Covid-19 Question-Answering System

Team: Shuyan Yu, Zhengjia Mao, Xiaoyu Liu **Project Mentor TA:** Sihao Chen

1. Abstract

In this project, we dived into one of the hottest Natural Language Processing (NLP) topics, a document ranking task specifically, and built a Covid question-answering (QA) system based on the Retrieve-and-Rerank framework. This task is important for efficiently extracting relevant information from a large amount of documents. Our primary target contribution is to optimize traditional QA systems that generally rely on a single NLP model and require expensive computing resources. We, instead, experimented document retrieval models, TF-IDF and BM25, with different tokenization approaches to narrow down the ranges of target documents and then utilized strong learner BERT to rerank the retrieved data to improve relevantivity of the recommended documents. This optimized QA system can be simple while effective even with limited computational resources. Given the untokenized ranking results by TF-IDF as a benchmark, we found that the combination of BM25 and BERT performed fairly well in recommending documents relevant to a given query.

2. Introduction

When a large amount of documents is presented, researchers and scientists usually get discombobulated searching for relevant information on a specific topic. Our project aims to make this task easier with less time and higher precision. As the user inputs a natural language query, our system is expected to recommend a list of documents that are semantically related to the query in the descending order of relevance.

Our QA system was established based on the open dataset published on Kaggle's COVID-19 Open Research Dataset Challenge. The original dataset consisted of 335k JSON files totaling 33.91 GB. Each file included a paper ID, a title, an abstract, body texts, bibliography entities, reference entities, and back matters. Due to computational limitations on Google Colab, we shuffled and drew 10000 random samples from the database for training. The project pipeline involved two stages: retrieving and reranking. In the retrieving phase, TF-IDF and BM25 were implemented to return a list of 500 files relevant to the given query. And then during the reranking phase, the retrieved files were passed to BERT which returns the top k files based on reranking scores (here k is less than or equal to 10). Note that all the models were trained on title, abstract, and body text sections.

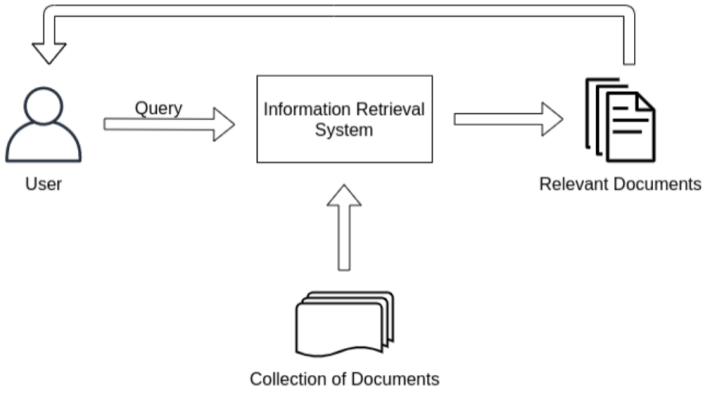

Figure 1: Question-Answering System Diagram

3. Background

A. Ubiquitous Knowledge Processing Lab, "Retrieve & Re-Rank", 2021, GitHub. Retrieved 4/1/2022 from https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/retrieve_rerank
This reference guided us through the idea of retrieve & re-rank and how to construct a BM25 model. We built our BM25 retrieving part with inspiration from this code and modified our model to fit the CORD-19 dataset.

B. Chris McCormick, "How To Build Your Own Question Answering System", 2021, McCormickML. Retrieved 4/1/2022 from https://mccormickml.com/2021/05/27/question-answering-system-tf-idf/
This reference constructed a BERT model for a different dataset based on a library that we are able to reproduce. We built our BERT reranking part with reference to this code.

4. Summary of Our Contributions

Contribution(s) in Code: Basically, we customized the entire project by ourselves with the adoption of the "retrieve & rerank" framework from PyGaggle [Castorini, 2021]. We first implemented the data preprocessing procedure by multiple tokenization approaches, then built a TF-IDF embedding from scratch to narrow down the range of relevant documents. We also modified the BM25 model to make it compatible with the CORD-19 context. On top of that, we applied BERT with tuned hyperparameters to rerank the retrieved articles.
Contribution(s) in Algorithm: Our main contribution in algorithm was to manually construct a TF-IDF and a BM25 retrieval model from scratch. As the vectorizer in Scikit-learn does not allow customization in terms of how to separate punctuations, we defined several rules for the algorithm better handling the CORD-19 text data.
Contribution(s) in Analysis: In addition to the traditional evaluation metrics such as recall, precision etc., we measured the macro-averaged performance of the reranked list of documents regarding each training query with the intention of treating all classes equally.

