Attn: Dr. Sun Aixin

AI6122 Text Data Management and Processing

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below. We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work.

Name	Signature / Date
Huang Yunfan	
Li Xuemeng	
Lu Pai	
Mao Jiangtian	

Important note:

Name must **EXACTLY MATCH** the one printed on your Matriculation Card. Any mismatch leads to **THREE (3)** marks deduction.

Review Data Analysis and Processing

G19

Huang Yunfan
G2201838F
Nanyang Technological University
Singapore
YHUANG063@e.ntu.edu.sg

Mao Jiangtian
G2202679E
Nanyang Technological University
Singapore
maoj0005@e.ntu.edu.sg

Li Xuemeng
G2202226E
Nanyang Technological University
Singapore
lixu0025@e.ntu.edu.sg

Lu Pai
G2102572F
Nanyang Technological University
Singapore
lupa0002@e.ntu.edu.sg

I. Introduction

This assignment consists of four parts: Data analysis, Development of a search engine, Development of a review summarizer, and Development of a sensitive word analysis application. In the Data analysis part, the writing style is analyzed by visualizing some sentences of the datasets and applying Pos Tagging. Sentences, words, and stems draw the distribution of the datasets. By calculating the relative word entropy, the most ten indicative words are found, which show the topic of each dataset. The search engine part was completed based on the given example code. It has fitted all the requirements from the assignment pdf. The indexer and searcher all work well and show no errors when running the programmer. The choice of Application is the Sentiment Analysis from reviews in the dataset, shown as 'Negative' for scores 1 and 2, 'Neutral' for scores 3, and 'Positive' for scores 4 and 5. The score was extracted from the 'Overall' part in JSON files.

II. Data pre-processing

A. Data selection

In this project, the two datasets are selected as "Sports and Outdoors" and "Cell Phones and Accessories". In each dataset, 200 samples are randomly chosen by their "asin"s. The two sampled datasets are the training datasets of this project. All analyses are done on these two datasets.

The JSON and CSV versions of the same dataset are built based on the tasks.

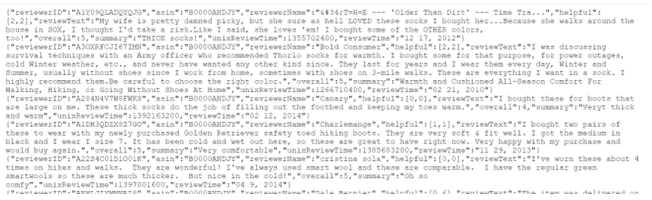
B. Json processing,

However, we find that the JSON file processed cannot be reorganized by JAVA. This is because the required data structure differs between the JSON parser python and java. Python JSON parser can handle the JSON file in a nested JSON structure. i.e., the JSON structures of each review are placed in a big JSON structure. The structure of this JSON format is shown below:

[{"reviewerID": "A1Y09GLADQYQJG", "asin": "B0000ANDJY", "reviewerName": ""T-M-E --- 'Older Than Dirt' --- Time Tra...", "helpful": [2, 2], "reviewText": "My wife is pretty damned picky, but she sure as hell LOVED these socks I bought her...Because she walks around the house in SOX, I thought I'd take a risk.Like I said, she loves 'em! I bought some of the OTHER colors, too!", "overall": 5.0, "summary": "THICK socks!", "unixReviewTime": 1355702400, "reviewTime": "12 17, 2012"}, {"reviewerID": "A3OXREJI67IMM", "asin": "B0000ANDJY", "reviewerName": "Bold Consumer", "helpful": [2, 2], "reviewText": "I was discussing survival techniques with an Army officer who recommended Thorlo socks for warmth. I bought some for that purpose, for power outages, cold Winter weather, etc., and never have wanted any other kind since. They last for years and I wear them every day, Winter and Summer, usually without shoes since I work from home, sometimes with shoes on 3-mile walks. These are everything I want in a sock. I highly recommend them.Be careful to choose the right color.", "overall": 5.0, "summary": "Warmth and Cushioned All-Season Comfort For Walking, Hiking, or Going Without Shoes At Home", "unixReviewTime": 1266710400, "reviewTime": "02 21, 2010"}, {"reviewerID": "A284N4V7W8FWK8", "asin": "B0000ANDJY", "reviewerName": "Charlemange", "helpful": [0, 0], "reviewText": "I bought these for boots that are large on me. These thick socks do the job of filling out the footbed and keeping my toes warm.", "overall": 4.0, "summary": "Veryt thick and warm", "unixReviewTime": 1392163200, "reviewTime": "02 12, 2014"}, {"reviewerID": "A1DM3QDXXE57QO", "asin": "B0000ANDJY", "reviewerName": "Charlemange", "helpful": [1, 1], "reviewText": "I bought two pairs of these to wear with my newly purchased Golden Retriever safety toed hiking boots. They are very soft & fit well. I got the medium in black and I wear I size 7. It has been cold and wet out here, so these are great to have right now. Very happy with my purchase and would buy again.", "overall": 5.0, "summary": "Very comfortable", "unixReviewTime": 1385683200, "reviewTime": "11 29, 2013"}, {"reviewerID": "A22S4C01D1O01K", "asin": "B0000ANDJY", "reviewerName": "cristina sola", "helpful": [0,

Figure 1. Text example with JSON format

However, the JAVA JSON parse can only handle the JSON in a listed JSON, i.e., the JSON structures of each review are placed in a list. This structure is shown below:



{"reviewerID":"A1Y09GLADQYQJG","asin":"B0000ANDJY","reviewerName":""T-M-E --- 'Older Than Dirt' --- Time Tra...","helpful":
[2,2],"reviewText":"My wife is pretty damned picky, but she sure as hell LOVED these socks I bought her...Because she walks around the
house in SOX, I thought I'd take a risk.Like I said, she loves 'em! I bought some of the OTHER colors,
too!","overall":5,"summary":"THICK socks!","unixReviewTime":1355702400,"reviewTime":"12 17, 2012"}
{"reviewerID":"A3OXREJI67IMM","asin":"B0000ANDJY","reviewerName":"Bold Consumer","helpful":[2,2],"reviewText":"I was discussing
survival techniques with an Army officer who recommended Thorlo socks for warmth. I bought some for that purpose, for power outages,
cold Winter weather, etc., and never have wanted any other kind since. They last for years and I wear them every day, Winter and
Summer, usually without shoes since I work from home, sometimes with shoes on 3-mile walks. These are everything I want in a sock. I
highly recommend them.Be careful to choose the right color.","overall":5,"summary":"Warmth and Cushioned All-Season Comfort For
Walking, Hiking, or Going Without Shoes At Home","unixReviewTime":1266710400,"reviewTime":"02 21, 2010"}
{"reviewerID":"A284N4V7W8FWK8","asin":"B0000ANDJY","reviewerName":"Canary","helpful":[0,0],"reviewText":"I bought these for boots that
are large on me. These thick socks do the job of filling out the footbed and keeping my toes warm.","overall":4,"summary":"Veryt thick
and warm","unixReviewTime":1392163200,"reviewTime":"02 12, 2014"}
{"reviewerID":"A1DM3QDXXE57QO","asin":"B0000ANDJY","reviewerName":"Charlemange","helpful":[1,1],"reviewText":"I bought two pairs of
these to wear with my newly purchased Golden Retriever safety toed hiking boots. They are very soft & fit well. I got the medium in
black and I wear I size 7. It has been cold and wet out here, so these are great to have right now. Very happy with my purchase and
would buy again.","overall":5,"summary":"Very comfortable","unixReviewTime":1385683200,"reviewTime":"11 29, 2013"}
{"reviewerID":"A22S4C01D1O01K","asin":"B0000ANDJY","reviewerName":"cristina sola","helpful":[0,0],"reviewText":"I've worn these about 4
times on hikes and walks. They are used wool and these are comparable. I have the regular green
smartwools so these are much thicker. But nice in the cold!","overall":5,"summary":"Oh so
comfy","unixReviewTime":1397001600,"reviewTime":"04 9, 2014"}

