Product Review Data Analysis and Processing

CE/CZ4045 Natural Language Processing
Group 13

Chat Anan Potcharaphol
U1521216H
pchatana001@e.ntu.edu.sg

Jin Zhi Yeong
U1522475F
jyeong003@e.ntu.edu.sg

Judith Tan Yi Ru
U1621452A
jtan311@e.ntu.edu.sg

Marathe Ajinkya Avinash
U1522716K
ajinkya001@e.ntu.edu.sg

ABSTRACT

In this product review data analysis and processing project, the use of natural language processing is demonstrated to analyse product reviews. Firstly, in the first section, we will illustrate the results of our data analysis on the given dataset using tokenization, segmentation and stemming. Next, we developed a noun-phrase detector to extract noun-phrase. The implementation of the noun phrase summarizer achieved low recall and high precision. Thirdly, in the third section, we discuss our approach on analysis of data using sentiment analysis. The results were illustrated with the top-20 positive and negative words used in product reviews. We found that, the most frequently used words in positive reviews are: “like”, “great” and “good”. In case of negative reviews words such as “problem”, “difficult” and “disappoint” are prevalent. Finally, in the fourth section, we develop an application to detect comparative words such as “better” and” more” and found that the top-5 comparative words are “more”, “better”, “less”, “longer” and “easier”.

1 Data Analysis

1.1 Popular Products and Frequent Reviewers

In the given dataset, we identify the top-10 products that attract the greatest number of reviews and the top-10 reviewers who have contributed the greatest number of reviews. The results are illustrated below in Figure 1 and 2 respectively.

We write a function to count the frequency of input attributes.

```
def find_most_frequent(data_list, key, top_n):
    """
    This function receives a list of json objects with keys and
corresponding values.
    It counts the number of json objects with identical value
of the specified key and sort the result in a list.
    Ex:
    my_list = [{"name": "John"}, {"name": "Marry"}, {"name":
"John"}]
    result = find_most_frequent(my_list, "name", None)
    >> result = [("John", 2), ("Marry", 1)]
    :param data_list: a list of json objects
    :param key: the key corresponding to value we want to count
    :param top_n: only return top_n result (if None it will
return all the result)
    :return: a list of tuple containing (value, count)
    """
    attr_list = [item[key] for item in data_list]
    result = Counter(attr_list).most_common()
    if top_n is None:
        return result
    if len(result) > top_n:
        return [result[i] for i in range(top_n)]
    return result
```

A usage of find_most_frequent fucntion is:

```
top_10_products = find_most_frequent(results, "asin", 10)
top_10_reviewers = find_most_frequent(results, "reviewerID",
10)
```

Here results is a json object obtained from raw data.

asin(Product ID)	No.of reviews
B005SUHPO6	836
B0042FV2SI	690
B008OHNZI0	657
B009RXU59C	634
B000S5Q9CA	627
B008DJIIG8	510
B0090YGJ4I	448
B009A5204K	434
B00BT7RAPG	431
B0015RB39O	424

Table 1: Top-10 products with the most reviews

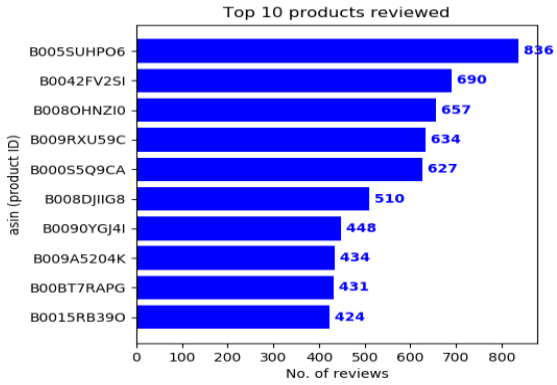

Figure 1: Depiction of the products and their respective reviews.

