IS450 Text Mining and Language Processing

Project Final Report

Prepared for:

Dr. Wang Zhaoxia

Prepared by:

G3T5

Ang Heng Di
Chen Yi Bing
Kylie Tan Jia Yun
Lim Chee Xiong
Teo Kai Lun Felicia

13 April 2022

Table of Contents

Abstract

Being one of the most severe public health emergencies, COVID-19 has been a popular topic all around the world, generating immense amounts of unorganised data. Coupled with a rise in fake news, it has the potential to result in people being misinformed. In order to help the general public get relevant and accurate information about COVID-19 with ease, using text mining techniques we attempt to create a bot to bring convenience to the general public. We chose to use topic modelling (Task 1) and question answering (Task 2) techniques to help us with this project.

For task 1, we identified latent topics talked about on Twitter during the start of the pandemic (year 2020) and compared it to our current situation (year 2021-2022). Topics were being output from two implementations of topic modelling - Latent Dirichlet Allocation (LDA)/LDA Mallet Model and Gibbs Sampling Dirichlet Multinomial Mixture (GSDMM). This analysis was done on close to 400,000 tweets that were scraped from Twitter from both time periods and then the keywords generated would be used to assess the interpretability of latent topics generated.

For task 2, we combined a deep learning model as well as COVIDASK - a question answering module - and integrated it into a bot using Telegram API to create a user-friendly interface. The COVIDASK module uses Dense Sparse Phrase Index (DenSPI) as the base model for query encoding, document retrieval and information retrieval. We utilise the COVID-QA dataset of 2,019 COVID-19 related Question and Answer pairs to train our model with Roberta-base-squad2 as the base model for our deep learning model. The model will process the article returned by the COVIDASK module that best matches the user input query and return either a word or a phrase that best answers the query. Both the COVIDASK module and Roberta-base-squad2 model will be imported into the Telegram bot python program where the answer will be returned to the user via the Telegram API.

1. Introduction

COVID-19 is by far one of the most severe public health emergencies in the world. Being the talk of the town and around for more than 2 years, there is an immense amount of unsorted data out there which lacks organisation. Making situations worse, according to the United Nations, almost every country has seen an increase in fake news in the context of COVID-19 [1]. This could result in people being misinformed, leading to undesirable consequences [2]. Although there has been research on how to organise COVID-19 research data for researchers and scientists, there have not been studies that aid in sorting general information about COVID-19 from various sources. Hence, we aim to provide information they need, from what to do if tested positive to the history of COVID-19; A system using NLP techniques that allows the general public to find any information they want to know about COVID-19.

To create this system, we broke this research down into two research questions:

1. What trends/topics can be inferred from English tweets from the start of pandemic (2020) to current situation (2021-2022)?

   This task would allow us to understand what the general public usually discusses about and what they might potentially want to know.

2. How can we use existing QA systems to aid the general public to find COVID-19 information that is accurate, relevant, and easy to understand?

   This task allows us to create a system which users can use to get accurate information they need about COVID-19.

2. Solution Details & Overview

2.1 Task 1: Topic Modelling over COVID-19 related Tweets

2.1.1 Literature Review

For successful topic modelling using LDA, we referred to details outlined in a research paper titled “Applying LDA topic modelling in communication research: Toward a valid and reliable methodology” [3]. This paper outlined various crucial stages that affected the reliability and precision of LDA topic modelling - pre-processing of corpus, selection of model parameters, model evaluation and valid interpretation of topics generated. This was achieved through the development of a practical guide on the application of LDA topic modelling.

The referenced paper placed heavy emphasis on the importance of interpretability in that it ought to be the main criterion in the selection of models or topics generated. This could be

achieved through various methods such as deriving the optimal number of topics and displaying words that are hardly used in each topic. In addition, the paper also highlights the need to exclude topics with top keywords that are inscrutable or too general and to only include topics with words deemed meaningful by domain experts.

Another research paper titled "A Query-Driven Topic Model" outlines a model where short and simple inputs by users can return related topics [4]. Through this model, it allows for insights into common specific topics that users may have in regards to an event or matter. Similarly, we could look into the application of this outlined model for COVID-19 related topics.

We found another paper titled "Understanding Weekly COVID-19 Concerns through Dynamic Content-Specific LDA Topic Modeling" [5] that was highly similar to our project. This paper follows the utilisation of LDA topic modelling to pinpoint certain COVID-19 related issues that social media users have voiced concerns on. This mirrors our project where we look to utilise Twitter data to observe changes in viewpoints or concerns in regards to COVID-19 overtime. Hence, we used this as a reference for our own project.

2.1.2 Dataset

The dataset used in this project for Task 1 was gathered from Twitter using Snscrape API using the keyword 'covid', filtering it to only include English tweets. Data was collected for a time period of six months for both the start of the pandemic and the current situation, with a limit of 35,000 tweets per month as shown in Figure 1. The total number of tweets is 385,146 which would then be preprocessed before the start of the experiment.

	Period	**Total Tweets**
Early Pandemic	Jan 2020 - Jun 2020	210,000
Current Situation	Aug 2020 - Jan 2022	175,146

Figure 1 . Time period and total tweets from the dataset

Preprocessing

The tweets collected from Twitter are vastly different from standard English and hence may be of low quality to train our model with. There are many elements that do not appear in standard English such as emojis, hashtags, URLs etc. In order to ensure that our dataset and features (tokens) would result in a better model, we will need to preprocess our dataset.

The table below, Figure 2, illustrates the steps taken to preprocess the dataset and the objective for each step

Removal of emoji	While emoji does convey a certain sentiment, we are not concerned with the sentiment of the tweet. Also, emoji cannot be read by the machine hence we have removed them.
Convert all text to lowercase	To reduce the likelihood of obtaining a sparse matrix when vectorising our text, converting all text to lowercase would help reduce our dimensions for a better model training process.
Expand contractions	Tweets contain contractions (eg “I’m”) and their counterparts (eg “I am”). Hence, to also reduce the dimensionality of our corpus, we have decided to expand contractions.
Remove URL, hashtags, mentions (@user)	Tweets contain certain features that do not add value to the model training process hence we decided to remove them.
Remove non-alphanumeric characters	While some numbers or special characters do play a role in topic modelling (eg. COVID-19 etc), we deemed most of them to not play a significant role in our task and decided to remove all non-alphanumeric characters.
Remove punctuations	While punctuations does help in sentiment analysis, it does not offer much value in topic modelling hence we have decided to remove them.
Remove stopwords	Not only does English stopwords not offer any value in discerning the difference in topics, they may also be noise in our topic modelling task since the high frequency in most tweets would ‘drown’ away any keywords that may be of value.
Lemmatisation	The need to normalise words is also important to reduce dimensionality through grouping root words in their many variations (plural form, various verbs and adjectives). We chose to use lemmatisation instead of stemming due to it having better performance [6] results through producing real dictionary words and having

	analysis that depends on a word's Parts-Of-Speech.

Figure 2 . Sequence and objective of each preprocessing step

2.1.3 Methodology

After the dataset has been preprocessed and ready to be trained on to produce a topic detection model, multiple techniques would be applied to it in order to pass it into the machine for model training. The general flow of the techniques are shown below in Figure 3.

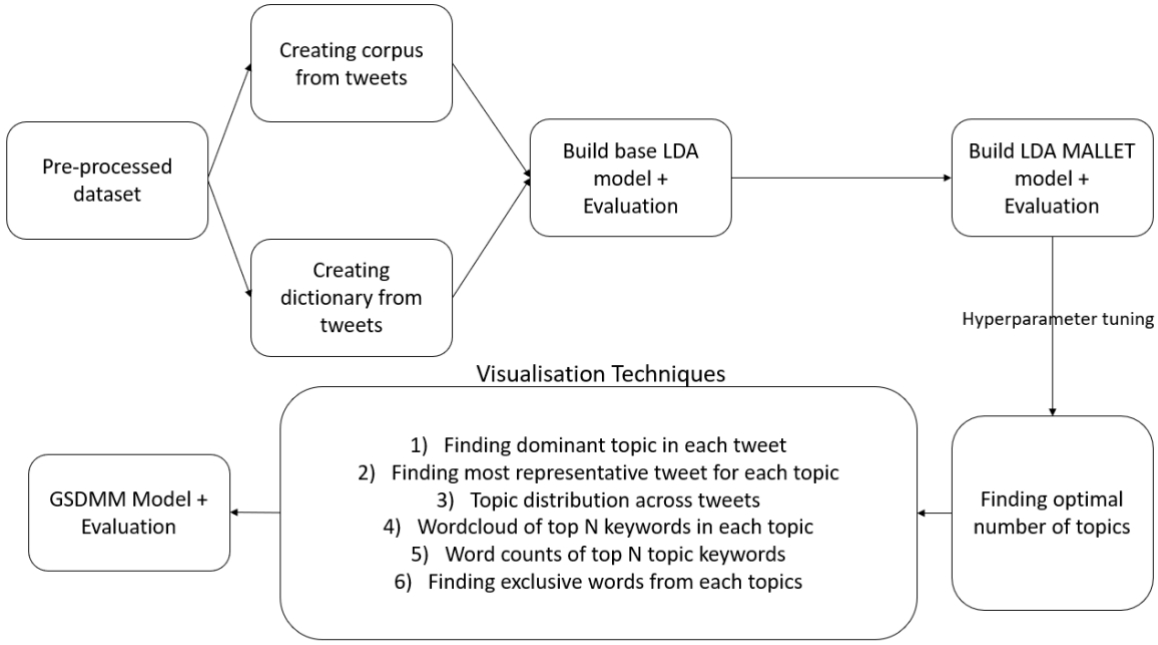


Figure 3 . Overview of methodology for Topic Modelling

The output of base LDA model, LDA MALLET model and GSDMM model is a list of topics which itself has a list of keywords and a score/weightage assigned to each keyword denoting how 'important' it is to the topic.