Figure 2. Text example for JAVA JSON parse

To solve this problem, we wrote a JSON converter to convert the nested JSON to listed JSON by using the Python pandas package and set the attributes of "DataFrame.to_json" to "orient=" records," lines=True". With this setting, the pandas will store the nested JSON as a listed JSON, which the JAVA JSON parser can process.

III. Main Tasks

We hereby describe the idea, the code logic, and the result of all four tasks.

A. Data Analysis

Data analysis includes the analysis of Writing Style, Pos Tagging, Sentence Segmentation, Tokenization and Stemming, and Indicative Words.

Several reviews are shown in Figure 3:

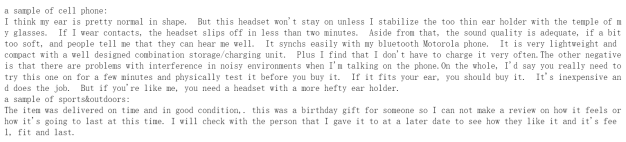


Figure 3 Some reviews of the datasets

After visualizing some reviews, in these datasets, the first word in a sentence is capitalized. Proper nouns are capitalized. There are a few misspellings, and the grammar could be improved. In the example shown in Figure 4, the "his" should be changed to "this" in the below sentence, and a "the" should be added before the word "Battery." Compared with news articles, the English in the dataset is not very formal.

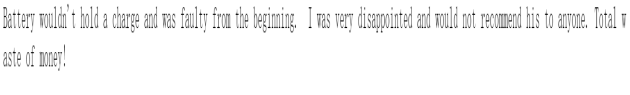


Figure 4. A wrong example

Pos Tagging:

Five sentences are randomly selected from the two datasets. The sentences and the corresponding POS tagging results are shown in Figure 5, 6, 7, 8, and 9.

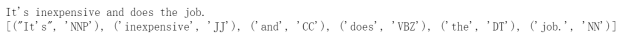


Figure 5. Pos tagging example 1

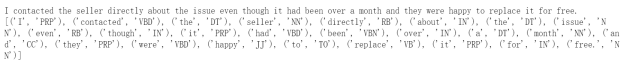


Figure 6. Pos tagging example 2

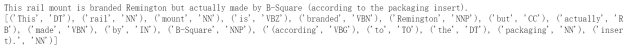


Figure 7. Pos tagging example 3

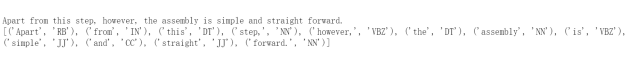


Figure 8. Pos tagging example 4

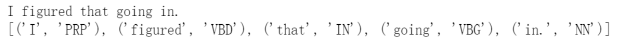


Figure 9. Pos tagging example 5

In this part, all tags are in the Penn Treebank tag set. The results are in a (word, tag) format. In most examples, the results are accurate.

Sentence Segmentation:

All reviews of the two datasets are performed through sentence segmentation. The result of the distribution is shown in Figure 10. The x-axis is the length of a review in the number of sentences, and the y-axis is the number of reviews of each length.

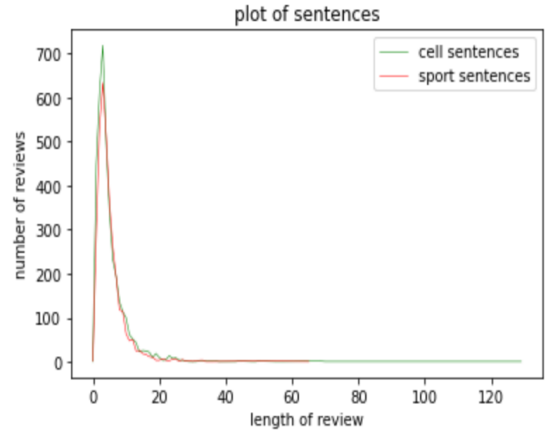


Figure 10. The distribution of two datasets by sentence length

From the figure, we can see that the distributions of the two datasets are quite similar.

The lengths of most reviews by the number of sentences are in the range of [1,10]. Only a few reviews have more than 20 sentences. In the "Cell Phone and Accessories" dataset, about 5% of the reviews are quite long (more than 50 sentences). A review has more than 100 sentences.

Tokenization and Stemming:

All reviews are tokenized, and the distributions are drawn by the number of words in a review. The figure of the two datasets' distributions is shown in Figure 11. The x-axis is the length of a review in the number of tokens, and the y-axis is the number of reviews of each length.

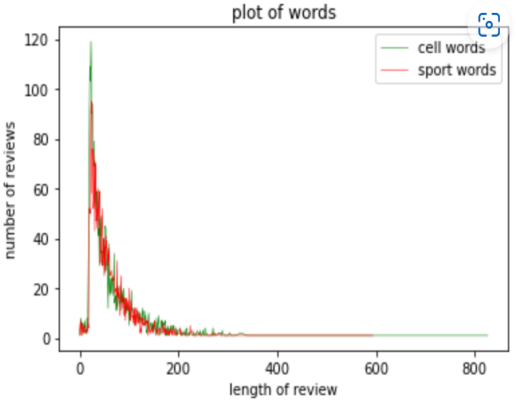


Figure 11. The distribution of two datasets by word length

From the figure, the length of most reviews by the number of tokens is less than 200. The modes of the two datasets are 24 and 25. A large number of reviews are in the range of [20,50]. In the "Cell Phone and Accessories" dataset, some reviews contain more than 600 words. The reason is that those reviews contain many sentences.

Without stemming, the 20 most frequent tokens in the "Cell Phone and Accessories" dataset and the corresponding figure are shown in Figure 12,13.

