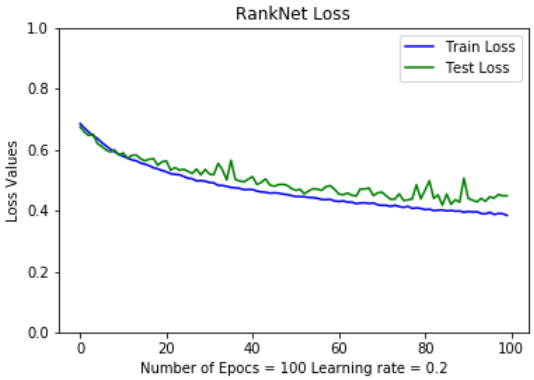
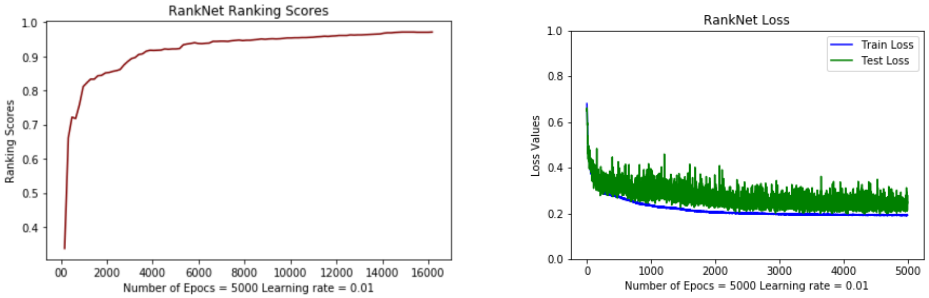


Figure 1: RankNet Loss values for Adadelta optimizer

Figure 1 shows the performance for learning rate 0.2 and Adadelta optimizer. As the scores were not very good, we then tried to train our model with a different learning rate and optimizer. We ran our model many times and the following figures depict the best ranking scores and loss output that we received from our model.



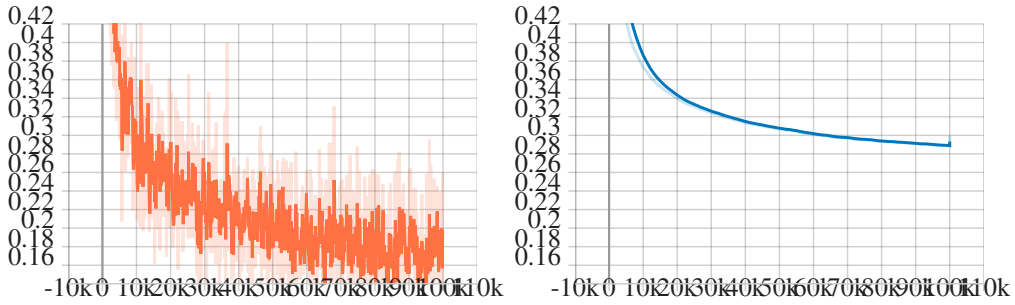
(a) RankNet Ranking Scores (b) RankNet Train and Test Losses

Figure 2: RankNet performance for Adam optimizer

The above figures depict the best-ranking scores that we have received from our RankNet model. The final loss values were 0.2018. For the final output, we used **Adam** as the optimizer and 0.01 as the learning rate value. The number of neurons used in the hidden layer was 50. In order to calculate the train and test losses, we have used **binary_crossentroy**. We have evaluated our RankNet model using the Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). We need to tune our model to make it better as from the above loss plot it is clear that, the model started to refuse to learn after approximately 2000 epochs. It is still required to modify the hyperparameters of the model for better performance in the future.

5.3 TF-Ranking

After tuning and training the model multiple times, we settle on the current version of our model, which has two hidden layers of size 64 and 32 respectively, as well as a dropout rate of 0.5 to avoid overfitting. The model is trained over 100,000 training steps with an initial learning rate of 0.005 using pairwise logistic loss and the Adagrad optimizer, which is suitable for sparse textual data [9].