2.1.4 Experiment Setup

Creating Corpus and Dictionary from the dataset

The two main inputs for a LDA topic model is a dictionary of unique words and a corpus made up of all the documents. To create a corpus is to simply combine all the preprocessed tweets from our dataset.

To create the dictionary for our model, we also considered including ngrams such as bigram and trigram. This is because some words that occur frequently together may belong to a specific topic as opposed to when they are split apart. Take for example the phrase 'Machine Learning'; 'Machine' alone could be put into construction or factory topics while 'Learning' could be represented as educational topics. But when we put 'machine' and 'learning' together, they would fall under a data science topic.

As such, we added bigrams into our dictionary that have a minimum count of 5 in the entire corpus as shown in Figure 4.

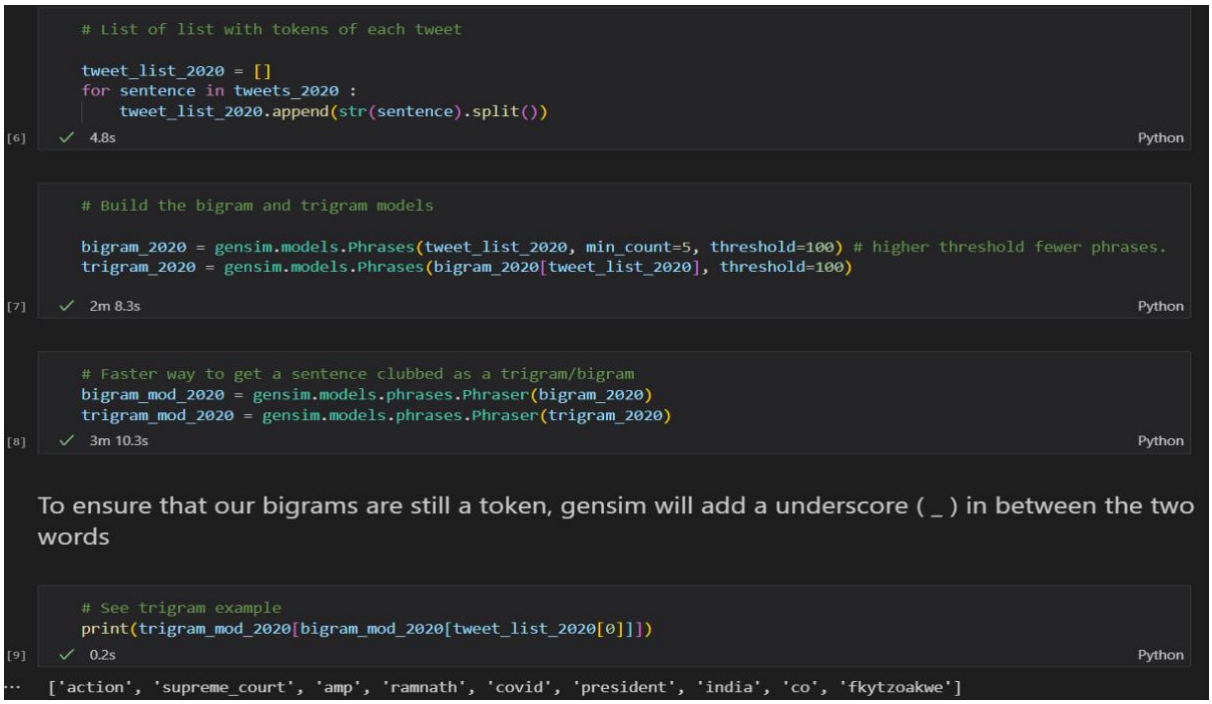


Figure 4 . Creating bigram/trigram model for recognition within the corpus

We have decided to not include trigram as it may cause our dictionary to have too many dimensions.

Since machines cannot comprehend text, we need to turn our corpus into numbers or matrices in order to train the model. We have decided to construct a term frequency matrix for every tweet as shown in Figure 5 .

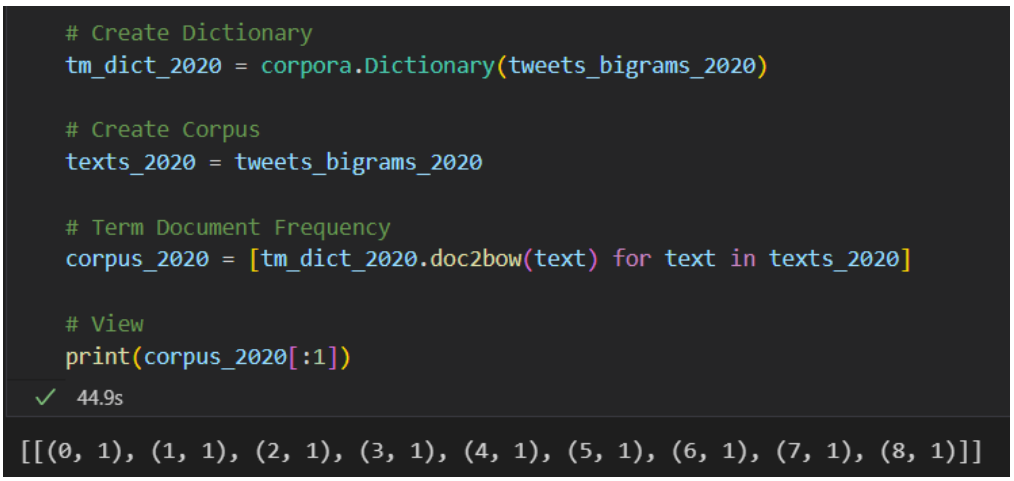


Figure 5. Vectorising tweets in (word_id, frequency) form

Each tweet would be vectorised into a list of tuples in the form of (word_id, frequency). So a tuple of (0,1) would be read as word_id 0 appeared 1 time in a given document/tweet.

Building our Base LDA Topic Model

Now that we have our two main inputs, corpus and dictionary, we can create our LDA topic model. We will use a randomly chosen topic number of 5 for our model for both years (2020 and 2021-2022) and save the model as shown in Figure 6.

```
# Build LDA model
lda_model_2020 = gensim.models.ldamodel.LdaModel(corpus=corpus_2020,
                                                 id2word=tm_dict_2020,
                                                 num_topics=5,
                                                 random_state=99,
                                                 update_every=1,
                                                 chunksize=100,
                                                 passes=1,
                                                 alpha='auto',
                                                 per_word_topics=True)


#saving model
# lda_model_2020.save('lda.model.2020')
```

Figure 6 . Base LDA Topic Model parameters

The output of the LDA model would be a list of topics where a topic is a combination of keywords with varying weights associated with each keyword.

The weightage of each keyword is correlated to the importance of those keywords to the topic generated.

Our main metric to evaluate the quality of the topic model is mainly Coherence Score (C_V). As shown in Figure 7 and Figure 8, the coherence scores of both years 2020 and 2021-2022 are 0.383 and 0.365 which is not very good.

```
# Compute Coherence Score
coherence_model_lda_2020 = CoherenceModel(model=lda_2020, texts=tweets_bigrams_2020,
coherence_lda_2020 = coherence_model_lda_2020.get_coherence()
print('\nCoherence Score: ', coherence_lda_2020)


Perplexity:  -8.168037789612342

Coherence Score:  0.38310763127256486
```

Figure 7 . Coherence score of year 2020

```
# Compute Coherence Score
coherence_model_lda_2022 = CoherenceModel(model=lda_2022, texts=tweets_bigrams_2022,
coherence_lda_2022 = coherence_model_lda_2022.get_coherence()
print('\nCoherence Score: ', coherence_lda_2022)


Perplexity:  -8.056932156397028

Coherence Score:  0.36558240692998956
```

Figure 8 . Coherence score of year 2021-2022

Visualisation of topics

To better understand how each topic is distinct from one another, we employed a library called pyLDAvis to create an interactive chart. Each topic is a bubble and its size is relative to how many documents it contains. Overlapping bubbles means that the topics are somewhat related to each other as shown in Figure 9 for both years.

