



## 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:

[illegible]

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:

[illegible]

Figure 2. Text example for JAVA JSON parse

### 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:

a sample of cell phone:  
I think my ear is pretty normal in shape. But this headset won't stay on unless I stabilize the too thin ear holder with the temple of my glasses. If I wear contacts, the headset slips off in less than two minutes. Aside from that, the sound quality is adequate, if a bit too soft, and people tell me that they can hear me well. It syncs easily with my bluetooth Motorola phone. It is very lightweight and compact with a well designed combination storage/charging unit. Plus I find that I don't have to charge it very often. The other negative is that there are problems with interference in noisy environments when I'm talking on the phone. On the whole, I'd say you really need to try this one on for a few minutes and physically test it before you buy it. If it fits your ear, you should buy it. It's inexpensive and does the job. But if you're like me, you need a headset with a more hefty ear holder.

a sample of sport/outdoors:  
The item was delivered on time and in good condition. This was a birthday gift for someone so I can not make a review on how it feels or how it's going to last at this time. I will check with the person that I gave it to at a later date to see how they like it and it's fee I. fit and last.

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.

Battery wouldn't hold a charge and was faulty from the beginning. I was very disappointed and would not recommend his to anyone. Total waste of money!

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.

It's inexpensive and does the job.  
[('It's', 'NNP'), ('inexpensive', 'JJ'), ('and', 'CC'), ('does', 'VBZ'), ('the', 'DT'), ('job', 'NN')]

Figure 5. Pos tagging example 1

I contacted the seller directly about the issue even though it had been over a month and they were happy to replace it for free.  
[('I', 'PRP'), ('contacted', 'VBD'), ('the', 'DT'), ('seller', 'NN'), ('directly', 'RB'), ('about', 'IN'), ('the', 'DT'), ('issue', 'NN'), ('even', 'RB'), ('though', 'IN'), ('it', 'PRP'), ('had', 'VBD'), ('been', 'VBN'), ('over', 'IN'), ('a', 'DT'), ('month', 'NN'), ('and', 'CC'), ('they', 'PRP'), ('were', 'VBD'), ('happy', 'JJ'), ('to', 'TO'), ('replace', 'VB'), ('it', 'PRP'), ('for', 'IN'), ('free', 'NN')]

Figure 6. Pos tagging example 2

This rail mount is branded Remington but actually made by B-Square (according to the packaging insert).  
[('This', 'DT'), ('rail', 'NN'), ('mount', 'NN'), ('is', 'VBZ'), ('branded', 'VBN'), ('Remington', 'NNP'), ('but', 'CC'), ('actually', 'RB'), ('made', 'VBN'), ('by', 'IN'), ('B-Square', 'NNP'), ('according', 'VBG'), ('to', 'TO'), ('the', 'DT'), ('packaging', 'NN'), ('insert', 'NN')]

Figure 7. Pos tagging example 3

Apart from this step, however, the assembly is simple and straight forward.  
[('Apart', 'RB'), ('from', 'IN'), ('this', 'DT'), ('step', 'NN'), ('however', 'RB'), ('the', 'DT'), ('assembly', 'NN'), ('is', 'VBZ'), ('simple', 'JJ'), ('and', 'CC'), ('straight', 'JJ'), ('forward', 'NN')]

Figure 8. Pos tagging example 4

I figured that going in.  
[('I', 'PRP'), ('figured', 'VBD'), ('that', 'IN'), ('going', 'VBG'), ('in', 'IN')]

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.

