Question Answering Tool

Cheuk Hang Ng (a1821087)

University of Adelaide, a1821087@student.adelaide.edu.au

This project aimed to build a question answering tool which is capable to output the top 10 most relevant sentences from the articles in the dataset with the corresponding sources of the articles. The core of the tool is a natural language processing model using word embeddings approach. Data preprocessing and exploratory data analysis followed by text preprocessing were done before building the model and accepting a query. The tool successfully showed 10 most relevant sentences from the articles together with the sources. Continuous parameters tuning is needed to improve the model.

Computing methodologies • Computing methodologies • Artificial intelligence • Natural language processing • Lexical semantics

Additional Keywords and Phrases: Text classification, Word Embeddings, Word2Vec

1 INTRODUCTION

This project aimed to build a question answering tool which is capable to search and show for the top 10 most relevant sentences in a set of articles with the corresponding sources of the articles given a query as an input. The set of articles is a subset of the dataset provided in the COVID-19 Open Research Dataset Challenge (CORD-19) competition on Kaggle. This project started with data preprocessing and exploratory data analysis, followed by text preprocessing before building the word embeddings model. Word embeddings is a common approach to natural language processing for text representation. The algorithm chosen for word embeddings was Word2Vec in this project. Each article was decomposed into sentences and all sentences were fed into the Word2Vec model to undergo unsupervised learning. The tool asks for user query and search for 10 most relevant sentences and sources of the sentences by cosine similarity. The performance of the model was evaluated by human judgement.

2 METHODOLOGY

2.1 Data Preprocessing and Exploratory Data Analysis

The full dataset was provided at Kaggle COVID-19 Open Research Dataset Challenge (CORD-19) competition page. Only the folder named “pdf_json” containing the articles and “metadata.csv” were downloaded and used in this project. The article files were in json format. They were first read as a list and then transformed to pandas dataframe object. The “metadata” file was also read as a dataframe. The dataframes were then merged by matching the ‘paper_id’ column and the ‘sha’ column in the dataframe objects respectively. To extract the main content of each article, the dataframe was iterated to unpack all texts under the ‘body_text’ column. The ‘country’ attribute under the column ‘metadata’ was also extracted with the same method, rows that threw exceptions in this step were skipped. This project used the country of the first author in each article to represent the country of origin of the article.

Articles without title were dropped. In addition, null rows under 'abstract' and 'main_content' were replaced with empty string. Some columns that were not used in this project were then dropped to reduce the size of the dataframe object. The index of the dataframe was then reset. The number of publications according to location was plotted in bar chart, and the number of publications after 2003, grouped by month, was plotted as line graph. The year 2003 was chosen because there was a virus called the Severe Acute Respiratory Syndrome coronavirus which is very similar to the Severe Acute Respiratory Syndrome coronavirus 2 causing COVID-19, started a pandemic in 2003 [1].

2.1.1 Word cloud

A word cloud was generated after text normalization. A word cloud visualizes the frequencies of words in a set of text data. The more frequent the word is, the bigger the size it appears in a word cloud. `WordCloud()` from

Figure 1: A typical word cloud, via Wikimedia Commons. (https://t.ly/ZdBX)

WordCloud was called and words from all preprocessed sentences were concatenated into a single list to generate the word cloud.

2.2 Text Preprocessing

Text data are usually preprocessed before used in training natural language processing models. Text preprocessing includes text cleaning and text normalization. All abstracts and main contents were iterated and cleaned before normalization. Three lists were created to contain the processed sentences. One contained all processed sentences while the other two contained processed sentences from abstracts and main contents respectively.

2.2.1 Text Cleaning

Cleaning the text to be trained with natural language processing models could help improving the performance of the model. Text cleaning strategies chosen were lowercasing all text and removing punctuations except full stops, Unicode text, all numbers and double spacing in the text. The following function takes string as an input, process the string with the above strategies, and return the cleaned text.

```
def text_process1(text):

    #lowercase all characters

    text = text.lower()
```

```
#remove punctuation
text = re.sub(r'[%s]' % re.escape(punc), ' ', text)
#remove unicode text
text = re.sub(r'[^\x00-\x7F]+', ' ', text)
#remove the numbers
text = re.sub(r'[0-9]', '', text)
#remove double space
text = re.sub(r'\s{2,}', ' ', text)
return text
```

2.2.2 Text Normalization

Text normalization aims at simplifying modelling process and improving model's performance. Typical text normalization includes but not limited to the following processes: removing Stop-words, stemming, lemmatization and tokenization. Text normalization processes chosen were removing Stop-words, lemmatization and tokenization. The set of Stop-words used was the English Stop-words set downloaded from NLTK with the addition of some common words in research articles such as "et", "al", "may" and "also". Lemmatization is the process of reducing a word to its base form by analyzing the morphological information of the word. Tokenization is another text normalization process which decomposes a sentence into a list of words.