Figure 9 . pyLDAvis for both years 2020 and 2021-2022

Building LDA MALLET Model

MALLET is a Java-based package that often gives better quality of topics than the default LDA model. As shown in Figure 10, the coherence score for both year 2020 and 2021-2022 improved.

```
# Show Topics
pprint(ldamallet_2020.show_topics(formatted=False))

# Compute Coherence Score
coherence_model_ldamallet_2020 = CoherenceModel(model=ldamallet_2020, texts=tweets_bigrams_2020,
coherence_ldamallet_2020 = coherence_model_ldamallet_2020.get_coherence()
print('\nCoherence Score: ', coherence_ldamallet_2020)

Coherence Score:  0.40871950915891375
```

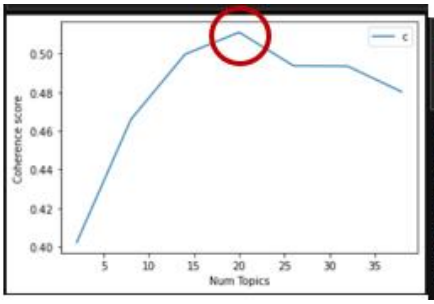
```
# Show Topics
pprint(ldamallet_2022.show_topics(formatted=False))

# Compute Coherence Score
coherence_model_ldamallet_2022 = CoherenceModel(model=ldamallet_2022, texts=tweets_bigrams_2022,
coherence_ldamallet_2022 = coherence_model_ldamallet_2022.get_coherence()
print('\nCoherence Score: ', coherence_ldamallet_2022)

Coherence Score:  0.43044402075775345
```

Figure 10 . Improvement in coherence score using MALLET Model

Finding optimal number of topics

To find the optimal number of topics, we will need to iteratively build LDA models and plot their coherence score on a graph. We will be using the MALLET MODEL from here onwards.

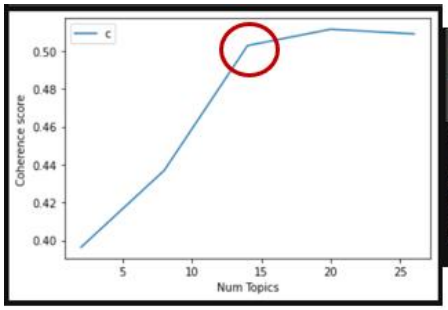
There are two ways to find the optimal number of topics : Find the highest coherence score when plotted on a graph OR find the point that marks the end of a RAPID increase in coherence score on the graph. Figure 11 and Figure 12 shows the optimal number of topics that we will use for both 2020 and 2021-2022 respectively.




```
# Print the coherence scores
for m, cv in zip(x, coherence_values_2020_mallet):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.4023
Num Topics = 8  has Coherence Value of 0.466
Num Topics = 14  has Coherence Value of 0.4998
Num Topics = 20  has Coherence Value of 0.5111
Num Topics = 26  has Coherence Value of 0.4939
Num Topics = 32  has Coherence Value of 0.4936
Num Topics = 38  has Coherence Value of 0.4803
```

Figure 11. Coherence score plot for year 2020




```
# Print the coherence scores
for m, cv in zip(x, coherence_values_2022_mallet):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.3963
Num Topics = 8  has Coherence Value of 0.4369
Num Topics = 14  has Coherence Value of 0.5029
Num Topics = 20  has Coherence Value of 0.5116
Num Topics = 26  has Coherence Value of 0.5091
```

Figure 12. Coherence score plot for year 2021-2022

From both graphs, the optimal number of topics for year 2020 and 2021-2022 is 20 and 14 respectively.

Improving Interpretability

We tried several ways to help improve interpretability of the topics generated. Firstly, we tried to find the dominant topic in each document. To do this, we found the topic number that contributed the highest percentage of keywords to the given document.

Secondly, we went to find the most representative document for each topic. We felt that it would be better if we knew which document contributed the most to a given topic so we can better understand what the topic is about.

Lastly, since LDA assumes that a document is a distribution of topics, we went to find the topic distribution across each tweet. This is to help us understand the volume and distribution of topics to judge how widely it is being discussed.

2.1.5 Results & Discussion

WordCloud of the top N keywords in each topic

Using a word cloud is an effective way to help interpret the result obtained from our topic model where the size of the words is proportional to the weight. Figure 13 shows the word cloud for the year 2020 for the first four topics. Figure 14 shows what we can infer from each topic generated.

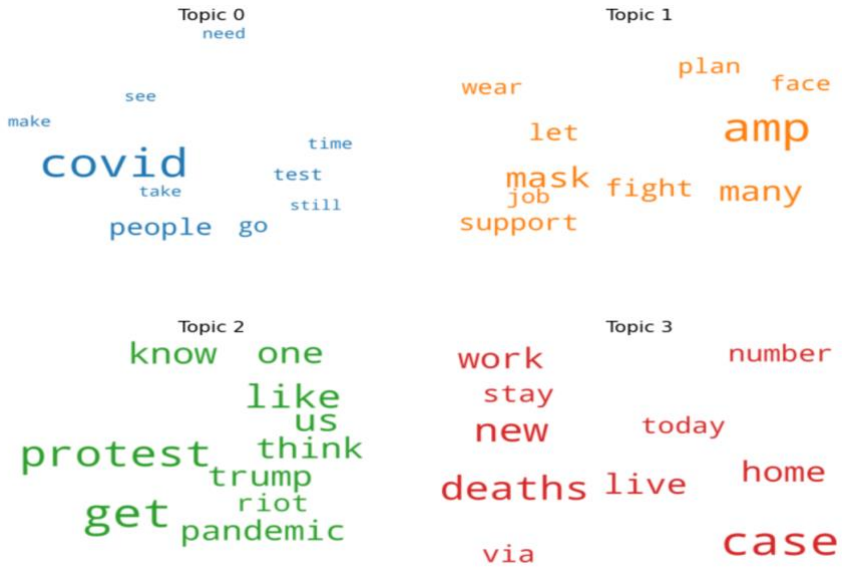


Figure 13. WordCloud for the first 4 topics of year 2020

	Possible topics
Topic 0	Covid tests may take time (ambiguous)
Topic 1	Job support plan and wearing of face masks
Topic 2	Protesting/Rioting in regard to trump amidst the pandemic
Topic 3	Total cases and deaths, staying at home to work

Figure 14. Inference of topic from year 2020

Whereas Figure 15 shows the word cloud for the year 2021 to 2022 for the first four topics and Figure 16 shows the inference for each topic.

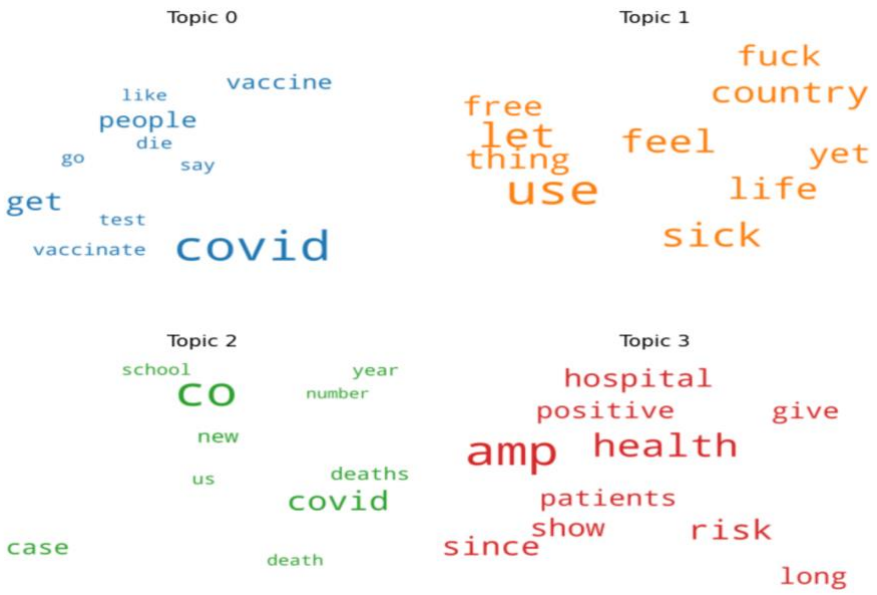


Figure 15. WordCloud for the first 4 topics of year 2021-2022

	Possible topics
Topic 0	Covid vaccines and people dying
Topic 1	People not feeling free in their country and they are sick of it
Topic 2	Covid deaths and cases in schools
Topic 3	Positive patients show risks for a long time

Figure 16. Inference for each of the first 4 topics of year 2021-2022

To better understand the difference in topics generated from the LDA MALLET model, we went to find the most representative document from the topic along with the word cloud of the topic to find out if there is any correlation. Figure 17 and Figure 18 shows the comparison between the word cloud and the most representative document for the given topic for year 2020 and year 2021-2022 respectively.

Figure 17. Comparison for the year 2020

Figure 18. Comparison for the year 2021-2022

From the two figures above, we can see that the most representative document does not show much relation with the word cloud of their topic. This shows that there is much more to be done to improve our model and our data preprocessing steps.

Word counts of topic keywords to their weightage

Even if there are words with a high weightage in any given topic, it may not be due to them being of high importance to the topic itself but rather due to it being commonly found in the global corpus.

We will consider a word as crucial to the topic IF it has both a relatively low number of word count and high weight in each topic. The figure below, Figure 19, shows the word count and weightage for the year 2020 while Figure 20 shows the word count and weightage for the year 2021-2022.