Reviewer ID	No.of reviews
A2NYK9KWFMJV4Y	152
A22CW0ZHY3NJH8	138
A1EVV74UQYVKRY	137
A1ODOGXEYECQQ8	133
A2NOW4U7W3F7RI	132
A36K2N527TXXJN	124
A1UQBFCERIP7VJ	112
A1E1LEVQ9VQNK	109
A18U49406IPPIJ	109
AYB4ELCS5AM8P	107

Table 2: Top-10 reviewers with the most reviews

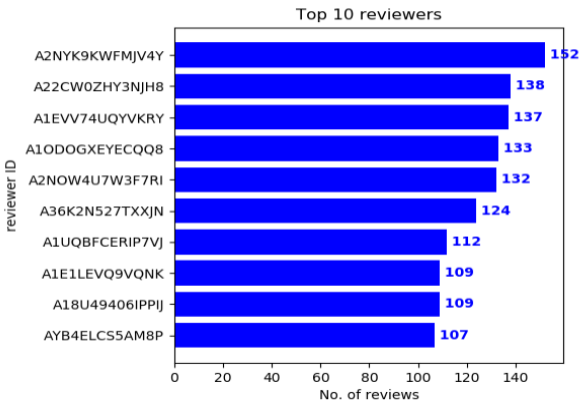

Figure 2: Depiction of the reviewers and the number of reviews contributed.

1.2 Sentence Segmentation

Next, we perform sentence segmentation on the reviews and show the distribution of the data in the Figure 3 below.

For sentence segmentation, we made use of ***sent_tokenize*** function from ***nltk.tokenize*** package.

Top-10 Reviews contributing the highest number of sentences

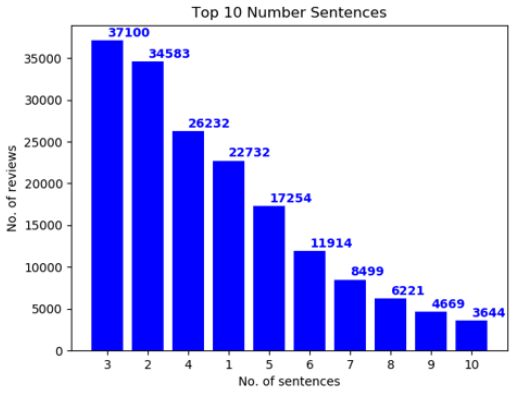

Figure 3: Depiction of the reviews and the number of sentences

Most reviews contain 2-3 sentences. As the number of sentences increases, the number of reviews with those number decreases. This suggests that most reviewers do not spend a lot of time contributing to the reviews. Most of them just stated their opinions straight to the point with a few sentences

5 samples of sentences segmented

1. A great seller.
2. its and ok product I can't complain I mean it could have been better, but it does what it supposed to do and that's it
3. On the down side, one of the strap clips broke the first time I used it.
4. Protects the expensive phone great!
5. If you follow the instructions and use the included dust tape and applicator it goes on smooth, no air bubbles, with minimal effort.

The segmentation algorithm largely depends on the use of big capital letters for the start of the sentence and punctuations like full stop and exclamation mark for the end of the sentence.
A short sentence like sentence 1 and 4 as well as a long sentence like sentence 3 and 5 are segmented correctly.

In the case where the start of the sentence is a small capital letter like sentence 2, the algorithm can detect this by using the end of the previous sentence. However, if there is no full stop at the end, the algorithm finds it difficult to detect the end of the sentence as seen by the wrongly segmentation in sentence 2.

1.3 Tokenization and Stemming

We made use of ***word_tokenize*** function from ***nltk.tokenize*** package for word tokenization and ***PorterStemmer*** class from ***nltk.stem*** package to perform stemming.

NLTK library is not able to handle some word for stemming. For example, it converts "charged" to "charg". In this case, we munually indicate the stemming process by defining a reference as a key-value pair in a dictionary like below:

```
my_stem_ref = {
    "charged": "charge",
    "charges": "charge",
    "charger": "charge",
    "charging": "charge",
    "batteries": "battery",
    "batterys": "battery",
    "verified": "verify",
    "verifies": "verify"
}
```

Table 3 and table 4 show the Top-10 words before and after stemming respectively.