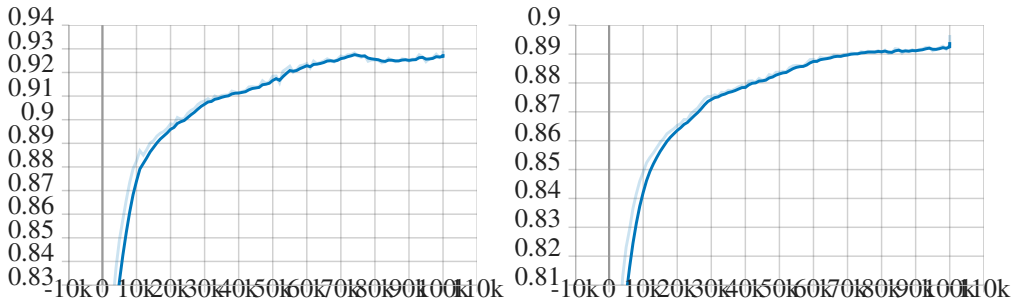


(a) Training loss

(b) Validation loss

Figure 3: TF-Ranking model losses

The model performs relatively well with a final validation loss value of 0.2932. To evaluate the model, we again employ the Mean Reciprocal Rank and Mean Average Precision metrics, which are available for use as part of TF-Ranking. Figure 4 displays the evaluation scores computed for the TF-Ranking model across the training steps. The model performs well with a final Mean Reciprocal Rank score of 0.9278 and Mean Average Precision score of 0.8941.



(a) Mean Reciprocal Rank

(b) Mean Average Precision

Figure 4: TF-Ranking model evaluation plots

5.4 Model comparison

Table 1 contains the different evaluation scores for the 3 models we implemented as part of our project.

Table 1: Evaluation comparison across the 3 models

Model	Evaluation metrics	
	Mean Reciprocal Rank	Mean Average Precision
BM25	0.5286	0.5418
RankNet	0.7632	0.7156
TF-Ranking	**0.9278**	**0.8941**

The TF-Ranking model performs the best out of the 3, taking the lead in both Mean Reciprocal Rank and Mean Average Precision scores, followed by RankNet, and finally BM25.

It was expected that RankNet and TF-Ranking both outperform BM25 because the latter is a traditional score-based method that only utilizes shallow features such as term/document frequency and document length, thus disregarding linguistic context. Because our learning-to-rank models employ GloVe word embeddings, which are adept at capturing linguistic cues [13], they prove to be better search

engines when it comes to dealing with Tweets. Furthermore, since RankNet and TF-Ranking are deep learning models, they are able to learn the connection between the features on a deeper level at the expense of explanability and training time.

Our implementation of TF-Ranking also significant outperforms our RankNet model. However, since RankNet is underperforming at the moment and in need of more hyperparameter tuning, we cannot yet clearly intuit reasons for the difference in performance. Once our RankNet model performs better, we hope to be able to analyze closely why TF-Ranking is more well-evaluated for the purpose of our project.

6 Conclusion

As the culmination of our work in the fall 2021 iteration of CS 5604 (Information Storage and Retrieval), we present an investigation into different relevance ranking methods on COVID-19-related Twitter data, specifically extracted from the Covid-HeRA dataset [7]. We implement and evaluate 3 relevance ranking models: Okapi BM25, RankNet, and TF-Ranking. We then evaluate their performance using the Mean Reciprocal Rank and Mean Average Precision metrics, and conclude that TF-Ranking performs the best out of the three for the purpose of our project.

There are a few possible future directions for our project. Firstly, we hope to build a front-end for our search engine, which would utilize all 3 of our models to retrieve Tweets from user-provided queries. At this point, although we have tentatively finished implementing our models, we do not yet have a graphical user interface for our search engine, and we aim to remedy that soon. Secondly, we want to continue tuning the hyperparameters for our models for the best performance possible. In addition, we hope to make the input data reflect the dataset better: as of now, for each query, the number of irrelevant documents is around twice the number of relevant documents within our input data. However, there are many more irrelevant Tweets in the real dataset. Therefore, we plan to modify our input data so that it is more proportional to real data—thus improving the model's performance with unseen queries—and retrain or reconfigure our models accordingly. Lastly, we look to elevate our models by implementing advanced mechanisms that could aid in their performance, such as using attention heads in our neural networks [17].