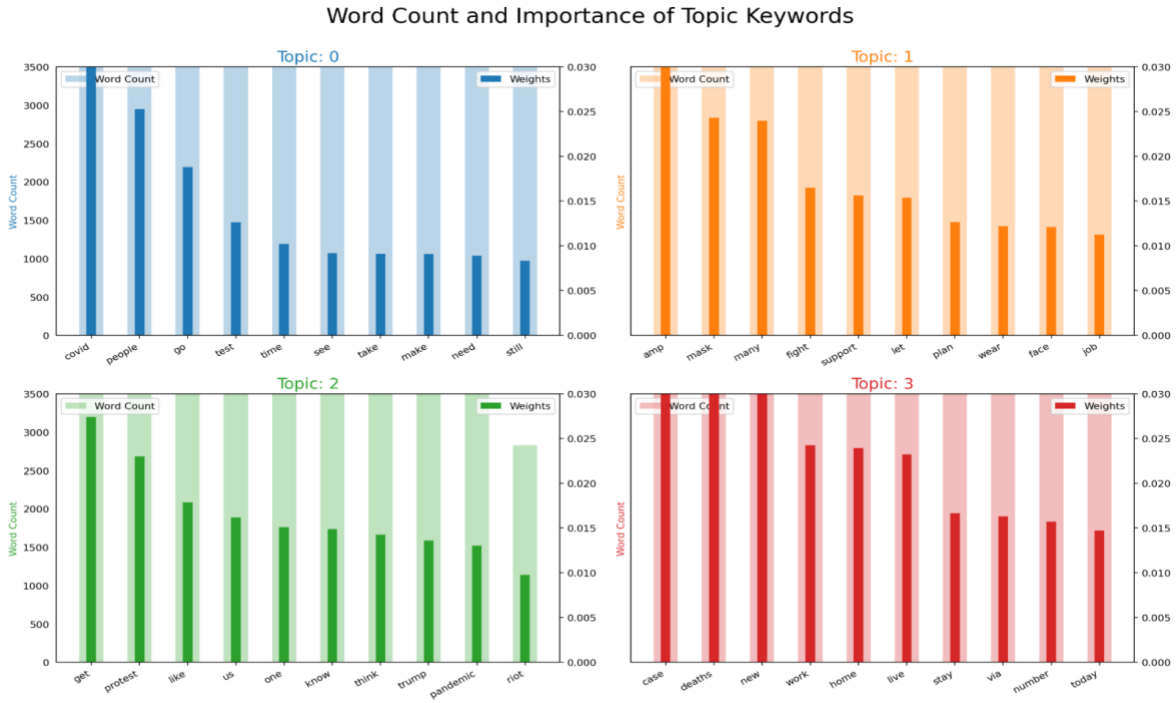


Figure 19. Word count and weightage for the first four topics of year 2020

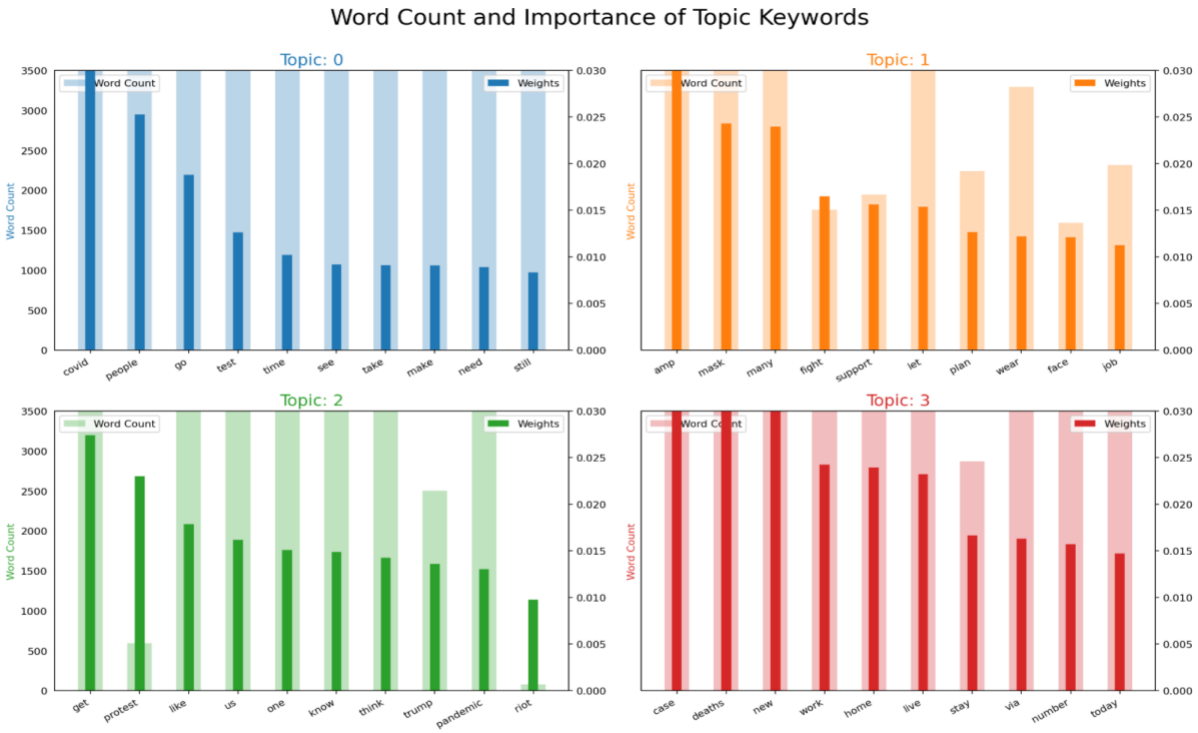


Figure 20. Word count and weightage for the first four topics of year 2021-2022

Finding exclusive words from each topic by tuning relevancy score

As mentioned above, a flaw with looking at the words with the highest weightage is that it may be considered a keyword due to it being a common word in the global corpus.

Another alternative to better understand topics is to show exclusive words from each topic which could give us more insight into what the topics are talking about and how they differ from one another.

By tuning the relevancy score, it helps to prioritise terms that are more exclusive to a given topic. For our experiment, we tuned the lambda value to 0.6. Figure 21 and 22 shows the first five topics with their exclusive words from the year 2020 and year 2021-2022 respectively.

	0	1	2	3	4	5	6	7	8	9
Topic 1	trump	hoax	call	president	americans	lie	blame	claim	handle	political
Topic 2	people	flu	rate	infect	kill	year	mortality	data	million	deaths
Topic 3	impact	support	pay	business	fund	crisis	money	relief	businesses	company
Topic 4	health	care	public	workers	medical	staff	healthcare	fight	national	safety
Topic 5	amp	watch	question	talk	join	video	listen	answer	discuss	experts

Figure 21. Exclusive words from topics of year 2020

	0	1	2	3	4	5	6	7	8	9
Topic 1	amp	lie	biden	trump	americans	vote	blame	media	control	leave
Topic 2	health	government	care	issue	mandate	public	pandemic	plan	support	rule
Topic 3	vaccine	vaccines	virus	study	prevent	treat	infection	treatment	ivermectin	dose
Topic 4	risk	vax	children	vaxxed	science	age	anti	chance	read	covid
Topic 5	case	deaths	school	report	number	omicron	variant	patients	hospitals	update

Figure 22. Exclusive words from topics of year 2021-2022

Interpretation of Results

We will mainly be using coherence score (C_V) to measure how interpretable the topics are. C_V score ranges from 0 - 1 and is a measure of how similar the keywords within a topic are to each other. Generally, a coherence score within the range of 0.55 - 0.70 is considered ideal. Figure 23 shows the change to coherence score throughout our experiment with a change in model and number of topics used.

	Year 2020	**Year 2021-2022**
Base LDA Model	**0.383**	**0.365**
LDA MALLET Model	**0.408**	**0.430**
LDA MALLET with ideal number of topics	**0.511**	**0.503**

Figure 23. Table of coherence score

Through the use of the MALLET Model and optimising the number of topics, we were able to reach a C_V coherence score of above 0.5 which may not be ideal, but it shows that the model is somewhat reliable in producing quality topics.

Gibbs Sampling Dirichlet Multinomial Mixture (GSDMM)

A limitation of LDA in the context of our task is that LDA performs better with long documents as opposed to short documents like tweets. Additionally, LDA assumes that in a given document, there is a mixture of topics. However, for a short text like tweets, it is more accurate to say that a single document or tweet would only talk about one topic. Hence, we decided to try out the Gibbs Sampling Dirichlet Multinomial Mixture (GSDMM) algorithm as it assumes that every document has only ONE topic assigned to it and it fares better with shorter documents like tweets. The exact algorithm is dubbed the MovieGroup Process and the research paper was published in 2014 [7] which proposed a DMM model based on Gibbs sampling.

The figure below shows the iterative process (10 iterations) of the GSDMM algorithm where it shuffles documents and clusters according to the MovieGroup algorithm with the alpha being 0.01 and beta being 0.01.

```
from gsdmm import MovieGroupProcess

mgp_2020 = MovieGroupProcess(K=20, alpha=0.01, beta=0.01, n_iters=10)

vocab = set(x for tweets in tweets_bigrams_2020 for x in tweets)
n_terms = len(vocab)
gsdmm_2020 = mgp_2020.fit(tweets_bigrams_2020, len(tm_dict_2020))
```

```
In stage 0: transferred 453259 clusters with 20 clusters populated
In stage 1: transferred 285797 clusters with 20 clusters populated
In stage 2: transferred 180033 clusters with 20 clusters populated
In stage 3: transferred 130032 clusters with 20 clusters populated
In stage 4: transferred 108577 clusters with 20 clusters populated
In stage 5: transferred 96836 clusters with 20 clusters populated
In stage 6: transferred 89953 clusters with 20 clusters populated
In stage 7: transferred 85900 clusters with 20 clusters populated
In stage 8: transferred 82909 clusters with 20 clusters populated
In stage 9: transferred 80592 clusters with 20 clusters populated
```

Figure 24. Iterative process of GSDMM

The output of GSDMM is somewhat similar to LDA in that it outputs a list of topics with their corresponding keywords. However, the keywords have a count frequency attached to them instead of a weightage and, like LDA, the higher the count the more important it is to the topic. The clusters are also ranked according to the number of documents inside them.