Word	Count
phone	171056
case	139891
like	69857
use	59856
screen	57641
great	56178
battery	53817
good	53078
well	46585
charge	43554

Table 3: Top-10 words before stemming

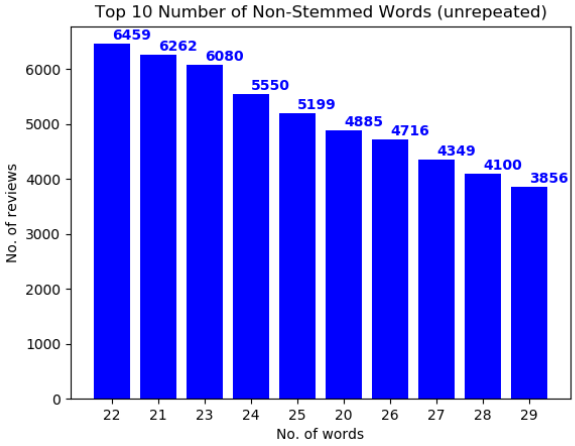

Figure 4: Top 10 Number of un-stemmed Words (unrepeated)

Word	Count
phone	189482
case	163278
charge	129333
use	116695
like	79627
work	75515
great	66011
battery	65066
screen	61069
good	58074

Table 4: Top-10 words after stemming

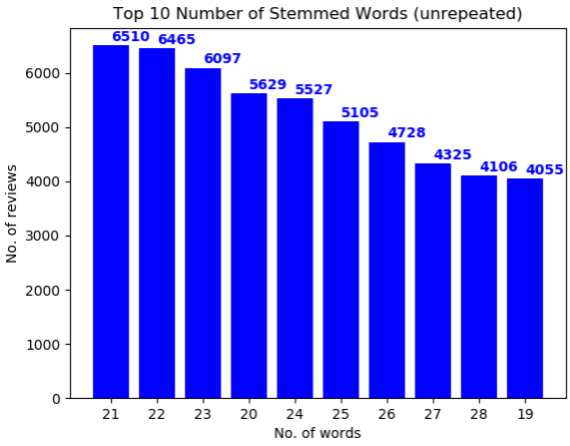

Figure 5: Top 10 Number of Stemmed Words (unrepeated)

In general, the nature of words suggest that the reviews are about smartphone accessories. It also shows that, in general, reviews are positive due to the use of words "great", "work", "like" and "good".

It can also be observed that the number of word count for the same word increases after stemming. This is because after stemming, different words with the same root are combined. Stemming also causes some words to move up the list such as the word “charge” and “battery”. Before stemming, the words “charged”, “charging”, “charges”, “charger”, “charge” are counted separately. As a result, the number of counts is low. After stemming, the counts are combined, causing the word “charge” to move up the list from last rank to the third rank.

1.4 POS Tagging

The following is a code snippet of how POS tagging is done:

```
# POS Tagging
n_samples = 5
print("\n{} POS Tagging".format(n_samples))
for i in range(5):
    sample_review = random.choice(results)
    sample_sentence_list =
sample_review["sentence_tokenize"]
    sample_sentence =
random.choice(sample_sentence_list)
    word_tokens = word_tokenize(sample_sentence)
    print(nltk.pos_tag(word_tokens))
```

We made use of ***pos_tag*** function from ***nltk*** package.

5 Samples of POS Tagging

1. Got (NNP) this (DT) for (IN) my (PRP$) dad (NN). (.)
2. There (EX) are (VBP) a (DT) lot (NN) of (IN) vendors (NNS) selling (VBG) this (DT) battery (NN) on (IN) Amazon (NNP).
3. Minimalist (NN) and (CC) good (JJ) looks (NNS). (.)
4. When (WRB) my (PRP$) phone (NN) charge (NN) is (VBZ) down (RB) to (TO) less (JJR) than (IN) 15 (CD) % (NN) , (,) I (PRP) just (RB) switch (VB) the (DT) battery (NN) out (RP) for (IN) a (DT) fresh (JJ) one (CD) , (,) place (VBZ) the (DT) drained (JJ) one (CD) in (IN) the (DT) battery (NN) charger (NN) , (,) and (CC) move (VB) on (IN) with (IN) less (JJR) than (IN) a (DT) 1 (CD) min (NN) interruption (NN) . (.)
5. This (DT) is (VBZ) a (DT) neat (JJ) accessory (NN) for (IN) any (DT) phone (NN) with (IN) a (DT) headphone (NN) jack (NN). (.)

