

# **Text Summarization and Sentiment Analysis of COVID-19 Related News Articles**

Wenhui Jin, Seungjoo Choi, Jihee Son

Department of Computer Science, Stony Brook University

CSE 351/519 Data Science

Prof. Pravin Pawar

June 19, 2020

## Table of Contents

Introduction	3
Machine Learning Techniques	3
2.1 Text Summarization	4
2.1.1 Extractive Text Summarization	4
Technique #1 Cosine Similarity	4
Technique #2 Term Frequency(TF) and Inverse Document Frequency(IDF)	5
2.1.2 Abstractive Text Summarization	6
Technique #3: Tensorflow text summarization	6
2.2 Sentiment Analysis	7
Technique #4 VaderSentiment of NLTK Module Library	7
Apply	8
Validation	12
Visualization and interpretation of Results	14
Conclusion	17
6.1 Limitations and Performance Concerns	17
6.2 Business Value	17
6.3 Reflection	18
Future Work	18
7.1 Further Research	<b>Error! Bookmark not defined.</b>
Work Cited and Appendix	18

## 1. Introduction

Ever since the first COVID-19 case was reported, the virus has affected the world and is now a pandemic. It is important for governments and related organizations to analyze and to understand how COVID-19 affects people economically, politically, psychologically, to carry out proper policy and counter plans to help people to get through various difficulties. Through this project, we find out people's reaction to COVID-19 in different aspects, and the trend of people's sentiment.

We have extracted COVID-19 related news from the news dataset. After that we summarized the extracted dataset which is COVID-19 related news articles. We conducted article summarization using two different techniques, extractive summarization and abstractive summarization. After, we conducted sentiment analysis based on the summarized news, and used Power BI to visualize the trend of sentiment as well as the most frequent words appearing in positive news and negative news respectively. We figured out the difficulties that people are going through by analyzing the frequently appeared words in negative news and summarized news articles.

## 2. Machine Learning Techniques

Text summarization and sentiment analysis are two primary techniques that we used for the project. In general, there are two types of summarization which are extractive summarization and abstractive summarization. First of all, the key point of extractive summarization is weighting the important part of sentences. The summarization is done by selecting a subset of words that retain the most important points. From the original text, the technique weights the sentences based on similarity between sentences and selects sentences with higher rank. The following is the general pipeline for extractive summarization

*Input document → sentences similarity → weight sentences → select sentences with higher rank*

Abstractive text summarization is the text summarization technique that generates words from the model through prediction instead of extracting the words from the samples itself. This technique is able to generate summarization with the words that do not appear in the original input file if it is necessary. Consequently, the interpretation and examination of text using advanced natural language techniques is required to generate a shorter text but conveys the most

critical information of the contents. Although there are many approaches to implement the generation of words, we chose RNN as our approach in this particular project. The following part is the general pipeline for abstractive summarization.

*Input document → understand context → semantics → create own summary.*

Sentiment analysis refers to an analysis of extracting subjective impressions, feelings, attitudes, and individual opinions on a topic from a text. The result of the analysis is generally expressed in binary form as yes/no or good/bad. The main task of sentiment analysis is to identify positives, negatives, or neutrality in a given text, which is called as polarities of the text.

For this project, we tried two extractive methods based on Cosine similarity and term frequency(TF) & Inverse Document Frequency(IDF) respectively. We also tried to use one abstract text summarization based on the tensorflow especially in sequence to sequence model. At last, we tried one technique of sentiment analysis based on the NLTK module from Python.

## **2.1 Text Summarization**

### **2.1.1 Extractive Text Summarization**

#### **Technique #1 Cosine Similarity**

Open Source: <https://github.com/edubey/text-summarizer>

Technique #1 uses an unsupervised learning approach to find the sentences similarity and rank them. This technique does not require a process of training and building a model for the summarization since it fully relies on Python built-in libraries such as Natural Language Toolkit (NLTK) and NetworkX. It follows below steps:

*Input news article → split into sentences → remove stop words → build a similarity matrix → generate rank based on matrix → pick top N sentences for summary.*

First of all, it preprocesses a news article by splitting it into sentences and removing the stop words. This process is done by using the Python functions such as open files, read lines, and split, and the stop words are obtained by the stopwords function from the NLTK module. Second, it makes sentences into a bunch of vectors and finds the similarity among sentences. The approach to finding the sentences similarity in technique #1 is Cosine similarity, which is a measure of similarity between two non-zero vectors of an inner product space. It measures the cosine angle between two vectors to find the similarity between them, and an angle of 0 indicates the sentences are similar. The Cosine similarity is obtained by utilizing the cosine\_distance

function from the NLTK module. After that, it ranks the sentences based on similarity among each other and this process is done based on the NetworkX module. At last, it outputs the summarized news article by picking top N sentences. N means the number of sentences that we want to summarize.