Figure 25 and Figure 26 shows the top 2 topics of the clusters output for year 2020 and year 2021-2022 respectively.

```
Number of documents per topic : [15159 60047 19416 22250 22370 32991 51919 18081 22608 17584 20189 46455
 18926 30807 13109 14890 22973 40853 17928 16472]
Most important clusters (by number of docs inside): [ 1  6 11 17  5 13 16  8  4  3 10  2 12  7 18]

Cluster 1 : [('covid', 61610), ('people', 13093), ('get', 12180), ('co', 11081), ('go', 10799), ('like', 6283), ('protest', 5296),
('know', 5156), ('say', 5070), ('think', 4853), ('amp', 4542), ('us', 4380), ('take', 4333), ('die', 4284), ('need', 4145), ('home',
4143), ('make', 4087), ('one', 4033), ('see', 4029), ('work', 3774)]

Cluster 6 : [('covid', 54498), ('co', 52207), ('case', 28603), ('coronavirus', 17317), ('state', 11557), ('new', 11295), ('confirm',
10599), ('first', 9410), ('report', 8762), ('health', 7659), ('death', 7311), ('test', 7236), ('deaths', 6074), ('say', 5433),
('update', 5423), ('county', 5357), ('washington', 5171), ('total', 4882), ('officials', 4736), ('positive', 4728)]
```

Figure 25. GSDMM output for the year 2020

```
Number of documents per topic : [ 5969  7472  3715 15163  4916 11457 29154 10222 20668 30035 10209 13187
  8444  4533]
Most important clusters (by number of docs inside): [ 9  6  8  3 11  5  7 10 12  1  0  4 13  2]

Cluster 9 : [('covid', 34021), ('get', 13983), ('people', 8556), ('vaccinate', 5826), ('vaccine', 4774), ('die', 4299), ('know',
3912), ('mask', 3798), ('still', 3517), ('co', 3333), ('say', 3329), ('go', 3274), ('like', 3263), ('spread', 3006), ('think', 2587),
('work', 2431), ('take', 2427), ('would', 2342), ('one', 2291), ('make', 2264)]

Cluster 6 : [('covid', 30929), ('co', 7575), ('people', 4684), ('get', 3959), ('amp', 3167), ('go', 3049), ('like', 2802), ('say',
2664), ('die', 2205), ('trump', 2155), ('think', 2141), ('us', 2096), ('make', 2001), ('would', 1964), ('biden', 1904), ('know',
1727), ('state', 1710), ('one', 1669), ('take', 1666), ('lie', 1589)]
```

Figure 26. GSDMM output for the year 2021-2022

2.2 Task 2: QA Telegram Bot

2.2.1 Literature Review

There are various COVID-19 related QA Systems, and each system uses a different methodology. The journal paper from IOP titled "Question Answering Systems for Covid-19" surveyed various COVID related QA systems by giving introductions to the various systems and comparing them in terms of performance (on significant parameters like Precision, Recall etc) with regards to answering COVID related questions raised by doctors and medical researchers [8]. Some systems mentioned by the articles include CAiRE (Center for Artificial Intelligence Research)-COVID system, CovidQA and COVIDASK.

COVIDASK is one of the QA systems that performed better in the aforementioned research paper and we proceed to study more about the system. The paper titled "Answering Questions on COVID-19 in Real Time" done by the creator of COVIDASK QA system explained in detail about why the system was created, how the system works and the evaluation of the performance of the system [9]. COVIDASK combines biomedical text mining and QA techniques to provide answers to questions in real-time using Information Retrieval Methods.

Besides looking at the QA systems, we also looked at the different methodology available. The research paper titled "EfficientQA: a RoBERTa Based Phrase-Indexed Question-Answering System" discussed about the possibility to transfer the natural language models into dense vectors representing questions and answer candidates to make the task of question-answering compatible with a simple nearest neighbour search task using BERT-based models [10]. This method will greatly improve the performance of the systems as answers can now be retrieved in a faster and more accurate manner.

2.2.2 Dataset

The datasets that are involved in this task includes firstly a pre-processed 37k COVID-19 Open Research Dataset (CORD-19) abstracts that can be found on Semantic Scholar site [11]. The dataset is being used to train the COVIDASK's DenSPI model.


```
"qas":[
  {
    "question":"What is the main cause of HIV-1 infection in children?",
    "id":262,
    "answers":[
      {
        "text":"Mother-to-child transmission (MTCT) is the main cause of HIV-1
        "answer_start":370
      }
    ],
    "is_impossible":false
  },
  {
    "question":"What plays the crucial role in the Mother to Child Transmission
    "id":276,
    "answers":[
      {
        "text":"DC-SIGNR plays a crucial role in MTCT of HIV-1 and that impair
        "answer_start":2003
      }
    ],
    "is_impossible":false
  },
```


Figure 27. Sample of COVID-QA *dataset*

	Context	Question	Id	Answer
0	Functional Genetic Variants in DC-SIGNR Are As...	What is the main cause of HIV-1 infection in c...	262	Mother-to-child transmission (MTCT) is the mai...
1	Functional Genetic Variants in DC-SIGNR Are As...	What plays the crucial role in the Mother to C...	276	DC-SIGNR plays a crucial role in MTCT of HIV-1...
2	Functional Genetic Variants in DC-SIGNR Are As...	How many children were infected by HIV-1 in 20...	278	more than 400,000 children were infected world...
3	Functional Genetic Variants in DC-SIGNR Are As...	What is the role of C-C Motif Chemokine Ligand...	316	High copy numbers of CCL3L1, a potent HIV-1 su...
4	Functional Genetic Variants in DC-SIGNR Are As...	What is DC-GENR and where is it expressed?	305	Dendritic cell-specific ICAM-grabbing non-inte...

Figure 28. Restructured COVID-QA *dataset using Jupyter Notebook*

Adding on to that, COVID-QA dataset [12] that can be found at the ACL Anthology repository is used for our deep learning Roberta-base-squad2 models. Figure 27 is a sample of COVID-QA dataset that is in SQuAD format with 2,019 QA pairs . The dataset that was originally in JSON format was loaded into jupyter notebook as data frames and restructured into 4 columns as shown in Figure 28 and unnecessary columns of data were removed from the dataset.

2.2.3 Methodology

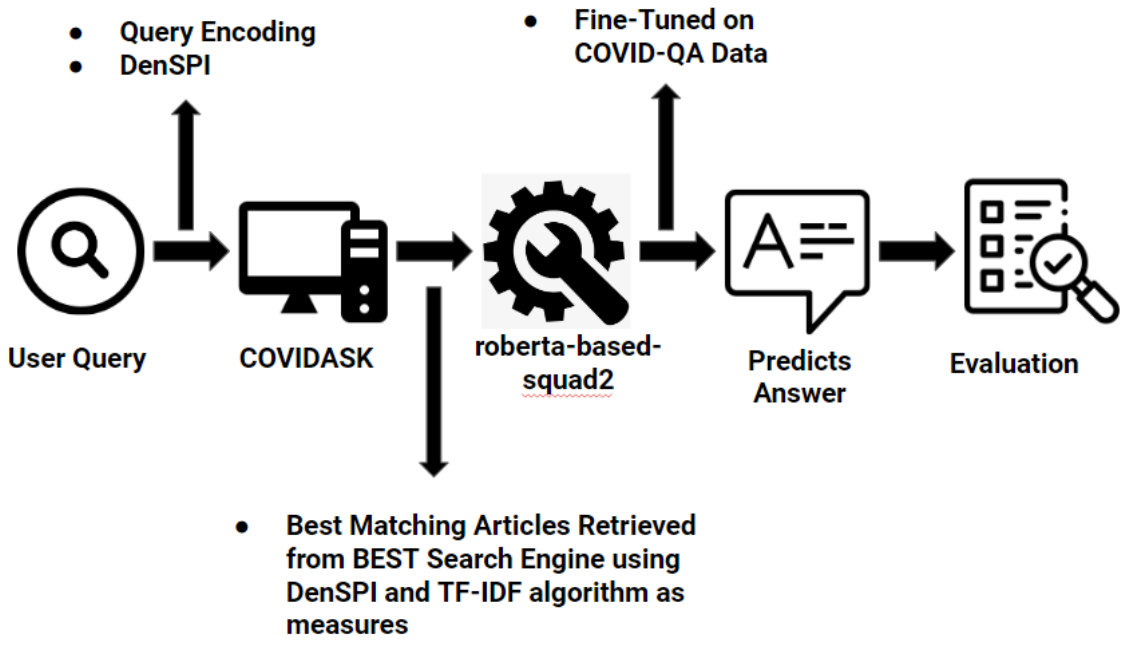


Figure 29. Summary of flow of different Methodology used in the process

Figure 29 shows the summary of the flow of the different methodology used in our process. When user input their query, the query was encoded in phrase indexes using DenSPI model under the COVIDASK Module, before the top best article was retrieved from BEST Search Engine by measuring the DenSPI indexes and using the TF-IDF algorithm as measures for comparison. The best article will then be passed to the roberta-based-squad2 deep learning model that is fine-tuned on COVID-QA dataset, and the model will output the answer as a phrase or word. The answer is then evaluated by the speed where the whole process takes from user input to answer output on the Telegram bot, as well as the legitimacy of the answer returned with regards to the user query.