The meaning for the word type can be seen in appendix II.

2 Noun Phrase Summarizer

2.1 Top-20 Noun Phrases

We define the noun phrase as a phrase containing a determinant if any, followed by adjective(s), followed by noun(s). Note that the presence of at least one adjective before the noun is mandatory according to our definition to avoid making the noun phrases common place and to better represent the reviews. In this experiment, we tag words of the reviews with a part of speech and then use regular expressions to extract noun phrases. We use the following regular expression:

(? :(?:\w+ DT)?(?:\w+ JJ)+)\w+ (?:N[NP]+|PRN)

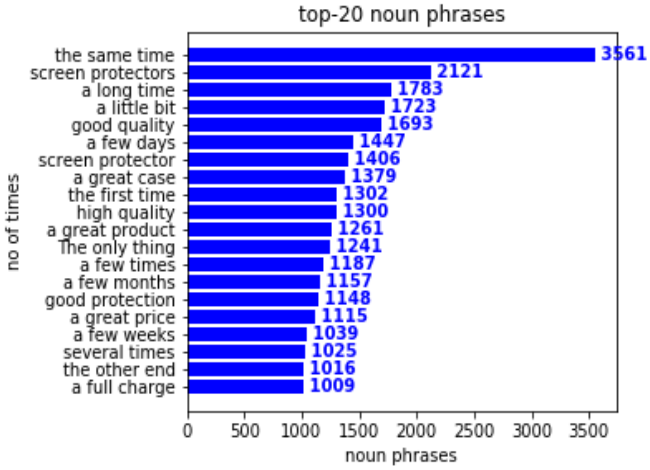

Figure 6: Depicts the top 20 noun phrases in the corpus

2.2 Top-10 Noun Phrases of Popular Products

1.	'B005SUHPO6'
2.	'B0042FV2SI'
3.	'B008OHNZI0'

Table 5: Top three products

We chose the following top three products in Table 1 from the data analysis section. The ‘representative noun phrases’ have been selected considering the frequency of the noun phrase in the product’s reviews and the frequency in all the reviews. A weighted sum of frequency in product reviews and frequency in all reviews in the ratio of (2:1) has been taken to decide the representative phrases.

Score= 2*(freq in this product) + (freq in products except this)

B005SUHPO6	the same time
	screen protectors
	a long time
	a little bit
	good quality
	a few days
	screen protector
	a great case
	the first time
	high quality

Table 6: Noun phrase results of product "**B005SUHPO6"**

B0042FV2SI	screen protectors
	a long time
	a little bit
	good quality
	a few days
	screen protector
	the first time
	high quality
	a great product
	The only thing

Table 7: Noun phrase results of product "**B0042FV2SI"**

B008OHNZI0	the same time
	screen protectors
	a long time
	a little bit
	good quality
	a few days
	screen protector
	the first time
	High quality
	a great product

Table 8: Noun phrase results of product "**B008OHNZI0"**

2.3 Noun Phrase Analysis

The representative noun phrases do not really represent the review of the product. They are just the most common phrases used in the reviews. Many of them are inflectional forms of stems like "screen protector" and "screen protector"

2.4 Result of Noun Phrase Detector

The underlined phrases in the Table 5, "Sentences" column are the manual annotated noun phrases in the reviews. The "Detected Noun Phrase" column represent the noun phrases detected by the noun phrase detector.

Review	Sentences	Detected Noun Phrase
Review 1	I haven't had any problems with it. It just like the original charger that comes with the Galaxy S2.	['the original charger']
Review 2	a couple problems with this headset, when using it with the Treo 650 -1. it does not automatically connect when a call is received or initiated2. the volume is way too lowcheck out the Jabra JX10, Cardo Scala, or Palm Treo headset ---- there are too many great headsets out there, to settle for this onehappy shopping!	['many great headsets', 'this onehappy shopping']
Review 3	Durable. Sounds great. Mic provides good sound for those on the other endof the connection. I like the wrap-around fit (walkman-style headphones look dorky - wrap around is better). Although.. after 30 minutes the tips of my ears can become sore which I am learning to cope with. I use it with Motorola L2 cell phone and Anycom 250 USB	['good sound', 'around fit', 'style headphones','the same place','stress areas' 'much smarter']

