Question Answering for COVID-19 Open Research Dataset Challenge

Manlu He, Yanfang Hou

22 July 2020

1 Introduction

In this project, we try to address multiple question answering subtasks given by the COVID-19 Open Research Dataset Challenge [2] using TF-IDF, Word2Vec and BlueBERT and evaluate their performances based on several metrics. The experiments run on DSlab clusters using Spark to distribute the data and computations. The methods, metrics, experimental settings and results are discussed in details in the following sections.

1.1 Dataset

The COVID-19 Open Research Dataset (CORD-19) [2] are scholarly articles about COVID-19, SARS, and related coronaviruses with a size of 13GB, provided by the White House and some research groups. It includes 138,794 records and 68,000 of them have full texts. The metadata includes 18 features which describe the data: sha, source, title, abstract, publish time, pmc id, etc. The full texts are provided in json format which consists of paper id, abstract, body texts and references.

1.2 Problem

In this project, we focus on several subtasks given by the kaggle challenge, which are more related to public concerns and require less clinical experience and expertise. Given a query(subtask) shown in table 1, the model will return a list of relevant papers along with the most related sentences. Also, we preprocess the original questions from the following aspects to retrieve more accurate results:

- Complete the question: To make the queries more specific, we slightly modify some of the original queries. For example, the original query "Range of incubation periods for the disease in humans" is changed to "What is the incubation period of COVID-19?". By doing this, "the disease" is specified as "COVID-19" in the new query. Also, we exclude words like "range" because such words are general and might cause noise in different context.

- Split long question into multiple queries: Some original subtasks contain mutiple queries, which makes the targeting of keywords more complicated. Hence, splitting a question into multiple queries facilitates the question-answer matching. For example, an original subtask "Data on potential risks factors: Smoking, pre-existing pulmonary disease, Co-infections and other co-morbidities, Neonates and pregnant women..." is divided into query 6 and query 7 shown in table 1 in our case.

Table 1: The queries for question answering

ID	Query
0	What is the incubation period of COVID-19?
1	What is the prevalence of asymptomatic COVID-19 infection?
2	Do temperature and humidity affect the transmission of SARS-CoV-2?
3	How is the persistence of the coronavirus on surfaces of different materials?
4	What diagnostics have been implemented for SARS-CoV-2?
5	Are movement control strategies effective to prevent transmission of SARS-CoV-2?
6	Is smoking a risk factor for COVID-19?
7	What are the risks of COVID-19 infection in pregnant women?
8	How is the immune response to SARS-CoV-2?
9	What percentage of patients with SARS-CoV-2 are hospitalized?

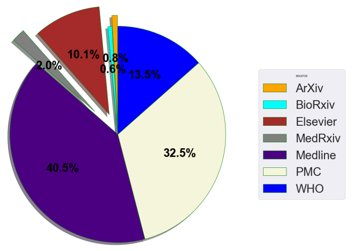
1.3 EDA

To gain an initial insight of the CORD-19 and generally uncover some properties of the dataset, we visualize the data on certain features locally.

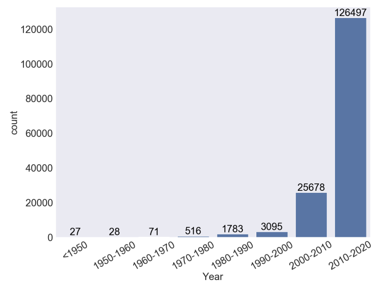
- **Sources**: Papers come from different sources, which are shown in Figure 1(a). It is obvious that most of the papers are Medline and PMC sourced.
- **Publish year**: According to the Figure 1(b), we are well informed that the number of the published scholarly articles about coronaviruses increases during these years and have surged within the last 20 years.
- **Abstract keywords**: We retrieve the most frequent words from the abstract of all the papers and generate the wordcloud in Figure 2 which presents words of more prominence in larger size.

2 Method

Referring to the approach described in [1], our method consists of two components: (1) retriever module for finding relevant documents to narrow search space. (2) reader module for extracting the most related sentences from a small collection of documents.



