AI6122 Product Review Data Analysis and Processing

He Linzi
heli0005@e.ntu.edu.sg

Ong Jia Hui
jong119@e.ntu.edu.sg

Irene Ng Yu Si
ing011@e.ntu.edu.sg

Tan Mengxuan
mtan099@e.ntu.edu.sg

1 DATA ANALYSIS

This section will describe how we conducted our exploratory data analysis on the given review dataset.

1.1 Popular Products and Frequent Reviewers

Our initial analysis tells us that there are a total of 190,919 records in the dataset. Since, a particular product can be reviewed by different reviewers and a reviewer can review a product multiple times, we find out that there are a total of 10420 unique products and 27874 reviewers.

Next, we plot the review distribution of the dataset from the both the products and users perspectives. It shows in Figure 1 that majority of the products received few reviews. From the users' perspective, majority of the reviewers gave few reviews (Figure 2). Both Figure 1 and 2 display characteristics of heavy tailed distributions.

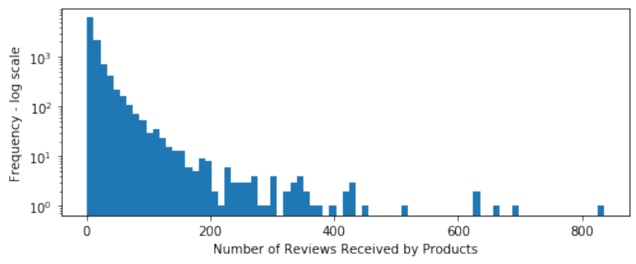

Figure 1: Review distribution of the dataset from the product perspective.

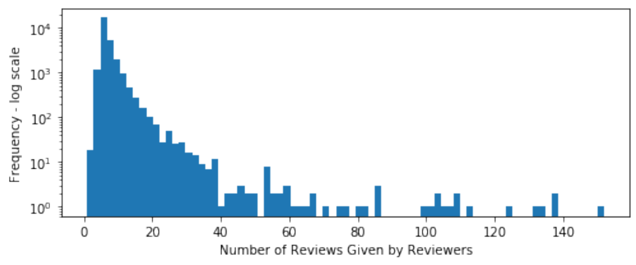

Figure 2: Review distribution of the dataset from the user perspective.

Furthermore, having the boxplots (Figure 3 and Figure 4) from both perspectives allows us to understand the spread of the data when arranged in increasing order of the number of reviews received or the number of reviews given. For instance, from Figure 3, common products received 6 to 18 reviews as indicated by the interquartile range (from the 25th to 75th percentile) of the plot. Figure 4 also tells us that common reviewers tend to give 5 to 7 reviews as shown by the interquartile range of the plot.

Subsequently, we can also use Figure 4 to derive the 5 groups of reviewers according to how active are they in posting reviews. Since the boxplot has already arranged the reviewers in sequential order according to how many reviews they gave, we find it intuitive to use this for our categorisation. After computations, the 5 groups of users are categorised as follows:

Table 1: Different Groups of Active Reviewers

Group	Percentile	% of total users	Reviews Posted
Not active	0 - 25	4	1 - 4
Not very active	25 - 50	42	5
Average	50 - 75	21	6
Active	75 - 92	25	7 - 10
Most active	92 - 100	8	11 - 125

As shown in Table 1, majority of the reviewers posted 5 to 10 reviews. The first 3 groups of not so active users lie within 0 to 75 percentile of the data. The gap between these 3 groups is exactly 1 quartile. The Figure 4 boxplot has also identified several 'outliers' beyond the high quartiles which may be used to represent the group of 'Most active' users. These points are located 1.5 times the interquartile range away from the third quartile. The first of such 'outlier' made 11 reviews and is at the 92th percentile. Hence, the 'Most active' reviewers are within the 92th and 100th percentile.

After categorising the reviewers, we can move on to plot their rating distributions.

Figure 5 displays the proportion of users that gave a particular rating for each group. All groups have a large proportion of users who gave a rating of 4 to 5 to products. The 'Most active' group contains the highest proportion of users who gave a rating of 4 to 5. We also show the cumulative distribution of the ratings given by each group.

From Figure 6, we can see that the cumulative distribution curve belonging to the 'Most active' group is lower than all the relatively less active groups. This shows that for each discrete value of rating, the probability that the users from the relatively less active groups giving a rating that is equal or lower to that rating, is higher than that of the 'Most active' groups. This implies that the cumulative distribution curve of the 'Most active' group first-order stochastically dominates [8] all the other groups. Therefore, we can conclude that active users tend to give higher ratings.