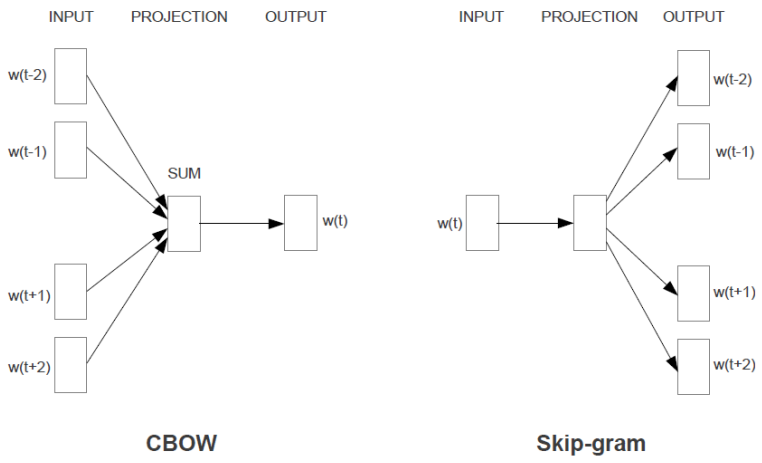

Figure 2: Two working principles of Word2Vec [3]

2.3 Word Embeddings

Word embeddings is a natural language processing technique where text from corpus are mapped as vectors. It is a type of learned representation where words with the same meaning will have the same representation

vector. Word embeddings has the advantage of representing word similarity in terms of context with relatively low dimensional vectors [2].

2.3.1 Word2Vec

Word2Vec is a word embeddings algorithm where the vector for each word is created by a shallow neural network with two hidden layers. The model is able to capture both semantic and syntactic information of words. The algorithm has two working principles, Continuous Bag-of-Words (CBOW) and Skip-Gram (SG). This project used the CBOW model. The CBOW model predicts the centre word from surrounding context. The model maximizes the probability of a word being in a particular context with the following form:

$$P(w_i | w_{i-c}, w_{i-c+1}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+c-1}, w_{i+c})$$

where w_i is a word at position i with c being the length of window. `Word2Vec()` from Genism was called to build the model, with vector size of 100, window length of 8 and not ignoring any words as the parameters, and the list that contained all tokenized sentences as the training data.

2.4 Similarity search

The tool prompts the user for a query. The query is then preprocessed by cleaning and normalization. The tool searches similar sentences in all articles by computing the cosine similarity between the vectorized preprocessed query and the vectors representation of all sentences. `cosine_similarity()` from Sklearn was called for computing similarity. 20 most similar sentences were returned and sorted in descending order by similarity. The reason of keeping 20 most similar sentences is to avoid duplicated sentences in abstract and main content from the same article. Only 10 distinct results will be output.

2.5 Evaluation

The user determines the accuracy and precision of the search results. The result from the tool is suitable to be judged by the user because it is a natural language processing problem where the user could easily determine whether the search results are answering their question accurately and precisely or not.

3 EXPERIMENTS AND RESULTS

3.1 Exploratory Data Analysis

The set of articles used in this study was obtained by extracting files from the folder described in section 2.1 at random. Number of files between 5,000 to 7,000 would be enough to build the word embeddings model and the number is suitable for completing this project on a personal computer in terms of computing power. A total of 5,928 articles in json format were extracted.

3.2 Data Preprocessing

The dataset was first preprocessed as described in section 2.1. After data preprocessing, 5,228 instances were left in the dataframe object which was still a suitable size of a dataset for this project. As outlined in section 2.1,

the number of articles against counties of origin and time were visualized. The word cloud generated in later step will also be shown below.

