CZ4045 Natural Language Processing – Assignment (G03)

Calvin Tan Sin Nian
U1521152K
ctan113@e.ntu.edu.sg

Chan Yi Hao
U1520808B
ychan019@e.ntu.edu.sg

Leung Kai Yiu
U1522848L
kleung003@e.ntu.edu.sg

Phang Jun Yu
U1522651D
jphang005@e.ntu.edu.sg

Soh Jun Jie
U1521123B
sohj0027@e.ntu.edu.sg

ABSTRACT

The objective of this report is to highlight the insights obtained from performing several Natural Language Processing (NLP) tasks on a dataset of Amazon reviews from the Cell Phones and Accessories category. In this paper, we list down the steps taken in achieving the stated results and evaluate the results obtained.

CCS CONCEPTS

• **Artificial Intelligence → Natural Language Processing → Information Extraction**

• **Applied Computing → Document Management and Text Processing**

KEYWORDS

NLP, POS tagging, Sentiment Analysis, Noun Phrase, Lemmatization, Negation, Segmentation, Tokenization

1 INTRODUCTION

In this project, we will be performing various steps in a typical end-to-end Natural Language Processing (NLP) pipeline. This includes: (i) dataset analysis, (ii) noun phrase summarizer, (iii) sentiment word detection, (iv) a simple NLP application that performs negation detection. We will be using various NLP libraries such as Natural Language Toolkit (NLTK) [1] and Stanford NLP [6].

1.1 Dataset Description

The dataset contains 190,919 reviews from the Cellphones and Accessories product category in Amazon [5]. Information such as reviewer ID, product ID, review text (in full), rating (out of 5), summary, review time are provided. All data except the last column were used.

1.2 Data Cleaning

While the given dataset has already been pre-processed to correct formatting errors, more data cleaning was required. The following lists the common data cleaning steps used throughout various parts of the project.

1.2.1 *Blank reviews*. There are 99 records where the review texts are blank. These records are removed as it will affect the statistics we generate from our analysis, especially for the sentence segmentation analysis.

1.2.2. *Website links*. The POS tagger from NLTK will not work well for website links as they will be parsed and tagged (e.g. 'http' is tagged as a noun). Thus, they are replaced in our dataset with a <http_url> token.

1.2.3 HTML Entities. The dataset contains many symbols that are displayed as HTML entities instead of ASCII encodings. For instance, the double quote symbol " is presented as ". This is a problem since the HTML entities are tagged as nouns when running a Parts of Speech (POS) tagger on the dataset. With high frequency of occurrence (at least 20,000 instances), they are a major influence on results produced by any analysis and thus should be removed. A Python library, html.parser, was used to convert all HTML entities into ASCII.

1.2.2 *Inconsistent Spacing*. At least 20,000 sentences end with a full stop but do not have a space after the full stop. This poses a problem to sentence segmentation. For example, a sentence like "I like NLP.It is very fun." could be treated as a single sentence and "NLP.It" may be tagged as a noun as its POS tag. A regular expression was used to identify these patterns and a space will be added after the full stop in the middle. This will allow sentence segmentation to be done properly and the correct POS tags will be assigned to the tokens.

2 DATASET ANALYSIS

Pandas [4] dataframe was used to perform dataset analysis because it has many built-in functions and thus allow easy manipulation of data. A parallelizing function (as shown below in Figure 2.1) was also created to speed up the application of Pandas functions. It makes use of the multiprocessing module in Python to create multiple processes that run concurrently.

```
def parallelize_dataframe(df, func):
    a,b,c,d,e = np.array_split(df, num_partitions)
    pool = Pool(num_cores)
    df = pd.concat(pool.map(func, [a,b,c,d,e]))
    pool.close()
    pool.join()
    return df
```

Figure 2.1: Pseudocode for the parallelizing function

2.1 Top-10 Products and Reviewers

Using Pandas' `value_counts().nlargest(10)`, the top-10 popular products and frequent reviewers are listed according to their frequencies in Table 2.1 and 2.2 respectively. A code snippet is shown below in Figure 2.2.

```
print(df.asin.value_counts().nlargest(10))
print(df.reviewerID.value_counts().nlargest(10))
```

Figure 2.2: Pseudocode for Top-10 frequencies

Table 2.1: Top-10 most reviewed products

asin	Frequency count
B005SUHPO6	836
B0042FV2SI	690
B008OHNZI0	657
B009RXU59C	634
B000S5Q9CA	627
B008DJIIG8	510
B0090YGJ4I	448
B009A5204K	434
B00BT7RAPG	431
B0015RB39O	424

Table 2.2: Top-10 reviewers

reviewerID	Frequency count
A2NYK9KWFMJV4Y	152
A22CW0ZHY3NJH8	138
A1EVV74UQYVKRY	137
A1ODOGXEYECQQ8	133
A2NOW4U7W3F7RI	132
A36K2N527TXXJN	124
A1UQBFCERIP7VJ	112
A1E1LEVQ9VQNK	109
A18U49406IPPIJ	109
AYB4ELCS5AM8P	107

2.2 Sentence Segmentation

The reviews were segmented into sentences using the `sent_tokenize()` function found in the NLTK library. It uses an instance of `PunktSentenceTokenizer` from NLTK's `tokenize.punkt` module. This instance has been trained on and works well for many European languages. It can ascertain punctuations and characters that mark the end of a sentence as well as the start of a new sentence.

Although our data cleaning resolves some issues where sentences are not properly separated (no space between a full-stop and a uppercase), there are unresolvable exceptions. For instance, no space between a full-stop and a lowercase, as well as abbreviations. A code stub for sentence segmentation is shown in Figure 2.3.

```
df = parallelize_dataframe(df,
parallelizegetsentlen)
   sentencecounts =
df['sentencelength'].value_counts().to_dict()

def parallelizegetsentlen(df):
   df['sentencelength'] = df.apply(lambda x:
len(sent_tokenize(x['reviewText'])), axis = 1)
   return df
```

Figure 2.3: Pseudocode for sentence segmentation

Most reviewers wrote 3 sentences in their reviews, followed by 2, 4, 1 and 5 in descending order. Also, from Figure 2.4, number of sentences decreases exponentially afterwards. However, there are outliers. For instance, reviewer A16I8SQQIA2B8C wrote a 5,272 words (310 sentences) review on comparing a product with iPhone.