5. Detailed Description of Contributions

1) Contributions in Code:

Initially, we proposed to utilize the neural ranking algorithm from PyGaggle as a baseline for our CovidQA. Nonetheless due to package dependency issues and the limitation of computational resources, we failed to reproduce the results from PyGaggle. Thus instead, we adopted its underlying "retrieve & rerank" idea and constructed the whole QA framework by ourselves. As the training set is stored in JSON format, we first created a data preprocessing pipeline to load and separate the text body in terms of queries and potential lists of relevant articles. The following preprocessing steps involved stopwords removal and documents indexing. Our initial experiment was implemented on the preprocessed corpus without tokenization, which was identified as a benchmark for our ranking models. Considering the existing vectorizers from Scikit-learn do not allow customization of the ways to handle different types of punctuations, we ended up designing our own tokenizers and document ranking models from scratch. The retrieving process mainly relied on TF-IDF and BM25.

The first tokenization approach consisted of removing common English stopwords, removing standard generalized markup language, regularizing date/time data, handling acronyms, abbreviations, numbers, possessive nouns, hyphenated phrases. On the contrary, the second method was hard-coded based on observations. On top of these two approaches, we manually computed term frequencies and document frequencies, then built a TF-IDF model, outputting scores that indicate the relavantivity of each article and the given query. Furthermore, with the intention of experimenting on different retrieval methods, we repeated the embedding process by the BM25 framework from

scratch. Upon the computation of document frequency and index frequency, we parsed the corpus and queries, then calculated the BM25 scores to rank each document for the first time.

2) Contributions in Algorithm:

One of the most important steps before NLP modeling tasks is to tokenize data. Considering that text data is usually skewed and not well-organized, we needed to standardize both input documents and queries to make sure there was no mismatch between words that refered to the same thing. In hopes of designing a specific preprocessing solution for the CORD-19 data context and going above and beyond the tokenization approaches in existing libraries such as NLTK, we set up a basic preprocessing pipeline to manipulate the training documents from scratch mainly using regular expressions. For each file, we extracted title, abstract, and body text sections from the JSON data source and stored them as a list of sentences. The corpus list was then fed into a customized tokenizer which separated punctuations among words and normalized the cleaned corpus. In respect of punctuation, we detailed multiple edge cases in natural language and created some principles as following:

.	Acronyms, abbreviations, numbers are not tokenized
‘	Expand while needed (e.g. can’t -> can not, Monday’s -> Monday ‘s)
-	Group the phrases separated by - together
,	Numbers are not tokenized

For normalization, we integrated different formats of the same word (e.g. jan, Jan. → January; de-accent words such as resume) and unified the date/time data to ensure them to be displayed in a consistent manner.

In addition, we built a TF-IDF model from scratch and performed the embedding on a sample set of 10,099 files from CORD-19. This TF-IDF solution involved tokenization described above as well as the process of constructing Vector Space Models through TF-IDF weighting scheme. We first computed the inverted index dictionary of all documents’ tokens using term frequency as shown in Eq. (1):

$$tf_{t,d} = \frac{f_{t,d}}{max\{f_{t,d}\}} \tag{1}$$

Then we combined the term frequency with inverse document frequency to obtain the TF-IDF weight of each term through Eq. (2), where df_t stands for the number of documents that contain *t*. To initialize the retrieving process, we also computed the weights of both queries and documents then compared the cosine similarity between them.

$$w_{t,d} = tf_{t,d} \cdot \log_{10}(\frac{N}{df_t}) \tag{2}$$

Another innovation in algorithm was the BM25 document retrieval model we built from scratch as well. Similarly, we computed the query term *q* and the inverse document frequency of the term *IDF(q)*. Note that the IDF component of BM25 measures how often a term occurs in all of the documents and penalizes terms that are common, which is slightly different from the IDF formula in TF-IDF shown in Eq. (3):

$$ln(1 + \frac{docCount - f(q_i) + 0.5}{f(q_i) + 0.5}) \tag{3}$$

Intuitively, the queries containing rarer terms generally result in greater multipliers, thus they contribute more to the final score. Then after the queries and document texts being parsed, the BM25 score of each article is obtained by Eq. (4):

$$\sum_i^n IDF(q_i) \frac{f(q_i, D) \cdot k_1 + 1}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{fieldLen}{avgFieldLen})} \tag{4}$$

where the term *b* indicates the proportional effect of the length of the document compared to the average length, and the term frequency saturation characteristic *k1* limits how much a single query term can affect the score of a given document.