(a) Sources of the papers



(b) Publish year of the papers

Figure 1: EDA Results

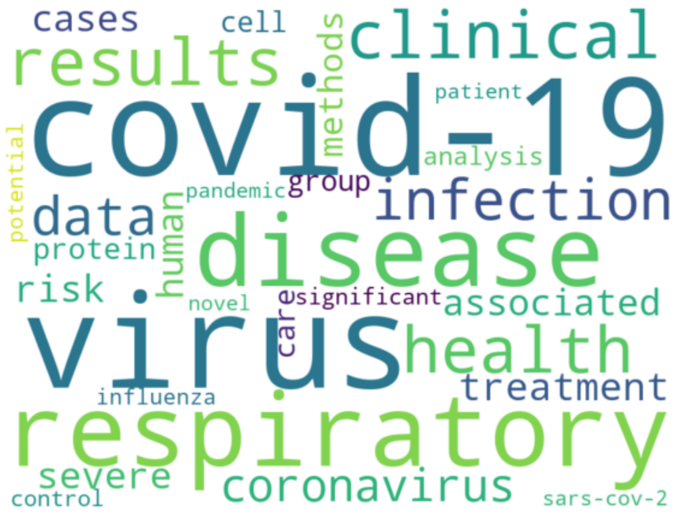


Figure 2: Word cloud of the abstract of all the papers

2.1 Retriever

We identify relevant documents from 68,000 articles by applying the TF-IDF model which assigns importance to each word in a document based on its term frequency and inverse document frequency. Intuitively, the documents where the less-common terms in a query occur frequently usually have high tf-idf scores. The main procedure are described as follows.

(a) Convert each article into a set of words;

(b) Identify articles that are relevant to COVID-19 based on whether they contain the specified keywords;

(c) Remove stop words from document words;

(d) Compute tf-idf value for each word in each document;

(e) Summing up the tf-idf values of query terms in a document as the document score.

(f) Select the top 100 articles with the highest tf-idf scores as candidate documents.

In the above steps, (a)(b)(e)(f) are processed by Spark UDF functions. (c)(d) are implemented by *StopWordsRemover*, *CountVectorizer* and *IDF* transformers provided by Spark MLlib. We write out the paper ids of candidate papers into disk for reuse.

2.2 Reader

After obtaining candidate articles, we utilize language models to produce embedding vectors for each sentence, which are used to analyze the similarity between each sentence and the given query. We try the two methods, one is Word2Vec [5] and the other is BlueBERT [6]. The next parts describe the detailed procedures.

2.2.1 Data preprocessing

Before feeding data into model, we first split text into sentences and delete irrelevant sentences to narrow down search space and reduce memory burden. The specific rules are as follows: (1) discard the paragraphs that do not contain any query words or their lemma forms; (2) remove sentences ending with "?"; (3) eliminate sentences shorter than 7 words. The splitted sentences are saved as multiple files into disk for the use of different models. Each query has an average of 44,695 candidate sentences.

2.2.2 Word2Vec

Sentence embeddings are generated by using Word2vec model that are capable of capturing context of a word and meaningful semantic relationships. First, we use the Word2Vec estimator from Spark MLlib to train the model on the whole corpus (14,953,775 sentences). To efficiently train the model, we increase the number of partitions to 10 and conduct multiple iterations for model accuracy. Then the trained model is used to transform each preprocessed sentence and query to a dense vector with 100 dimensions. Finally, we calculate the cosine similarity between each sentence and the given query and return the 20 most similar sentences.

2.2.3 BlueBERT

In another method, we embed each sentence by using BlueBERT [6] model which is pre-trained on PubMed abstracts and clinical notes. An important advantage of the BlueBERT model over Word2vec is the capacity of creating word representations that dynamically change with the word context. However, the BlueBERT model is very computationally heavy and requires more memory resources. We adopt the following strategies to address out-of-memory issue and speed up the program.

Batch processing in parallel For each query, there are 100 candidate sentence files and we use *sc.textFile* to load data to obtain a RDD with 100 partitions. Each thread holds one partition for predicting the similarity of sentences and this process is implemented by *rdd.mappartitions* function. Besides, we use pre-trained BlueBERT model with Huggingface transformers [8] to predict the embeddings of sentences with a batch size of 32. Such matrix multiplication can make it run faster.