	dongle for Skype. This is the one to get. I had a competing Plantronics product -- in fact two of them -- but they both broke in the same place due to the use of plastic in high-stress areas. The Motorola design is much smarter.	
Review 4	I really don't have a lot of time. That is why I shop online, So if I'm purchasing an item that is to be used to make life more convenient, like a Bluetooth headset, it really is obnoxious to receive one that is "worse than sucks". This item did not even charge. Plantronics really needs to pull this item because it is really damaging to their reputation to have this out there for sale. I choose Plantronics because as a Regional Sales Manager/Road Warrior, I've always had luck with Plantronics. Too bad that this one was a lemon."	['a Regional Sales']
Review 5	Great ! these things are super good and the batteries keep just as long as the OEM! would recommend and the NFC works!	[ø]

Table 9: Results of sample reviews using developed Noun phrase detector

2.5 Evaluation of Noun Phrase Detector

A numerical value for precision and recall has not been calculated as it involves counting the total number of phrases (noun or otherwise) in the review. Instead, the performance of the noun phrase extractor in terms of false positives and false negatives has been discussed.

2.5.1 Precision

In the observed reviews, the noun phrase extractor does not detect random phrases that are not noun phrases. Overall, the false positives are low, and the extractor has a high precision.

2.5.2 Recall

In the observed reviews, the noun phrase extractor leaves some words out of a noun phrase. Like the extractor only detected “style headphones” out of “walkman-style headphones” or “Regional Sales” out of “Regional Sales Manager”. It also misses out some phrases “a couple problems” due to weak performance of the POS tagger. The noun phrase has a low recall.

3 Sentiment Word Detection

In this section, the use of natural language processing will be illustrated to extract top 20 positive and negative keywords from Amazon product review. Firstly, we explored the given dataset by visualizing the data a little more by plotting some graphs with the Seaborn library to see any relationship between our newly created text length feature and the overall rating.

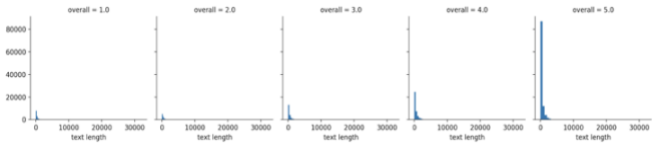

Figure 7: Distribution of text length across all five ratings

From the Figure 7 above, seems like overall, the distribution of text length is similar across all five ratings. However, the number of text reviews seems to be skewed a lot higher towards the 4-star and 5-star ratings. This may cause some issues later on in the process.

After processing the aforementioned task, data cleaning and preprocessing (Tokenization, Sentence segmentation, Parts of speech tagging and removal of stop words), we segregated the reviews according to their overall ratings – positive reviews and negative reviews as seen in Table 10 below.

Types of Reviews	Overall Ratings
Positive	4.0 and 5.0

Negative	1.0 and 2.0

Table 10: Type of reviews by ratings

Next, we label and rank a piece of text as positive, negative or neutral using a lexicon of positive and negative words using NLTK's built-in ***Vader Sentiment Analyzer.*** We utilize this tool by first creating a ***Sentiment Intensity Analyzer (SIA)*** to categorize our reviews, then we'll use the polarity scores method to get the sentiment. Finally, we'll append each sentiment dictionary to a results list, which we'll transform into a list as seen in Figure 9 below.

```
#Sentiment Analysis

allPosLower = [item.lower() for item in pos]
allNegLower = [item.lower() for item in neg]
allNeuLower = [item.lower() for item in neu]

sid = SentimentIntensityAnalyzer()
pos_word_list=[]
neu_word_list=[]
neg_word_list=[]

for word in stemmed_words:
    word=word.lower()
    if (sid.polarity_scores(word)['compound'] >= 0.2) and (word in allPosLower):
        pos_word_list.append(word)
    elif (sid.polarity_scores(word)['compound']) <= -0.2 and (word in allNegLower):
        neg_word_list.append(word)
    else:
        if(word in allNeuLower):
            neu_word_list.append(word)
```

Figure 8: Snippet of codes for sentiment analysis

Our list consists of four columns from the sentiment scoring as seen from Figure 8 which is neu_word_list, neg_word_list, pos_word_list and compound. The first three represent the sentiment score percentage of each category in our reviews, and the compound single number that scores the sentiment. 'compound' ranges from -1 (Extremely Negative) to 1 (Extremely Positive).