3) Contributions in Analysis:

The major evaluation metrics employed for our system are macro-averaged precision and macro-averaged recall. In general, a macro-average computes the metric independently for each query and then takes the average, thus weighing all queries/classes equally. Mathematically, precision and recall are both computed over individual documents in the prediction against those in the true answer. Precision is the ratio of the number of shared documents to the total number of documents in the prediction, while recall is the ratio of the number of shared documents to the total number of documents in the ground truth. The macro-averaged performance is simply averaging the scores by the number of queries,

$$MAP/MAR = \frac{1}{Q}\sum_{q=1}^{Q} precision/recall(q) \tag{5}$$

where *Q* is the total number of queries, *AP(q)* is the average precision for query *q*.

5.1 Method

Retrieve-and-Rerank: The tasks "retrieve" and "rerank" are different only on how large the candidate pool is. It refers to using simple models to retrieve a narrowed list of candidate documents related to a query from a larger pool for optimization purposes, then using more sophisticated models to rerank the selected candidates with advanced scoring schemes.

TF-IDF: Known as Term Frequency Inverse Document Frequency, it is an algorithm that transforms natural language text into a numerical representation as indexed values. The main idea is to give more weights to frequent words: "tf" (term frequency) refers to the frequency of a term in a document, and "idf" (inverse document frequency) refers to the frequency of documents that contain the term [Mukesh Chaudhary, 2020].

BM 25: It can be seen as an improved version of TF-IDF, which uses the TF-IDF structure but adds more complexity to it. BM25 adds term saturation parameter, multi-term evaluation, document length parameter [Rudi Seitz, 2020].

BERT: Known as Bidirectional Encoder Representations from Transformers, it is a more sophisticated embedding model developed by Google. After sentences are tokenized (BERT has its own tokenizer), the words are converted into high-dimensional tensors through a multi-layer network [Chris McCormick, 2019]. Unlike sequential models, BERT is bi-directional and pretrained with two NLP tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM randomly masks a word in a sentence and predicts it using transformers from both sides at the same time, and NSP takes two adjacent sentences each time to learn the contextual relations between them. More intuitively, BERT differs from Word2vec in that one word can have different embedding representations in different contexts [Pavan Sanagapati, 2021].

5.2 Experiments and Results

Experiment Problem Setup

Our project aims to build a QA system with experiments on comparing multiple models (TF-IDF, BM25, BERT) that can help return a list of documents relevant to a given query. BERT is known for one of the most advanced NLP models for building a QA system [Dai & Callan, 2019]. As the limited computing resources were not able to support BERT to retrieve documents from the entire mega pool with 300k documents, we decided to implement the Retrieve & Rerank method for optimization purposes. We aimed to validate the hypothesis that the Retrieve & Rerank method is advantaged in consuming less computing resources while achieving satisfying results that are good enough to beat the baseline.

Notice all three models basically do the same task of selecting the most relevant documents from a pool. We differentiated the stages"retrieve" and "rerank" only based on how large the pool is. First, we manually selected 10k random JSON files from a 300k-document pool. Our baseline solution is to retrieve a narrowed down the candidate documents related to a query from the 10k pool simply using TF-IDF without any tokenization. To beat the baseline solution, we used TF-IDF and BM25 with tokenization to narrow down the candidate documents for the reranking stage, we retrieved top 10, 50, 100, and 500 files for each query and calculated the metrics to compare and contrast performances of different models. During the reranking stage, we performed BERT on the retrieved files from the previous step and zoomed in to top 1, 5, and 10 files to evaluate model performance.

Standards and Metrics

Given its best results produced by human specialists' literature review, we compared our performance metrics with respect to it. The evaluation metrics we employed include macro averaged precision (details discussed in the contribution section). The answers/standards we used here were the literature review results used in PyGaggle models. There were 27 queries in total along with 155 document IDs as the standard answers to queries. Notice that the distribution of given answers was imbalanced as query 1 had 26 marked answers but query 27 only had 1 document. We utilized the metadata file to construct links between the file IDs retrieved by standard datasets and the filenames of training data, which prepared us for future analysis.

4) Results

A. Retrieving Stage

Model	Metric	Top 10	Top 50	Top 100	Top 500
TF-IDF (no tokenization) ~30min	Precision	3.7%	1.78%	1.37%	0.39%
	Recall	10.93%	20.25%	36.75%	42.82%
	# Files	172	272	523	2172
TF-IDF (tokenization 1) ~5min	Precision	4.81%	1.85%	1.30%	0.41%
	Recall	8.75%	15.50%	28.22%	38.63%
	# Files	49	211	418	1770
TF-IDF (tokenization 2) ~5min	Precision	4.07%	1.85%	1.37%	0.47%
	Recall	14.63%	20.78%	37.73%	47.37%
	# Files	72	274	525	2192
BM25 ~3min	Precision	7.04%	2.30%	1.30%	0.41%
	Recall	13.08%	23.32%	26.23%	44.99%
	# Files	164	691	1211	4021

To locate down on a range of candidates, we chose TF-IDF with tokenization 2 as the final retrieved model since it balanced the performance with running time and also returned a reasonable amount of unique files. Notice that the TFIDF without any tokenization will take too long to execute but still with less recall and BM25, although good in performance, retrieved too many documents (4021 compared to 2192).