2.2.4 Experiment Setup

Overview of the process

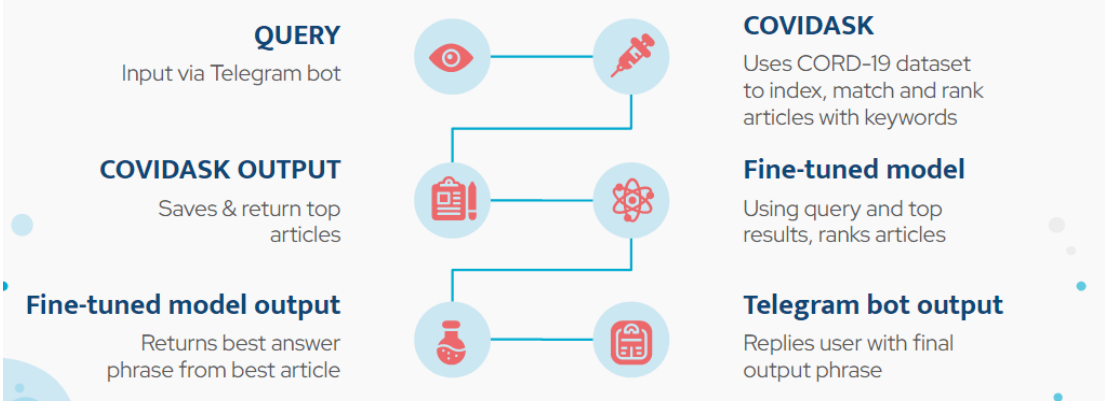


Figure 30. Overview of the QA Telegram Bot Process

Figure 30 is the summary of our QA Telegram Bot Process. For our process, we made use of COVIDASK to perform query encoding as well as document retrieval, before adding our own deep learning model to return the best answers from the best article.

Firstly, the user will input their query via Telegram bot and the COVIDASK module will encode the query, index, match and rank the articles with keywords and return the top article that best matches the user query. After that, our deep learning model that uses Roberta-base-squad2 will further analyse the top articles to return the best answer phrase from that article, and finally the answer will be returned via the bot to the user.

COVIDASK

The first part of our QA Telegram Bot process involves the COVIDASK module, where it handles the user query and returns the article that best matches the query that the user input. The module uses the base model known as the Dense Sparse Phrase Index model (DenSPI), and the model is trained using phrase dumps from 37k Cord-19 abstracts for faster and more accurate document and information retrieval process [13]. The abstracts were enumerated, embedded and indexed into phrases to prepare for pure phase-retrieval problems.

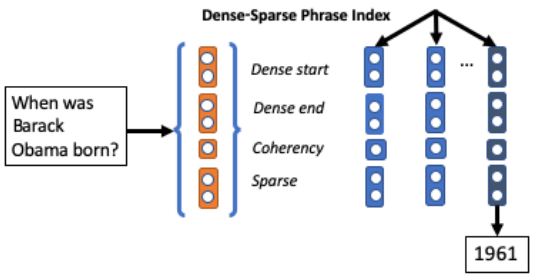


Figure 31. Screenshot of DenSPI Methodology that COVIDASK is based on

When users input their query, the query is encoded into indexes by tokenizing the query into query feature in phrase form, then enumerating, embedding and indexing the phrases.

```
# Stopwords and filtering for BEST queries
if not os.path.exists(os.path.join(os.path.expanduser('~'), 'nltk_data/corpora/stopwords')):
    nltk.download('stopwords')
if not os.path.exists(os.path.join(os.path.expanduser('~'), 'nltk_data/tokenizers/punkt')):
    nltk.download('punkt')
stop_words = set(stopwords.words('english') + ['?'] + ['Why', 'What', 'How', 'Where', 'When', 'Who'])
entity_set = [
    'COVID-19', 'SARS-CoV-2', 'hypertension', 'diabetes', 'heart', 'disease', 'obese', 'death',
    'HCoV-19', 'HCoV', 'coronavirus', 'symptoms', 'incubation', 'periods', 'period', 'quarantine',
    'asymptomatic', 'transmissions', 'fecal', 'excretion', 'decline', 'Wuhan', 'mortality',
    'patients', 'stay', 'reproduction', 'risk', 'factor', 'factors', 'pregnancy', 'interval', 'absent',
    'reported', 'length', 'diagnosed', 'United', 'States', 'isolated', 'CDC', 'WHO', 'vaccine',
    'negative', 'animals', 'airbone', 'spread', 'blood', 'sanitizer', 'controlled', 'illness', 'friends',
]
query_tokens = word_tokenize(query)
```

Figure 32. Screenshot of best_serach function from BEST.py file that will retrieve relevant articles

The BEST Model API is then being called to get a list of COVID-19 related articles from the biomedical entity search engine (BEST). Making use of the DenSPI model, a batch search for the top 10 articles that best matches the DenSPI of the user query is executed. By setting the entity_set values that are related to COVID in the best_search function, articles that are tagged with entity values that match the entity_set will then be returned. The articles will then be enumerated, embedded and indexed into phrases (DenSPI) and be compared against the user query phrase's denSPI index in the previous step, before TF-IDF algorithms are used to rank the top 10 articles in order of relevance to the query index. The article with the highest tf-idf score is returned by the model and passed to the deep learning model

Roberta-base-squad2

After obtaining the article from the previous step, our group used another model extracting the answer phrase from the article. We used the Roberta-base-squad2 as our pretrained model, and fine-tuned it on our own dataset (COVID-QA) with Huggingface API [14].

In the finetuning steps, firstly we reformat the data into SQUAD format. At the same time, our project cut the long context to a shorter one to cater for the pre-trained model's input limitation. During the process, we made the answers located in a random position in the context to avoid dataset biases.

```python
# cut context, leave a random number of character in front of the answer
def cut_context(start, length, context):
    # input: start,answer len, and context;
    # output new start and context

    # random number of char before answer
    num1 = random.randint(200,600)
    # after ans
    num2 = 1100 - num1
    # start and end of context cutting
    s = start - num1 if start - num1 >= 0 else 0
    e = start + length + num2 if start + length + num2 <= len(context) else len(context)
    # new answer start
    S = num1 if start - num1 >= 0 else start
    return S, context[s:e]
```

Figure 33. Screenshot of cut_content function from train.py file that will cut the context

```python
def reformat_data_into_squad(datapath,datapath_after):
    data = pd.read_json(datapath)
    covidqa = {}
    contents = []
    for i in tqdm(range(data.shape[0])):
        topic = data.iloc[i, 0]['paragraphs']
        for sub_para in topic:
            context = sub_para['context']
            title = sub_para['document_id']
            for q_a in sub_para['qas']:
                # cut context
                S, cut = cut_context(q_a['answers'][0]['answer_start'], len(q_a['answers'][0]['text']), context)
                content = {}
                content['context'] = cut
                content['title'] = title
                content['answers'] = {}
                content['answers']['answer_start'] = [S]
                content['answers']['text'] = q_a['answers'][0]['text']
                content['question'] = q_a['question']
                contents.append(content)
    covidqa['data'] = contents
    with open(datapath_after, 'w') as fp:
        json.dump(covidqa, fp, indent=4)
```

Figure 34. Screenshot of reformat_data_into_squad function from train.py file

Then, the processed data was put into the data loader to match the data input pattern from the Huggingface platform.

```python
# reformat data into squad format and load it
reformat_data_into_squad(datapath,datapath_after)
json_data = load_dataset('json',data_files=datapath_after, field='data')
```

Figure 35. Screenshot of reformatting steps from train.py file

After which, the data was tokenized and split into train and validation sets at 30%.

```
# tokenize context and answer
tokenized_squad = json_data.map(preprocess_function,
                                batched=True,
                                remove_columns=json_data["train"].column_names)
# train_val_split
tokenized_squad = tokenized_squad['train'].train_test_split(test_size=0.3)
```

Figure 36. Screenshot of data processing from train.py file that will tokenize and split the data

After data processing, we defined the data collator to make the dataset into batches and loaded the pre-trained model and tokenizers.

```
# data collator -- batch the data
data_collator = default_data_collator

# load pretrained model and tokenizer
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Figure 37. Screenshot of loading model and tokenizer from train.py file

Finally, we defined the training arguments, collaborated every section together and fed them into the Trainer. Figure 38 shows we ran for 3 epochs with batch size at 1 with regards to the limited computation power. Our model reached the F1 score at 0.67 as seen in Figure 39.