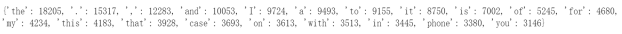


Figure 12. Most 20 frequent tokens in "Cell Phone and Accessories"

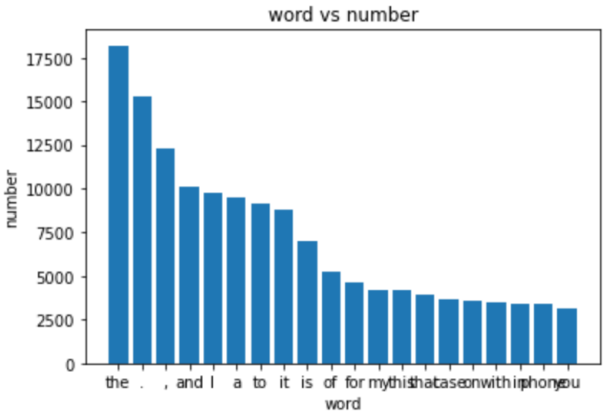


Figure 13. Histogram of most 20 frequent tokens in "Cell Phone and Accessories"

Without stemming, the 20 most frequent tokens in the "Sports and Outdoors" dataset and the corresponding figure are shown in Figures 14,15.

{'.': 12833, 'the': 12616, ',': 9622, 'a': 8294, 'and': 8174, 'I': 7908, 'to': 7511, 'it': 6637, 'is': 4964, 'of': 4341, 'for': 3907, 'th
is': 3348, 'in': 3132, 'on': 2917, 'my': 2667, 'you': 2635, 'that': 2634, 'with': 2614, 'have': 2220, 'but': 2184}

Figure 14. Most 20 frequent tokens in "Sports and Outdoors"

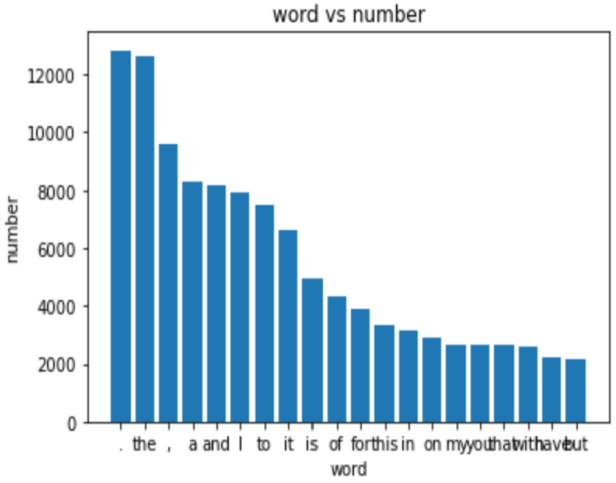


Figure 15. Histogram of most 20 frequent tokens in "Sports and Outdoors"

From the figure. We can see that some symbols occur many times. After moving some symbols, the 20 most frequent words and corresponding figures in the "Cell Phone and Accessories" dataset is shown in Figure 16,17. The 20 most frequent words and corresponding figures in the "Sports and Outdoors" dataset are shown in Figures 18,19.

{'the': 18205, 'and': 10053, 'I': 9724, 'a': 9493, 'to': 9155, 'it': 8750, 'is': 7002, 'of': 5245, 'for': 4680, 'my': 4234, 'this': 4183,
'that': 3928, 'case': 3693, 'on': 3613, 'with': 3513, 'in': 3445, 'phone': 3380, 'you': 3146, 'but': 2684, 'have': 2635}

Figure 16. Most 20 frequent tokens in "Cell Phone and Accessories" after removing symbols

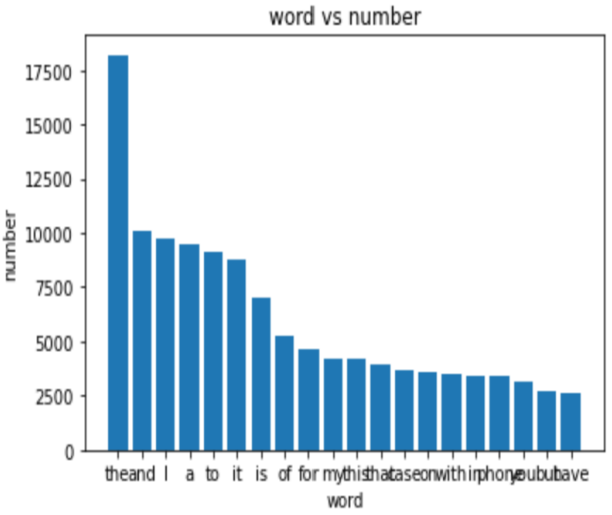


Figure 17. Histogram of most 20 frequent tokens in "Cell Phone and Accessories" after removing symbols

{'the': 12616, 'a': 8294, 'and': 8174, 'I': 7908, 'to': 7511, 'it': 6637, 'is': 4964, 'of': 4341, 'for': 3907, 'this': 3348, 'in': 3132,
'on': 2917, 'my': 2667, 'you': 2635, 'that': 2634, 'with': 2614, 'have': 2220, 'but': 2184, "n't": 1780, 'not': 1764}

Figure 18. Most 20 frequent tokens in "Sports and Outdoors" after removing symbols

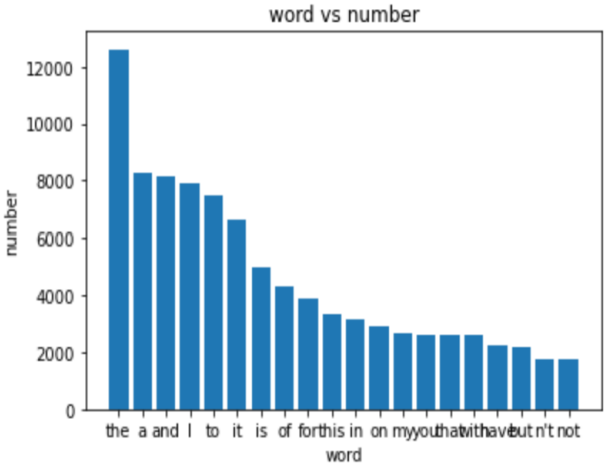


Figure 19. Histogram of most 20 frequent tokens in "Sports and Outdoors" after removing symbols

After applying stemming, before moving symbols, the 20 most frequent stems and the corresponding figures are shown in Figures 20, 21, 22, and 23.

{'the': 20429, '.': 15377, ',': 12283, 'it': 11285, 'i': 10277, 'and': 10260, 'a': 9671, 'to': 9218, 'is': 7050, 'of': 5272, 'this': 502
3, 'for': 4840, 'my': 4506, 'case': 4362, 'that': 4106, 'phone': 3750, 'on': 3678, 'with': 3632, 'in': 3581, 'you': 3401}

Figure 20. Most 20 frequent stems in "Cell Phone and Accessories"

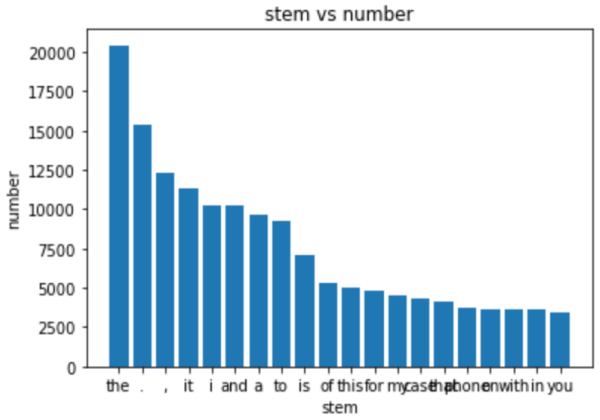


Figure 21. Histogram of most 20 frequent stems in "Cell Phone and Accessories"