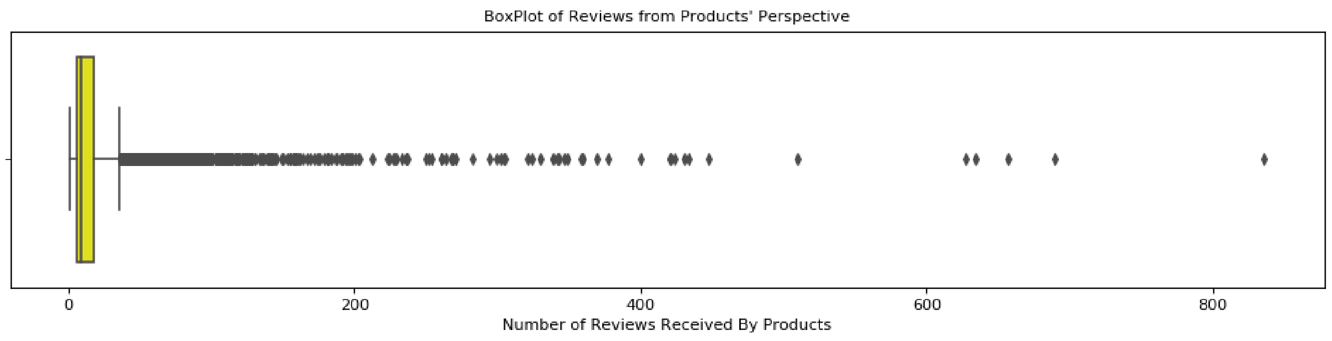

Figure 3: Boxplot from the products' perspective.

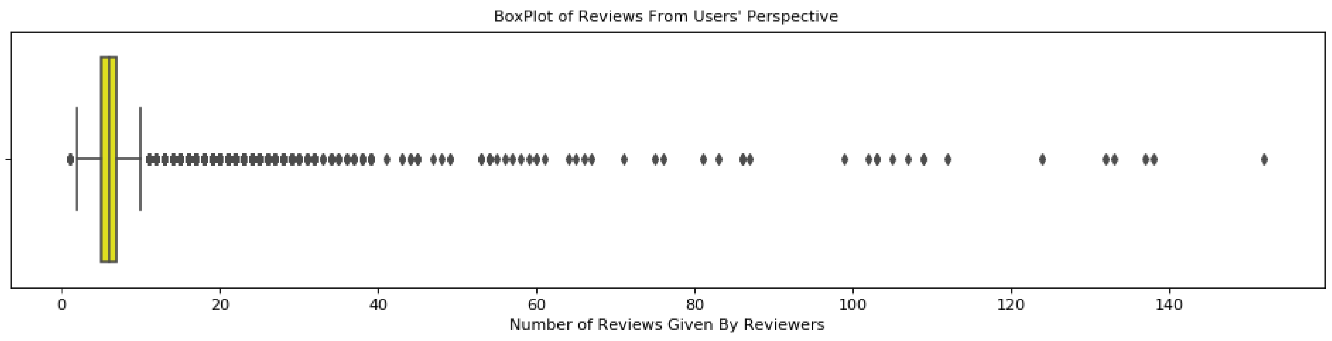

Figure 4: Boxplot from the users' perspective. This plot is also used to derive the 5 groups of active reviewers.

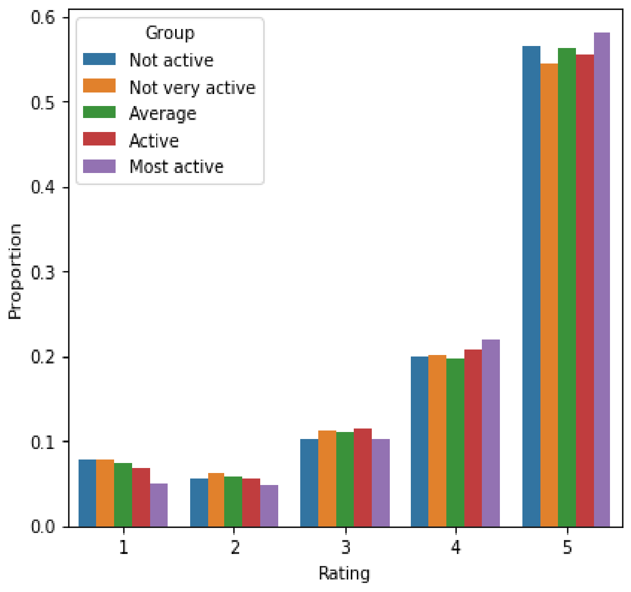

Figure 5: Ratings given by different groups.