```
# collaborate everything into training args
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_steps=32
)

# define trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_squad["train"],
    eval_dataset=tokenized_squad["test"],
    data_collator=data_collator,
    tokenizer=tokenizer,
)

# fine tune it!
trainer.train()
```

Figure 38. Screenshot of setting training arguments and trainer from train.py file

```
print(F1_score)

0.6731940679474255
```

Figure 39. Screenshot of f1 score from train.py file

Telegram Bot and Combination of COVIDASK, Deeplearning Models and TelegramBot

The Telegram bot is created using python Telegram package and the Telegram API function as an evaluation of our solution. Using BotFather to obtain the bot token, a simple python Telegram bot code is used and the COVIDASK module is being imported to the bot module. User input is obtain and passed to COVIDASK function as shown in Figure 40 and the best article returned from the query result is then parsed to the *get_response* function as shown in Figure 41 with the user input query, where the Roberta-base-squad2 deep learning model will return a word or phrase that best answer the user query as a reply for the user query.

```
def echo(update, context):
    """Echo the user message."""
    user_input = str(update.message.text)
    requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

    query = user_input
    results = covidAsk(query)
    context = results['ret'][0]['context']

    return_ans = get_response(model_dir, query, context)


    # INSERT ANY FUNCTION HERE

    update.message.reply_text(return_ans)
```

Figure 40. Screenshot of echo function of the Telegram bot

```
def get_response(model_name, query, context):
  # Load model & tokenizer
  model = AutoModelForQuestionAnswering.from_pretrained(model_name)
  tokenizer = AutoTokenizer.from_pretrained(model_name)
  # model.cuda()
  # build model
  # nlp = pipeline('question-answering', model=model, tokenizer=tokenizer, device=0)
  nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)

  # get predictions
  inputs = {
          'question': query,
          'context':context
      }

  predict = nlp(inputs)['answer']
  return predict
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
model_dir = "drive/MyDrive/TM Project/model"

query = "Where do I live?"
context = "My name is Sarah and I live in London"
```

```
get_response(model_dir, query, context)
```

```
'London'
```

Figure 41. Screenshot of get_response function using Roberta-base-squad2 Model

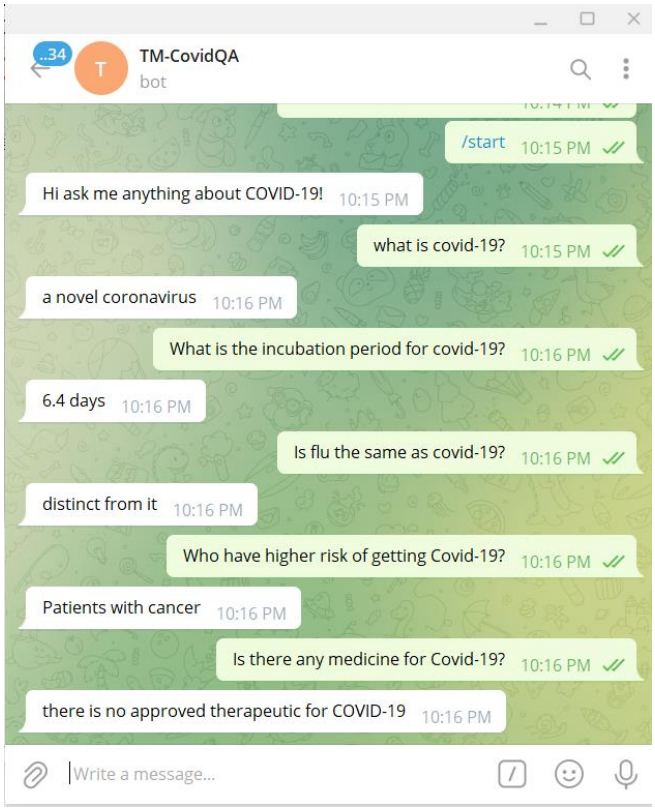


Figure 42. Screenshot of the responses returned by the Telegram bot when COVID-*19 related questions are being asked*

2.2.5 Results & Discussion

The bot seems to return legit and accurate answers when the question is asked using accurate medical terms and is unable to reply accurately to questions asked in a natural (human) way. An example will be when the question "What is Covid-19?" is asked, the correct answer is returned as seen in Figure 42. However, when the question is being asked in a more natural way (natural question) as to what general user will ask "what is covid?", the bot could not return an answer that makes sense.