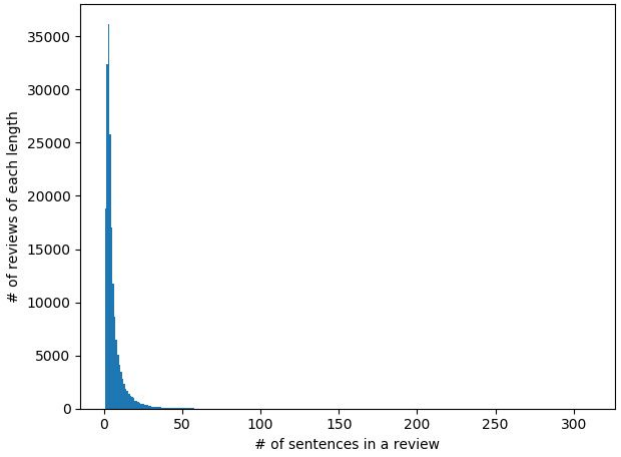

Figure 2.4: Plot for Sentence Segmentation

3 short and 2 long reviews were randomly sampled using `DataFrame.sample()` and exported into a .csv file. Short reviews are arbitrarily defined to contain less than 5 sentences, and long reviews consist of more than 15 sentences. Refer to Appendix A or `/results/Dataset Analysis/sample_sentence_lengths.csv` for the sample sentence lengths.

The sentence tokenizer correctly identifies ellipses as a sentence-internal punctuation (found in the first sample), as opposed to a terminal punctuation mark. It was also able to identify the tilde symbol (~) and colon as non-terminal punctuations marks in the fifth sample.

However, the sentence tokenizer seems to crudely segment the text "No dust under the screen. At. All." in the fourth sample into 3 sentences. A complete sentence, even a one-word sentence, needs to have a noun and a verb. Thus it seems incorrect to merely segment the sentences primarily based on terminal punctuation marks. Perhaps POS tagging can be integrated into the algorithm to more accurately tokenize text into sentences.

2.3 Tokenization and Stemming

The `word_tokenize` method of NLTK was used to segment reviews from the `'reviewText'` column into words. The tokenizer uses the `PunktSentenceTokenizer` model for word segmentation. Figure 2.5 shows the word_tokenize method being used in a parallelizing function.

```
def parallelizetokenizetext(df):
   df['tokenizedwords'] = df.apply(lambda x:
nltk.word_tokenize(x['reviewText']), axis = 1)
   return df
def parallelizestemtokens(df):
   df["stemmedwords"] = df['tokenizedwords'].apply
(lambda x: [ps.stem(y) for y in x])
   return df
```

Figure 2.5: Pseudocode for the parallelizing function

After getting the word tokens with `word_tokenize`, the Porter stemmer (`ps.stem`) was used to remove common morphological and inflexional token endings. The number of reviews was subsequently plotted against the associated number of tokens.

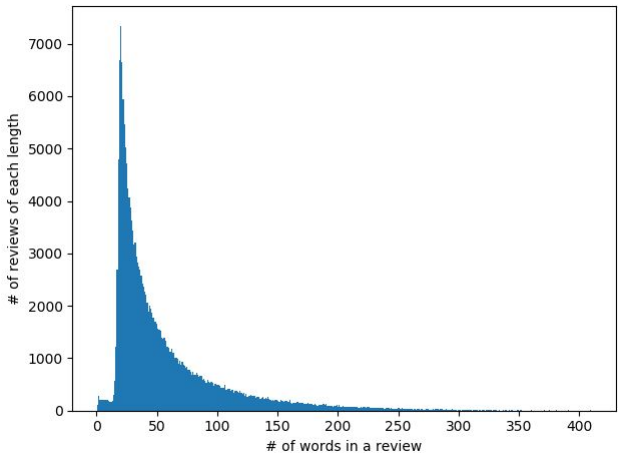

Figure 2.6: Tokenized words without stemming

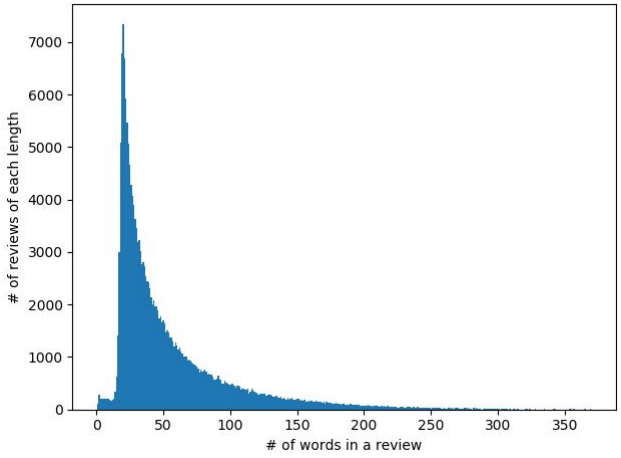

Figure 2.7: Tokenized words with stemming

Figure 2.6 and 2.7 shows that most of the reviews have 16 to 66 words (with more than 1000 occurrences) despite stemming. The length with the highest number of reviews is 20 words, with more than 7000 occurrences. In comparison, the top 3 largest number of words in reviews has decreased from 409, 391 and 381 to 369, 365 and 357 after stemming. This is because stemming has reduced inflected or derived words to their word stem. Therefore, these type of redundant words are removed.

Top 20 most frequent words before and after performing stemming are listed in Table 2.3 and 2.4. NLTK's `stopwords.words('english')` was used to exclude the stopwords. Since there is no universal list of stopwords in NLP research, we find words that are deemed as stopwords elsewhere in our results, such as "one" or "would". A list of stopwords used can be found in Appendix B or in the source code at: `/results/Dataset Analysis/stopwords.txt`.

Table 2.3: Top-20 most frequent words, pre-stemming

Words	Frequency count	Words	Frequency count
phone	174,958	well	51,149
case	146,050	iphone	47,691
one	86,759	get	46,380
like	71,853	charge	44,897
great	66,874	charger	38,646
would	66,809	product	38,223
use	61,676	really	38,055
screen	60,145	also	37,145
good	58,601	time	36,661
battery	57,976	works	32,743

Table 2.4: Top-20 most frequent words, stemmed

Words	Frequency count	Words	Frequency count
phone	192,494	screen	61,812
case	165,228	get	61,243
use	118,087	good	58,823
one	92,736	look	52,093
charg	92,539	well	51,161
like	79,743	iphon	50,521
work	76,325	fit	50,296
great	66,912	time	48,381
would	66,814	charger	45,281
batteri	66,237	protect	44,734

"charge" and "charger" are thought to be stemmed to the same word but Porter Stemmer seems to be able to differentiate them, not stemming the word "charger".