## **Technique #2 Term Frequency(TF) and Inverse Document Frequency(IDF)**

Open Source: <https://github.com/Shakunni/Extractive-Text-Summarization>

Technique #2 uses the TextRank algorithm to find the important sentences in each text file. TextRank is a graph based ranking algorithm for NLP, which decides the “importance of a vertex within a graph, based on a global information recursively drawn from the entire graph”(Sareen, 2018). TF is the number of times a word appears in the current sentence, and IDF means the number of a word appears in the entire set of all sentences. This technique is similar to technique #1 except for the step of tokenizing the text and building the similarity matrix. Besides, this technique does not make people decide how many sentences the final summarization includes. This technique follows below steps:

*Input news article → tokenize the document → generate term-document matrix(TD matrix)  
→ generate rank based on matrix*

Different from the previous technique which uses Python built-in function to tokenize the sentences, this approach uses the PunktSentenceTokenizer from the NLTK module. It is a default sentence tokenizer based on a pre-trained model which splits a text file at every period symbol.

Instead of making a matrix manually, this approach uses CountVectorizer and TfidfTransformer class and fit\_transform function from the sklearn module in Python to create the normalized word-sentence matrix. It converts a collection of text to a matrix of token counts. The fit\_transform method of CountVectorizer() class learns the vocabulary dictionary and returns a word-sentence matrix, and TfidfTransformer is used for normalizing the matrix so that the value comes out in the range 0 - 1

In addition, instead of picking the number of sentences in the summarization, this approach makes a threshold for picking the sentences. It takes the mean value of normalized scores, and any sentence with the normalized score 0.2 more than mean value is considered to be

threshold. Then it separates out the sentences that satisfy the criteria of having a value above the threshold.

### **2.1.2 Abstractive Text Summarization**

#### **Technique #3: Tensorflow text summarization**

Open Source : <https://github.com/dongjun-Lee/text-summarization-tensorflow>

Technique #3 is tensorflow implementation of abstractive text summarization by using seq2seq library. The seq2seq library which is the abbreviated form of the sequence to sequence, is a combined concept of RNN and encoder and decoder. The reason why we chose his technique is that it is the best for our current dataset which consists of numerous txt forms of articles and is able to adjust with it.

Its machine learning model consists of 4 different parts, which are word embedding, encoder, decoder, attention mechanism. The machine learning technique, especially the RNN is used in encoder and decoder. According to the author of the open source, encoder is used LSTM cell with `stack_bidirectional_dynamic_rnn` and Decoder uses LSTM Basic decoder for training, and BeamSearch Decoder for inference.

As we mentioned earlier, there are other ways to implement abstract text summarization by using another model of neural networks. However, we are specifically looking for the RNN model because of the characteristics of our dataset. Since our dataset consists of numerous articles related to one specific concept which is COVID-19, many synonyms related to COVID-19 repeatedly appear throughout the dataset. However, if we use a simple neural network, it is able to identify the specific word in different parts of the text. On the contrary, the RNN which stands for recurrent neural network is able to identify the repeatedly advent vocabulary throughout the whole dataset if it is found in different parts of the text.

In addition to this concept of RNN, we need to use the encoder and decoder. Since our data inputs which are the txt form of articles, consist of different lengths. Also, our newly generated sentences have different length as well. Therefore, we have another network that takes input of length, and generates another output of different length. The encoder and decoder is the architecture that enables these functions.

The basic pipeline for this technique is like the following.

*Input news article* → *prep\_data* → *train* → *test* → *save result*

This technique provides the pre-trained model, which we used for this project but it also provides a custom training option as well. We tried to perform transfer learning with the provided dataset from the open source and our dataset together, but due to lack of cpu performance of our machine and lack of time, we decided to leave it as a future work. Therefore we used only a pre-trained model for our dataset.

## 2.2 Sentiment Analysis

### Technique #4 VaderSentiment of NLTK Module Library

Technique #4 uses the VaderSentiment of NLTK Module Library. Vader stands for Valence Aware Dictionary and Entiment Reasoner. The general pipeline of VaderSentiment is as follows.

*Open the lexicon file → filter one letter words → check punctuation → calculate sentiment score → normalize score*

The first step is to read the lexicon file. The file consists of the average sentiment score of each word, derived from 10 people. Next, before calculating the sentiment score of the input document, the module filters one letter words in the input document and checks for punctuation characters such as period, question mark and exclamation mark. If the length of a word is less or equal to 1, the word is excluded from calculation.