```
(covidAsk) cx@cx-Virtual-Machine:~/Downloads$ python TM_CovidQA_Telebot.py
2022-04-10 22:43:58,955 - apscheduler.scheduler - INFO - Scheduler started
The time of execution is:  6.276210308074951
```

Figure 43. Screenshot of the time taken for the query to be answered

From Figure 43, we can see that the bot takes about 6.2 seconds to return the results from the point where the user query is received. Using other QA systems such as google search as a benchmark, the average response time from the point where users finish inputting the query, to the point where the answer is returned and shown is 800ms to 1.6 seconds. Other COVID QA systems such as DrQA take about 3-4 seconds. Therefore, the current Telegram bot's response time is slightly slow as compared to the other system.

3. Challenges Faced

While exploring and trying out the various available COVID-related QA systems that are available to the public, we experience difficulty in deploying the QA System as there are different environments which different QAs require. We worked on deploying and trying out 3 different QA systems, DrQA, CAiRE-COVID and COVIDASK in the respective sequence in a trial and error attempt since these 3 systems are rated quite highly based on the research paper mentioned in the literature review section. With no prior knowledge, lots of time is spent on exploring the different deployment as different systems require deployment on different OS.

One major issue that we faced is that there are many errors encountered when we are trying to install and run the QA systems. When trying to install DrQA system on the virtual machine on linux, there is an error when the requirement.txt was run, where the module named "pexpect" has a function update and the async function named "_async" is used as an reserved word in python 3.7 and the DrQA's code is built on older python version. When the python version was downgraded, it solved the error but more errors occurred due to the result of downgrading.

Similarly, we tried installing CAiRE-COVID system and met with an issue with the "faiss" package and even though the faiss package is successfully installed, when the program was run, there is an error showing that the subpackage in faiss named "swigfaiss" is not found. Reinstalling and debugging did not solve the issue and in the end, COVIDASK was used as it was installed and deployed without major issue.

4. Conclusion & Future Work

The aim of our project is to bring convenience to others in terms of finding accurate information about COVID-19. In task 1, our team managed to uncover the specific topics frequently talked about through ensuring that the word chosen for the topic holds high weight despite low count. This directly addresses the target problem identified earlier which was the massive cluttered information in the Internet today. In task 2, through using deep learning models and COVIDAsk, we were able to integrate that with Telegram Bot which the general public can have access to to ask questions about COVID-19. However, both tasks came with their own set of limitations and recommendations for the future.

For task 1, we felt that our dataset collection and preprocessing steps were not ideal. Firstly, there was not enough data being collected from Twitter so we could increase the dataset for both time periods. Secondly, abbreviations, slangs and acronyms were left in our data to be trained which would have affected the quality of our results. We propose to use the website Urban Dictionary to map through word embeddings [15] on our tweets. This would help us better deal with elements not found in Standard English. Lastly, there were some important phrases that were removed due to the removal of non-alphanumeric characters, such as COVID-19, which could have provided better insights. Hence, we propose to create a dictionary with important contextual phrases to be retained for future works.

Given that the current model that we use (LDA and GSDMM) uses a Bag-Of-Words model, where the order of the word is ignored and contextual meaning is lost, the results obtained may not be the best. We want to improve on this by proposing the use of word embeddings [16].

Word embeddings offer quality learning of word semantic relations in our corpus which will help us address the limitation of both LDA and GSDMM where semantic relations between words are ignored.

Both GSDMM and LDA have another limitation which we intend on addressing in our future work which is the assumption that a document is a distribution of topics (LDA) or that a document belongs to only one topic (GSDMM). While it is true that shorter texts, like tweets, tend to only talk about one topic, it is wrong to generalise and we should let the model decide whether it should be classified into one or many topics. That is why we propose the use of a Poisson-based Dirichlet Multinomial Mixture Model (PDMM) [17] that allows each document to be either classified as one or many topics.

Improving the model can only bring you so far in your analysis, if you cannot interpret the result it does not bring much value. Hence, we are also looking into automatic generation of topic labels to improve interpretability of our outputs. Automatic topic labelling is still a relatively new field but we can look into existing research done using sequence-to-sequence neural-based approach [18] which, when trained over a large synthetic dataset, can address previous limitations which restrict generated topic labels to a small set.

For task 2, for training the model, we propose to use question answering slicing window technique to truncate the long context into sub-query context pairs for the model to learn. This enables mapping of sub-query context pairs to original context more easily when inferencing [19]. This is a better alternative solution to the issue of the length of the context exceeding the maximum input length of the model.

With regards to data improvements, we propose to scrape updated data from official websites such as WHO, or government websites with updated COVID-19 data. Additionally, to enable the model to answer natural questions, we would train the model over natural questions corpus [20]. This would enable improvements of the bot in terms of answering natural language questions (i.e. know that COVID and COVID-19 refer to the same thing).

Other relevant tasks to look into include abstractive summarisation to determine how well a query matches a generated summary of a document [21] as well as information extraction using Name-Entity Recognition (NER) which can help with improving accuracy of answer through identifying entities in the queries to match with the entities in the underlying knowledge base [22]. These would help with increasing accuracy of the answers returned. Additionally, using NER to improve the QA system can reduce the amount of data being used, improving the efficiency of the system, and potentially generating faster responses for users.

5. Reflections

Name	Reflection
Ang Heng Di	Having some background in NLP projects, I felt good coming into the project. While I did face many issues, my teammates were very patient and understanding throughout the project duration. I had never done Topic Modelling tasks before but having to research and understand every aspect of Topic Modelling to complete the code and explanation had provided me with the best learning experience (outside of classroom) I could ever ask for. Coding, documentation and explanation aside, I feel that I have truly understood the importance of communication and accountability to everyone in the team to do your part as best as possible and if not ask for help. Our team had split into mainly two groups for two tasks but we had to still communicate and integrate each other's parts into our own as we were all working towards a common goal. It is really easy to just focus on your part and pay no heed to the other parts that do not concern you but I would say that defeats the purpose of doing two tasks as a group together. Overall I enjoyed the process of learning through coding it out yourself and researching (especially going beyond what we had learnt in class) new ideas or methodologies that would improve the model and I would love to do such projects/research again as a team/pair.
Chen Yi Bing	In the proposal phase of the project, I used text mining techniques and tried to apply the code from the lab in class to task2. In this process, I also learned the usage of the new package. In the second half of the project, the project allowed me to experiment with the new deep learning api and accomplish the task of not only inference but also running models with my own data. During the process, I encountered various problems, which were solved by searching for information and consulting prof. It was a very valuable experience. The communication and work allocation within the group was nice. Everyone is very competent in different aspects. The weekly group discussion was very efficient. All in all, it was a pleasant journey.
Kylie Tan	The learning curve for this project was quite steep for me because I do not have a background in NLP/Text Mining. Hence, a lot of time was spent on researching and understanding the techniques. Nonetheless, this project allowed me to step outside my comfort zone and challenge myself to learn and expose myself to some NLP techniques. Additionally, I also learnt about how to implement to Telegram bot as well as the codes that came with it. For our team collaboration, initially things were a little unclear and messy in the beginning. However, we eventually sorted out who was responsible for what helped with the progress of our group project. Also, our weekly meetings did help with checking in on each other as well as getting feedback and being open to ask for help when we needed it. Overall, I learnt that communication is very important because although

	we were all working on our own things for the project, eventually we had to link it back to one another and that takes frequent check-ins and updates. Overall, I really liked this project because this project allows us to help others to get information at their convenience by just using Telegram. Additionally, it also allowed us to learn about the people's sentiments on COVID-19. However, I did feel that our data could use a more updated one to fit the current context more. Nonetheless, given the time we had, I feel that what we have done was truly incredible.
Lim Chee Xiong	Having some prior experience in NLP projects, text mining and especially the building and understanding of QA systems to me is on another higher level. Majority of the time was spent researching and trying out the codes as my prior projects were run on Windows OS and majority of the QA project is built on Linux OS. This broadens my knowledge and experience in NLP and it's a great experience to have. For team collaboration, I felt that our team has done a great job in communicating. Weekly meetings were held for us to update one another on the work progress with all of us being highly-cooperative. This ensures that work can be completed and done on a smooth experience. Overall, I benefited much from this project and I look forward to the end of this COVID pandemic such that this and other similar COVID-related projects will never be needed again.
Teo Kai Lun Felicia	Having had no prior experience in text mining, it was a rather eye opening experience in regards to the numerous materials available - datasets, systems and models - to achieve a variety of text mining goals. As the team was highly cooperative and constantly met internally set deadlines, the project experience was rather pleasant. I liked how our project was relevant, applicable and possibly useful should it be properly implemented. However, as COVID-19 has played a significant part of our lives in the previous 2 - now going 3 years - it was rather draining to continue discussing details about this pandemic in addition to our daily lives.

References

[1] United Nations. (2020, April 30). *5 ways the UN is fighting 'infodemic' of misinformation*. https://www.un.org/en/un-coronavirus-communications-team/five-ways-united-nations-fighting-%E2%80%98infodemic%E2%80%99-misinformation

[2] Barua, Z., Barua, S., Aktar, S., Kabir, N., & Li, M. (2020). Effects of misinformation on COVID-19 individual responses and recommendations for resilience of disastrous consequences of misinformation. *Progress in Disaster Science*, *8*, 100119. https://doi.org/10.1016/j.pdisas.2020.100119

[3] Daniel Maier, A. Waldherr, P. Miltner, G. Wiedemann, A. Niekler, A. Keinert, B. Pfetsch, G. Heyer, U. Reber, T. Häussler, H. Schmid-Petri & S. Adam (2018) Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology, Communication Methods and Measures, 12:2-3, 93-118, DOI: 10.1080/19312458.2018.1430754

[4] Zheng, Z. F., He, Y. H., & R.P. (2021, June). *A Query-Driven Topic Model*. https://doi.org/10.48550/arXiv.2106.07346

[5] Zamani, M., Schwartz, H. A., Eichstaedt, J., Guntuku, S. C., Ganesan, A. V., Clouston, S., & Giorgi, S. (2020). Understanding Weekly COVID-19 Concerns through Dynamic Content-Specific LDA Topic Modeling. Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, 2020, 193–198. https://doi.org/10.18653/v1/2020.nlpcss-1.21

[6] D.K., S.B.S., N.N.M., & D.B.M. (2021, January). *An Interpretation of Lemmatization and Stemming in Natural Language Processing*. ResearchGate. Retrieved April 12, 2022, from https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing

[7] J.Y., & J.W. (2014, August). *A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering*. DB Group. Retrieved April 12, 2022, from https://dbgroup.cs.tsinghua.edu.cn/wangjy/papers/KDD14-GSDMM.pdf

[8] Gupta1, P., Garg2, R., & Kaur3, A. (2021, November 1). IOPscience. Journal of Physics: Conference Series. Retrieved April 12, 2022, from https://iopscience.iop.org/article/10.1088/1742-6596/2062/1/012027

[9] Lee, J., Yi, S. S., Jeong, M., Sung, M., Yoon, W., Choi, Y., Ko, M., & Kang, J. (2020, October 9). Answering questions on COVID-19 in real-time. arXiv.org. Retrieved April 12, 2022, from https://arxiv.org/abs/2006.15830

[10] Chaybouti, S., Saghe, A., & Shabou, A. (2021, January 30). EfficientQA : A Roberta based phrase-indexed question-answering system. arXiv.org. Retrieved April 12, 2022, from https://arxiv.org/abs/2101.02157

[11] Download Cord-19. Semantic Scholar. (n.d.). Retrieved April 12, 2022, from https://www.semanticscholar.org/cord19/download

[12] Möller, T., Reina, A., Jayakumar, R., & Pietsch, M. (n.d.). Covid-QA: A question answering dataset for covid-19. ACL Anthology. Retrieved April 12, 2022, from https://aclanthology.org/2020.nlpcovid19-acl.18/

[13] Seominjoon. (n.d.). Seominjoon/denspi: Real-time open-domain question answering with dense-sparse phrase index (DENSPI). GitHub. Retrieved April 12, 2022, from https://github.com/seominjoon/denspi

[14] *deepset/roberta-base-squad2 · Hugging Face*. (2022, January 25). Huggingface. https://huggingface.co/deepset/roberta-base-squad2 *Download Cord-19*. Semantic Scholar. (n.d.). Retrieved April 12, 2022, from https://www.semanticscholar.org/cord19/download

[15] S.R.W., W.M., B.M., & Kiran Garimella. (2020, May). *Urban Dictionary Embeddings for Slang NLP Applications*. Aclanthology. Retrieved April 12, 2022, from https://aclanthology.org/2020.lrec-1.586.pdf

[16] C.L., Y.D., H.W., Z.Z., A.S., & Z.M. (2018, April). Enhancing Topic Modeling for Short Texts with Auxiliary Word Embeddings. Digital Library Acm. Retrieved April 12, 2022, from https://dl.acm.org/doi/10.1145/3091108

[17] J.Q., Z.Q., Y.L., Y.Y., & X.W. (2019, April 13). *Short Text Topic Modeling Techniques, Applications, and Performance: A Survey*. Arxiv. Retrieved April 12, 2022, from https://arxiv.org/abs/1904.07695

[18] A.A., N.A., & M.S. (2020a, May 29). *Automatic Generation of Topic Labels*. Arxiv. Retrieved April 12, 2022, from https://arxiv.org/ftp/arxiv/papers/2006/2006.00127.pdf

[19] *SQuAD question answering task based on BERT pretrained model*. (2022, March 2). Moon Inn. https://www.ylkz.life/deeplearning/p10265968/

[20] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M. W., Dai, A. M., Uszkoreit, J., Le, Q., & Petrov, S. (2019). Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, *7*, 453–466. https://doi.org/10.1162/tacl_a_00276

[21] Esteva, A., Kale, A., Paulus, R., Hashimoto, K., Yin, W., Radev, D., & Socher, R. (2021). COVID-19 information retrieval with deep-learning based semantic search, question answering, and abstractive summarization. *Npj Digital Medicine*, *4*(1). https://doi.org/10.1038/s41746-021-00437-0

[22] Toral, A., Noguera, E., Llopis, F., & Muñoz, R. (2005). Improving Question Answering Using Named Entity Recognition. *Natural Language Processing and Information Systems*, 181–191. https://doi.org/10.1007/11428817_17