Writing out multiple files We give up using *collect* function which forces Spark to collect all data into the driver. Instead, each executor parallely write down the result of partitions into separate files. This is achieved by *rdd.saveAsSingleTextFile* function. Besides, before sorting the similarity, we first filter out the sentences whose similarity are less than 0.75.

Tuning configuration parameters (1) There are 8 workers nodes and each node has 16 CPU cores and 22.5G memory available. We assign two executors to each node and each executor is allocated to 7 cores and 10G memory. We do not use all cores since the current number of cores are sufficient to handle 100 partitions and also provide more memory for each task. (2) We extend the executor heartbeat interval and network timeout appropriately to prevent executors from being killed wrongly by driver. (3) We adjust *spark.memory.storageFraction* to a lower value to allocate more working memory for execution.

Table 2: Spark configuration setting

Parameter	Value
Number of workers	8
spark.driver.memory	50G
spark.executor.instances	2
spark.executor.memory	10G
spark.memory.storageFraction	0.1
spark.executor.heartbeatInterval	300s
spark.network.timeout	600s

3 Evaluation

We leverage the following three approaches to evaluate the quality of the embeddings and compare the methods.

3.1 Multidimensional scaling

Multidimensional scaling (MDS) is a method to visualize the distance among sets of points for analyzing similarity [7]. A dimensionality reduction is conducted while the original distance among points are well maintained [4]. Generally, closer points are more similar. In our case, after obtaining the results of each query, we map the sentence vectors into 2 dimensions for visualization.

3.2 K-means clustering and purity

K-means clustering is a method for cluster analysis. The algorithm divides all the observations into a pre-specified number of clusters described by the mean(cluster centroid) of the observations within the cluster [3], trying to minimizing within-cluster sum-of-squares defined in the Equation 1. Here the N denotes number of samples x, C is the set of clusters and μ is the mean.

$$\sum_{i=0}^{n} min_{\mu_j \in C}(\|x_i - \mu_j\|^2) \tag{1}$$

Since it is expected that answers for the same query belong to the same cluster answers as much as possible while answers for different queries are assigned into different classes after K-means clustering, we leverage purity given in Equation 2 as a metric to measure if clusters contain only observations of a single class.

$$Purity(\Omega, C) = \frac{1}{N} \sum_{k} max_j |\omega_k \cap c_j| \tag{2}$$

Here the N denotes the number of observations, Ω=(ω_1, ω_2, ..., ω_k) represents division of the clusters, C=(c_1, c_2, ..., c_j) is the real division of the classes.

3.3 Human rated score

Apart from the above two metrics that evaluate the answers intuitively, we evaluate answers using human sense, which can directly reflect the quality of the answers. We label each answer as '1' if it is relevant to the query otherwise it is labeled as '0' indicating irrelevant. Here we consider an answer as 'relevant' when the answer can directly answer the given query. An answer is considered as a correct answer only when '1' is assigned by both of us. After scoring all the answers, an average score is computed over all the queries for each method.

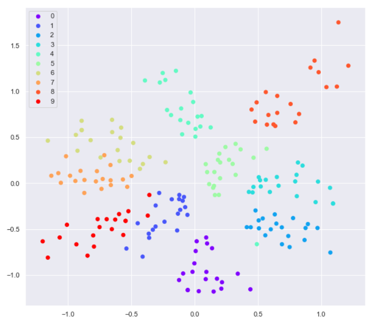
4 Results

In general, the model using Word2Vec obtains better results than the BlueBERT model. We discuss this performance difference mostly results from the fact that the BlueBERT model is not tuned on our corpus. A simplified example of the model output which excludes the paper id and the embedding is given in Table 5.

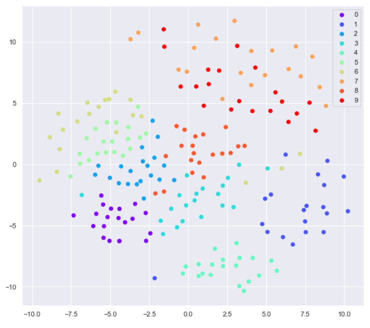
- **Multidimensional scaling**: According to the results shown in Figure 3, it is obvious that the Word2Vec model answers represented as points tend to stay closer and form separate clusters with apparent boundaries. As for the BlueBERT model answers, the distances among points within the same cluster are larger and the distances among different clusters are smaller, indicating the returned answers for the same query share less common features.