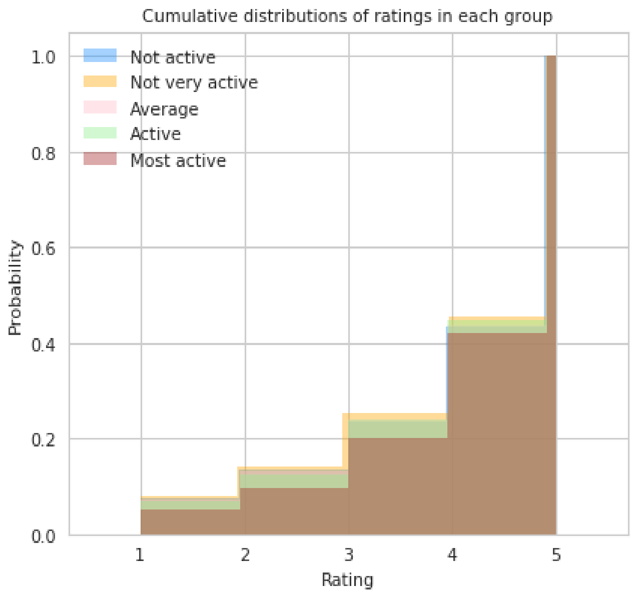

Figure 6: Cumulative distribution of ratings from each group of users.

1.2 Sentence Segmentation

In order to perform sentence segmentation, we exploit the function sent_tokenize() in NLTK [7] toolkit to segment raw review text. As shown in Figure 7, majority of the reviews contain 0 to 20 sentences and there are little reviews containing more than 20 sentences.

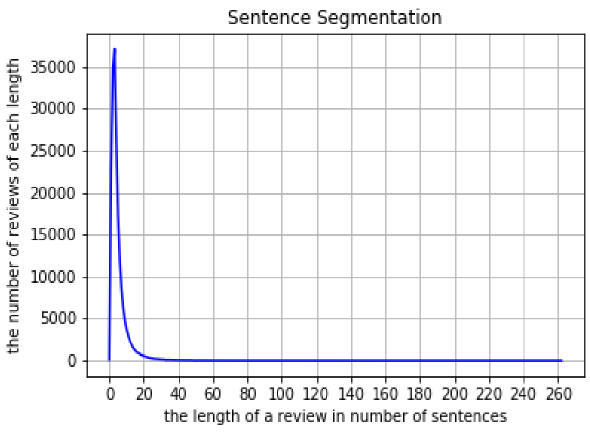

Figure 7: Overall distribution of sentences number over number of reviews.

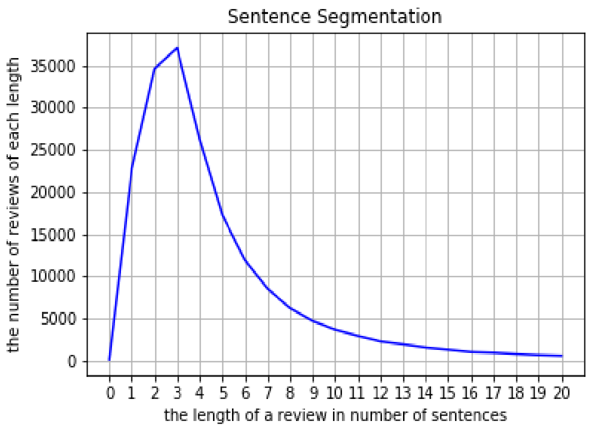

Figure 8: Partial distribution of sentences number over number of reviews between 0 to 20.

Furthermore, we narrow the x-axis range to 0-20 and the distribution is illustrated in Figure 8. It reveals that the number of reviews of each length begins a climb reaching a peak at around 36000 where the number of sentences in a review is at 3. After that, there is a steady decrease approaching 0.

1.3 Tokenization and Stemming

For this step of data analysis, we examine the number of unique words used in all the reviews. The reviews are first tokenized using the 'word_tokenize' function from 'nltk.tokenize' package. Subsequently, the tokens are then stemmed using 'PorterStemmer' and 'LancasterStemmer' from the 'nltk.stem' package. For both the tokens and the stemmed tokens, the number of reviews for each unique number of word count were extracted and plotted in the Figure 9.