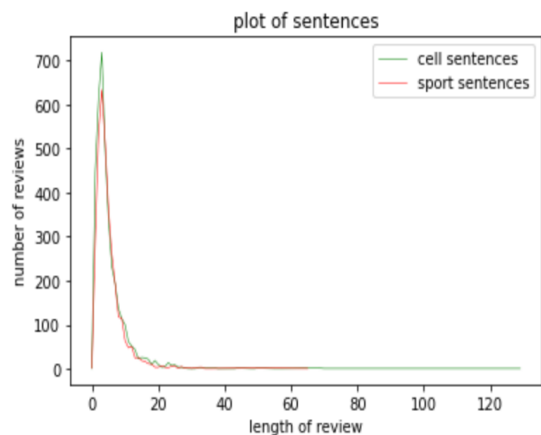


Figure 10. The distribution of two datasets by sentence length

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.

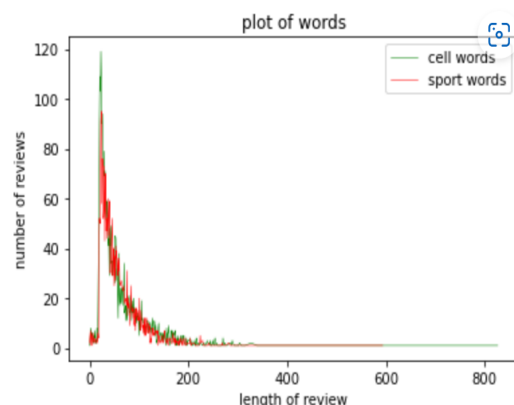


Figure 11. The distribution of two datasets by word length

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.

[('the': 18205, 'i': 15317, 'it': 12283, 'and': 10653, 'l': 9724, 'a': 9499, '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)]

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

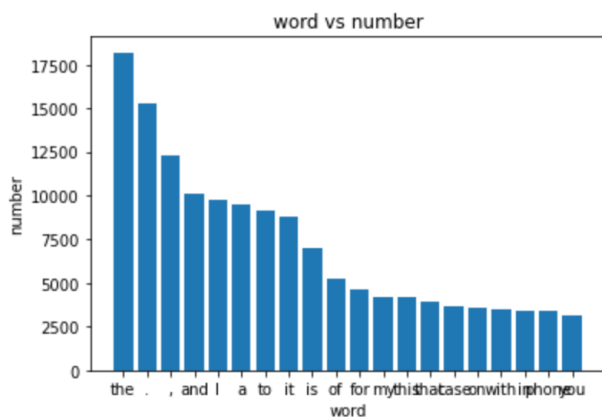


Figure 13. Histogram of most 20 frequent tokens in “Cell Phone and Accessories”

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”

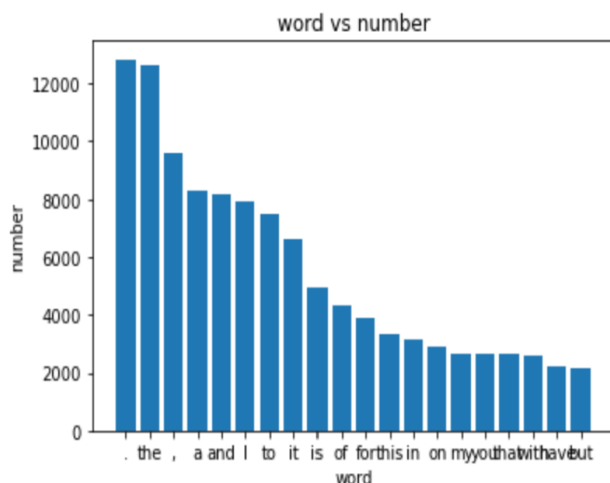


Figure 15. Histogram of most 20 frequent tokens in “Sports and Outdoors”

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.