In comparison, the most frequent words after stemming provide richer insights for the dataset. Meaningful keywords such as ''fit' and "protect" are discovered. Both words may imply that reviewers were looking for accessories that fit well and can protect their cell phones. Furthermore, the higher frequency counts made the results more legitimate.

Given the nature of the dataset, we expected words such as "phone", "good" and "great" to be commonly used in the context of cell phone accessories and indeed they were listed. With the list of popular keywords, we have also discovered some of the more popular cell phone accessories such as "case", "battery", and "charger". These accessories were not as expected as cables, earphones and selfie sticks.

2.4 POS Tagging

The POS tagging was performed on 5 sentences from the dataset using the NLTK's `pos_tag()` function. A code stub for POS tagging is shown below in Figure 2.8.

```
sampledf = df.sample(n=5)
```

```
sampledf['selectedSentence'] =
sampledf.apply(lambda

x:random.choice(sent_tokenize(x['reviewText'])),
    axis = 1)
sampledf['selectedSentencePosTag'] =
sampledf.apply(lambda x:nltk.pos_tag(
    nltk.word_tokenize(x['selectedSentence'])),
    axis=1)
```

Figure 2.8: Pseudocode for POS tagging

```
"bought for my security system, and works just
like the original" =>
[('bought', 'NN'), ('for', 'IN'), ('my', 'PRP$'),
('security', 'NN'), ('system', 'NN'), ('and',
'CC'), ('works', 'VBZ'), ('just', 'RB'), ('like',
'IN'), ('the', 'DT'), ('original', 'JJ')]
```

Figure 2.9: POS tags on a sampled sentence

Figure 2.9 shows POS tagging on a sample sentence. Although `pos_tag()` manages to tag the tokens in the sentences correctly, there are a few exceptions. For instance, in "...find an outlet for your phone or anything usb related...", the POS tagger tags "usb" as "JJ" or adjective instead of "NNP" or proper noun. This is likely because the NLTK's POS tagger has not been trained on a corpus that features technical terminologies.

3 DEVELOPMENT OF A NOUN PHRASE SUMMARIZER

3.1 Noun Phrase Detector

According to the Oxford Guide to English Grammar, a noun phrase is defined as a combination of a noun and optionally other words that leads to the following structure: "Quantifier - Determiner - Adjective modifier - Noun modifier - Noun - Other modifiers" [10]. Other sources, such as the Oxford Dictionary of English Grammar, raises the possibility of including pronouns [11]. Although pronouns can be considered as noun phrases, they are excluded because of the ambiguity and lack of meaning carried by them. For example, when two different reviews use the word 'it', they could refer to different things. Furthermore, including them will simply result in a list of top 20 noun phrases that does not provide much insights.

We thus define noun phrase as a combination of one or more words that (i) always contain at least 2 nouns OR 1 adjective and 1 noun, (ii) does not contain verbs, (iii) exclude pronouns. This is formally defined in Figure 3.1.

3.1.1 Approach. In choosing the dataset, we opted to use both the summary and the review text so as to get as much information as possible. Also, users were more likely to express their emotions through adjectives in the review so we felt that including the summary would be necessary for an insightful analysis.

For subsequent sections where the noun phrase detector will be used, the methodology is similar, varying only in parameters used. There are three stages: Parsing, Counting and Lemmatizing.

3.1.2 Parsing. In parsing, we first define a grammar structure using regular expressions. The data is then tokenized and tagged with POS tags and the noun phrases that match the defined structure are extracted.

```
NBAR: {<DT>*<NN.*|JJ.*>+<NN.*>}
NP: {<NBAR>}
    {<NBAR><IN><NBAR>}
```

Figure 3.1: Grammar rules used for noun phrases

This means that the first pattern to be matched is the NBAR and it requires one or more noun or adjectives and an additional noun. The determinant is optional. The "." indicates that the noun can be a proper noun or the different varieties of nouns since there there are different tags for different types of nouns.

The noun phrase is then defined to be either one NBAR term or two NBAR terms with a preposition in between. In our implementation, we made the conscious approach to have the parser include phrases with adjectives using the "+" symbol. This allows us to get more representative information out of the dataset as adjectives more accurately depict the product's use for the user.

3.1.3 Counting and Lemmatization. In our extraction of most frequent noun phrases, we also converted the words into lowercase and lemmatized them so that both lower and upper-case words would be counted together. The lemmatization removed double counting of certain phrases. For example, "products" and "product" were both identified as valid noun phrases by our algorithm but we only needed it to appear once since both mean the same thing. Hence, we lemmatized all the phrases so that the counts for similar cases could be aggregated and a more accurate representation of the information hidden in the data could be captured. This also allowed for a wider variety of phrases that could be used to capture the representation of the reviews. *In the noun phrases captured by our detector, the noun phrases output will be in the lemmatized form but this will not affect the understanding by the user in any way.*

Lastly, the extraction was performed on both the review text and summary of the products because the summary provides valuable insights that should be included.

Using this definition, the top-20 most frequent noun phrases identified are presented in Figure 3.2:

```
[('screen protector', 15653), ('great case',
5836), ('great product', 5667), ('battery life',
4154), ('same time', 3674), ('sound quality',
3563), ('cell phone', 3942), ('great price',
3464), ('usb port', 3336), ('good case', 2846),
('good product', 2828), ('nice case', 2497),
('power button', 2466), ('car charger', 2308),
('phone case', 2276), ('long time', 2216), ('good
quality', 2188), ('battery pack', 2181), ('only
thing', 2148), ('usb cable', 1903)]
```

Figure 3.2: Top 20 noun phrases by frequency

We believe that these terms are mostly satisfactory in explaining the products being reviewed in this dataset and

also offer an overview of features that users look out for in their products. There is a good range of features as users talk about screens, battery life, sound quality, usb ports, casings, buttons, car chargers and more. The only term that might be slightly different would be "only thing" as it is more a linguistic expression with which users say "the only thing I would …". In this example, this phrase does not tell us much other than the users' pattern of speech. However, the other phrases are generally representative and a good indication of the reviews in the dataset.