When calculating the sentiment score, there are five major steps. First, the VaderSentiment checks for negative verbs such as aren't or cannot. If there's any, the score is weighted with -0.7. Next, the module considers uppercase letter words as emphasizing meanings and adds 0.733 if the sentiment of the word is positive and subtracts 0.733 if the sentiment of the word is negative. Third, when there's a modifier, we weight the noun by adding or subtracting 0.293 based on sentiment of the noun. Moreover, nouns are multiplied by weights based on their location in a sentence. So closer the noun is to the modifier, the bigger value of weight will be multiplied. After, when there's the word "but", the module considers as reverse of the content and multiplies the sentiment score by 0.5 if "but" comes before the content and multiplies by 1.5 if "but" comes after. Lastly, exclamation marks are considered as emphasis and each mark is weighted with 0.295.

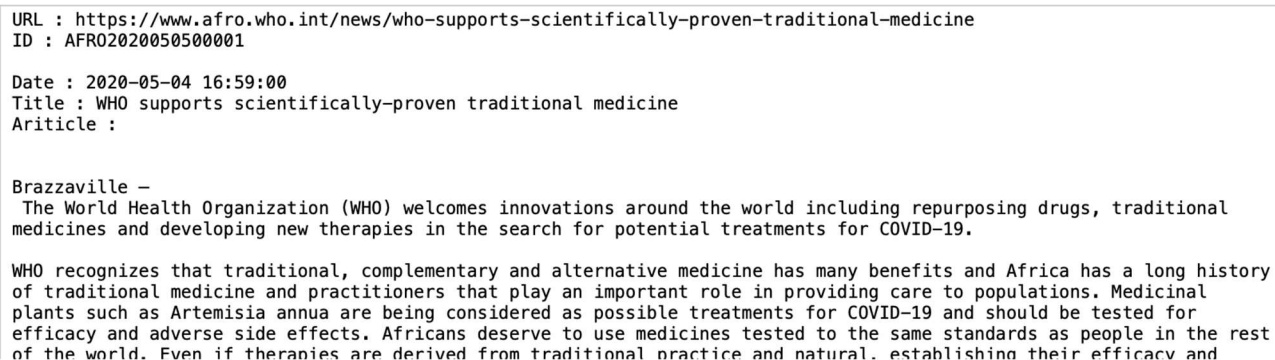
After the calculation is done, the sentiment score is normalized to a value between -1 and 1. Then the module computes the ratio of the calculated sentiment scores. The ratio is computed as positive score / ( positive score + negative score + neutral score).

### 3. Apply

#### a. Technique #1 Cosine Similarity

The open source that we chose does not perfectly fit our project in terms of the format of input news content and the format of output result. Therefore, we write our own Python script based on the open source to be customized to our project.

First of all, we deleted the line of URL, ID, Date, Title, and Article since they are unimportant information that increase the performance of text summarization. After several times of testing and digging deep into each news article, we find out that all of the news have the same format which start with lines of unimportant information such as URL and ID. Figure 3.1 is a screenshot of part of an example news article. Since the program reads through each news article line by line, lines such as “Date: 0000-00-00 00:00:00” and “Article:” will be recognized as a single line and be represented as vectors for further comparison. Thus, we deleted those lines to maximize the performance of the text summarization process.



```
URL : https://www.afro.who.int/news/who-supports-scientifically-proven-traditional-medicine
ID : AFR02020050500001

Date : 2020-05-04 16:59:00
Title : WHO supports scientifically-proven traditional medicine
Article :
```

Brazzaville –  
The World Health Organization (WHO) welcomes innovations around the world including repurposing drugs, traditional medicines and developing new therapies in the search for potential treatments for COVID-19.

WHO recognizes that traditional, complementary and alternative medicine has many benefits and Africa has a long history of traditional medicine and practitioners that play an important role in providing care to populations. Medicinal plants such as Artemisia annua are being considered as possible treatments for COVID-19 and should be tested for efficacy and adverse side effects. Africans deserve to use medicines tested to the same standards as people in the rest of the world. Even if therapies are derived from traditional practice and natural, establishing their efficacy and

Figure 3.1 screenshot of part of the example news contents

After a deep exploration of the results, we found out that there are news that are consist with sentences with many “|” symbol and no actual meanings. They are the tabs of certain websites with useless information. Therefore, we deleted summarized sentences that has more than two “|” symbol. In addition, we also found out that there are sentences that are meaningless. Those kinds of sentences always have less than 6 words. Thus, we decided to only regard sentences with more than 5 words as sentences that are meaningful.