[',': 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': 3390, 'you': 3146, 'but': 2684, 'have': 2635]

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

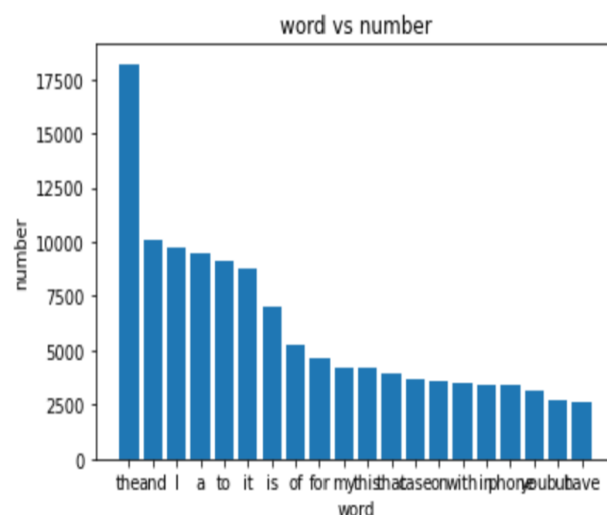


Figure 17. Histogram of most 20 frequent tokens in “Cell Phone and Accessories” after removing symbols

[',': 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

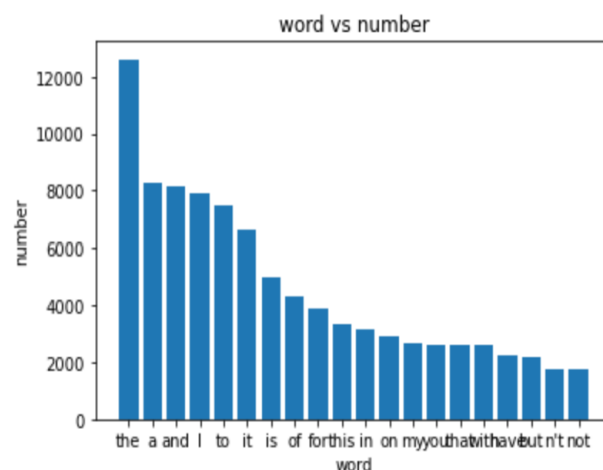


Figure 19. Histogram of most 20 frequent tokens in “Sports and Outdoors” after removing symbols

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

[',': 39429, ',': 15377, ',': 12283, 'it': 11285, 'i': 10277, 'and': 10260, 'a': 9671, 'to': 9218, 'is': 7650, 'of': 5272, 'this': 5023, 'for': 4840, 'my': 4506, 'case': 4362, 'that': 4106, 'phone': 3730, 'on': 3678, 'with': 3632, 'in': 3581, 'you': 3401]

Figure 20. Most 20 frequent stems in “Cell Phone and Accessories”

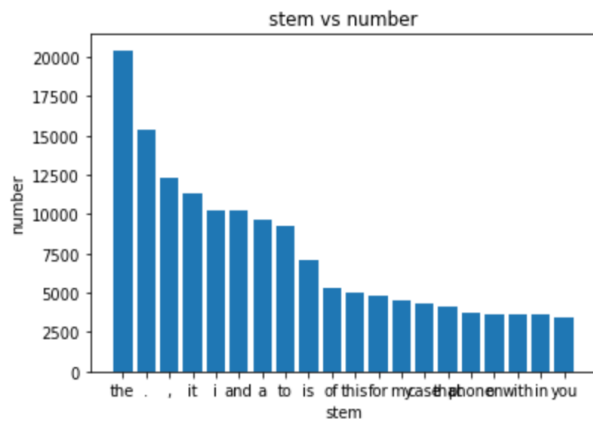


Figure 21. Histogram of most 20 frequent stems in “Cell Phone and Accessories”

[ 'the': 14124, '.': 12875, ',': 9023, 'a': 8471, 'it': 8395, 'and': 8321, 'i': 8255, 'to': 7562, 'is': 5018, 'of': 4375, 'for': 4041, 'th  
is': 4033, 'in': 3252, 'on': 2974, 'you': 2879, 'my': 2866, 'that': 2732, 'with': 2671, 'but': 2499, 'have': 2392]