{'the': 14124, '.': 12875, ',': 9623, 'a': 8471, 'it': 8395, 'and': 8321, 'i': 8255, 'to': 7562, 'is': 5018, 'of': 4375, 'for': 4041, 'this': 4033, 'in': 3252, 'on': 2974, 'you': 2879, 'my': 2866, 'that': 2752, 'with': 2671, 'but': 2409, 'have': 2392}

Figure 22. Histogram of most 20 frequent stems in "Sports and Outdoors"

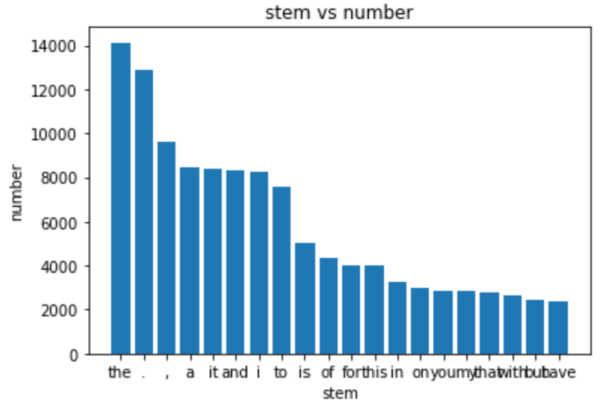


Figure 23. Histogram of most 20 frequent stems in "Sports and Outdoors"

By visualizing the result, the most frequent words and stems are quite similar, which are DET, PRONOUN, PREP, and other closed-class words. Those words cannot indicate the characters of the datasets. The words or stems that may show the characters of datasets are NOUN, VERB, and other open-class words. Thus, we must remove stop words from finding meaningful words in each dataset. A large number of tokens or stems are the least frequent words. About 5% of tokens and 10% of stems occurs more than 100 times.

After moving symbols and stop words, the results are shown in Figures 24,25,26,27.

{'case': 4362, 'phone': 3750, 'use': 2307, 'one': 1879, 'like': 1739, 'charg': 1710, 'screen': 1673, 'veri': 1669, 'iphon': 1545, 'protect': 1349, 'batteri': 1273, 'great': 1267, 'get': 1266, 'work': 1258, 'look': 1245, 'good': 1209, 'would': 1198, 'doe': 1175, 'fit': 1133, 'protector': 1019}

Figure 24. Most 20 frequent stems in "Cell Phone and Accessories" after removing symbols and stop words

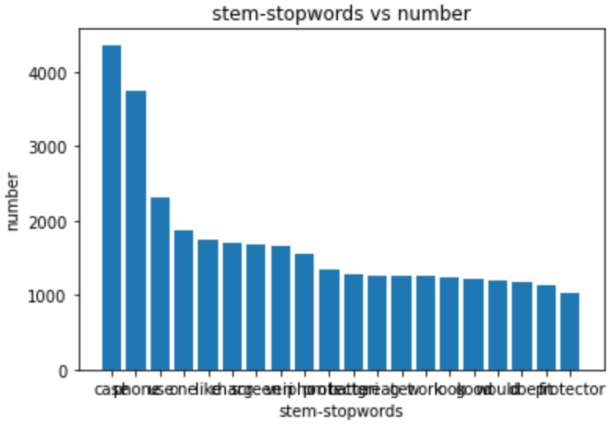


Figure 25. Histogram of most 20 frequent stems in "Cell Phone and Accessories" after removing symbols and stop words

{'use': 1955, 'one': 1404, 'veri': 1283, 'work': 1130, 'like': 1117, 'great': 1014, 'would': 1000, 'well': 999, 'get': 952, 'good': 928, 'light': 758, 'littl': 737, 'fit': 733, 'need': 679, 'knife': 678, 'time': 641, 'doe': 603, 'onli': 597, 'look': 590, 'price': 582}

Figure 26. Histogram of most 20 frequent tokens in "Sports and Outdoors" after removing symbols and stop words

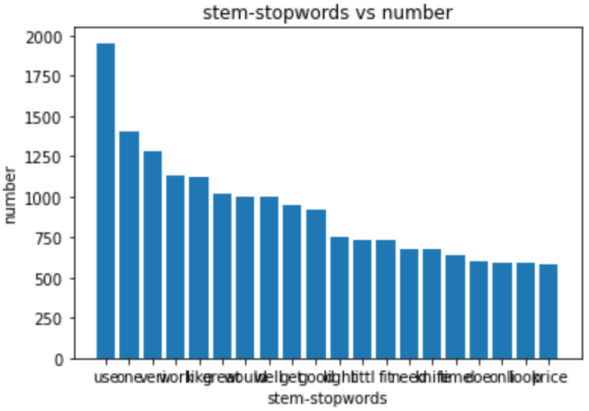


Figure 27. Histogram of most 20 frequent tokens in "Sports and Outdoors" after removing symbols and stop words

After applying the stemming, the number of word types decreases. The result is shown in Table 1:

Number of times each token appears	Words(tokens)	Words after stemming	Words after stemming first and then removing stop list words and symbols
Cell phone & Accessories	20984	14731	14603
Sports & Outdoors	20491	13914	13782

Table 1. The number of times each token appears

The symbols are selected as Figure 28:

```
stoplistsymbol =[',','.','...','?','!','(',')','&',';','``','n\'t','\'s']
```

Figure 28. the list of removing symbols

By visualizing the result, after applying stemming and removing stop list words and symbols, some meaningful words occur in the most frequent words list. In the "Cell Phone & Accessories" dataset, there are stems such as "phone", "charg", "screen", "iphon", "batteri" and "protector". Those tokens may indicate the type of the dataset.

In the "Sports & Outdoors" dataset, the result is not that obvious. There are some stems such as "light" and "knife". Those tokens may be relevant to the dataset topic.

Indicative words:

The method to find the indicative words is calculating the relative entropy of words. The relative entropy is defined in Figure 29:

$$P(w|D_1) \times log\left(\frac{P(w|D_1)}{P(w|D_2)}\right).$$

Figure 29. relative entropy equation

Many words occur in only one dataset which causes the denominator equal to zero. Therefore, all numbers of occurrences of words are added 1. The frequency of words is calculated correspondingly.

The result of the top-10 most indicative words and stems in each of the two datasets are listed below:

Top-10 words:

"Cell Phone and Accessories":

Phone, case, screen, iPhone, protector, charge, the, charger, battery, protection (before removing stop-list word)

Phone, case, screen, iPhone, protector, charge, charger, battery, protection, USB (after removing stop-list word)

"Sports and Outdoors":

knife, stove, bike, scope, recoil, rifle, blade, fork, a, gun (before removing stop-list word)

knife, stove, bike, scope, recoil, rifle, blade, fork, gun, spork (after removing stop-list word)

Top-10 stems:

"Cell Phone and Accessories":

Phone, case, iphon, screen, protector, charg, protect, charger, the, batteri (before removing stop-list word)

Phone, case, iphon, screen, protector, charg, protect, charger, batteri, devic (after removing stop-list word)

"Sports and Outdoors":

Stove, knife, bike, scope, rifl, spork, mount, recoil, blade, gun

The indicative words and stems are closely related to the topics of the two datasets.