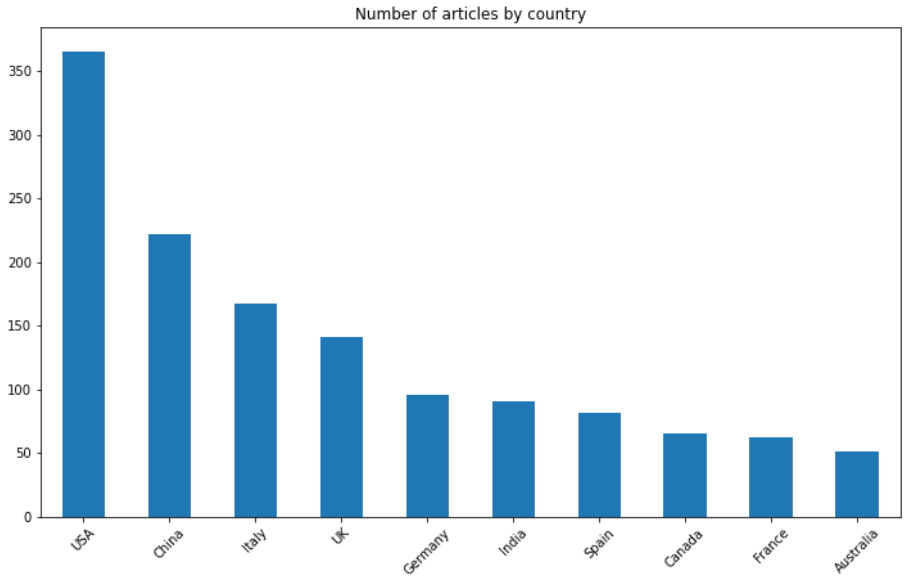

Figure 3: Number of articles from different countries

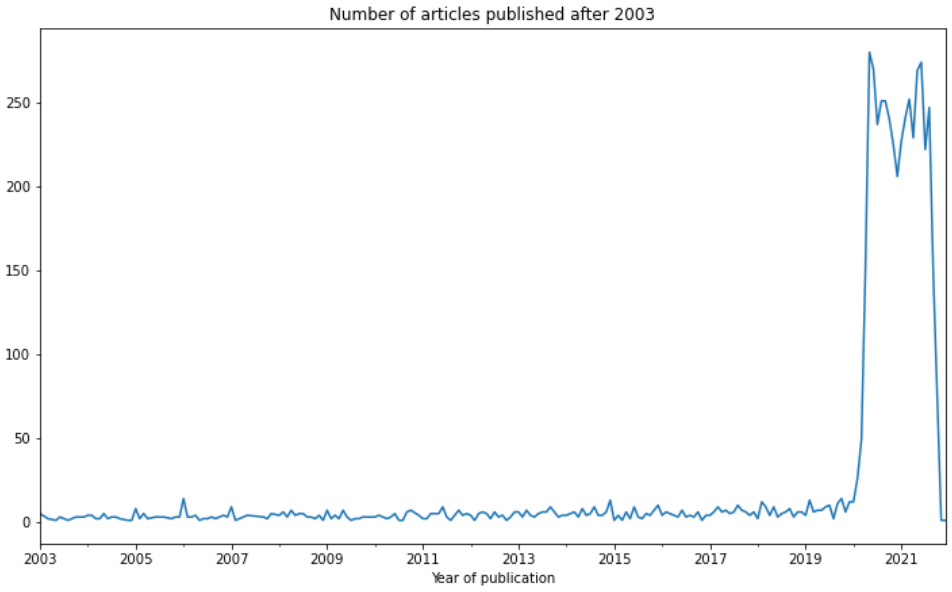

Figure 4: Number of articles published after 2003

Figure 5: Word cloud generated from the dataset obtained

3.3 The models

A Word2Vec model with Continuous Bag-of-Word training algorithm was built with the list containing all tokenized sentences from the abstract and main content in all articles. A total of 1,043,560 tokenized sentences were provided to train the model. Example of tokenized sentences are shown below.

```
[['laboratory', 'confirmed', 'case', 'respiratory', 'viral', 'infection', 'among', 'hospitalized', 'patient'], ['categorized', 'hospital', 'acquired', 'infection'], ['overall', 'incidence', 'hospital', 'acquired', 'respiratory', 'viral', 'infection'], ...]
```

3.4 Similarity Search

“How long does it take to cure COVID” as the input query was used to test the question answering tool. The cosine similarities between the vectorized query and vector representation of all sentences in the dataset were then computed and the top 20 most similar sentences were selected and sorted in descending order by the corresponding similarity. Top 3 most similar tokenized sentences of the sample query are shown below.

```
['known', 'long', 'take', 'covid', 'patient', 'recover', 'infection']
['sick', 'leave', 'long', 'covid']
['new', 'zealand', 'recovery', 'covid', 'long', 'difficult']
```

3.5 The Result

The question answering tool successfully showed the top 10 most relevant article snippets containing the most similar sentences as the output. The top 3 most relevant article snippets of the sample query are show below as an example of the output. The search result was satisfying.