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

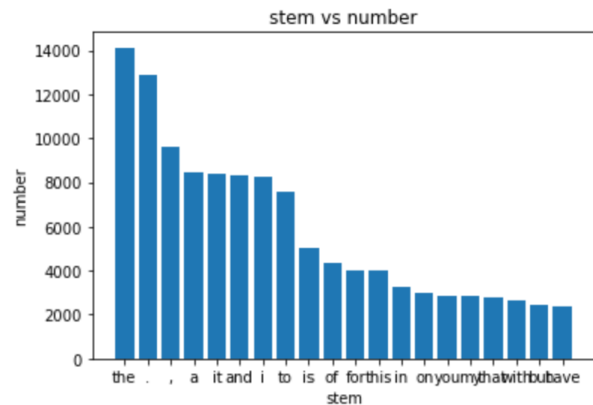


Figure 23. Histogram of most 20 frequent stems in “Sports and Outdoors”

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, 'ipho': 1545, 'prot  
e': 1349, 'batteri': 1273, 'great': 1267, 'get': 1266, 'work': 1258, 'look': 1245, 'good': 1209, 'would': 1198, 'dow': 1175, 'fit': 1133, 'protector': 1019]

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

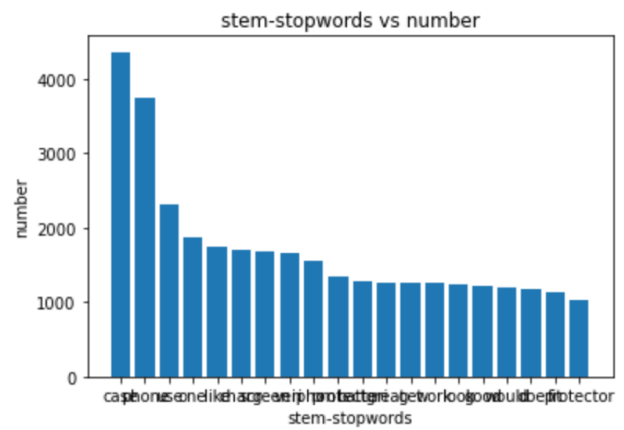


Figure 25. Histogram of most 20 frequent stems in “Cell Phone and Accessories” after removing symbols and stop words

[ '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, 'dow': 603, 'onl': 597, 'look': 590, 'price': 582]

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

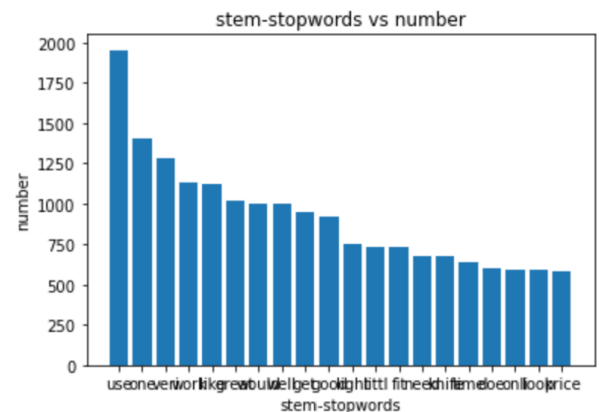


Figure 27. Histogram of most 20 frequent tokens in “Sports and Outdoors” after removing symbols and stop words

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 stemmin g	Words after stemmin g first and then removing stop list words and symbols
Cell phone & Accessorie s	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 three fields, we all support three searching models for each document field, i.e., the user can choose to search in "review text," "reviewer," and "asin" 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-set 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