Article :

2 Feb 2020: 17 187 / ... / 80 024 / ... / 81 589 / ... / 82 830 / 82 836 / 82 858 / 82 862 / 82 874 / Mainland China2 Feb 2020: 15 / .  
Worldwide | COVID-19 | Humans  
Tajikistan | COVID-19 | Humans  
Comoros | COVID-19 | Humans  
Shenzhen, Guangdong Province, China | COVID-19 | Humans

### Figure 3.2 contents with tabs

Date : 2020-05-02 13:45:52

Title : J Glob Health. Impact of coronavirus outbreak on psychological health

Article :

J Glob Health.

2020 Jun;10(1):010331. doi: 10.7189/jogh.10.010331.

Impact of coronavirus outbreak on psychological health.

Khan S

1,

2

,

Siddique R

1,

2

,

Li H

1,

2

,

Ali A

3

,

Shereen MA

4

,

Bashir N

3

,

Xue M

1,

2

### Figure 3.3 contents with short and meaningless sentence

URL : <https://flutrackers.com/forum/forum/-2019-ncov-new-coronavirus/united-state>  
ID : FLUT2020050500018

Date : 2020-05-05 09:00:12

Title : Arkansas boy battling cancer beats coronavirus

Article :

### Figure 3.4 screenshot of an example of empty contents

As we added more and more conditions to the summarization progress, we found out that some of the news articles might have no output or have less than N ranked sentences left. In this case, we save the title of the news and add it to the final output as a part of summarization since the title of a news contains many information about the news.

After customizing the open source to our project, we add functions to read through all the COVID-19 news contents and summarize them one by one, and save them into an Excel file for further sentiment analysis. These processes were done with the xlwt, xlrd, xlutils modules from Python library.

## **b. Technique #2 Term Frequency(TF) and Inverse Document Frequency(IDF)**

Instead of only using the PunktSentenceTokenizer function from NLTK module, we combined it with the token method that we wrote for technique #1 because the pre-trained model is not perfectly fit to our input data. We changed the first element of the tokenized sentence list that generated from PunktSentenceTokenizer to the first element that generated from the token method we wrote. The reason was that the first element of PunktSentenceTokenizer generated sentence list was everything starting from the first word until the first period. However, take Figure 3.1 as an example, the first period appears until the first paragraph ends, which means the first element in the generated sentence list from PunktSentenceTokenizer function includes a bunch of unimportant lines that will decrease the performance of the text summarization. Therefore, we replaced the first element with the lines without those unimportant information. As a result, we use the code in Figure 3.5 to select the final sentence list to be returned. FirstSentence is the first element in the token method we wrote, and sentences\_list is the list that is generated by PunktSentenceTokenizer. If the length of sentences\_list is 1, we use the sentences\_list as the final sentence list; and if firstSentence is already in the sentences\_list, we just use the sentences\_list as the final sentence list as well; otherwise we replace the first element in sentences\_list with firstSentence and use it as final token sentence list.

```
if len(sentences_list) == 1:
    return firstSentence,title
elif firstSentence == sentences_list[1]:
    return sentences_list,title
else:
    return firstSentence + sentences_list[1:],title
```

Figure 3.5 Process of token sentence list selection

Since some of the news contents have no meaningful contents just like Figure 3.3 and Figure 3.4, we just use title as the summarization in those kinds of situations. And we also add code regarding saving and looping. We add title to the summarization if the summarization is insufficient, and we loop through all the news contents and do summarization one by one, and save the results to an Excel file for further sentiment analysis

### c. **Technique #3 Tensorflow text summarization**

This open source is made for generating titles of the articles. Therefore, it does not perfectly fit our original purpose which is generating a summary of each article. We have made some modifications of the source code to adjust our data condition. This open source consists of 5 parts which are `prep_data.py`, `model.py`, `train.py`, `utils.py`, and `test.py`.

The `prep_data.py` prepares open source's dataset to the format of tensorflow training. The dataset of open source consists of two parts which are training and validation. As we stated earlier, the purpose of this open source is generating titles of the articles. They used actual title data for the validation and used article data for training dataset. However, we did not make any changes on `prep_data.py`, since we used a pre-trained model.

The `utils.py` contains the utility functions including parsing, building dictionary, dataset, batch and embedding functions of `test.py` and `train.py`. We did not make any modification on the functions that are used in `train.py` for the same reason. For the functions that used in `test.py`, we slightly modified the functions to adjust our dataset which includes over six thousands txt files in one directory. The code from open source only opens one text file which holds about 60,000 lines. Once it iterates the text file and parses through the `clean_title` function, it returns only the article section of the text file to the tensorflow model in `test.py`.