3.1.4 Noun Phrase Classifier. Alternatively, a simple binary classifier can be trained to detect noun phrases. The output of one such classifier - from NLTK's NP Chunker - is presented in Appendix C. The results contain noun phrases that are not in our defined rules (contains pronouns, singletons) but also has legitimate ones.

3.2 Representative Noun Phrases

In this section, we were tasked to find 10 representative noun phrases for the 3 most popular products. The 3 most popular products are shown in Table 3.1 below.

Table 3.1: Top 3 most reviewed products

Asin	Frequency count
B005SUHPO6	836
B0042FV2SI	690
B008OHNZI0	657

In an initial analysis, the noun phrases returned mostly matched static definitions of the product. While that was good, we felt that in order for a noun phrase to be "representative", it should also capture what the user wanted to convey in their review. Hence, in the grammar structure, we added a rule that *apart from just being a noun, the first term could also be an adjective*. This allowed us to capture more descriptive noun phrases.

For example, Figure 3.3 shows noun phrases returned without the new grammar rule for asin B005SUHP06.

```
[('phone', 703), ('case', 695), ('iphone', 333),
('otterbox', 248), ('product', 97), ('protection',
90), ('color', 82), ('screen protector', 76),
('time', 76), ('price', 69)]
```

Figure 3.3: Representative noun phrase without new grammar rule for Asin B005SUHP06

While this is somewhat informative, it was not necessarily representative of the product on this review page as a review is an expression by users of the product. If we wanted a static definition of the product, the dataset could be obtained from the official product page instead.

Figure 3.4 shows the improved results after adding new grammatical rules. We feel that these 10 phrases are indeed very representative of the products. One point to note is that both 'great product' and 'good product' appear in the latter 2 cases. While users may use 'good' and 'great' interchangeably, we did not combine the counts of these 2 phrases as the proportion of "great" to "good" could also include important information about user's reviews of the product in other datasets. Lastly, this also confirmed our hypothesis that including the summary was important as many users included adjectives in their review summary as emotion is key to communication and we wanted to capture these emotions as an accurate representation of the product.

```
asin=B005SUHP06
[('screen protector', 78), ('great case', 69),
('great product', 53), ('otter box', 41), ('great
protection', 37), ('otterbox defender', 36),
('otterbox case', 33), ('best case', 32), ('good
case', 29), ('belt clip', 23)]

asin=B0042FV2SI
[('screen protector', 172), ('matte finish', 44),
('great product', 30), ('finger print', 20),
('anti glare', 19), ('great price', 15), ('good
product', 14), ('matte finishing', 13), ('air
bubble', 12), ('screen cover', 11)]

asin=B008OHNZI0
[('screen protector', 311), ('tech armor', 89),
('great product', 61), ('great screen protector',
37), ('home button', 28), ('best screen
protector', 26), ('air bubble', 25), ('good
product', 23), ('retina display', 22), ('matte
finish', 20)]
```

Figure 3.4: Representative noun phrase using new grammar rules for top 3 most reviewed products

3.3 Analysis on 5 Sample Reviews

We randomly picked 5 reviews from the Amazon dataset. Annotated noun phrases are highlighted in green and detected noun phrases are in blue in Figure 3.5.1 to 3.5.5.

```
I purchased this hoping to replace a cheaper phone
case purchased from amazon. However, the cheaper
one is a better product. ok

Annotated Noun Phrases: cheaper phone case,
cheaper one, better product

Detected: cheaper phone case, cheaper one, better
product

Precision: 100%, Recall: 100%
```

Figure 3.5.1: Noun phrases of Sample 1

```
dont bother with those tempered glass protectors.
This is what you need right here. Enough hardness
so it doesnt get scratched when you rub it down
with a cloth. No visual distortions and the pen
feels great on it. 3 pack is great if you screw up
or need to replace it down the road. Dont waste
money on more expensive bs. feels really good and
looks great!

Annotated Noun Phrases: tempered glass protectors,
no visual distortions, expensive bs

Detected: dont bother with tempered glass
protector, enough hardness, no visual distortion,
pen feel, dont waste money, expensive b
```

Precision: 33.33%, Recall: 66%

Figure 3.5.2: Noun phrases of Sample 2

The parser recognized the word ‘dont’ as a noun instead of the verb since the ‘ was omitted. ‘Waste’ and ‘bother’ were also recognized as nouns instead of verbs. ‘pen feel’ was detected because ‘feel’ was detected as a noun instead of a verb.

The first thing that struck me when I got this was the size. It's very thin. The nice part about this is it easily slips into my laptop bag, my pocket, or many other places without being too much of a bother. The downside is that means it doesn't have all that much juice.I've had a few of these battery packs, so I'll go through how I think this one stands up. The Pro's- Thin, fits easily in a pocket for day use- Contains a micro-usb cable, huge plus for portability. - Sturdy, feels quite well built, solid.The Con's- Power at only ~1400mAh it's not going to give you a full charge on most things, but is enough to get a few more hours use out of most things. - Price, for this price there are (bulkier) 5000 mAh packs, but here you're paying a price for thinness, and the convenience of a cable built in. Tiny with built in cable

Annotated Noun Phrases: first thing, nice part, laptop bag, many other places, battery packs, micro-usb cable, huge plus for portability, full charge on most things, few more hours, most things

Detected: first thing, nice part, laptop bag, many other place, much juice, battery pack, pro's- thin, micro-usb cable, huge plus, con's- power, full charge on most thing, few more hour, most thing, mah pack

Precision: 71.43% , Recall: 90%

Figure 3.5.3: Noun phrases of Sample 3

Over here, we notice that converting the strings to lowercase causes some of the noun phrases to lose their original meaning. mAH was supposed to be milli-amperes but it can be taken to be an informal slang of the word “mine” as “mah” here.

I must have at least 50 cases in my house, for,my IPhone 5s. I know what I want, but no one has even able to give me a case that I think is perfect. Finally, this case comes very close. This is a hybrid case that combines soft TPU sides with a hard polycarbonate back. It has a small lip on the case so you can feel safe that the screen is protected. It also allows the phone to have a very slim profile, with an easy grip and access to all ports. What I really love is that it has a clear matte back so I can actually, finally, see my beautiful IPhone! I have found most cases with clear backs are not really "clear", but this one is.Also, it
does not remove or scratch my screen saver, which is a real plus to me.It looks like I have found a new favorite case. Thank you Totallee, for allowing me to review your case! Excellent case that allows you to see your iPhone...