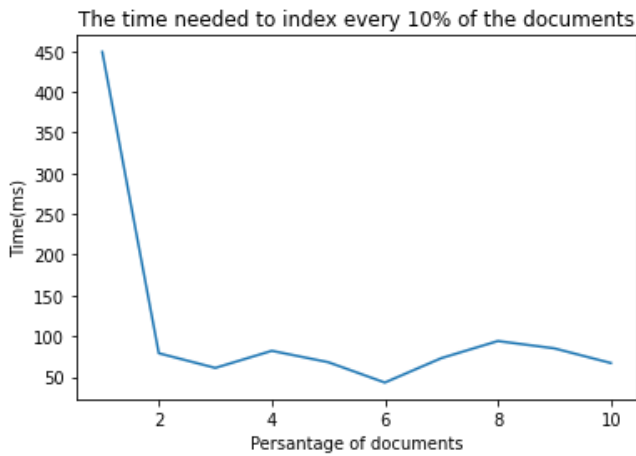


Figure 30. The consumption for indexing each 10% of document

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

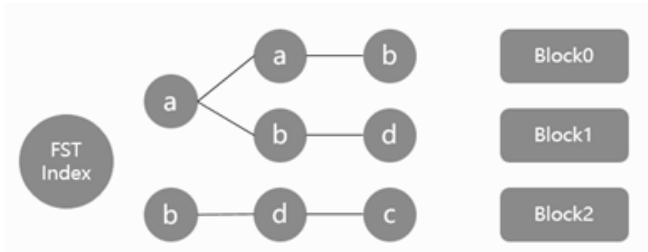


Figure 31. Structure of FST

As the new word 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

```

Result 1 DocID: 299 Score: 2.6894252
ID: A82KFGFYLHE1K
ASIN: B003ZFQWQD
Related information: i really enjoy using this solar panel made well good charging ability portabl

Result 2 DocID: 2624 Score: 2.6877834
ID: A2QZP7IMWC3G3D
ASIN: B00ANALDQW
Related information: good product not what i expected but does a good job protecting my phone its

Result 3 DocID: 788 Score: 2.5839481
ID: A2NSCZA69AEEYK
ASIN: B004W1QW3U
Related information: this was a good product just that i was looking for clear adhesive tape and t

```

Figure 32. Search result 1

Query: "very good" finished in 26ms

```

Result 1 DocID: 658 Score: 4.6274985
ID: A2QBDSAQZZITR
ASIN: B004UHMIZ4
Related information: a very good and reliable battery to use the battery is very easy to use good and re

Result 2 DocID: 3428 Score: 4.3948244
ID: ANLT6S7JMBE9F
ASIN: B00F1SHFJA
Related information: its a really nice full-featured and good looking charger its made of very good plas

Result 3 DocID: 358 Score: 4.3608674
ID: A1HH2KZIFLXRJL
ASIN: B004ZU9AT6
Related information: this stylus is very good in my opinion its just as good as the targus stylus for app

```

Figure 33. Search result 2

Query: "looks good" 28ms

```

Result 1 DocID: 1697 Score: 5.888174
ID: A1NAI2HOMTSTPT
ASIN: B008BK0MSG
Related information: i love the color of this case it looks good only real problems are it lo

Result 2 DocID: 883 Score: 5.7482586
ID: A2AH1G199JHSS0
ASIN: B005EM7ZS4
Related information: i like it good fit looks good does not interfere with any of the phones

Result 3 DocID: 1829 Score: 5.7482586
ID: A1Q75R11TBWJ4K
ASIN: B008LE7XD4
Related information: very good protector i have tried several type and this one was not diffi

```

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
B009SD2F40    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.



```
In [5]: df1_asin = df1.groupby('asin').size().sort_values(ascending=False).reset_index()[['asin']][0]
df1_topprod = df1[df1.asin == df1_asin]
df1_topprod.head()
```

```
Out[5]:
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixreviewTime	reviewTime
2039	A2Y2BVGZG9PM	B00SGDQF4D	A. Drowns "Closest of Five Samples"	[0, 0]	Love the case, it's simple and easy to get on...	5	Love it!	1370820400	08/10/2013
2039	A7UXKX03G0Z7P	B00SGDQF4D	Adrian	[0, 0]	This is the second case I bought for this and...	4	It's a great case but be warned!	1372354800	06/26/2013
2039	A2BFL3G11YNA	B00SGDQF4D	Aldabawan	[0, 0]	I ordered this case so I could stand up my Not...	5	Outstand'ing	1378870400	09/19/2013
2031	A1R4H02ZYFW12	B00SGDQF4D	Amanda B.	[0, 0]	Very thin, stylish and protective. Bought this...	4	Nice case	1368532400	04/21/2013
2032	A20R0PVEE1JRS	B00SGDQF4D	Amazon Customer "dwd"	[0, 0]	Product was as stated in the add. Delivered on...	5	came as ordered	1360972800	02/16/2013