Showing top 10 relevant articles

Assessment and Management of Diabetic Patients During the COVID-19 Pandemic3532

... in COVID-19 pathogenesis. [36] [37] [38] Therefore, disease intensity in COVID-19 patients is due both to the virus and the response of the host. 39 It is not known how long it takes for a COVID-19 patient to recover from infection. The infectivity period is estimated by detecting the virus or its ssRNA in respiratory tract samples. Though, presence of ssRNA of virus does not approve the contagious virus occurrence. Reports suggest that transmission of the virus may occur in the early stage of COVID-19 infection, as high virus loads are found in respiratory samples shortly after the onset ...

Original document at: https://www.ncbi.nlm.nih.gov/pubmed/34262317/; https://doi.org/10.2147/dmso.s285614

Patterns and predictors of sick leave after Covid-19 and long Covid in a national Swedish cohort1245

... to Covid-19. RESULTS: A total of 11,955 people started sick leave for Covid-19 within the inclusion period. The median sick leave was 35 days, 13.3% were on sick leave for long Covid, and 9.0% remained on sick leave for the whole follow-up period. There were 2960 people who received inpatient care due to Covid-19, which was the strongest predictor of longer sick leave. Sick leave the year prior to Covid-19 and older age also predicted longer sick leave. No clear pattern of socioeconomic factors was noted. CONCLUSIONS: A substantial number of people are on sick leave due to Covid-19. Sick leave ...

Original document at: https://www.ncbi.nlm.nih.gov/pubmed/34059034/; https://doi.org/10.1186/s12889-021-11013-2

On New Zealand's weak, strong and muddled management of a COVID-19 epidemic3327

... mental health problems, using e-health, virtual health teams and community health workers who are trained for this purpose and micro-credentialed.New Zealand's recovery from COVID-19 will be long and difficult. Best practice, both invented and borrowed, is essential. An explicit pandemic plan, which recognises the country's contextual strengths and weaknesses, and a national public health agency are essential ...

Original document at: https://doi.org/10.1111/imj.14928; https://www.ncbi.nlm.nih.gov/pubmed/32881265/

4 DISCUSSION AND CONCLUSIONS

This project was conducted on a personal notebook with Jupyter Notebook Python environment with a running time of around 45 minutes. With restricted storage and computing power, only around 5,000 articles were used to build the model. This project was my first text classification problem with natural language processing algorithms and there are room for improvements in terms of model performance and coding though the tool built accomplished the task satisfyingly.

First of all, a data selection could be performed before data preprocessing. For instance, keep extracting files until the designated numbers of articles with non-empty title, abstract, main content and country attributes were obtained. However, this involves a lot of computing power because it have to extract and unpack all json files prior to storing the file in the list. The data preprocessing strategies could also be improved. For example, the country of origin of the article could be represented by the mode of countries of all authors.

The text preprocessing steps were performed as expected. The noises were removed successfully, and the sentences were tokenized smoothly. One thing to mention is that the list containing all sentences was big for a personal computer. Every time I ran the notebook, large portion of running time was spent on iterating through the list.

For the word embeddings model, it was hard to evaluate how well was the model performing. Unlike regression model or other deep learning model, there was not any built-in function to quantify the performance of the word embeddings model like F-score, accuracy and absolute error in those models. Therefore, I could only test the model by running the Notebook from the beginning for multiple times to test for the quality of the outputs. Although this repeated process took time, it gave me insights on handling empty tokenized sentences or some unexpected output. In addition, the output was also optimized so that the tool presents the result in a more readable layout.

The tool could be improved by continuous parameters tuning. Parameters such as vector size and window length are worth trying. By tuning these parameters, the model may perform differently. Another possible way to improve the results is to preprocess the text with different approaches. For instance, stemming, spelling correction, part of speech tagging and negations handling could be introduced to the text preprocessing step.

REFERENCES

[1] El Zowalaty, M. E., & Järhult, J. D. 2020. From SARS to COVID-19: A previously unknown SARS- related coronavirus (SARS-CoV-2) of pandemic potential infecting humans – Call for a One Health approach. *One Health*, 9, 100124–100124. DOI: https://doi.org/10.1016/j.onehlt.2020.100124

[2] Naseem, U., Razzak, I., Khan, S. K., & Prasad, M. 2021. A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models. *ACM Transactions on Asian and Low-Resource Language Information Processing, 20(5), 1–35*. DOI: https://doi.org/10.1145/3434237

[3] Mikolov, T., Chen, K., Corrado, G., & Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv: 1301.3781. Retrieved from https://arxiv.org/abs/1301.3781