## 1. Write a paragraph about what you have learned in this project, and what you found most challenging.

Through this project, I learned the following two things: the importance of dealing with unnecessary words/punctuation and how to write a large program.

Firstly, when we analyze words, it is important to clear the text in advance, such as omitting the subject, articles (a, an, the), conjunctions (and, or), and so on. The reason is that we can focus on the words which we want to analyze. Suppose we would like to infer a focus point or trend word from several news articles by obtaining the most frequent word. If we analyze the word frequency of each article without clearing its text, it is a high probability that a computer returns the conjunction (and/or), period, or comma as the most frequent word. Especially, it will happen when we use the split function in Python to acquire the words one by one. Thus, clearing unnecessary words boosts the efficiency of analyzing data.

Secondly, I realized a crucial point of creating or writing computer programs: we should define several functions step by step. In other words, we should write a function for a specific purpose rather than unifying all or multiple purposes into one function. One of its advantages is that it is easier to develop a huge program in teams. If a group decides that one function must be only one role, its members can distribute a lot of tasks to complete a huge program. On the other hand, if a group decides to create a huge program with one large function, they will face some difficulties. For example, every person will use different the name of variables in a function, so they will consume a huge amount of time to check every variable one by one. Also, their productivity may decrease because group members are likely to work on the same task with multiple members individually. Its cause is that each of their tasks is uncertain and not determined. Therefore, I learned that splitting one task into one function improves group work productivity.

In conclusion, throughout this project, I have learned the importance of clearing the text as preparation for efficient data analysis and defining one function against one purpose.

## 2. Write a paragraph about some of the applications of document indexing and text processing.

Description of indexing() function:

This function reads the text file (filename.txt) and creates the nested dictionary *global_index_file*. This dictionary is composed of a term, the document number that its term is contained, and its frequency in each document. This document number is determined by the order of reading the text file from the dataset. If we apply a specific term to this dictionary such as global_index_file["Ontario"], it will return to another dictionary that contains its document number and frequency. Then, we apply a specific document number to the previous result, we can obtain its frequencies.

More specifically, the function indexing() is composed of 5 functions: readFolderContent(), punctuationRemoval(), stopWordRemoval(), appendTermDocFreq(), getIndex().

Firstly, the function indexing() opens the text file to note the term, its document number, and its frequency from the database documents. Its file is named "TermDocFile.txt" and manipulated as a variable *TermDocFreqFile*. Next, the computer obtains all documents from the database by running readFolderContent() function. After completing this phase, each document is stored as a string format in the list "files" The order of this list corresponds with the order of reading documents from the dataset. For example, the computer reads '0.txt', '1.txt', '10.txt', '100.txt', and so on. Thus, the document of each text is added to the list "files" in the same order as them.

The phase punctuationRemoval() and stopWordRemoval() are run in order to deal with unnecessary words or symbols. More specifically, punctuationReomval() replaces specified punctuation with the blank. And then, stopWordRemoval takes the result removed the punctuation as an input value and returns the word list that contains the term excluded stop words extracted from *Stop_Words.txt*. In other words, applying two functions to all documents consecutively can prepare for creating the index.

In the phase of appendTermDocFreq(), the computer retrieves a term, its document number, and its frequency for each document. These documents are extracted from the *files* one by one, which is the list created by readFolderContent() function. Also, the document number stored in *TermDocFile.txt* corresponds with the order of extracting documents from the *files*.

In the last phase getIndex(), the computer creates the nested dictionary, *index_file*, that is easy to access the document number that a specific term appears and its frequency. Also, the output of indexing() function is the same format as the output of getIndex() function.

### 3. Why is it a good idea to remove stop words and punctuations?

The purpose of removing stop words and punctuation is to focus on the important words or some specific topics. If people try to analyze a considerable number of texts by computer, they will utilize the word's frequency or the proportion of specific words against the whole text. However, in this phase, the computer will consider some unnecessary marks (punctuation) as words. Obviously, periods and commas are the objects that analysts do not want to focus on. Also, the increase of those unnecessary words can lead to the decline of the proportion of some important words against the whole document. It is because the denominator (the whole document) will increase although the numerator (focus words frequency) unchanged. Therefore, stop words and punctuation will decline the performance of the text analysis. That's why removing unnecessary words is a good idea to analyze the words or texts.

### 4. In document retrieval, why do we use TF*IDF method? Why using only term frequency is not a good idea?

First of all, IDF is calculated by the log of the total number of documents divided by the number of documents that contain a specific term (retrieved from query words one by one). Considering the log is merely the exponent, IDF indicates how infrequent a specific word is against the whole number of documents. To illustrate, IDF has a numerator (total documents) and a denominator (a word's frequency). That is, the fewer a specific word is included in the whole document, the more the outcome

of IDF increases. And the bigger the total documents in a collection, the more the outcome of IDF increases. Thus, we use the TF*IDF method to focus on how rare a term is.

If we do not use IDF, we will only focus on how frequent a term is against the whole document. It means that the more a query word is detected in a document by the computer, the bigger the score for its document. For example, if someone writes “document” as a query, the computer will return a document that contains a lot of the word “document”. It may work well when a query word is a common term. However, if query words contain a common term as well as a technical term that is used in a specific field, the computer tends to ignore the importance of the technical term. It is because the computer only focuses on how frequent a term is against the whole document. Therefore, using only the term frequency will not be an appropriate idea to develop the searching algorithm.