```
In [6]: df2_asin = df2.groupby('asin').size().sort_values(ascending=False).reset_index()[['asin']][0]
df2_topprod = df2[df2.asin == df2_asin]
df2_topprod.head()
```

```
Out[6]:
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixreviewTime	reviewTime
2044	A6D9GK1GVYKUB	B00SG1HARU	NaN	[0, 0]	Nice way to have extra few shells, I like it.	4	Four Stars	1405488800	07/18/2014
2044	A27VXG3RNCZ7E	B00SG1HARU	10behavior	[0, 0]	It does what the ad says, it holds everything nice...	5	Does what the ad says	1357204200	01/11/2013
2046	A1H1N0E3ZDWAJ	B00SG1HARU	AC	[0, 0]	This handcase was exactly all I knew it was go...	5	perfect	1358888400	01/01/2013
2047	A304YFVGE0WAF	B00SG1HARU	Adam L. Rogers	[0, 0]	For this price I was not expecting much quality...	5	Random it up!!	1380589600	10/1/2013
2048	A1W5D9G2DMH01	B00SG1HARU	AJ Turner	[0, 0]	This shogun handcase is by far one of the be...	5	Must have for target shooters, hunters, and th...	1361864000	02/24/2013

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.

```
In [29]: df1_topprod.reviewText.apply(lambda x: clean_text(x)).to_list()
```

```
Out[29]:
```

['love case simple easy get phone laser gun get stuck case back get cleaned often normal phone use though wise awes  
ome even kick stand help keep phone propped side',  
'second case bought like give heads back cover phone get scratched dust debris gets via opening stand found permanen  
t ring back cover took easy money sale',  
'ordered case could stand note ii sturdy anticipate serve purpose long time hinge solid stand strong pretty handy',  
'this stylish protective brought case friend says really enjoys hand taken since got',  
'product stated add delivered time functions perform phone looks great well switch case case lot one favorite thinki  
ng getting real one well',  
'case fits perfectly pocket kick stand works well sleek slender price vs well worth',  
'also case fits note well great grip though stand bit flimsy works fine price cant beat',  
'really liked this light note skeptical case wanted protection really like product colorful easy grip stand didnt ex  
pect use quite useful watching video reading recipes connect little hard volume on/off slide really push phone fits a  
fine snugly phone wallet fear sliding well',  
'case fits note well stand works great buttons holes placed correctly dont case extra hole top right cornerthere use  
leaves phone vulnerable corner case note extra opening',  
'ive read comments product let tell love case one alot mix reviews came want put match outfit going casual event smethn s  
atur want use case everyday use kind case mild protective dont go around testing tough kickstand frail would watch movi  
uld watch movies tv kickstand means performance hands styles view phone kickstand good case especially like show ever  
more phone looks multiple colors case would try',

Figure 38. The cleaned text of reviewText.