We will consider reviews with a compound value greater than 0.4 as positive and less than -0.4 as negative. There's some testing and experimentation that goes with choosing these ranges. However, there is a trade-off to be made here. If we choose a higher value, we might get more compact results (less false positives and false negatives), but the size of the results will decrease significantly.

Top 20 Positive Sentiment Words

Word	Word Count
great	58611
good	45810
love	34139
nice	27312
better	16676
perfect	14060
best	9242
hand	9242
excel	7003
help	6417
friend	5062
clean	4990
wish	4944
free	4916
care	4260
super	4181
cute	4129
easier	4003
kind	3836
impress	3743

Table 11: Top-20 positive sentiment words

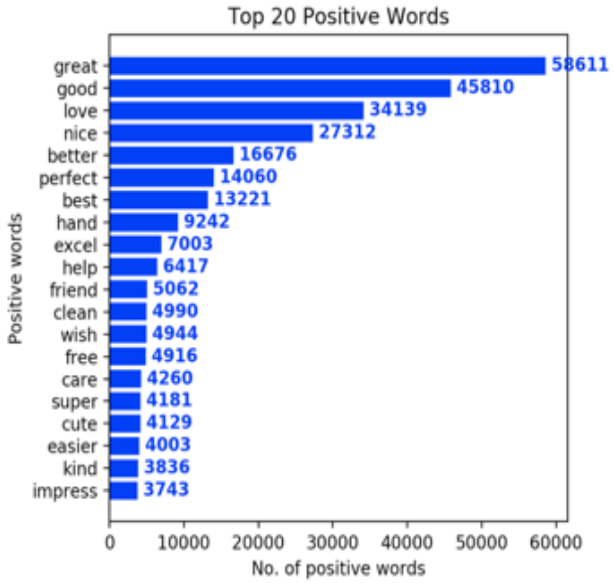

Figure 9: Top 20 Most frequently used words in Positive reviews

Word	Word Count
problem	2672
bad	1850
disappoint	1720
broke	1644
poor	917
wrong	689
useless	654
fail	648
broken	631
annoy	525
worst	496
hate	482
die	454
lose	448
suck	390
dead	354
pain	257
fake	270
weak	229

fault	218

Table 12: Top-20 negative sentiment words

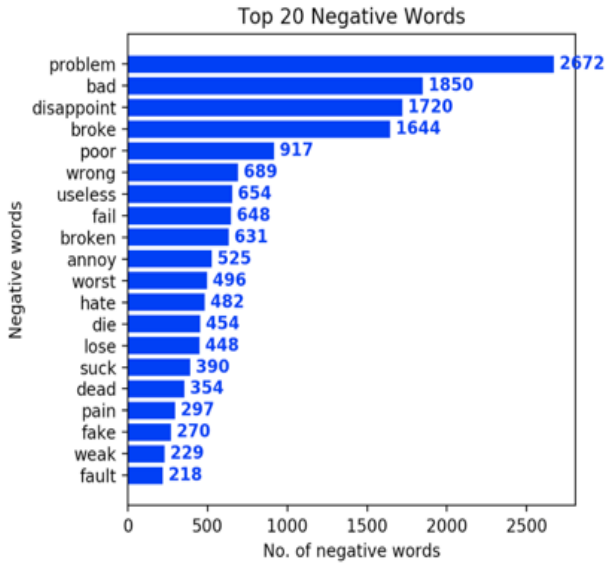

Figure 10: Top 20 Most frequently used words in Negative reviews

As seen in Figure 9, the most frequently used words in positive reviews are: "like", "great" and "good". In case of as seen from Figure 10, negative reviews words such as "problem", "difficult" and "disappoint" are prevalent.