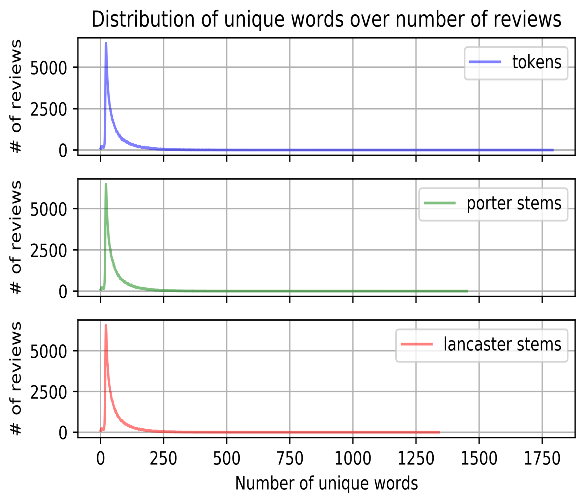

Figure 9: Unique word count over number of reviews.

A few observations can be made from the Figure 9. Firstly, the dataset contains large number of reviews that are of around 50 to 200 unique words, while a smaller number of reviews contain large number of unique words. The graphs all demonstrated a long-tail phenomenon, whereby a small number of events occur with high frequency and a large number of events occur with low frequency. This holds true even after stemming. Secondly, after token stemming by Porter, the number of reviews with more than 1,400 words were reduced to zero. This shows that in reviews with large number of words, many of them contain words of similar stems. Lastly, the maximum number of unique word count also depends on the stemmer used, Lancaster stemmer appears to be a more aggressive stemmer than Porter as it further reduce the maximum number of unique words to less than 1,300.

```
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer

tokens = [word_tokenize(review) for review in reviewtexts]
porter_stems = [[porter.stem(word) for word in tokenlist] for
↪  tokenlist in tokens]
lancaster_stems = [[lancaster.stem(word) for word in tokenlist]
↪  for tokenlist in tokens]
```

1.4 POS Tagging

In this segment, multiple methods were explored and analysed by performing 'pos_tag' function from 'nltk.pos_tag' package on different pre-processed functions. The json file is first read into a data frame, pre-processed to different categories, tokenized, stemmed followed by pos tagged on the different stemmed results

```
from nltk.stem.snowball import SnowballStemmer
stemmer_snowball = SnowballStemmer('english')
```

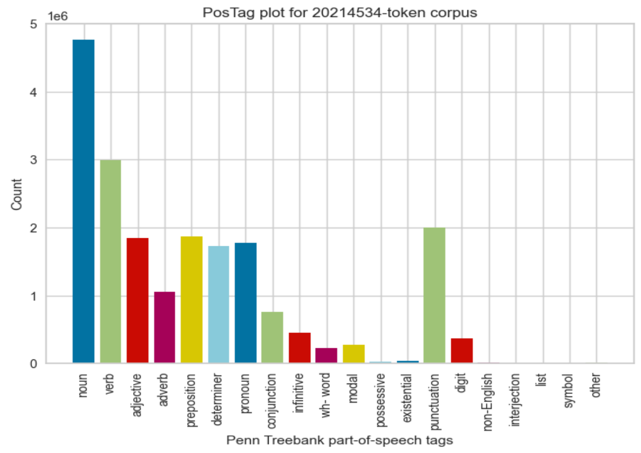

Figure 10: Pos tag categorisation for Porter Stemmed Words

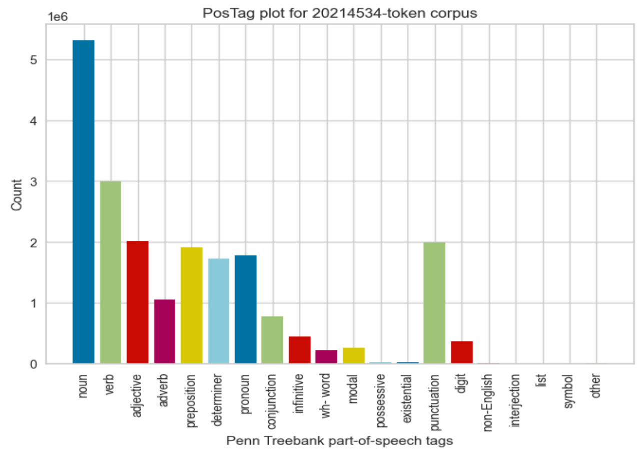

Figure 11: Pos tag categorisation for Lancaster Stemmed Words

```
    for text in text_list:
        tokens = nltk.word_tokenize(text)
       portstemmed = [stemmer_porter.stem(token) for token in tokens]
        lancasterstemmed = [stemmer_lancaster.stem(token) for token
        ↪  in tokens]
       SBstemmed = [stemmer_snowball.stem(token) for token in tokens]
        ptagged = nltk.pos_tag(portstemmed)
        ltagged = nltk.pos_tag(lancasterstemmed)
        stagged = nltk.pos_tag(SBstemmed)
```

These pos tag results were then clearly illustrated and better summarised with Postagvisualizer from yellowbricks package [1], using Penn Treebank part-of-speech tags.