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

```
In [12]: cv = CountVectorizer()
word_count_vector = cv.fit_transform(docs)
tf = pd.DataFrame(word_count_vector.toarray(), columns=cv.get_feature_names())
tfidf_transformer = TfidfTransformer()
X = tfidf_transformer.fit_transform(word_count_vector)
idf = pd.DataFrame(tfidf_transformer.idf_, columns=cv.get_feature_names())
# print(idf)
# tf_idf = pd.DataFrame(X.toarray(), columns=cv.get_feature_names())
# print(tf_idf)
idf.sort_values(by = 'idf_weights', ascending=False).head(50)
```

```
Out[12]:
```

	feature_name	idf_weights
0	absolute	5.150167
437	nicely	5.150167
639	nicity	5.150167
640	nonstopcom	5.150167
641	normal	5.150167
642	normally	5.150167
644	noticed	5.150167
645	obviously	5.150167
646	occasionally	5.150167
649	ultra	5.150167
650	okay	5.150167
652	ok	5.150167
655	cruff	5.150167
656	operation	5.150167
662	opposite	5.150167

Figure 39. The keywords generated of product I

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.

```
In [10]: df1_topprod.reviewText.apply(lambda x: clean_text(x)).to_list()
```

```
Out[10]:
```

['love case simple easy get phone laser gun get stuck case back get clean often normal phone use though wise awesome  
on kick stand help keep phone prop side',  
'second case bought like give heads back cover phone get scratch dust debris get via open stand found perman ring back  
cover took easy money sale',  
'order case could stand note ii sturdy anticip serv purpos long time hinge solid stand strong pretti handi',  
'this stylish protecte brought case friend say really enjoy hand taken since got',  
'product state add deliv time function perfect phone look great well switch case case lot one favorit think get red  
one well',  
'case fits perfectll pocket kick stand work well sleek slender price vs well worth',  
'also case fits note well great grip though stand bit flimsy work fine price cant beat',  
'really like this light note skeptic case want protect really like product color easi grip stand didnt expect use qu  
it use watch video read recip drawback litt hard volum on/off slide really push phone fits snugly phone wallet fee  
r slide well',  
'case fits note well stand work great button hole place correctll dont case extra hole top right cornerthere use leav  
phone vulner corner case note extra open',  
'ive read comment product let tell love case one alot mix review case want put match outfit go casual event smethn s  
atur want use case everyday use kind case mild protect dont go around test tough kickstand frail would watch movi  
kickstand mean perform hand styls view phone kickstand good case especi like show enorm phone look multipl color case  
would tri',

Figure 41. The newly cleaned text of reviewText.

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



```
In [10]: doc1 = df1_topprod.reviewText.apply(lambda x: clean_text(x)).to_list()

In [11]: cv = CountVectorizer()
word_count_vector1 = cv.fit_transform(doc1)
tf = pd.DataFrame(word_count_vector1.toarray(), columns=cv.get_feature_names())
tfidf_transformer = TfidfTransformer()
X = tfidf_transformer.fit_transform(word_count_vector1)
idf = pd.DataFrame({'feature_name': cv.get_feature_names(), 'idf_weights': tfidf_transformer.idf_})
# print(idf)
# tf_idf = pd.DataFrame(X.toarray(), columns=cv.get_feature_names())
# print(tf_idf)
idf.sort_values( by = 'idf_weights', ascending=False).head(50)
```

```
Out[11]:
```

	feature_name	idf_weights
459	line	5.135167
544	movie	5.135167
518	music	5.135167
519	music	5.135167
520	myterial	5.135167
521	natr	5.135167
522	nacwood	5.135167
523	necessary	5.135167
525	nag	5.135167
527	naghen	5.135167
531	real	5.135167

Figure 42. The new keywords generated of product1

```
In [12]: doc2 = df2_topprod.reviewText.apply(lambda x: clean_text(x)).to_list()

In [13]: cv = CountVectorizer()
word_count_vector1 = cv.fit_transform(doc2)
tf = pd.DataFrame(word_count_vector1.toarray(), columns=cv.get_feature_names())
tfidf_transformer = TfidfTransformer()
X = tfidf_transformer.fit_transform(word_count_vector1)
idf = pd.DataFrame({'feature_name': cv.get_feature_names(), 'idf_weights': tfidf_transformer.idf_})
# print(idf)
# tf_idf = pd.DataFrame(X.toarray(), columns=cv.get_feature_names())
# print(tf_idf)
idf.sort_values( by = 'idf_weights', ascending=False).head(50)
```

```
Out[13]:
```

	feature_name	idf_weights
375	hawk	5.18205
395	intim	5.18205
422	unk	5.18205
397	hd	5.18205
396	hawk	5.18205
406	mod	5.18205
393	rapper	5.18205
391	rapper	5.18205
405	unkess	5.18205
400	regardess	5.18205
410	regret	5.18205
411	rel	5.18205