On one side, there is a large number of neutral reviews, a total of 186964, which is mainly due to two reasons:

1. The assumption that we made earlier where reviews with compound value between 0.4 and -0.4 are considered neutral. The higher the margin, the larger the number of neutral reviews.
2. We used general lexicon to categorize reviews. The more correct way is to use a reviews-specific lexicon, but for that we would either need a human to manually label data, or we would need to find a custom lexicon already made. Another possibility is that our analyzer produced a lot of false negatives.

To sum up, Amazon's product review platform shows that most of the reviewers have given 4-star and 5-star ratings to the product reviews. The average length of the reviews comes close to 230 characters. We also uncovered that lengthier reviews tend to be more helpful and sentiment analysis shows that positive sentiment is prevalent among the reviews and in terms of emotions, 'great', 'good and 'love have highest scores.

4 Application

4.1 Objective of Application

The application we developed is to detect comparative words in a user provided corpus. The definition of comparative words are words that are used to compare the differences between 2 objects in a sentence. A sentence with comparative words usually has a structure in this form: Noun (subject) + verb + comparative adjective + *than* + noun (object). Examples of comparative words include: less, more, bigger and easier.

4.2 Formation of comparative words

For single syllable words, comparative words can be formed by adding an *-er* at the end. If the adjective has a consonant + single vowel + consonant spelling, the final consonant must be doubled before adding the ending. An example of a consonant + single vowel + consonant spelling is the word big where the final consonant 'g' is doubled before adding an *-er* at the end.

One Syllable	
Word	**Comparative form**
tall	taller
big	bigger
hard	harder

Table 13: One Syllable

Two syllable words can be changed into their comparative forms by adding an *-er* at the end or by adding the word "more" before the word. An example of adding the word "more" to make a comparative is the phrase "more studious". Adding an *-er* at the end of "studious" would be incorrect and so to create the comparative form, we add the word "more" to form the phrase "more studious". For words that end in *y*, change the y to an i before adding an *-er* at the end. For words that end with an 'e', simply add a "r" to form the comparative version.

Two Syllables	
Word	**Comparative form**
happy	happier
simple	simpler
studious	more studious

Table 14: Two Syllables

Three or more syllable words can be changed into their comparative forms by adding the word "more" before the word.

Three or more Syllables	
Word	**Comparative form**
interesting	more interesting
exciting	more exciting
appropriate	more appropriate

Table 15: Three or more Syllables

There are some words where the rules above do not apply when transforming to the comparative form. Some examples are shown below.

Irregular Comparatives	
Word	**Comparative form**
good	better
bad	worse
far	further/farther

Table 16: Irregular Comparatives

4.3 Methodology

Given that the rules for the formation of comparative words are governed mainly by the number of syllables a word possesses, it would be very difficult to come up with a regular expression that is able to encompass most of the words. Further complicating this is the existence of irregular comparatives that do not conform to the syllable rules. Furthermore, there exist many words that also contain the *-er* at the end but are not comparative words. Examples of these words include basketballer, soccer, water and so on. Hence, we decided to go with using Part-of-speech (POS) tagging to better identify these words. Using NLTK's POS tagging library, we tagged words in a sentence and used the tags to identify comparatives. The tags we used were JJR (for comparative adjectives) and RBR (for comparative adverbs). Using POS tagging, we were better able to identify comparative words.

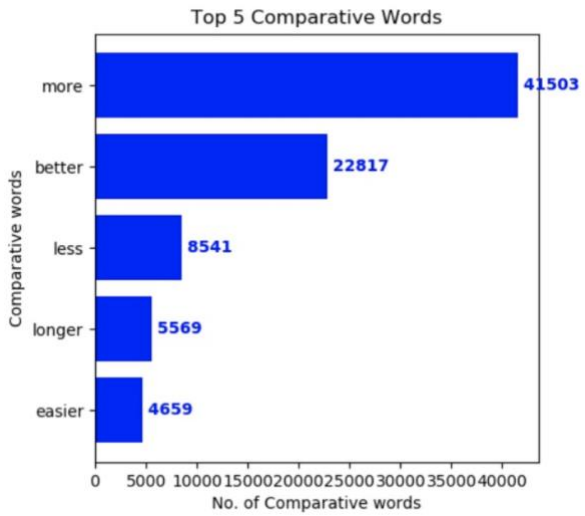

Figure 10: Top 5 Comparative words

The following figure above represent the top 5 comparative words. We can see that words like "more", "better" and "less" are the commonly used words when users used in the product reviews when comparing between two entities or product.