- **Purity**: As can be seen from the Table 5, the purity results obtained after k-means cluster is consistent with the above MDS results where the Word2Vec model outperformances the BlueBERT model. The 1.0 purity achieved by the Word2Vec model indicates the answers for the same query are assigned to the same cluster and the model has the ability to discriminate different queries.

- **Human rated score**: According to the results shown in Table 5, the Word2Vec model has an average score of 0.555, indicating more than half of the answers can properly answer corresponding queries. The BlueBERT model fails to produce better results, as the pre-trained model might not be able to capture certain words' features, especially for newly coined terminologies like 'COVID-19'. For example, the BlueBERT model outputs an answer related to asymptomatic infection but irrelevant to COVID-19 given query 1. Also, there are great performance difference among queries because some queries are more general while some are more specific. For example, the word 'humidity' from query 2 is a terminology without ambiguity and barely has synonyms hence being helpful in capturing word features.



(a) MDS of Word2Vec result



(b) MDS of BlueBERT result

Figure 3: Multidimensional scaling results

Table 3: Simplified version of model output given a query: what is the incubation period of COVID-19?

Sentence	**Similarity**
There is a 2 to 14-day incubation period for SARS-CoV-2.	0.917
"what is the incubation period of COVID/normal coronavirus/SARS-CoV-2/nCoV".	0.915
The incubation period for COVID-19 is considered to be 14 days.	0.907
The distribution of the cases having a longer incubation period.	0.898
Our study initially demonstrated asymptomatic transmission of COVID-19 in the incubation period, especially in the last three days of incubation period, by estimating the incubation period with the use of accurate exposure history of confirmed cases.	0.897
The incubation period of a Covid-19 case is the time between infectious exposure and symptom onset.	0.897
The people in the infection period are those who have undergone the incubation period.	0.894
The incubation period is the time from infection to the onset of the disease.	0.891
From our definition of incubation period, his incubation period will be 28 days, which is surprisingly high.	0.883

Table 4: Purity of Word2Vec and BlueBERT results

Model	Purity
Word2Vec	1.0
BlueBERT	0.815

5 Conclusion

In this report, we study a distributed implementation for question-answering task. The experiments show that in our case the Word2Vec model performs better than pre-trained BlueBERT model in finding the sentences which match to the given query. That is mainly because the Word2Vec model has been trained on our corpus while the BlueBERT model is not further tuned using our data. Meanwhile, the efficiency of our program greatly benefits from distributed computation.

For question-answering task, it is necessary to clean data in advance so that we can focus on the potential answers, which brings much progress in performance. Query processing also plays an important role in boosting accuracy. In our experiment, we found that a concise and specific question can efficiently improve the matching of question and answers.

For the use of Spark, if the data size is not very large, running program in multi-thread on local mode is always efficient than in cluster mode, since the latter case has to manage resources and interact with workers. When running BlueBERT model, the memory issues do not occur on local mode but on clusters, maybe because of the limited memory on worker nodes. Tuning the Spark on cluster mode is complicated. We need to choose suitable Spark architecture and configuration to fit our data and problem.

In the future, we would like to fine-tune the BlueBERT model using our

Table 5: Human rated score of Word2Vec and BlueBERT results

	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Average
Word2Vec	0.30	0.35	0.75	0.6	0.6	0.5	0.6	0.75	0.65	0.45	0.555
BlueBERT	0.15	0	0.5	0.05	0.3	0.25	0	0.05	0.3	0.05	0.165

corpus to improve its performance. Additionally, since BlueBERT model prediction involves large matrix multiplication, it could be more efficient to run BlueBERT model on multiple GPUs in parallel.

References

[1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[2] Kaggle. Covid-19 open research dataset challenge (cord-19), 2020.

[3] Scikit learn developers. K-means user guide, 2019.

[4] Scikit learn developers. Multi-dimensional scaling (mds) user guide, 2019.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)*, pages 58–65, 2019.

[7] Wikipedia. Multidimensional scaling, 2020.

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.