`Test.py` function is where the actual testing is performed. It is implemented in the open source data file format, we need to adjust the python script to our format. Therefore, we functionalize the test part to receive the text file directory as an input. Since we do not need to get the model each time we iterate the new text file from the directory, before the iteration starts we get the model first. After that, pass the text file to the util function for parsing. After that it gets predictions output as a result from the tensorflow model. Since it is abstractive text summarization, we do not want to get duplicated words. Therefore we check the duplicated

words. After all the processing is done, we save summarized text into the excel format for sentimental analysis in the later part.

#### d. Technique #4 VaderSentiment from NLTK module in Python

Based on the sentiment score derived from the sentiment analysis using VaderSentiment module, we classified the articles into three categories. We labeled the article as positive if the score was larger than 0.5, neutral if the score is between 0 and 0.5, and negative if the score is less than 0.

### 4. Validation

As our project regards natural language, we had to use human power to validate the results. We created a google form with 10 randomly selected articles. The google form consists of 4 questions for each article. Two of the questions ask testees to rate the summarization derived from the extractive methods, one asks testees to rate the summarization derived from the abstract method, and the last question asks testees about the sentiment of the original article. For questions regarding the summarization, testees give one point if the summarization is very poor, and five points if the summarization is very good.

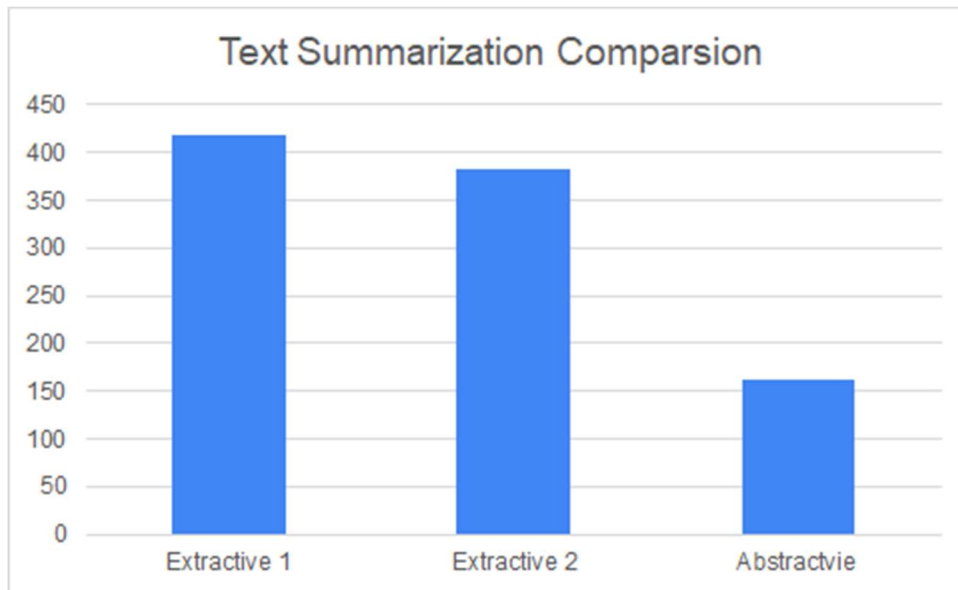


Figure 4.1 Result of rating summarization methods

The figure 4.1 shows the response of testees regarding three different summarization. The x-axis describes the used summarization method and the y-axis describes the sum of scores of

10 articles that the testees rated. The extractive 1 refers to extractive summarization method using cosine similarity and extractive 2 refers to extractive summarization method using TF and IDF. The score of each article ranges between 1 and 5, 1 representing poorly summarized and 5 representing excellently summarized. From the result, we can conclude that the extractive summarization method using cosine similarity was voted as the most powerful and accurate method and the abstractive method as the least powerful and accurate method.