It can be observed that among the huge database of reviews, there are more nouns and adjectives quantified after Lancaster, figure11 and Snowball stemming, figure 12 as compared to Porter stem, exemplified in figure 10. Snowball stemming has shown higher counts of verbs and determiners than Lancaster and Porter stemming as well. It can be conclusively analysed that among the reviews, verbs

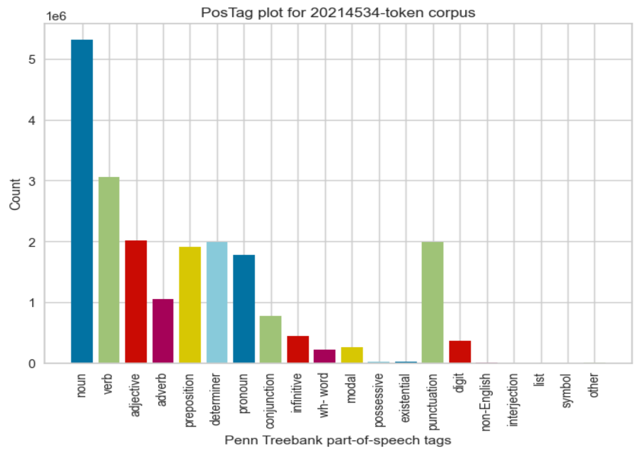

Figure 12: Pos tag categorisation for Snowball Stemmed Words

and adverbs are popular choice of words to describe the products which are reasonably nouns, thereby, depicting identifiable highest tag counts. Determiners and prepositions are not used commonly as compared to nouns but both are on almost equal utilisation frequency. Determiners are also on close counts to pronouns which may be referenced that the sentence structure of all the reviews could be written in 50-50 usage probability with either components.

2 DEVELOPMENT OF A REVIEW SUMMARIZER

The section elaborates on design and implementation of the review summarizer developed by the team. We will first describe what an ideal product summary is based on our understanding and the challenges in achieving it given the dataset. Next, we talk about our data preprocessing techniques followed by our proposed solution. Then, several baseline models implementation details will be given. Lastly, we will present the evaluation results of our proposed solution in comparison with the baseline models.

2.1 The Ideal Summary and Challenges

The review texts written by users on a particular product contain mainly opinions related to features of the product, sentiments towards usage of the product and overall evaluation or some recommendations for other users.

For popular products that received hundreds of reviews, it is difficult for potential customers to read them and make informed decisions. It may also be hard for sellers to understand customers' perspectives without a good summary. Therefore, having these in mind, we believe that an ideal product summary should be able to meet the following criteria:

- Capture the gist about the features of the product.
- Accurately reflect the general sentiments of users towards the product.
- The summary should contain few sentences that express diverse meanings.

- The summary should have a decent structure such that there is readability flow.

Now, we use a summary template to illustrate the structure of an ideal product summary. Assuming that we have already obtained the best sentences that meet the above criteria, the summary of a product should have the following structure and flow:

1. Topic Sentence:
The summary begins with this sentence. It should provide the reader an overview of the main feature of the product.
2. Sentiment Sentences:
This part of the summary should describe the positive or negative sentiments of users related to the main feature mentioned in the Topic Sentence. It may be one or two sentences long.
3. Detail Sentence:
This sentence should provide some details of the product that are not available in the previous sentences. It is also used to inject diversity to the summary.
4. Concluding Sentence:
A short sentence that reflects how users feel about this product as a whole followed by a possible recommendation to potential customers.

After defining the structure of our ideal summary, the problem now becomes finding the best sentences for each portion of the summary.

One of major challenges of finding the right sentence for our summary is working with an unlabelled dataset. As target summaries are not given, an unsupervised approach has to be used. Furthermore, since we have decided on a sentence-level extraction approach, we also have to ensure that the model we are building does not constantly pick the longest sentence as the representative sentence. This issue will surface when using a pure TF-IDF model without length normalization or clustering. In addition, to make the extracted sentences sound like a summary of all the reviews, we will need to take in consideration how a normal review is structured when designing our model.

2.2 Data Processing

We will be using TF-IDF score to embed our sentences and reviews. To prepare the dataset for TF-IDF scoring, each 'reviewTexts' are merged and stored as one long review passage using the product's unique ID, 'asin'. Our team mainly uses Spacy for text processing for this task. Spacy [3] is a powerful tool that automatically creates an informative "doc" from text inputs after passing through its NLP pipeline. Most of the time, certain parts of the pipelines were explicitly disabled in the function calls to improve efficiency.