Annotated Noun Phrases: Iphone 5s, hybrid case, soft TPU sides with hard polycarbonate back, small lip, slim profile, easy grip, clear matte back, beautiful IPhone, clear backs, screen saver, real plus, new favorite case, excellent case

Detected: hybrid case, soft tpu side with hard polycarbonate, small lip, slim profile, easy grip, clear matte, beautiful iphone, most case with clear back, screen saver, real plus, new favorite case, excellent case

Precision: 91.67% , Recall: 84.62%

Figure 3.5.4: Noun phrases of Sample 4

The issue in this sample is that ‘back’ is recognized as a verb so it is omitted from the detection when it is in fact a noun in this use case. However, we have decided to count the detected phrases as valid as ‘back’ is not an essential noun since the reader can easily infer that a hard polycarbonate or matte material can only refer to the ‘back’ of a phone. Interestingly, the tagger recognizes ‘grip’ and ‘plus’ as nouns in this case when they can also be verbs. ‘Most’ is also recognized as a noun when it is in fact an adverb.

Stylish and attractive. Good seal around the edges. Buttons work fine. Excellent drop protection. Works fine with the RND Dock. She-a-looks-ana-feels like a Ferrari

Annotated Noun Phrases: Good seal, excellent drop protection, RND Dock

Detected: good seal, button work fine, excellent drop protection, rnd dock

Precision: 75% , Recall: 100%

Figure 3.5.5: Noun phrases of Sample 5

In the above example, ‘buttons work fine’ was wrongly classified as a NP.

The overall precision for identifying noun phrases from the 5 samples of reviews is 74.29% and the recall is 88.33%.

The algorithm has a much higher recall than precision. The high number of false positives is primarily due to the lack of formatting and grammar or precise spelling used in online review terminology. This problem is ultimately mitigated by having a large dataset and picking terms that appear consistently. Also, if the term is consistently spelt wrongly, it would not really matter since our algorithm would still capture it and it can easily be deciphered by a human reader.

The results also reflect the inability of the tagger to correctly capture the right tags of certain tokens. In some cases, the tagger guesses wrongly and this generates false negatives and false positives. The tagger in NLTK is *PerceptronTagger*, which is a greedy averaged

perceptron tagger that adjusts the weights on input features and average the adjustments over number of iterations. Being trained on the Wall Street Journal corpus, the weights are likely to be an ill fit for our dataset (since it has more informality and shorter sentence structures). With inaccurate POS tags, our grammar rules will not be able to pick up those misclassified tags. However, humans are able to disambiguate these tags easily, hence the precision and recall are not at 100%. To bring it closer to perfection, the POS tagger needs to be trained on a review-based dataset.

Lastly, these sample reviews further confirm our hypothesis of using adjectives to define representative noun phrases. By taking adjectives into account, we clearly get an accurate idea of the product in review, instead of just taking lots of filler noun terms.

4 SENTIMENT WORD DETECTION

The result of the sentiment word detection solution is to produce the top 20 representative words for expressing positive and negative feelings. For this, we chose to perform natural language processing on the *review's summary section instead of the full review text*. The reason are three folds - (1) summary text is more concise and thus less computationally intensive, (2) summary text capture the true essence of the review, thus better reflecting candidate words that represent positive and negative feelings, and (3) shorter text require less data cleaning effort.

The third point is important as the full review text is riddled with tokens not included in the English dictionary e.g. "160x160 pixel", "&#34". Handling these unclean text have unintended consequences when performing POS tagging since removing word tokens can result in breaking the existing sentence structure.

4.1 Data Cleaning

Before analyzing words in the review to quantify the level of positivity and negativity, we perform the following data cleaning procedures.

Step 1: Any URL links included in the review summary are replace with a "http_url" word token.

Step 2: Common negation words such "no" and "not" are combined with the word following it to form a new token. For example "not good" would be converted to "not_good". This is important because the fundamental usage of the word following after the negation word ("good" in this case) has changed.

Step 3: The regular expression "\w+" is used to identify valid word tokens. This helps in removing invalid tokens such as full stop and comma.

Step 4: Part-of-speech tagging is performed on the partially cleaned summary text. Words tagged as noun (NN), will be remove from the summary text as a noun does not express feelings.

Step 5: Finally, stop words such as "and", "but", "then" are removed from the review summary text.

The remaining text that are filtered out from the cleaning process are valid words to be considered for calculation of sentiment scores.

4.2 Computing Sentiment Score of a Word

The key idea behind our approach for computing sentiment score was first to assume that each word, W, in a particular review, R_i, has equal probability in contributing to its review score, S_i. More formally, we can define it as the following.

$$\forall\ W_j \,\varepsilon\, R_i\ , 0 < j < wordcount(R_i),\ P(S_i \mid W_j) = P(S_i \mid W_{j+1})$$

j refers to the position of the word in the review R_i and i refers to the position of the review R_i in the review dataset.

The above definition may not make sense at first, since a word in a review with favourable rating would have more positive sentiment than the same word appearing in a review rated badly. We therefore compute the sentiment value of a given word, W, by averaging its influence across all reviews that the word was mentioned.

```
1 word = "word"
2 word_influence = 0
3 word_mentioned_count = 0
4 for review in AmazonReviews:
5     while review.summary.find(word):
6         word_influence += review.overall
7         word_mentioned_count += 1
```

Figure 4.1: Pseudocode for calculating word influence of a given word.

The above pseudo code snippet in Figure 4.1 demonstrate computation of total influence of a given word. From line 5, when a given word is mentioned in the review, we increase its current influence by the review overall score. The while loop ends when the review found all the instances of the word W in the review summary. Similarly, other words mentioned in the same review would also have their influence increased by the same review overall score. This is in accordance with our previous assumption that all words in the same review have equal probability to contribute to the current review score.

However, 1 thing that is done differently from what is stated previously, is that we also keep track of the number of times the given word is mentioned across multiple reviews. The sentiment score can then be computed by dividing the word total influence by number of times the word was mentioned. The equation is given as follows.

$$Sentiment(w) = \left[\sum_{i=0}^{N} S(w,\ R_i) \right] \div N$$

The function $S(w,\ R_i)$ refers to the influence of the word w for a review R_i and it is same as the overall score of the review R_i. A sum function is used to obtain the aggregate influence of the word w across all reviews. N

refers to number of times the word w was mentioned across multiple reviews instances. One thing to note here is that if the same word was mentioned twice in the same review and once in another, then N will be 3 instead of 2 since the word was mentioned a total of three times. Similarly, assume that the first review that the word was mentioned twice had an overall score of 4 and the other review had an overall score of 2, then the total influence would be evaluated as 10 i.e. 4 + 4 + 2. The sentiment of the word would be 10 / 3 = 3.333.