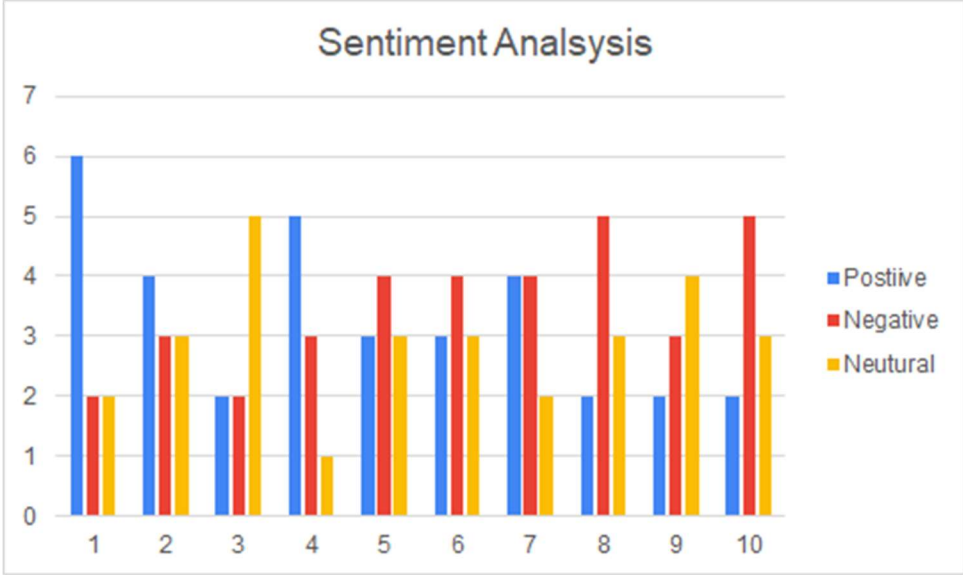


Figure 4.2 Result of answered sentiments of 10 articles

Article 1	positive
Article 2	positive
Article 3	negative
Article 4	positive
Article 5	negative
Article 6	neutral
Article 7	positive
Article 8	negative
Article 9	negative
Article 10	neutral

Figure 4.3 Result of sentiment analysis on the selected articles

The figure 4.2 and 4.3 shows the result of validation of the sentiment analysis. The figure 4.2 shows how the testees identified the sentiment of each article and the figure4.3 shows the sentiment of each article we have derived from our project. Currently, compared to the testees reponses, the result of sentiment analysis using the vader module shows approximately 50 percent of accuracy. According to researchers, when evaluating the sentiment of a given text document, human analysts tend to agree around 80-85% of the time, and that is also the baseline researchers try to meet when they train a sentiment scoring system. (Barba, 2019). Therefore, the accuracy of our result is overall acceptable and there is room for improvement.

### 5. Visualization and interpretation of Results

After analyzing the result of sentiment analysis and the visualization of the result in Power BI, several phenomena and conclusions were found. First of all, people have a negative attitude to the situation in general, and the percentage of negative news is over two times than positive news. According to Figure 5.1, in the period of late April and early May, the percentage of negative and positive news is 57.3% and 23.88% respectively, which means over half of the news in the dataset are negative, and the number of negative news is around two times larger than the number of positive news. Following figure 5.2 shows the difference between positive and negative news in each month. The number of negative news is around two times larger than positive news in each month as well. In Figure 5.3, it is easy to find out that people’s negative sentiment grew much faster than neutral and positive sentiment.