4.4 Evaluation of application

```
Sample 3.
Review analysed :  Works wonders for newer phones that need m
ore &#34;juice&#34; for charging like my Galaxy S4. The charg
er with the ports together on the top has a much brighter blu
e LED light that encompasses the square top and is the primar
y one I use, just because the 2.1A port is labeled and it mak
es it easier to identify that the charger is properly plugged
-in in my car. The other charger with the ports on the sides
(sort of) that are separate has a smaller blue LED and doesn'
t light up as much but that may be nicer for those driving at
 night. It gets used mostly by my girlfriend and she hasn't r
eported any issues from her Galaxy S3. I would highly recomme
nd this pack if you need two chargers (they also both come se
parately too if you only need one). A great bonus is the Micr
o-USB cable that comes with each, along with the adapter to c
harge certain devices that need a different cable layout for
drawing the right amount of power (Galaxy Tab is listed as th
e device that needs it, would work on others that need the re
-route I imagine: I hear HTC devices need this too but I don'
t have one to test so you're on your own there).
Comparative words found : 4
The words are : newer, more, easier, smaller
```

Figure 11: Screenshot of results for comparative detector

The Figure 11 above illustrates the application retrieving the comparative words from randomly sampled reviews using CellPhoneReview.json and displaying the words to the user. In conclusion, this application experiments the sentiments expressed in comparative sentences and achieve highly accurate results.

5 CONCLUSION

In this product review and analysis, we show how one can use natural language processing to

1. Perform data analysis using major task such as Sentence Segmentation, Tokenization, Stemming and POS Tagging,
2. Implementation of noun phrase summarizer and evaluated our results of the noun phrases,
3. Implementation of sentiment analysis and evaluated our results
4. Implementation of a comparative word application and evaluated our results

Appendix I: List of stop words

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?', '@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~', "n't", 'one', 'two', "'s", 'would', 'get', 'very']

Appendix II: POS Tagging Categories

CC = coordinating conjunction
CD = cardinal digit
DT = determiner
EX = existential there (like: “there is” … think of it like “there exists”)
FW = foreign word
IN = preposition/subordinating conjunction
JJ = adjective ‘big’
JJR = adjective, comparative ‘bigger’
JJS = adjective, superlative ‘biggest’
LS = list marker 1)
MD = modal could, will
NN = noun, singular ‘desk’
NNS = noun plural ‘desks’
NNP = proper noun, singular ‘Harrison’
NNPS = proper noun, plural ‘Americans’
PDT = predeterminer ‘all the kids’
POS = possessive ending parent’s
PRP = personal pronoun I, he, she
PRP$ = possessive pronoun my, his, hers
RB = adverb very, silently,
RBR = adverb, comparative better
RBS = adverb, superlative best
RP = particle give up
TO = to go ‘to’ the store.
UH = interjection, errrrrrrrm
VB = verb, base form take
VBD = verb, past tense took
VBG = verb, gerund/present participle taking
VBN = verb, past participle taken
VBP = verb, sing. present, non-3d take
VBZ = verb, 3rd person sing. present takes
WDT = wh-determiner which
WP = wh-pronoun who, what
WP$ = possessive wh-pronoun whose
WRB = wh-abverb where, when

6 REFERENCES

[1] Jones E, Oliphant E, Peterson P, *et al.* SciPy: Open Source Scientific Tools for Python**,** 2001 , http://www.scipy.org/ [Online; accessed 2018-11-05].

[2] John D. Hunter. Matplotlib: A 2D Graphics Environment**,** Computing in Science & Engineering, **9**, 90-95 (2007), DOI:10.1109/MCSE.2007.55(publisher link).

[3] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O’Reilly Media Inc.