B. Development of a Simple Search Engine

- Implementation:

In this task, we use Lucene to build a simple search engine. We use the same structure of the search engine as the sample code we download from NTU learn, but we also make our modifications to make it able to do the search task on our dataset.
We design the indexer part first. Before we did the indexing, we did some pre-processing on our records. Firstly, we do the parsing/linguistic processing (pre-processing) on the dataset; the details of the pre-processing will be discussed in the discussion section. Then we choose three fields to do the indexing, which are: "review text," " reviewerID," and "asin." The reason for choosing these three fields will be discussed in the next section of this report.
After the documents are indexed, we design the search engine. As we index for tress fields, we all support three searching models for each document field, i.e., the user can choose to search in "review text," "reviewer," and "as in" before they input the query. After the search finishes, we print the output of our search result.
Finally, our search engine can support free text keyword queries to top N results (N is pre-stetted by the user). It will print each result document's rank, scores, doc ID, and doc content. Meanwhile, it also supports "asin" searching so that users can search for reviews specific to a product by inputting the asin value of this product to our search engine in "asin" searching mode.

- Discussion:

D1: What field(s) shall be indexed and searchable?
As discussed in the previous section, we indexed three fields: "reviewText," "reviewerID," and "asin." We decided to make these three fields indexed and searchable because we want to make our users able to search for the reviews of a particular product (by asin). And if they want to find what product has the attribute they invested in, they can search for a review text. Finally, we keep "reviewerID" because if the merchant wants to find a potential customer's interest, they can search reviewerID to know what this potential customer likes most.

D2: Detail our choice of parsing/linguistic processing on the words/terms in the chosen fields.
Unlike the data parsing/linguistic processing (pre-processing) in the first part of this assignment, we only do the lowercase and punctuation removal for the search engine indexer. The reason why we choose not to do the word stemming here is that we think that the word stemming will lead to false positive matching when we handle the query.

D3: Discuss our findings on the indexing time.
Firstly, we collect the time needed to index every 10% of the documents. The time for indexing each 10% of documents is 449ms, 79ms, 61ms, 82ms, 68ms, 43ms, 73ms, 94ms, 85ms, and 67ms. To make them more accessible for observation, we plot them out in Figure 30

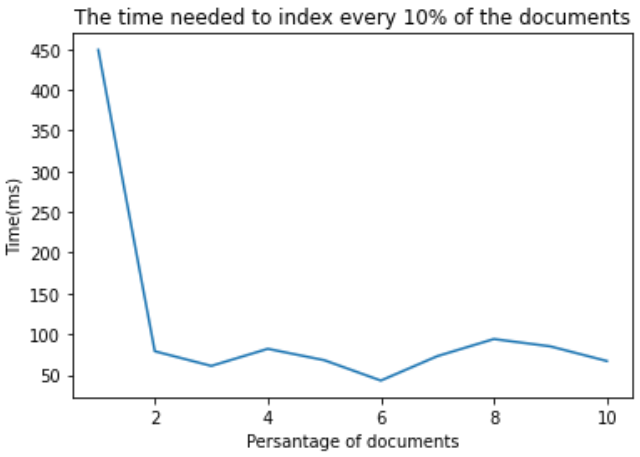


Figure 30. TIe consumption for indexing each 10% of document

From this figure, we can find that the indexer spends the most time in the first 20% of documents, and the time consumption decreases rapidly after the first 20% of documents are indexed. We think the time consumption for the first 20% is that Lucene is using the FST (Finite State Transducers) index. The structure of the FST indexer is shown in Figure 31

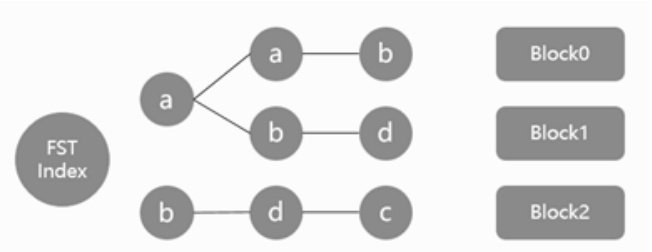


Figure 31. Structure of FST

As the new world frequently appears in the first 20% of documents, the FST algorithm needs more time to handle these new words. As a result, the time consumption of indexing the first 20% of data will become higher.

D4: Whether the results returned by the search engine are as expected and the time taken to process a query.
We try the queries for "good," "looks good," and "not good" in the review search, as shown in Figures 32, 33, 34:
Query: "good" finished 20ms

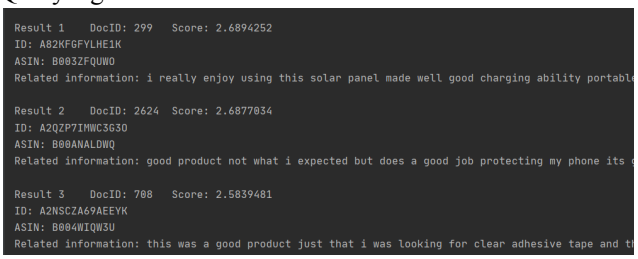


Figure 32. Search result 1

Query: "very good" finished in 26ms

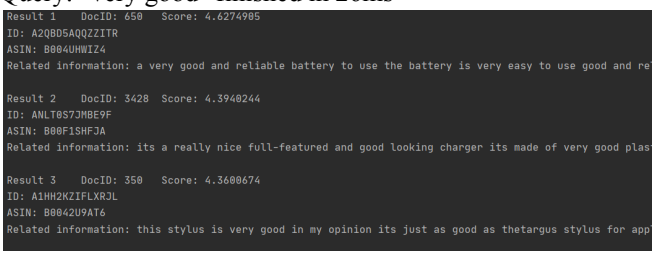


Figure 33. Search result 2

Query: "looks good" 28ms

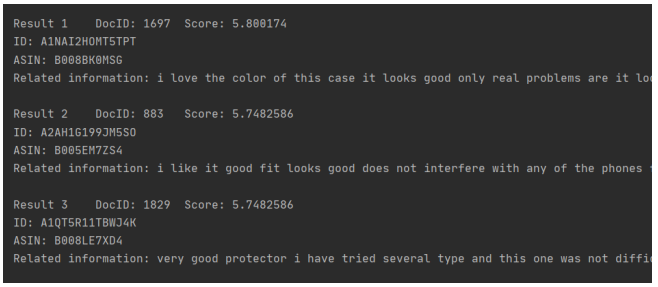


Figure 34. Search result 3

From this result, we can find that our search engine works well. For both single keyword and phrase queries, it gives us the result, including the query we want. From time rescoring, we find that searching for a single keyword query is faster than phrase queries.