B. Reranking Stage

Model	Metric	Top 1	Top 3	Top 5
TF-IDF (tokenization 2)	Precision	0.0%	1.23%	3.7%
	Recall	0.0%	0.15%	4.93%
	# Files	7	20	38
TF-IDF (tokenization 2) + BERT	Precision	14.81%	11.11%	7.41%
	Recall	3.42%	9.16%	10.09%
	# Files	22	40	57

In the reranking stage, while retrieving and targeting final answers, adding BERT as the reranking stage showed a clear improvement compared to simply TF-IDF (tokenization 2) to limit answers within range of 5.

6. Compute/Other Resources Used

We mainly used Colab Pro for this project. Given the limited computational capacity and flexibility, we manually selected 10,000 random samples from the 300k documents pool to optimize the process.

7. Conclusion

Our final results have reached the benchmark achieved by the untokenized TF-IDF baseline model, and gone above and beyond in implementing multiple tokenizing and retrieving techniques, with the best performing combination selected for our QA system. We did find this learning process fruitful and challenging. Our initial intention at the beginning of the semester was to build a QA system by virtue of a Knowledge Graph embedding, while as the project progressed, we became aware that the lack of referenceable resources and the lack of our own in-depth knowledge towards this advanced topic would hinder us from moving forward. Meanwhile, we noticed the results of one of the only few available resources from Github was unable to be reproduced on our end given the limited computational resources and package dependency issues on Google Colab. Thanks to our mentor, Sihao, who has been absolutely responsive and helpful throughout the semester, we were guided to refine our focus on the retrieve-and-rerank framework and build the algorithms from scratch.

For future work, we would consider experimenting on more robust ranking models such as monoT5, to improve our system performance. We would also test on more different combinations of models to set up a good pipeline of document retrieval. In terms of computational limits, we would look into other dedicated servers such as AWS, and other database frameworks that support fast compilation with NLP models, like ElasticSearch and Hugging Face. So far, our QA system has succeeded in recommending COVID articles for input queries. As we integrate more advanced algorithms and feed more training samples into the system, we believe our project would be useful as a COVID document searching tool one day.

References (Exempted from page limit)

1. Pavan Sanagapati, “Knowledge Graph & NLP Tutorial-(BERT,spaCy,NLTK)”, 2021, *Kaggle*. Retrieved 2/15/2022 from https://www.kaggle.com/pavansanagapati/knowledge-graph-nlp-tutorial-bert-spacy-nltk

2. Sandyvarma, “Covid-19: BERT + MeSH Enabled Knowledge Graph”, 2020, *Kaggle*. Retrieved 2/15/2022 from https://www.kaggle.com/sandyvarma/covid-19-bert-mesh-enabled-knowledge-graph#Abstract

3. Shahules, “CORD : Tools and Knowledge graphs”, 2020, *Kaggle.* Retrieved 2/15/2022 from https://www.kaggle.com/shahules/cord-tools-and-knowledge-graphs/notebook

4. Mukesh Chaudhary, “TF-IDF Vectorizer scikit-learn”, 2020, *Medium.* Retrieved 3/20/2022 from https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a

5. Ragini Patil, “Question Answering System using A TF-IDF Method”, 2020, *International Journal of Science, Engineering and Technology*. Retrieved 3/20/2022 from https://www.ijset.in/wp-content/uploads/IJSET_V8_issue4_257.pdf

6. Dhilip Subramanian, “Content-Based Recommendation System using Word Embeddings”, 2020, *KDnuggets.* Retrieved 3/20/2022 from https://www.kdnuggets.com/2020/08/content-based-recommendation-system-word-embeddings.html

7. Chris McCormick, “BERT Word Embeddings Tutorial”, 2019, *McCormickML*. Retrieved 3/20/2022 from https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/#what-is-bert

8. Ubiquitous Knowledge Processing Lab, “Retrieve & Re-Rank”, 2021, *GitHub*. Retrieved 4/1/2022 from https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/retrieve_rerank

9. Chris McCormick, “How To Build Your Own Question Answering System”, 2021, *McCormickML*. Retrieved 4/1/2022 from https://mccormickml.com/2021/05/27/question-answering-system-tf-idf/

10. Rudi Seitz, “Understanding TF-IDF and BM-25”, 2020, *KMW Technology*. Retrieved 4/1/2022 from https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/

11. Zhuyun Dai & Jamie Callan, “ Deeper Text Understanding for IR with Contextual Neural Language Modeling”, 2019, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Retrieved 4/1/2022 from https://arxiv.org/abs/1905.09217

12. Castorini, “PyGaggle: Neural Ranking Baselines on CovidQA”, 2021, GitHub, Retrieved 3/20/2022 from https://github.com/castorini/pygaggle

Broader Dissemination Information:

Your report title and the list of team members will be published on the class website. Would you also like your pdf report to be published?