4.3 Computing the Adjusted Sentiment Score of a Word

The need for the computation of an adjusted sentiment score arise due to the fact that the Amazon review dataset contains an imbalanced number of positive reviews and negative reviews. We define a review as positive or negative as the following.

$$PosNeg(R) = \begin{cases} positive & if\ S(R) \geq 3, \\ negative & if\ S(R) \leq 2 \end{cases}$$

The function S retrieve the overall score of the review R. In addition, we also keep track of the number of times a given word W appear in both a positive review and negative review by the following.

```
word = "word"
pos_review_mentioned = 1
neg_review_mentioned = 1
for review in AmazonReviews:
    while review.summary.find(word):
        if review.overall >= 3:
            pos_review_mentioned += 1
        if review.overall <= 2:
            neg_review_mentioned += 1
```

Figure 4.2: Pseudocode for finding number of times a word is mentioned in positive and negative reviews

From the above pseudo code snippet in Figure 4.2, we first initialise the number of positive and negative review mentioned as 1 to avoid a division by zero problem later on when we calculate the adjusted sentiment.

Then let pm be the count of positive reviews mentioned and nm be the count of negative reviews mentioned. Note that if a word W, appears twice in the same positive review, pm will be 2 instead of 1 since the word was mentioned more than once in the same review.

Also let $totalpm$ denote the total number of reviews that are positive and $totalnm$ denote the total number of reviews that are negative. Our adjusted sentiment score can then be calculated by the following equation.

$$AdjSentiment(w) = Sentiment(w) \times \frac{pm}{nm} \times \frac{totalnm}{totalpm}$$

Here, we scale the calculated sentiment value first by a factor of pm/nm to reflect the word w appearing in more positive reviews than negative reviews or vice-versa. We also scale the calculated sentiment value by a factor of $totalnm/totalpm$ to account for the fact that there may be more positive reviews in the dataset than there are negative reviews or vice-versa. For the given dataset, 166,952 reviews were positive and 23,967 reviews were negative base on the criteria defined previously.

Once we have calculated the adjusted sentiment value for all possible words in the review dataset, we then sort them in descending order. The top 20 words with the highest sentiment value are the top 20 positive words and the bottom 20 words in the list are the top 20 negative words.

4.4 Discussion of Results

This section illustrate the top 20 positive and negative words using the formula for adjusted word sentiment.

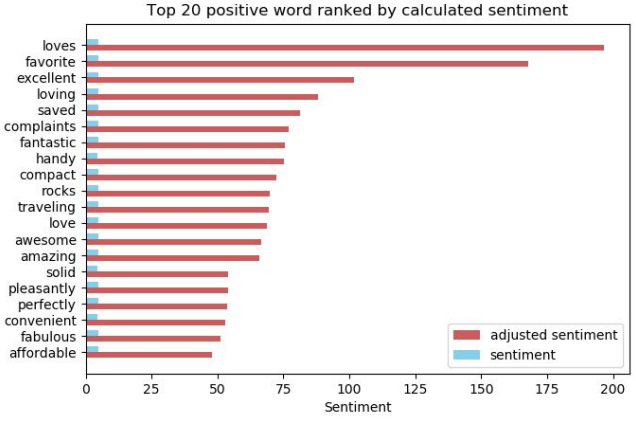

Figure 4.3: Top 20 positive words ranked in descending order by adjusted sentiment

Some words listed in the top 20 positive words in Figure 4.3 may look odd, for example, “traveling” and “solid”. However, upon inspecting reviews that contains these words, it was found that “travelling” and “solid” is something positive according to our human common sense. For example, people usually want products that are great for traveling. On the other hand, people used “solid“ to express reliability and durability of products.

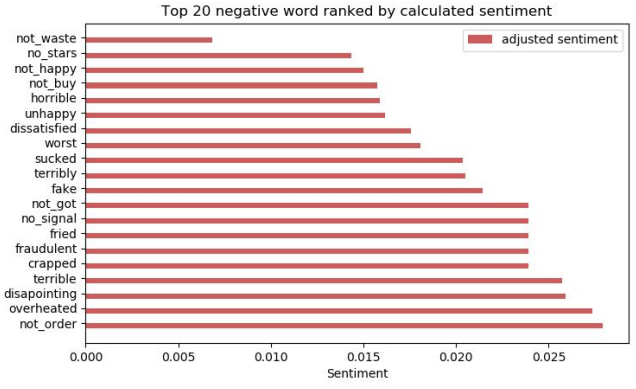

Figure 4.4: Top 20 negative words ranked in ascending order by adjusted sentiment

For the list of top 20 negative words in Figure 4.4, the word “not_waste”, “not_got”, “not_order” may seems doubtful since they do not make much sense by

themselves. More contextual information are required to interpret them.

From the dataset, the word “not_waste” was used to express the buyer desire to “not waste” his valuable resources, i.e. time or money, on the reviewed item. The word “not got” was used by buyers to show dissatisfaction for not receiving something expected. The word “not order”, on the other hand, were frequently used to indicate buyer’s intention to “not order” the item in the future.

Overall, most of the top 20 positive and negative words generated by our word sentiment detection algorithm made much sense. However, words that are ambiguous in nature which relies on more context are also picked up by our algorithm. For future work, one way to handle this issue could be to perform chunking of sentiment words with contextual words. Doing so enhances understanding and intuition behind the word token captured.

5 APPLICATION

The simple NLP application that we developed attempts to tackle the problems that negation poses to NLP tasks. Many sentences contains words like ‘no’ and ‘not’ to nullify or give the opposite meaning of the word it is paired with. However, in basic NLP processes, text is often tokenized into individual words and processed independently. For example, a simple frequency-based driven topic modeller will consider a phrase like “big but not bulky" in a category ‘large items’, although the key message is how it is not oversized. Basically, the semantics of the phrase are not captured. For more accurate information extraction, we need to clean the data so that negations can be captured.