C. *Development of a Review Summarizer*

The review summarizer is developed based on the *reviewText* column of the dataset. For this task, I select all the reviews for two products, clean the text, and summarize the indicative keywords using IDF (Inverse Document Frequency).

In previous work, we have randomly selected 200 products from each dataset. The *asin* column is the indicator of a unique product. Hence, I count on the number of individual reviews for each asin value and select the most frequently reviewed product from each dataset.

```
In [3]: df1_asinno = df1.groupby('asin').size().sort_values(ascending=False)
        df1_asinno.head()

Out[3]: asin
        B009SD2F4O    124
        B00AFSK28M    104
        B00E2TT8W6    100
        B004I58ZVY     87
        B00E2YI052     82
        dtype: int64

In [4]: df2_asinno = df2.groupby('asin').size().sort_values(ascending=False)
        df2_asinno.head()

Out[4]: asin
        B00551HARU    130
        B00419J4X0    105
        B001BR1O74     96
        B000KIMTCY     87
        B00CULFR4Q     71
        dtype: int64
```

Figure 35. The "asin" of the most frequent reviewed product from each dataset.

Figure 36. The sample of selected data frames of the most frequent reviewed product from each dataset.

Then the text is cleaned by converting all text to lowercase, removing the punctuation and stop words. After this, I found that there were some keywords containing numbers that are not what I want. Hence, I add in a function to delete all numbers from the text.

```
In [7]: def lowercase(text):
            text = [word.lower() for word in text.split()]
            return " ".join(text)
        def remove_punctuation(text):
            translator = str.maketrans('', '', string.punctuation)
            return text.translate(translator)
        def remove_num(text):
            remove_digits = str.maketrans('', '', string.digits)
            return text.translate(remove_digits)
        def remove_stopwords(text):
            stop_words = set(stopwords.words('english'))
            word_tokens = word_tokenize(text)
            filtered_sentence = []
            for word in word_tokens:
                if word not in stop_words:
                    filtered_sentence.append(word)
            return ' '.join(filtered_sentence)
        def clean_text(text):
            text = lowercase(text)
            text = remove_punctuation(text)
            text = remove_num(text)
            text = remove_stopwords(text)
            return text
```

Figure 37. The function of cleaning text.

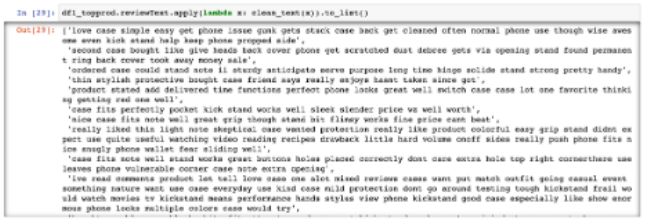


Figure 38. The cleaned text of reviewText.

Then keywords are generated using IDF. From sklearn modules, we import TfidfTransformer and CountVectorizer.

Figure 39. The keywords generated of product1

However, from the results, we can see there are some keywords like “normal” and “normally” having similar meaning. Stemming and lemmatization are then added in the text cleaning process.

```
In [8]: def lowercase(text):
            text = [word.lower() for word in text.split()]
            return " ".join(text)
        def remove_punctuation(text):
            translator = str.maketrans('', '', string.punctuation)
            return text.translate(translator)
        def remove_num(text):
            remove_digits = str.maketrans('', '', string.digits)
            return text.translate(remove_digits)

        def remove_stopwords_and_lemmatization(text):
            stop_words = set(stopwords.words('english'))
            word_tokens = word_tokenize(text)
            lemmatizer = WordNetLemmatizer()
            ps = PorterStemmer()
            filtered_sentence = []
            for word in word_tokens:
                if word not in stop_words:
                    temp = ps.stem(word)
                    temp = lemmatizer.lemmatize(temp)
                    filtered_sentence.append(temp)
            return ' '.join(filtered_sentence)
        def clean_text(text):
            text = lowercase(text)
            text = remove_punctuation(text)
            text = remove_num(text)
            text = remove_stopwords_and_lemmatization(text)
            return text
```

Figure 40. The new function of cleaning text.

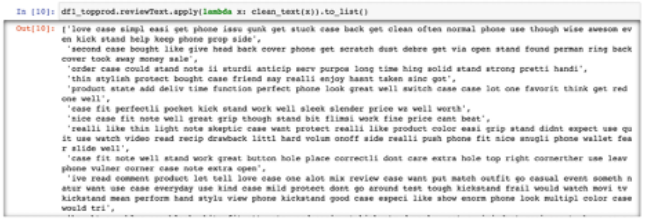


Figure 41. The newly cleaned text of reviewText.

Finally, the keywords of each product are generated by using the IDF function.

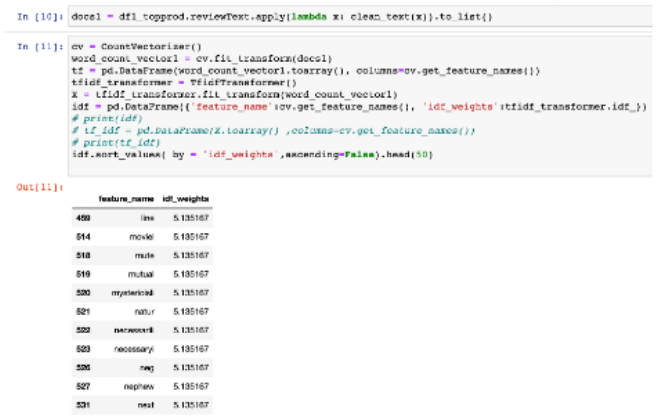


Figure 42. The new keywords generated of product1

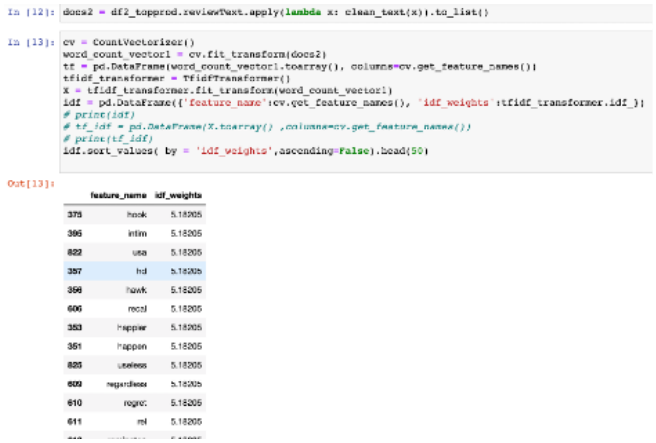


Figure 43. The new keywords generated of product2

IDF is measuring the distinctiveness of the word. Hence, we use IDF to generate the indicative keywords as the summary.