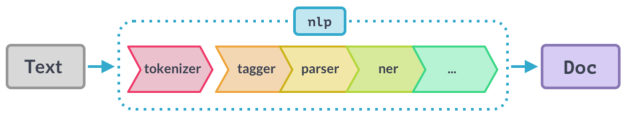

Figure 13: Spacy NLP Pipeline

As shown in Figure 13, Each word in the reviews are first tokenized. Subsequently, each token will then be converted to lower case and lemmatized. Punctuations and stop words are also removed from the tokenized list our team wishes to disregard them in the TF-IDF scoring of each sentence.

```python
sentnlp = spacy.load("en_core_web_sm",
    disable=['parser', 'ner', 'tagger'])
sentnlp.add_pipe(nlp.create_pipe('sentencizer'))

# Remove personal pronouns, stopwords, and punctuations
def cleanup_text(text, stopwords, punc):
    texts = []
    global sentnlp
    doc = sentnlp(text) # only do tokenization and sentencizer
    for sentence in doc.sents:
        sen_text = nlp(sentence.text, disable=['parser', 'ner'])
        tokens = [tok.lemma_.lower().strip() for tok in sen_text
        ↪  if tok.lemma_ != '-PRON-']
        tokens = [tok for tok in tokens if tok not in stopwords
        ↪  and tok not in punc]
        tokens = ' '.join(tokens)
        texts.append(tokens)
    return str(texts)
```

2.3 Our Model: TF-IDF Hybrid POS Summarizer

There exist two main text summarization approaches; extractive and abstractive. The former approach focuses on extracting words or phrases or sentences, concatenating them into one single summary. The latter methodology requires learning of the language representation to paraphrase the intent of the original text into one fluent summary.

Extractive text summarization techniques are being used in many state-of-the-art text summarization systems [6, 11, 12] due to its effectiveness in scoring and selecting the sentences in the original text. Similarly, our task in finding the best sentences for each part of the product summary aligns well with the extractive text summarization methodology. Therefore, we have decided to adopt an extractive approach in creating our product summarizer.

We named our model the TF-IDF Hybrid POS Summarizer because the model makes use of a mixture of techniques (i.e., vector space and clustering) together with POS tagging to pre-select sentences. The details will be explained in the following section.

2.3.1 The Algorithms. Moving on, we will describe and explain the extractive algorithm used at the sentence-level by our summarizer for selecting the best sentences for each part of the summary according to the summary template mentioned in Section 2.1.

Extracting the Topic Sentence (Step 1) : To extract the first sentence for our summary that contains the main feature information of the product, we first do POS taggings on all the raw sentences from all reviewers on this product. Next, we identify the most occurring noun from these sentences and propose that this noun represents the main feature of the product. Moving on, we extract the TF-IDF representations (Equation 1) of all sentences that contain this noun and form a cluster of sentences. Lastly, we extract the sentence closest to the cluster's centroid by measuring the Euclidean distance (Equation 2) between each sentence and the centroid. This sentence will act as a central representation that gives an overview of the main feature of the product.

$$tfidf(\mathbf{t}, \mathbf{d}, \mathbf{D}) = (1 + \log_{10} \mathbf{tf}_{t,d}) \times \log_{10} \frac{\mathbf{N}}{\mathbf{df}_t} \tag{1}$$

```python
import numpy as np
def get_tf_idf(sentences):
    "get tf-idf vectors for sentences"
    num_of_docs = len(unique_id)
    sent_tf_vecs = []
    for sentence in sentences:
        sent_tf_vec = np.zeros(len(word_frequencies))
        sent_word_frequencies = _word_frequency(sentence)
        for idx, word in enumerate(sent_word_frequencies):
            if (word in word_doc_frequencies):
              idx = all_words_ls.index(word)
              tfidf_value = (1 +
              ↪   np.log(sent_word_frequencies[word])) *
              ↪   np.log(num_of_docs/word_doc_frequencies[word])
              sent_tf_vec[idx] = tfidf_value
        sent_tf_vecs.append(sent_tf_vec)
    return sent_tf_vecs
```