Count of Sentiment by Sentiment

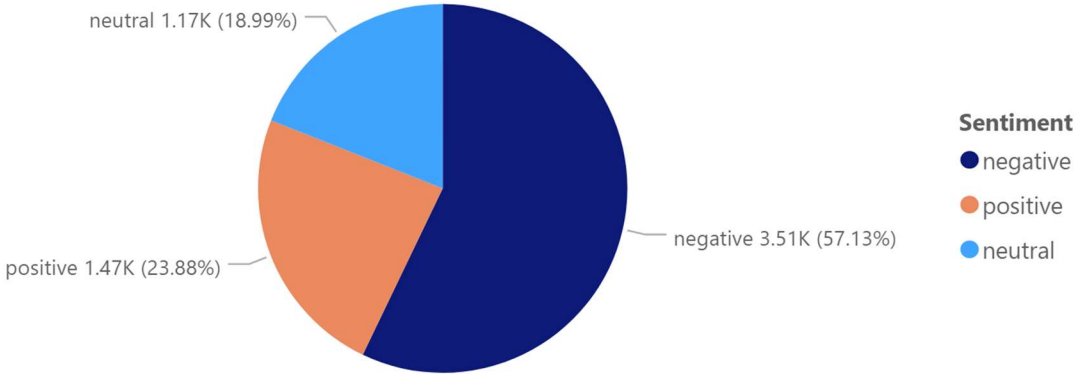


Figure 5.1 The percentage of neutral, positive, and negative news

### Count of Sentiment by Month and Sentiment

Sentiment ● negative ● neutral ● positive

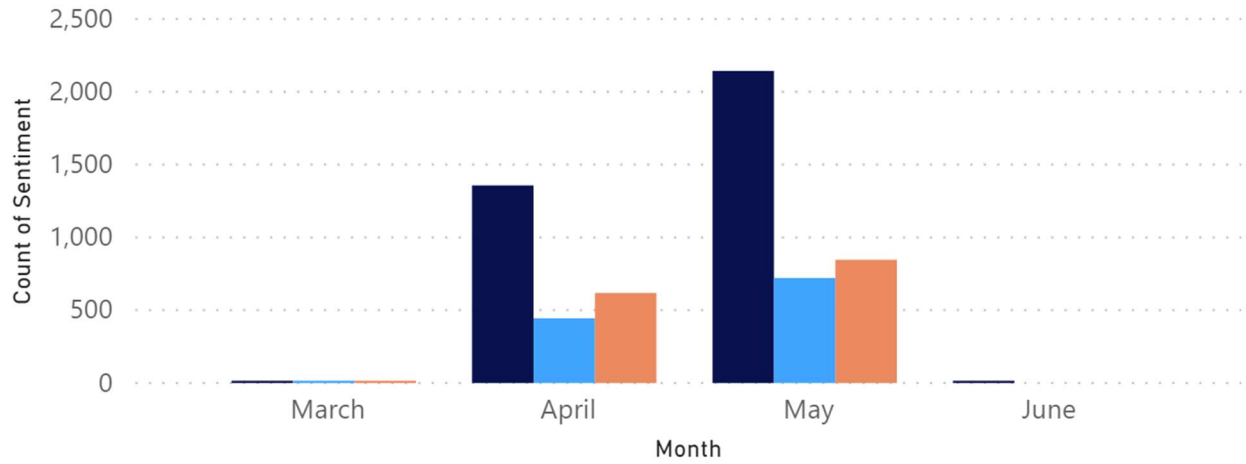


Figure 5.2 The number of neutral, positive, and negative news in April and May

### Count of Sentiment by Month and Sentiment

Sentiment ● negative ● neutral ● positive

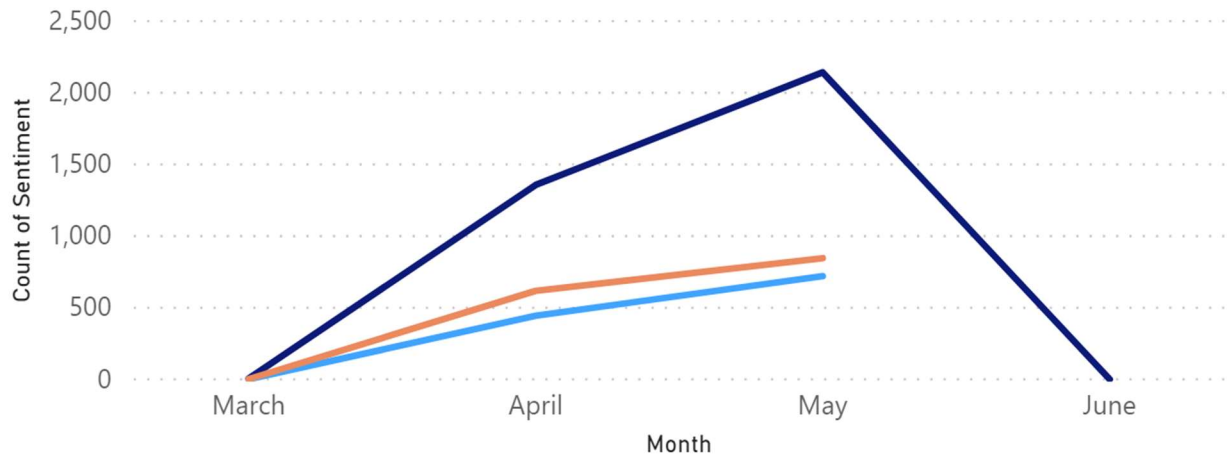


Figure 5.3 The growth of neutral, positive, and negative news respectively





economics, Trump, lockdown, and so on. Conversely, the words in positive news are in similar size except for some typical words such as Coronavirus and positive. People have a positive attitude to government, hospital, health, treatment, recovery, and so on. Several words have a high frequency in both positive and negative news, such as health and government. However, since the number of negative news is about two times larger than positive news, people's sentiments are negative to those subjects in general.

## **6. Conclusion**

### **6.1 Limitations and Performance Concerns**

One of the limitations that we encountered while doing the project was that the news articles are not real-time data. Since K.T provided the data monthly only from late December 2019 to early May 2020, text summarization and sentiment analysis will only be available during that time. Thus, it will be hard to find out people's latest sentiment regarding COVID-19. In addition, since KT does not provide the data after May 5, searching for proper data after that date might be another difficulty.

In the case of abstractive text summarization, we could not train our and build our own model due to lack of machine specification and lack of cpu performance. We spent over two days trying to train and build our model, but we gave it up and decided to use a pre-trained model due to the time limitation. That is the reason we have seen many <ink> values. The pre-trained model does not contain the words such as COVID-19 and WHO. If we perform transfer learning upon the pre-trained model, the accuracy of abstract summarization will be increased.