The negation problem is best exemplified by cases where semantics play a huge role in representing the meaning of the sentence. Such cases are listed in Table 5.1.

Table 5.1: List of phrases with negation

Negation Phrases extracted from the Amazon Dataset
1. ...doesn't look cheap and no stones are missing.
2. Not very durable at all!!
3. ... put crappy dysfunctional roms on their phones and offer no tech support or customer service.
4. Doesn't come bogged down with a bunch of useless apps that nobody wants.
5. Never liked holstering larger phones anyway.

For our dataset, the major use case of negation detection is to allow for more accurate sentiment analysis to be performed. When looking for representative positive or negative words in product reviews based on rating scores attached to the reviews, a keyword search that does not take negated words into account will produce misleading results. Armed with better negation detection, we will be able to capture semantics that expresses negations, which is prevalent in sentences with negative sentiments.

5.1 Types of Negations

To tackle negation, we first need to define the types of negations. There are 3 classes of intra-sentential negative expressions - (1) Not-negators (no, not, -n’t), (2) N-negators (never, neither, nor), and (3) Affixal (a-, dis-, im-, in-). Not-negators are the most common negation terms [12].

Our work will focus on non-affixal negations because affixal negations are not problematic in traditional NLP workflows - the affix attached to the word already gives the word the opposite meaning. A list of Not-negators and N-negators is presented in Appendix D.

5.2 Current State of the Art

From our literature review, negation detection is a widely known sub-problem in many NLP tasks. However, it is still not completely resolved: not much research has been done to perfect the art or to propose any best practices.

Most of the well-known NLP tools for negation detection comes from the healthcare industry as negations are prevalent in clinical text (e.g. “no swelling, distention observed”) and the only available publicly available dataset was BioScope. Notable algorithms include Negex [8] and NegBio [9]. Negex adopts a rule-based approach that generates regular expressions. While effective, these algorithms do not have a good performance speed due to the intensive lookup needed. NegBio, a more recent development, takes it further by including dependency parsing so as to leverage on syntactic structures. This creates stricter and more accurately generalizable rules.

5.3 Our Solution

Our solution gives a different edge from Negex and NegBio by providing both look-ahead and look-back functionalities. Given an input string, an output string with negations will be produced with the negated words replaced with the negated form of the word.

The PennTree Bank tokenizer and ssplit annotators from Stanford CoreNLP [6] are used to help us split the paragraph into sentences and words. Every word is iterated through and during each iteration, the words that occur before and after it are analyzed. The pseudocode is presented in Figure 5.1 and further explained below.

Almost all cases of negation has the negators preceding the word that is negated. Thus, words occurring before the current word are checked against a list of negators (by the function `InNegationWords()` in `checkNegation()`) to decide whether the current word should be negated. If found, a flag will be set to ‘true’.

If the flag is true, further analysis will be performed on words occurring after the current word. This is done to capture patterns such as “no static, cracks or pops”. Given this pattern, traditional negation detectors will only label ‘static’ as ‘no_static’ and neglect the rest. However, our solution provides a lookahead function which labels ‘cracks’ as ‘no_cracks’ and ‘pops’ as ‘no_pops’.

This algorithm will encounter a problem when faced with patterns such as “no instructions, but did include the charger”. In general, words like ‘but’ and ‘yet’ un-negates the sentences. Thus, additional checks will need to be and this is done by the `checkButYet()` function.

If `checkButYet()` returns false, the current word will then be negated. If the current token is 'and', 'or' or ',', it will be replaced with 'no1'. Else, if the current word is not 'and', 'or' or 'no', it will be prepended with 'no_'.

```
function main()
for word in sentence:
    if (checkNegation(word)):
        appendNo(word)
function checkNegation()
flag = false
for each word until three_words_before:
    if InNegationWords():
        flag = true

if flag == true:
    for each word until four_words_after:
        if checkButYet(): return flag
        else:
            negateWord()
            changeToNo()
```

Figure 5.1: Pseudocode for negation detection

With this application, we will perform semantic analysis better as more semantics of the sentence is captured.

5.4 Evaluation

Although ambiguities in negation detection remains, such a simple negation detection algorithm should be able to capture most of the cases. For the cases that are not captured, more sophisticated algorithms that has a more diverse dictionary of negators and take into account the parts of speech of the words. One example of how the current algorithm fails in in the case when the sentence has a "not only… but…" structure. To resolve this, more thorough treatment of conjunctions needs to be done.

Our algorithm requires the user to set two hyperparameters: the extent of look-ahead and look-back. Conjunctions and adverbs like 'but' and 'yet' gives us a clear idea of when we should terminate the look-ahead for cases where the sentence contains both positive and negative points. However, unipolar cases that only contain negativity (e.g. no discount, tax rebate), there is no known generalisable way to know when to stop negating the words. Casting the net too wide might result in spurious negations and from experimentation, our group feels that this hyperparameter should not go above 5 words. However, this threshold should be set carefully, depending on the nature of the text.

While the output is a good fit to NLP workflows, it isn't always human interpretable. If human interpretability is desired, a possible approach would be to replace the negated words with their antonyms, possibly with the help of a lexical database like WordNet [7].

6 CONCLUSIONS

In summary, we have performed various steps in an NLP pipeline. One of the key learning points our group has gained from working on this project is the importance of data cleaning. Real world data contains a lot of inconsistencies and errors which will significantly affect any rudimentary analysis that does not resolve them. However, caution must be made to not be overly engrossed at getting a perfectly cleaned data. Other than the fact that it is not possible with the current tools we have, different tasks require the data to be cleaned in different ways. For example, a noun phrase summarizer will not want to have a dataset with stop words removed as pronouns are an integral part of noun phrases.

7 CONTRIBUTIONS

In terms of report contributions, the team members contributed towards the corresponding parts of the source code they written. The contribution is as follows:

Calvin Tan Sin Nian	Dataset Analysis
Chan Yi Hao	Application
Leung Kai Yiu	Dataset Analysis
Phang Jun Yu	Noun Phrase Summarizer
Soh Jun Jie	Sentiment Word Detection

REFERENCES

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

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

[3] Travis E, Oliphant. (2006) A guide to NumPy. USA: Trelgol Publishing.

[4] Wes McKinney. (2010) Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51-56

[5] R. He, J. McAuley (2016) Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW.

[6] Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

[7] Princeton University "About WordNet." WordNet. Princeton University. 2010.