- YES

If your answer to the above question is yes, are there any other links to github / youtube / blog post / project website that you would like to publish alongside the report? If so, list them here.

- PyGaggle: Neural Ranking Baselines on CovidQA: https://github.com/castorini/pygaggle/blob/master/docs/experiments-covidqa.md

Proposed Schedule (Exempted from page limit)

PERSON (S)	TASK (S)	Wk5				Wk6				Wk7				Wk8				Wk9			
		MAR				APR															
		S 25	M 27	W 28	F 30	S 3	M 4	W 6	F 8	S 10	M 11	W 13	F 15	S 17	M 18	W 20	F 22	S 24	M 25	W 27	
All	Midpoint Report																				
Zhengjia, Shuyan	Reproducing Baseline																				
Zhengjia	Trouble Shooting																				
Xiaoyu, Shuyan	Data Preprocessing																				
Zhengjia, Xiaoyu	TF-IDF																				
All	Checkpoint with Mentor																				
Xiaoyu	BM25																				
Zhengjia	BERT																				
Shuyan	Model Tuning																				
All	Final Report Wrapup																				

Midterm Report (Exempted from page limit)

Evaluation Question [Select all pages] 7 / 7 pts

✔ + 1 pt	Does the report follow the provided template including the 4-page limit (excluding exempted portions), with reasonable responses to all questions?
✔ + 2 pts	Has feedback from the last round been effectively addressed?
✔ + 1 pt	Has the team identified a clear topic and viable new target contribution, as per the project specifications provided in class?
✔ + 1 pt	Has the team moved in a non-trivial way towards their target contribution?
✔ + 2 pts	Has a clear and systematic work plan been formulated for the remaining weeks?

Great progress!

I don't have many other high-level things to suggest since we last met, but here are a few ideas that might help with the technical side of the project.

1. As I mentioned -- The way document retrieval with neural model works is usually called Retrieve-and-Rerank: You use BM25 of TF-IDF to retrieve a large list (say ~100) of candidate documents related to a query. Then you use neural models (e.g. BERT, monoT5) for re-ranking the 100 candidates. For each candidate, the BERT model will output a logit/score, then you use the score to rank the 100 candidates.

2. On how to run BM25 TF-IDF document retrieval on the scale of 300k+ documents -- If you have a dedicated server such as AWS, you might want to look into elasticsearch, an open-source database framework that supports very fast retrieval with TF-IDF and BM25. If you don't have AWS credits, I'm actually happy to set it up using our one of our research groups servers, and I'll let you know how to programmatically access the APIs. Let me know.

3. monoT5: Good find! but I'm worried that you don't have enough computational resources to use this model, as T5 models are not memory-optimized for GPUs. If that's truly the case, consider switching back to a BERT-like model.

Happy to discuss further.

Covid-19 Question-Answering System

Team: Shuyan Yu, Zhengjia Mao, Xiaoyu Liu. **Project Mentor TA:** Sihao Chen

1. Introduction

Problem Setup

When a large amount of documents and articles is presented, researchers and scientists usually get discombobulated searching for relevant information on a specific topic. Our project aims to make this task easier. As the user inputs a natural language query, our system is expected to recommend a list of documents that are semantically related to the query.

Technically, we propose to build a COVID-19 QA system with TF-IDF, word2vec, BERT, and Knowledge Graph embeddings, meanwhile performing a compare-contrast analysis. If time allows, we will try to combine models to achieve a better performance. An open dataset published on Kaggle's COVID-19 Open Research Dataset Challenge will be adopted to our model. Our training set consists of 335k JSON files totaling 33.91 GB. Each file includes a paper ID, a title, an abstract, body texts, bibliography entities, reference entities, and back matters. The models will only be trained on title, abstract, and body text sections.

Motivation

The proposed QA model has been commonly adopted to search engines and recommendation systems. While it aims to automatically provide solutions to given queries, it can also mislead audiences to a list of topics that are not originally desired. Otherwise, we believe that there are no bad social impacts related to this project regardless of whether it's successful or not. If it succeeds, it can be beneficial to the community who care about COVID-19 and the professionals who want to be more efficient in searching papers.

2. How We Have Addressed Feedback From the Proposal Evaluations

One of the key feedbacks that we received from our mentor is to focus on existing datasets instead of exploring other fields. While digging deeper in the area of NLP and QA tasks, we understand that labeling a new dataset related to cancer requires proficient knowledge in the biomedical field as well as a large amount of work. In this case, we will switch to use CORD-19, the open-source dataset on Kaggle, as our main training dataset and utilize CovidQA as the golden standard where our models will compare and contrast with.