$$eucd\,(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (\mathbf{q}_i - \mathbf{p}_i)^2} \tag{2}$$

```python
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import euclidean_distances
def get_center_vec(sent_tf_vecs):
    "get central sentence representation"
    est = KMeans(n_clusters=1,
    ↪   random_state=0).fit(np.array(sent_tf_vecs))

    closest_centers_idx = []
    indexes = np.arange(len(sent_tf_vecs))
    cluster_ids = est.labels_
    centroids = est.cluster_centers_

    # calculate each sentence's dist from its centroid
    dist_from_centroids = []
    for idx in range(len(sent_tf_vecs)):
       dist = euclidean_distances(sent_tf_vecs[idx].reshape(1,-1),
       ↪   centroids[cluster_ids[[idx]], :])
        dist_from_centroids.append(dist[0][0])

    dist_arr = np.concatenate([np.array([dist_from_centroids]),
    ↪   cluster_ids.reshape(1,-1), indexes.reshape(1,-1)], axis =
    ↪   0 )
    dist_arr = dist_arr.T

    # find sentences with shortest dist to centroids
    for cluster_id in range(len(centroids)):
        dist_arr_id = dist_arr[dist_arr[:, 1] == cluster_id]
        row_idx = dist_arr_id[:, 0].argmin()
        sent_idx = dist_arr_id[row_idx][-1]
        closest_centers_idx.append(sent_idx)

    best_vec = sent_tf_vecs[int(closest_centers_idx[0])]

    return best_vec, int(closest_centers_idx[0])
```

Extracting the Sentiment Sentences (Step 2) : Two sentences that describe the main feature of the product will be extracted here. Using the Pos Tags obtained in the previous step, we identify all sentences that contain adjectives and extract their corresponding TF-IDF sentence representation. These are all the descriptive sentences in the reviews. Next, for each of the descriptive sentence extracted, we compute its Cosine similarity (Equation 3) with the Topic Sentence extracted in the previous step. Finally, the two descriptive sentences with the highest Cosine similarity scores will be extracted. These two sentences will be used in our summary because they are the most relevant sentences describing the main feature of the product. Furthermore, these descriptive sentences can also express users' sentiments towards the product.

$$\cos(\mathbf{p}, \mathbf{d}) = \frac{\mathbf{pd}}{\|\mathbf{p}\|\|\mathbf{d}\|} = \frac{\sum_{i=1}^{n} \mathbf{p}_i \mathbf{d}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{p}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{d}_i)^2}} \tag{3}$$

```python
from numpy import dot
from numpy.linalg import norm
def cos_sim(a, b):
    return dot(a, b)/(norm(a)*norm(b))
```

Extracting the Detail Sentence (Step 3) : Next, we will look for the sentence that provides some essential product details which are not available in the previous parts of the summary. To do this, we simply pick the sentence with the highest average TF-IDF score from the remaining pool of unselected sentences. Empirically, we have found that doing so will result in a slightly longer sentence that contains unique details of users' experiences with the product. This sentence also adds diversity to the entire summary.

Extracting the Concluding Sentence (Step 4) : Lastly, we will search for a decent concluding sentence to end off the summary. We gather all the last sentences from all reviewers of this product. Similar to how we extract the Topic Sentence, we form a cluster of last sentences and pick a central representative. This concluding sentence will show users' overall thoughts about the product and possible recommendations.

Table 2 illustrates the summary output from our model for a phone case product. It also explains how each of the output sentence fulfils the criteria of an ideal product summary specified in Section 2.1. In general, we can see from the output that the summary is concise in describing the phone case and users' opinions. Most importantly, the sentences are coherent and the summary has good readability flow.

2.4 Baseline Models

In order to compare our TF-IDF Hybrid POS Summarizer, we also implemented two baseline models that make use of vector space and clustering respectively. Similar to our TF-IDF Hybrid POS Summarizer, the baseline models also use TF-IDF for sentence embedding. These models will attempt to pick sentences that best reflect the opinions of all reviewers. Vector space and centroid-based text summarization models are widely used [4, 10] in the community and we believe that they are reasonable baseline models for comparison. Although our models' implementations do not follow exactly that of [4, 10], we borrow their general ideas for our baseline models.

2.4.1 *TF-IDF Vector Space Summarizer.*

The first baseline model is the Vector Space model. For a particular product, we treat all the reviews received as a document and create a TF-IDF vector to represent this document. Then, for each TF-IDF sentence vector in the document, we measure its Cosine similarity with the document vector. Lastly, we extract the top five sentences with the highest Cosine similarity as the summary.

Table 2: Summary output from the TF-IDF Hybrid POS model for a phone case.