[8] Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., & Buchanan, B. G. (2001). A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. Journal of Biomedical Informatics, 34(5), 301–310. https://doi.org/10.1006/jbin.2001.1029

[9] Peng, Y., Wang, X., Lu, L., Bagheri, M., Summers, R., & Lu, Z. (2017). NegBio: a high-performance tool for negation and uncertainty detection in radiology reports. ArXiv:1712.05898 [Cs]. Retrieved from http://arxiv.org/abs/1712.05898

[10] Eastwood, J. (1994). Oxford guide to English grammar. Oxford University Press.

[11] Noun phrase - Oxford Reference. (n.d.). Retrieved November 4, 2018, from http://www.oxfordreference.com/view/10.1093/acref/9780199658237.001.0001/acref-9780199658237-e-975

[12] Tottie, G. (1991). Negation in English speech and writing: A study in variation. San Diego: Academic Press.

APPENDIXES

Appendix A: Sample Sentence Length

Sample No.	reviewText	Length
1	the product is nice, cute, but not as i expected ...i think there are nicer stuff out there that you can choose form	1
2	This was a gift for the father-in-law & he's not always easy to please. As always Seidio does a great job. The case is slim, buttons press easily & it looks great as well as protects.	3
3	I have these plugged in all over my house and never had a problem with them. I like to get OEM chargers because they seem to last longer than knockoffs.	2
4	I seriously buy a lot of phone cases and screen protectors. These screen protectors last the longest for me so far. Easy to put on and they stay. In. Place. You get three but the one has been on my phone for months. Spend the money on these. They're worth it. They are easy to install. You'll need the right environment so a construction site is a no-no. They supply you with the dust remover tape so that is a giant plus. Edit Sept 10, 2013:I've just removed my first one since it was a little beat up. I've dropped a few things on my screen (not on purpose, c'mon...)and I had to replace it. Took 3 minutes. No dust under the screen. At. All. Hint: After you clean your screen. Use the provided tape remover tabs to get the stragglers and you shouldn't have to lift up the protector.	19
5	Out of the box there was one thing I really liked about the M2 and M3 Bolse series was the fact that they had digital displays indicating the amount of charge left in the battery pack. I was pleased because this innovative design will allow the user to know exactly how much juice is left in the battery. The LEDs are nice but as a rule, you'll only see three or four of them and the range of charge can be wide. I'm used to working with these batteries and keep them charged up. However, if you have one and only have 1 LED remaining, you may have a 1% charge or a 24% charge. If you had a fuel gauge like that in a car, you might be driving on 1/4 of a tank or fumes. Novel idea and I really liked it. I liked the promise of this battery, as well as the M2, until I started to test them. I cannot speak for anyone else, but I test every battery I get ... in real-time circumstances. I use my iPad a lot and when it's powered down sufficiently I start testing. I figured a 6600mAh battery would be able to do a sufficient job powering up an iPad with a 46% charge remaining. The M3 struggled from the start with an immediate drop in charge from 97% down to 65% in fifteen minutes. It was then I decided to watch and record very closely what was going on. A casual approach is usually my thing, but instead of a stellar performance I'd expect from a 6600mAh battery pack, I only got an average one. Bolse is usually a performer, but for some reason the M2 and M3 battery packs fell a bit short of my expectations. The performance was OK, but nothing more. If you want a Bolse, which is an excellent brand, I'd look at the AON series, which really performed spectacularly. I'm rating this one a three, which is OK, on the Amazon scale. Bolse M3 (6600mAh) - 97% charged at startiPad - 46% charge at start15 minutes - 51% (65%)30 minutes - 56% (58%)45	19

minutes - 61% (55%)60 minutes - 66% (51%)75 minutes - 71 % (39%)90 minutes - 76% (37%)120 minutes - 82% (19%)135 minutes - 85% (LO)150 - The M3 stopped charging at this pointWHAT'S IN THE BOX:~ 1 Bolse M3 External Battery~ 3 connectors~ 1 Micro USB power cable~ 1 velvet-like travel pouch~ Quick start guideSPECS:~ Battery type: Li-ion battery~ Life cycle: ~500 cycles~ Input current: 800mAh~ Output current:Output USB1: for iPad / iPhoneOutput USB2 for All Smartphones2A total max~ Capacity: 6600mAh~ Size: 4.13 x 2.36 x 0.71 inches~ Weight: 5.76 ounces~ Certifications: CE & FCC & ROHSWarranty: 12 months limitedSample provided for review.

Appendix B: List of stop words

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

Appendix C: NP Chunker output

Review number: 30475

Review Text and summary: The first thing that struck me when I got this was the size. It's very thin.The nice part about this is it easily slips into my laptop bag, my pocket, or many other places without being too much of a bother. The downside is that means it doesn't have all that much juice.I've had a few of these battery packs, so I'll go through how I think this one stands up.The Pro's- Thin, fits easily in a pocket for day use- Contains a micro-usb cable, huge plus for portability.- Sturdy, feels quite well built, solid.The Con's- Power at only ~1400mAh it's not going to give you a full charge on most things, but is enough to get a few more hours use out of
most things.- Price, for this price there are (bulkier) 5000 mAh packs, but here you're paying a price for thinness, and the convenience of a cable built in. Tiny with built in cable

representative noun phrases are: {'day', 'a cable', 'all', 'that much juice.i', 'a few more hour', 'i', 'plus', 'my laptop bag', 'me', 'the first thing', 'it', 'a pocket', 'a full charge', 'most thing', 'the size', 'that', 'thinness', 'you', 'portability.- sturdy', 'thin.the nice part', 'the downside', 'this price', 'there', 'these battery pack', 'many other place', 'my pocket', 'bulkier', 'a bother', '5000 mah pack', 'a micro-usb cable', "up.the pro's- thin", 'well', "con's- power", 'a price', 'the convenience', 'things.- price', 'huge', 'a few', 'this', 'only', 'of', 'this one'}

Appendix D: Negation Words

Not-negators: No, Not, n't (including verbs like Aren't, Can't, Doesn't, Don't, Hadn't, Hasn't, Haven't, Isn't, Shouldn't, Won't, Wouldn't)
N-negators: Barely, Few, Hardly, Little, Neither, Never, Nobody, Nor, None, Nothing, Nowhere, Rarely, Scarcely, Seldom
Affixes and Suffixes (Not used): de-, dis-, il-/im-/in-/ir-, mis-, non-, un- ; -less