Other feedback is related to performance metrics and the complexity of models. For main evaluation metrics, we formalize what we thought in the proposal and decide to use both F1 scores and MAP, the two common metrics in NLP tasks, to evaluate the performance of our models. We will consider using plots, such as the ROC curve, to visualize the performance. For models, we shift our focus from tuning the models so that they can be applied to different fields to constructing a baseline model and comparing/contrasting different document retrieval models. Hence, we decided to build and test at least three different models, TF-IDF, Word2vec, and BERT, and expect to find a way of combining multiple models to achieve better performance.

3. Prior Work We are Closely Building From

1) Pavan Sanagapati, "Knowledge Graph & NLP Tutorial-(BERT,spaCy,NLTK)", 2021, Kaggle. Retrieved 2/15/2022 from https://www.kaggle.com/pavansanagapati/knowledge-graph-nlp-tutorial-bert-spacy-nltk.

This reference gives a detailed explanation and implementation of BERT tokenization, training, and verification for general purposes. We may start with building a simple BERT model based on this source and implement it on the selected CORD-19 dataset.

2) Sandyvarma, "Covid-19: BERT + MeSH Enabled Knowledge Graph", 2020, Kaggle. Retrieved 2/15/2022 from https://www.kaggle.com/sandyvarma/covid-19-bert-mesh-enabled-knowledge-graph#Abstract
This reference constructs a specified BERT model for the selected CORD-19 dataset, and we will mostly build from this code. Additionally, this model combines Knowledge Graph with MeSH ontology into BERT embedding, which can be the potential improvements if the time allows.

4. What We are Contributing

Contribution(s) in Code: We customized the data preprocessing procedure by implementing different tokenization approaches. In terms of modeling, We have built a TF-IDF model from scratch as our baseline model and tested it on the CORD-19 dataset.
Contribution(s) in Analysis: We will reproduce models on a cleaned subset of CORD-19 and summarize the performance of each model through measuring the metrics discussed below.

5. Detailed Description of Each Proposed Contribution, Progress Towards It, and Any Difficulties Encountered So Far

1) Contribution(s) in Code:
Before sending data into models, one of the most important steps is to preprocess and tokenize the data. We have performed word tokenization using the NLTK library. Since the words are usually not well-organized and skewed, we need to standardize both the input documents and queries so that there is no mismatch between words representing the same thing. For now, we have set up a basic pipeline of preprocessing the input documents from scratch mainly using regular expressions. Our original data source is stored in JSON format. For each file, we extract the text parts of title, abstract, and body text sections and store them as a list of words. It is then sent to a customized tokenizer so that the tokenizer can separate punctuations from the words and normalize them. In the part of handling punctuations, we are careful about various cases in natural language and stick on some of the rules as following:

.	Acronyms, abbreviations, numbers are not tokenized
'	Expand while needed (e.g. can't -> can not, Monday's -> Monday 's)
-	Group the phrases separated by - together
,	Numbers are not tokenized.

For normalization, we create equivalent classes and replace when necessary (e.g. jan, Jan. -> january, de-accent words such as resume). We also browse to find some other powerful libraries which can be used on tokenizing and plan to apply some of them during experiments in the future.

In addition, we have built a TF-IDF model from scratch with the inspiration of the BERT solution from the existing code, and have it tested on a sample set of files in CORD-19. We will further implement a word2vec model and a BERT model in the remaining time. The current model includes a tokenization described above and

the process of constructing Vector Space Models through TF-IDF weighting scheme. We first compute the inverted index dictionary of all documents' tokens using term frequency:

$$tf_{t,d} - \frac{f_{t,d}}{max\{f_{t,d}\}}$$

Then we combine it with inverse document frequency to obtain the tf-idf weight of each term through $w_{t,d} = tf_{t,d} \cdot \log_{10}(\frac{N}{df_t})$ where df_t stands for the number of documents that contain t. To start the retrieving process, we compute the weights of both queries and documents and compare the cosine similarity between them.

Furthermore, we plan on implementing a more advanced word2vec model to perform embedding on the corpus. The detailed steps will be discussed in the following weeks.

3) Contribution(s) in Analysis:

The main evaluation metrics employed for our system are F1 score and mean averaged precision (MAP). F1 is suitable for our models as we care equally about precision and recall. Mathematically, F1 is computed over the individual words in the prediction against those in the true answer. The number of shared words between the prediction and the truth is the basis of the F1 score: precision is the ratio of the number of shared words to the total number of words in the prediction, and recall is the ratio of the number of shared words to the total number of words in the ground truth.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

In addition, MAP measures the average precision across multiple queries. Mathematically, this is given by

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(q)$$

where Q is the total number of queries, AP(q) is the average precision for query q.