References

[1] C. Baziotis, N. Pelekis, and C. Doulkeridis. DataStories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[2] M. Bendersky and X. Wang. Advances in tf-ranking. *Google Research*, 2021. URL `https://ai.googleblog.com/2021/07/advances-in-tf-ranking.html/`.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Conference*, 2005. URL `https://doi.org/10.1145/1102351.1102363`.

[4] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Technical Report*, 2010. URL `https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/`.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), may 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL `https://doi.org/10.1145/1961189.1961199`.

[6] L. Cui and D. Lee. CoAID: COVID-19 healthcare misinformation dataset. *CoRR*, abs/2006.00885, 2020. URL `https://arxiv.org/abs/2006.00885`.

[7] A. Dharawat, I. Lourentzou, A. Morales, and C. Zhai. Drink bleach or do what now? Covid-HeRA: A dataset for risk-informed health decision making in the presence of COVID19 misinformation. *CoRR*, abs/2010.08743, 2020. URL `https://arxiv.org/abs/2010.08743`.

[8] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum. An empirical study on learning to rank of tweets. *Conference*, 2010. URL https://dl.acm.org/doi/10.5555/1873781.1873815.

[9] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

[10] R. K. Gupta, A. Vishwanath, and Y. Yang. COVID-19 twitter dataset with latent topics, sentiments and emotions attributes. *CoRR*, abs/2007.06954, 2020. URL https://arxiv.org/abs/2007.06954.

[11] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. *Conference*, 2017. URL https://arxiv.org/abs/1710.05649.

[12] R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, M. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2970–2978, 2019. URL https://doi.org/10.1145/3292500.3330677.

[13] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[14] P. Roxby. Coronavirus confirmed as pandemic by world health organization. *Newspaper*, 2020. URL https://www.bbc.com/news/world-51839944/.

[15] Y. Saraf. Search relevance infrastructure at twitter. *Blog*, 2016. URL https://blog.twitter.com/engineering/en_us/topics/infrastructure/2016/search-relevance-infrastructure-at-twitter.

[16] K. M. Svore and C. J. C. Burges. A machine learning approach for improved bm25 retrieval. *Microsoft Research Technical Report MSR-TR-2009-92*, 2009. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/LearningBM25MSRTechReport.pdf.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[18] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018. doi: 10.1126/science.aap9559. URL https://www.science.org/doi/abs/10.1126/science.aap9559.

[19] X. Wang, C. Li, N. Golbandi, M. Bendersky, and M. Najork. The lambdaloss framework for ranking metric optimization. *Conference*, 2018. URL https://doi.org/10.1145/3269206.3271784.

Contributions

Swati Adhikari

- Contributed to the course project proposal.
- Contributed to the project presentation.
- Contributed to the final report: Introduction (Sections 1.1, 1.2 and 1.3), Background and motivation (Section 2.1), Okapi BM25 (Sections 4.1 and 5.1).
- Computed TF-IDF scores from the input data during the experimental stages of the project.
- Implemented and evaluated the Okapi BM25 model.

Provakar Mondal

- Contributed to the course project proposal.
- Contributed to the project presentation.
- Contributed to the final report: Related works (Section 2.2), RankNet (Sections 4.2 and 5.2).
- Computed TF-IDF scores from the input data during the experimental stages of the project.
- Implemented and evaluated the RankNet model.

Kiet Nguyen

- Contributed to the course project proposal.
- Contributed to the project presentation.
- Contributed to the final report: Abstract, Novelty (Section 1.4), Dataset (Section 3), TF-Ranking (Sections 4.3 and 5.3), Model comparison (Section 5.4), Conclusion (Section 6).
- Implemented code to fetch Tweet data using the Twitter API.
- Implemented code to preprocess Tweet data.
- Implemented code to construct input data for all the models.
- Implemented and evaluated the TF-Ranking model.