The last problem we had during this project was we weren't able to get ideal accuracy on sentiment analysis which is about 70 to 80 percent. Since the polarity score is calculated based on the existing word dictionary in the python library, the sentiment of the words is not customized with COVID-19 issues. Some positive words in the positive word dictionary might specifically become negative in the COVID-19 news. For example, the word China is neutral in general, but it is actually negative in most of the COVID-19 news.

### **6.2 Business Value**

Ideally, the result of the project could be useful for governments and global non-profit organizations to help people overcome the difficulties that come from COVID-19. This project

can find out how people react to the virus and the difficulties people are going through worldwide and help to prepare a counterplan based on the experiences from countries that suffered COVID-19 earlier. For example, if a country finds out that its citizens are suffering from economics, the government of the country can give out stimulus checks just like what the Korean and American governments did.

### **6.3 Reflection**

Through the project, we successfully finished extractive text summarization in two different techniques, and partially succeeded on the abstractive text summarization due to several limitations. We also had an overall acceptable performance on sentiment analysis. We believe that we could have a higher accuracy if we have more time and testers.

In the meantime, we are able to find out the trend of people's sentiment regarding COVID-19 and subjects that people have a positive and negative attitude to respectively. In general, people have a negative attitude regarding the situation of COVID-19, because the growth of negative sentiment is almost more than twice than the growth of positive sentiment. In addition, people have a negative attitude toward subjects like university and economics.

## **7. Future Work**

As we stated in the limitation, the accuracies of abstractive summarization and sentiment analysis can be improved. For the sentiment analysis, readjusting the positive and negative word banks to the words related to COVID-19 words such as lockdown and COVID-19 would increase the performance of it. We expect to have up to 80 percent of accuracy as our future goal of ideal outcome of sentiment analysis. For abstractive summarization technique, since words such as WHO or name of the disease weren't included in the pre-trained datasets, performing transfer learning to the pre-trained model would minimize the appearance of <unk> values and increase the performance overall.

## **8. Work Cited and Appendix**

Alsaqer, A., & Sasi, S. (2017). Movie Review Summarization and Sentiment Analysis using RapidMiner *NetACT*, 329-335. doi: 10.1109/NETACT.2017.8076790

Barba, P. (2019, July 10) Sentiment Accuracy: Explaining the Baseline and How to Test It. Retrieved June 18, 2020, from <https://www.lexalytics.com/lexablog/sentiment-accuracy-baseline-testing#:~:text=Setting%20a%20baseline%20sentiment%20accuracy,training%20a%20sentiment%20scoring%20system.>

Bhargava, R., Sharma, & Y., Sharma, G. (2016). ATSSI: Abstractive Text Summarization using Sentiment Infusion. *Procedia Computer Science*, 404-411. doi: 10.1016/j.procs.2016.06.088

Dubey, P. (2018, December 23). Understand Text Summarization and create your own summarizer in python. Retrieved June 18, 2020, from <https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70>

Hamzah, F., Lau C .H., Nazri, H., Ligot, D. V., Lee, G., Tan, C. L., . . . Salunga, R, E. (2020). CoronaTracker: World- wide COVID-19 Outbreak Data Analysis and Prediction. *Bull World Health Organ*, doi: <http://dx.doi.org/10.2471/BLT.20.255695>

Jahanbin, K., & Rahmanian, V. (2020). Using twitter and web news mining to predict COVID-19 outbreak. *Asian Pacific Journal of Tropical Medicine*, 13.

Opidi, A. (2019, April 15). A Gentle Introduction to Text Summarization in Machine Learning. *Floydhub*. Retrieved from <https://blog.floydhub.com>

Sareen, S. (2018, August 16). Text Summarisation with Gensim (TextRank Algorithm). Retrieved June 17, 2020, from <https://medium.com/@shivangisareen/text-summarisation-with-gensim-textrank-46bbb3401289>

Urologin, S. (2018). Sentiment Analysis, Visualization and Classification of Summarized News Articles: A Novel Approach. *IJACSA* 9(8), 616-625. doi:

10.14569/IJACSA.2018.090878

[http://www.aihub.or.kr/problem\\_contest/covid19](http://www.aihub.or.kr/problem_contest/covid19)

<https://github.com/edubey/text-summarizer/blob/master/text-summarizer.py>

<https://github.com/Shakunni/Extractive-Text-Summarization>

<https://github.com/dongjun-Lee/text-summarization-tensorflow>

<https://www.absentdata.com/power-bi/sentiment-analysis-in-power-bi/>

[https://github.com/Stephaniejinn/Test\\_Summarization-Sentiment\\_Analyze](https://github.com/Stephaniejinn/Test_Summarization-Sentiment_Analyze)