5.1 Methods

In the big picture, our model involves: analyzing query, retrieving documents, selecting answers. To find the similarity between query and documents, cosine similarity technique finds the dimensional distance between vectorized documents and query [Ragini Patil, 2020]. Therefore, natural language texts need to be represented in a numerical/vector form to calculate the cosine similarity. The following three methods are proposed, ordered by complexity ascending.

TF-IDF: Known as Term Frequency Inverse Document Frequency, it is an algorithm that transforms natural language text into a numerical representation as indexed values. The main idea is to give more weights to frequent words: "tf" refers to the frequency of the term in a document, and "idf" refers to the frequency of documents that contain the term [Mukesh Chaudhary, 2020].

Word2vec: Word2vec is a single-layer neural network model. While word2vec also represents text numerically as TF-IDF does, it uses a more complex technique called embedding that vectorizes words and their adjacent words in a probability manner. The vectorized words are in a higher dimensional space, and similar words have similar vectors pointing towards the similar directions [Dhilip Subramanian, 2020].

BERT: Known as Bidirectional Encoder Representations from Transformers, it is a more sophisticated embedding model developed by Google. After sentences are tokenized (BERT has its own tokenizer), the words are converted into high-dimensional tensors through a multi-layer network [Chris McCormick, 2019]. Unlike sequential models, BERT is bi-directional and pretrained with two NLP tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM randomly masks a word in a sentence and predicts it using transformers from both sides at the same time, and NSP takes two adjacent sentences each time to learn the contextual relations between them. More intuitively, BERT differs from Word2vec in that one word can have different embedding representations in different contexts [Pavan Sanagapati, 2021].

5.2 Experiments and Results

In reference to the introduction section, our project aims to build a QA system with experiments on multiple embedding approaches. Our baseline solution, as suggested by Sihao, implemented neural reranking with random, BM25 and monoT5 on the exact same CovidQA dataset as ours. Given its best results produced by monoT5, we will compare our performance metrics against it. The evaluation metrics we employed include F1 score and mean averaged precision.

So far, we have experimented on the customized TF-IDF approach and performed the training-retrieving on a subset of original datasets. We have input 10 queries along with around 100 files and output the result containing retrieved documents along with their cosine similarity scores in the format of (query_id, paper_retrieved_id, score). We have attached partial results in the appendix section. Notice that because of the limitations mentioned in section 6, we failed to perform the metrics measure of our model for now.

We have also utilized the metadata file to construct the link between the id of files retrieved by standard datasets and the filenames of training data, which help us prepare for future analysis.

6. Risk Mitigation Plan

Given the complete CORD-19 dataset is enormous and preprocessing is time-consuming, our minimum viable project is to build at least a TF-IDF-based QA system on a subset of the complete dataset. With this simplified version constructed, we can increment the model by incorporating more advanced techniques such as word2vec, BERT, KG, or even combining the techniques. Even if upgrading from the minimum viable version fails, this project based on TF-IDF is still useful in searching for relevant documents given a query. At this point, one of the main issues we encountered is the lack of computing resources.

The original dataset consists of nearly 335k json files and it takes a significant amount of time to compile on Colab. We plan on reproducing the baseline results on a subset of its original training set, then performing our own solutions on the subset as well so we can compare our output with the "gold standard". Once our models are tested and refined, we may consider applying them to the complete dataset therefore to obtain more generalized training parameters. We would upgrade Google Colab or rely on other resources with better computational power like AWS.

Supplementary Materials (Exempted from page limit)

Main Work Repository:
https://drive.google.com/file/d/1KgzpsEGf63kI7F2BhRXu7QgZpSrMHSxe/view?usp=sharing