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 “necessary” 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.

```
# Add column for then
df['Subjectivity'] = df['token'].apply(getSubjectivity)
df['Polarity'] = df['token'].apply(getPolarity)
df['Analysis'] = df['Polarity'].apply(analysis)
df['Overall'] = read_file.loc[:, 'overall']
#Show
df
```

	Reviews	token	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 dalli 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	token	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 dalli 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 reall push get b...	Positive
3768	My daughter is really into animal print so she...	daughter realli anim print filp surpris one ...	Positive

3769 rows x 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 *totchttext dataframe* data. One printed set showed as:

	Reviews	tokens	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 dalli hour everl...	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 pleasanti 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 x 3 columns

Figure 46. Example of preprocessed 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).

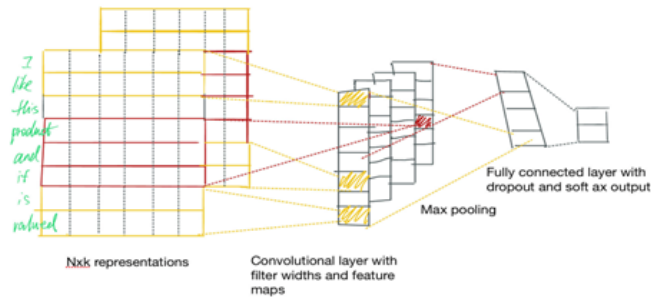


Figure 47. A general CNN model for this task

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:

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

$$W_{out} = \left\lceil \frac{(W_{in} + 2 * P - W_{kernel})}{S} \right\rceil + 1$$

$$CH_{out} = K$$

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-IDF 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 from 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.

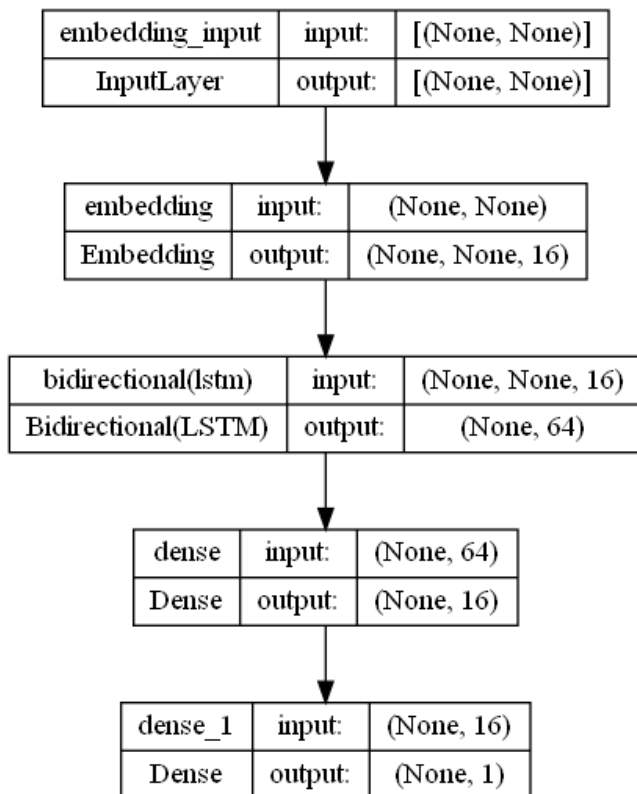


Figure 49. The structure of LSTM model

#### 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>