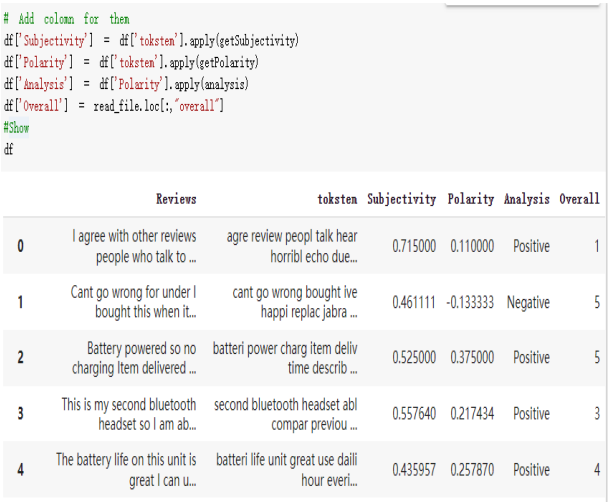
The limitation of this approach can be seen from the result. Some of the keywords like “necessarili" and “necessaryi" have some type errors. So for future works, we can put in more effort to add in some features like correcting features when processing the text.

D. *Application*

Libraries:used: Selenium, BeautifulSoup, Pandas, matplotlib, nltk, requests, sklearn, spacy, torch and torchtext //textBlob

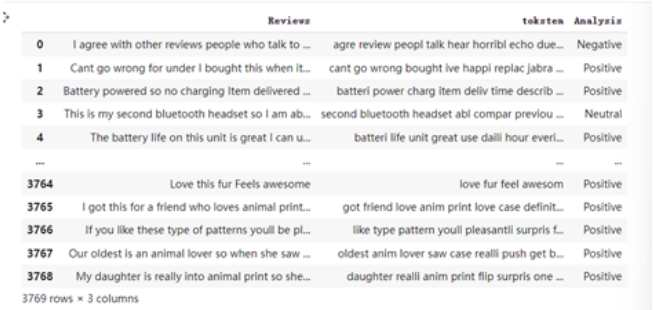
We decided to do the sentiment word analysis for the application task, and CNN was chosen as the training model. Sentiment Analysis is for showing people’s opinions which are extracted from the text. From the given dataset, we can directly get the sentiment based on the ‘overall’ branch, which is the mark from clients that shows their feelings about the product. Here we split the markings into three sets: when the overall mark is 1 and 3, we define them as labeled ‘Negative’, when it is 4 and 5, we suppose they are with label ‘Positive’, also when it is number 3, we give the label ‘Neutral’.

Obviously, as a part of NLP, the first step should be pre-processing the dataset. To go through it, we took the basic steps, firstly removing everything except the words(alphabet) and ‘ ’, then we did tokenization, stemming, and removed all stopwords for words that were all lowercase. For comparison, we put an additional column into the data frame. The next step was to add the sentiment Analysis column. As mentioned before, we got this data directly from the dataset file. We also tried to use subjectivity and polarity to get the analysis, but the results were not satisfactory.



	Reviews	toksten	Subjectivity	Polarity	Analysis	Overall
0	I agree with other reviews people who talk to ...	agre review peopl talk hear horribl echo due...	0.715000	0.110000	Positive	1
1	Cant go wrong for under I bought this when it...	cant go wrong bought ive happi replac jabra ...	0.461111	-0.133333	Negative	5
2	Battery powered so no charging Item delivered ...	batteri power charg item deliv time describ ...	0.525000	0.375000	Positive	5
3	This is my second bluetooth headset so I am ab...	second bluetooth headset abl compar previou ...	0.557640	0.217434	Positive	3
4	The battery life on this unit is great I can u...	batteri life unit great use dailli hour everi...	0.435957	0.257870	Positive	4

Figure 44. The image shows the analysis with the values of subjectivity and polarity. For the first row, it shows the polarity id over zero with analysis ‘Positive’, but actually, the mark is 1, which should be analyzed as ‘Negative’, Row 2, and row 4 also analyzed wrong with these two values.



	Reviews	toksten	Analysis
0	I agree with other reviews people who talk to ...	agre review peopl talk hear horribl echo due...	Negative
1	Cant go wrong for under I bought this when it...	cant go wrong bought ive happi replac jabra ...	Positive
2	Battery powered so no charging Item delivered ...	batteri power charg item deliv time describ ...	Positive
3	This is my second bluetooth headset so I am ab...	second bluetooth headset abl compar previou ...	Neutral
4	The battery life on this unit is great I can u...	batteri life unit great use dailli hour everi...	Positive
...	...	...	...
3764	Love this fur Feels awesome	love fur feel awesom	Positive
3765	I got this for a friend who loves animal print...	got friend love anim print love case definit...	Positive
3766	If you like these type of patterns youll be pl...	like type pattern youll pleasantli surpris f...	Positive
3767	Our oldest is an animal lover so when she saw ...	oldest anim lover saw case realli push get b...	Positive
3768	My daughter is really into animal print so she...	daughter realli anim print flip surpris one ...	Positive

3769 rows × 3 columns

Figure 45. Otherwise, the right image shows the correct analysis with each review.

For splitting the training and testing set, the coefficient used was 0.2, then further spitted out a validation set also with the same number. So in total, we have 2412 training data, 603 test data, and 754 test data. This step also did the conversion from the *pandas* data frame to the torch dataset. The fields of this data frame contained text and labels. The contents inside, which are the train, test, and validation test then loaded into *totchtext dataframe* data. One printed set showed as:

	Reviews	toksten	Analysis
0	I agree with other reviews people who talk to ...	agre review peopl talk hear horribl echo due...	Negative
1	Cant go wrong for under I bought this when it...	cant go wrong bought ive happi replac jabra ...	Positive
2	Battery powered so no charging Item delivered ...	batteri power charg item deliv time describ ...	Positive
3	This is my second bluetooth headset so I am ab...	second bluetooth headset abl compar previou ...	Neutral
4	The battery life on this unit is great I can u...	batteri life unit great use daili hour everi...	Positive
...	...	...	...
3764	Love this fur Feels awesome	love fur feel awesom	Positive
3765	I got this for a friend who loves animal print...	got friend love anim print love case definit...	Positive
3766	If you like these type of patterns youll be pl...	like type pattern youll pleasantli surpris f...	Positive
3767	Our oldest is an animal lover so when she saw ...	oldest anim lover saw case realli push get b...	Positive
3768	My daughter is really into animal print so she...	daughter realli anim print flip surpris one ...	Positive

3769 rows × 3 columns

Figure 46. Example of precessed data

{'text': [' ', 'slim', 'armor', 'simpl', 'anti', 'shock', 'design', 'come', 'three', 'differ', 'color', 'button', 'cover', 'orang', 'black', 'white', 'could', 'complet', 'kit', 'ad', 'clear', 'screen', 'protector', 'nice', 'tight', 'iphon', 'black', 'bodi', 'sleev', 'chang', 'come', 'mani', 'color', 'mine', 'gray', 'does', 'nt', 'show', 'dirt', 'scratch', 'hardli', 'notic', 'like'], 'label': 'Positive'}

Then maybe we can suppose that the label 'Positive' comes with the words 'clear', 'happy' and maybe more. Moreover, after building the vocabulary and creating the iterator for loading data, we turned it into Convolutional Neural Network (CNN).