Experiment Results:

```
tfidf-tfidf
Top 1
  retrieved(relevant) = 0
  retrieved(total) = 27
  precision(macro-avg) = 0.00%
  recall(macro-avg) = 0.00%
total 7
Top 5
  retrieved(relevant) = 5
  retrieved(total) = 135
  precision(macro-avg) = 3.70%
  recall(macro-avg) = 4.93%
total 38
Top 10
  retrieved(relevant) = 11
  retrieved(total) = 270
  precision(macro-avg) = 4.07%
  recall(macro-avg) = 14.63%
total 72
Top 50
  retrieved(relevant) = 25
  retrieved(total) = 1350
  precision(macro-avg) = 1.85%
  recall(macro-avg) = 20.78%
total 274
Top 100
  retrieved(relevant) = 37
  retrieved(total) = 2700
  precision(macro-avg) = 1.37%
  recall(macro-avg) = 37.13%
total 525
Top 500
  retrieved(relevant) = 56
  retrieved(total) = 13500
  precision(macro-avg) = 0.41%
  recall(macro-avg) = 47.37%
total 2192
Top 1000
  retrieved(relevant) = 65
  retrieved(total) = 27000
  precision(macro-avg) = 0.24%
  recall(macro-avg) = 53.65%
total 3739
```

```
BM25
Top 1
  retrieved(relevant) = 3
  retrieved(total) = 27
  precision(macro-avg) = 11.11%
  recall(macro-avg) = 4.78%
total 21
Top 5
  retrieved(relevant) = 10
  retrieved(total) = 135
  precision(macro-avg) = 7.41%
  recall(macro-avg) = 8.03%
total 91
Top 10
  retrieved(relevant) = 19
  retrieved(total) = 270
  precision(macro-avg) = 7.04%
  recall(macro-avg) = 13.08%
total 164
Top 50
  retrieved(relevant) = 31
  retrieved(total) = 1350
  precision(macro-avg) = 2.30%
  recall(macro-avg) = 23.32%
total 691
Top 100
  retrieved(relevant) = 35
  retrieved(total) = 2700
  precision(macro-avg) = 1.30%
  recall(macro-avg) = 26.23%
total 1211
Top 500
  retrieved(relevant) = 56
  retrieved(total) = 13500
  precision(macro-avg) = 0.41%
  recall(macro-avg) = 44.99%
total 4021
Top 1000
  retrieved(relevant) = 56
  retrieved(total) = 27000
  precision(macro-avg) = 0.21%
  recall(macro-avg) = 44.99%
total 4021
```

```
BERT
Top 1
  retrieved(relevant) = 4
  retrieved(total) = 27
  precision(macro-avg) = 14.81%
  recall(macro-avg) = 3.42%
total 22
Top 5
  retrieved(relevant) = 10
  retrieved(total) = 135
  precision(macro-avg) = 7.41%
  recall(macro-avg) = 10.09%
total 57
Top 10
  retrieved(relevant) = 12
  retrieved(total) = 270
  precision(macro-avg) = 4.44%
  recall(macro-avg) = 10.76%
total 91
```

```
Input question: What are the risks of COVID-19 infection in pregnant women?
Top-3 lexical search (BM25) hits
        23.999  b2696d577d29fb5656f54ad052d6251c2c2ed084        In December 2019, a series of pneumor
        23.615  4e3e70755c327f0207dc992bc6ec837fc21e5a87        surge across the globe, the approval
        23.334  99e06a62a0540449852f2f0ff519a75ba31121d4        Shortly following the emergence of th

-------------------------

Top-3 Bi-Encoder Retrieval hits
        0.739   99e06a62a0540449852f2f0ff519a75ba31121d4        Coronavirus disease 2019 (COVID-19) h
        0.728   99e06a62a0540449852f2f0ff519a75ba31121d4        Pregnant women with coronavirus disea
        0.717   99e06a62a0540449852f2f0ff519a75ba31121d4        The two opening decades of the third

-------------------------

Top-3 Cross-Encoder Re-ranker hits
        0.728   af25f32f5916b5dcd17f7197b18c7d154f4a9230        Pregnant women with coronavirus disea
        0.739   af25f32f5916b5dcd17f7197b18c7d154f4a9230        Coronavirus disease 2019 (COVID-19) h
        0.717   af25f32f5916b5dcd17f7197b18c7d154f4a9230        The two opening decades of the third
```

```
query_from_user = input("Enter a query related to COVID-19: ")
QA(query_from_user)
```

```
Enter a query related to COVID-19: what are the symptoms?
========================================
1 2f5e6026c5f4241d0636dfd05e0c230f65d58021
Title:  Persistent viral shedding of SARS-CoV-2 in faeces - a rapid review
Abstract:  Background and aims In addition to respiratory symptoms, patients with COV
========================================
2 02f0425d5b797650c643ec9e9d6a37dbbe2ab01c
Info not found
========================================
3 ec517ec7deec114e40cd1a5f538ea3565c0c400d
Title:  Defining the role of asymptomatic SARS-CoV-2 transmission: a living systemati
Abstract:  Background Reports suggest that asymptomatic individuals (those with no sy
========================================
4 192a10bcf3d863608281994484de0d96a2f41dea
Title:  The role of asymptomatic SARS-CoV-2 infections: rapid living systematic revie
Abstract:  Background: There is substantial disagreement about the level of asymptoma
========================================
5 4fabdcf18948e7fd575dee30654f1bb01ac0b6ee
Title:  Outbreak of COVID-19 in a family, Wenzhou, China
Abstract:  Since December 2019, China has experienced a widespread outbreak of COVID-
```