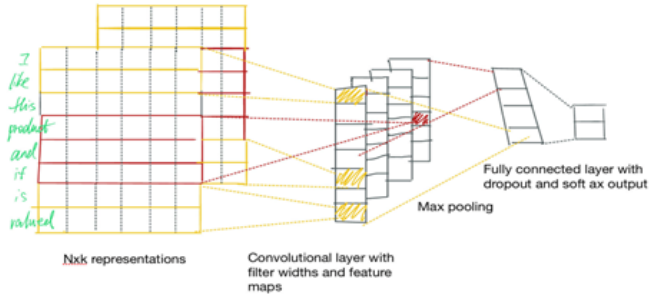


Figure 47. A general CNN model for this task

Since all words in the review are just 1 dimensional, we used word embeddings, which helps words to perform like 2D. The filter (or kernel) used here is [3,4,5], which could cover the whole important words. In CNN, we always choose the pooling with the maximum value on the layers, which is seen as the most important part of sentiment analysis (also the most important n-gram). The output size is calculated by:

$$H_{out} = \left[\frac{(H_{in} + 2*P - H_{kernel})}{S}\right] + 1$$

$$Wout = \left[\frac{(W_{in} + 2*P - w_{kernel})}{S}\right] + 1$$

$$CH_{out} = K$$

By going through training and evaluating, we got the final loss and accuracy of training, validation, and testing.

```
- Epoch: 01 - Train Loss: 0.620 - Train Acc: 71.59% - Val. Loss: 0.584 - Val. Acc: 76.13% -
- Epoch: 02 - Train Loss: 0.447 - Train Acc: 78.57% - Val. Loss: 0.506 - Val. Acc: 75.10% -
- Epoch: 03 - Train Loss: 0.306 - Train Acc: 77.92% - Val. Loss: 0.447 - Val. Acc: 74.07% -
- Epoch: 04 - Train Loss: 0.156 - Train Acc: 77.97% - Val. Loss: 0.485 - Val. Acc: 74.73% -
- Epoch: 05 - Train Loss: -0.013 - Train Acc: 77.96% - Val. Loss: 0.454 - Val. Acc: 74.35% -

- Test Loss: 0.454 - Test Acc: 75.24% -
```

Figure 48.

From this figure, we can find that the model is overfitting because training data accuracy is higher than validation data. To solve this problem, we decided to downgrade our CNN model to a KNN model to avoid overfitting.

In this step, we change the stemming in pre-processing to lemmatization and add one more step in word pre-processing, which is TF-IDF. We use the TF-IDS model from the sk-learn package and remain all other pre-processing steps the same. With the help of TF-IDF, we can get the reviews in vector form, which is suitable for classification by a KNN model.

To train the KNN model, we tried different k values, i.e., k=3,5,7,9, and got the following result.

K value	Train Acc	Test Acc
3	83.90%	76.53%
5	81.85%	79.05%
7	80.58%	80.37%
9	79.99%	79.98%

Table 2. Accuracy of different K value

From this result, we can find that when k=3,5 the KNN model is overfitting, because k value is too small, and the model can only notice the very nearby points. As a result, in this case, we can notice that the training accuracy is higher than the testing accuracy. When k=7, the model is well trained, as the training accuracy is near testing accuracy, and both of them are quite high. However, when we increase the k value to 9, the model becomes underfitting: although the training accuracy is near testing accuracy, both of them become lower than the case when k=7. The reason for this observation is that for the KNN model, k is like a regularization term; a higher k means a model more focused on the overall case rather than the nearby sample.

Besides KNN, we also tried using the SVM model to do this task. However, unlike KNN, SVM without OvO or OvR algorithm only supports binary classification. As both OvO and OvR are time-consuming, we decided to use the original SVM to do this task. As a result, in this case, we decided to ignore the 'Neutral' class and only do the binary classification for the other two classes. We use the SGD Classifier form SKlearn, and do the Scalar Normalization of TF-IDF vectors before we feed them into the SVM. Finally, the trained SVM results in a training accuracy of 97.83% and a testing accuracy of 89.41%. Although, in this case, the training accuracy is higher than the testing accuracy, and the model seems slightly overfitting, the testing accuracy is already high enough.

We want to try the LSTM model with the following structure for future work. However, this model never converges while training on our dataset, even with our best effort.

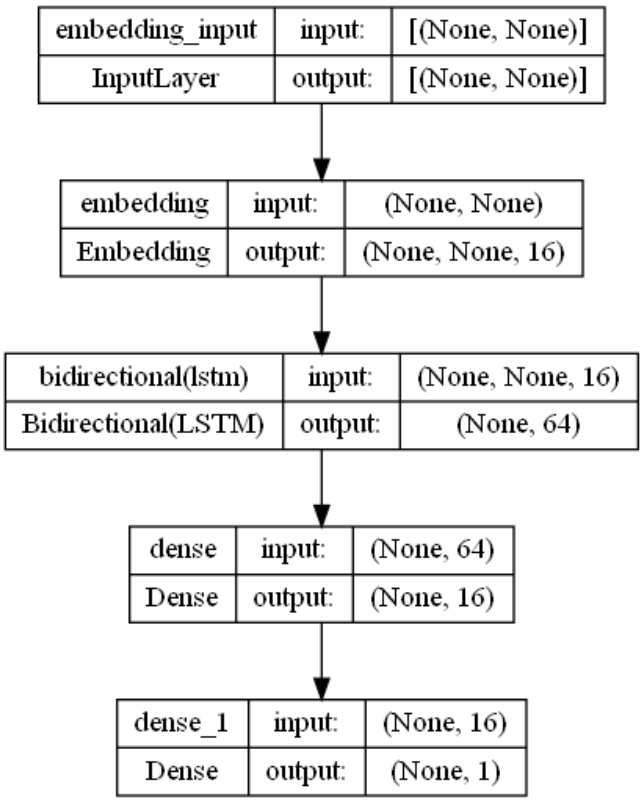


Figure 49. The structure of LSTM model

IV. Conclusion

In conclusion, for this assignment, we chose 200 datasets and pre-processed the data for further usage in main tasks. We also complete all four tasks, including Dataset Analysis, the Development of a Simple Search Engine, the Development of a Review Summarizer, and the Development of an Application. It is a great learning and exploring experience for us all regarding Text and Data Management and Processing topics.

V. Contribution

Mao Jiangtian did part one for Data Analysis, Li Xuemeng worked on the coding of Part 2 of the development of the Search Engine, and Huang Yunfan wrote the report of this part. Lu Pai worked on Part 3, which is the Summarizer of Reviews. In the end, Li Xuemeng and Huang Yunfan did the Application part jointly.

Although the allocated team size was 5, we are sorry to say that we couldn't get in touch with one of our original group members, Han Yaodong, by all ways and means since he/she did not reply to any email or Teams Message until the last week before the due date. Hence this assignment was completed by the remaining 4 group members.

References

CNN model:
https://towardsdatascience.com/cnn-sentiment-analysis-9b1771e7cdd6