Parts of the summary	Model Output	Remarks
Topic Sentence	I love this case!	The word ’case’ is identified as the most occurring noun and this opening sentence gives reader an impression on what the product is.
Sentiment Sentences	I have many as I love. Another pretty phone case that I really love.	Adjectives such as ’love’ and ’pretty’ show reviewers’ positive sentiments towards the product.
Detail Sentence	The "rubberized" cover feels a little greasy (like it was armor-all’d), but the design is gorgeous and a little 3d looking!	This sentence provides specific details on the case’s texture and design. It also reveals what is not so good about the product.
Concluding Sentence	Overall, would recommend.	A short concluding statement giving recommendation to potential buyers.

2.4.2 TF-IDF K-Means Summarizer. Another baseline model we can use is the centroid-based model. We extract the TF-IDF representations of all sentences from all the reviews users wrote on a product. Next, we use the implementation of K-Means clustering algorithm by Scikit-learn [9] to identify five clusters of sentences. Finally, for each cluster, we pick one sentence that is closest in terms of Euclidean distance to the cluster’s centroid. These sentences are also the most representative sentences for each cluster and they come together to form the product summary.

2.5 Results and Evaluation

The team evaluated our Hybrid POS summarizer against the baseline models : Vector Space and K-Means summarizers through a manual human rating system via Google Forms[1]. We asked 21 participants to rate the summaries of each product based on two metrics, namely conciseness and readability. This is in line with our definition of the ideal summary whereby we want a review summary to contain as much gist about the features of the product as possible within a pre-defined number of sentences.

For this evaluation, we first selected three random product reviews (denoted as #1, #2, #3) to generate summaries of five sentences using the three models. In order to see the effectiveness of summarizing long review texts, the team also generated summaries for the product (denoted as #4) found with the most and longest total reviews length. In order to add more variance to the reviews, we also added a fourth summary model for each product reviews that randomly selects five sentences from all the product reviews. We denoted this model “Random”.

```python
from random import choice
prod_reviews_texts = eval(prod_reviews['reviewSentences'][index])
range_of_indices = list(range(len(prod_reviews_texts)))
summary_sentences = []
for i in range(num_of_sentences):
    randIdx = choice(range_of_indices)
    summary_sentences.append(prod_reviews_texts[randIdx])
    range_of_indices.remove(randIdx)
```

These summaries along with the original product reviews are then added to the Google Form. For each product, the respondents were first asked to read through the original full review text (maximum of 100 sentences). Subsequently, the participants will give a rating of either “Terrible”, “Bad”, “Average”, “Good” or “Superb” for the conciseness and readability of each of the summaries based on the original text they just read. This evaluation process will repeat for all four product reviews. To prevent any form of biasness, the participants do not see the summaries in the same order. Instead, the sequence of the summaries shown to them is randomized for all products each time.

After collecting the survey results from all 21 respondents, the team processed the ratings from textual sentiment form to numerical form. We represented the negative sentiment to positive sentiment towards the conciseness and readability of the summaries using 1 to 5. After counting and tabulating the results, the team found some interesting observations from the evaluation surveys.

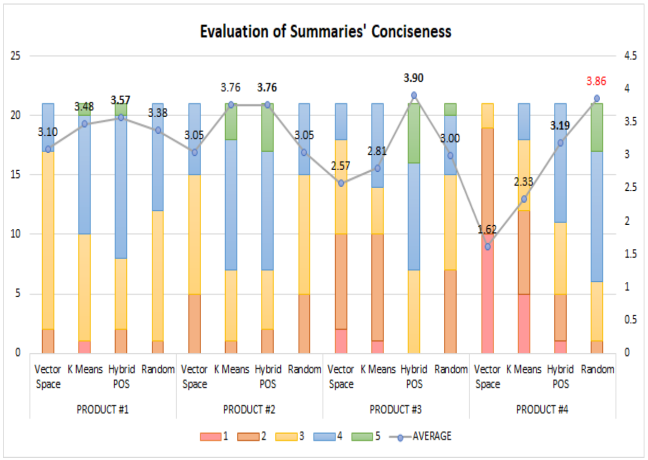

Figure 14: Evaluation results in conciseness metric

As seen in Figure 14, our Hybrid POS summarizer generally performed better than the other alternative models in terms of conciseness. This is true except for the anomaly case of Product #4, which is the product with the longest original review text. For product #4, our “Random” model coincidentally managed to choose five shorter length sentences to form a summary. Due to the long length of the original text, the five representative sentences chosen by our Hybrid POS model contained more references to the different

[1]https://forms.gle/EPZrkViFRJVPQeEp6

features of the product, which therefore lead to a slightly longer summary than the random summary. As speculated, the Vector Space model fared badly in terms of conciseness. This is because it focuses on selecting sentences of the highest word frequencies (TF-IDF), which are often lengthy. The K-Means model produces better results than the Vector Space model as it takes the centroids into consideration.

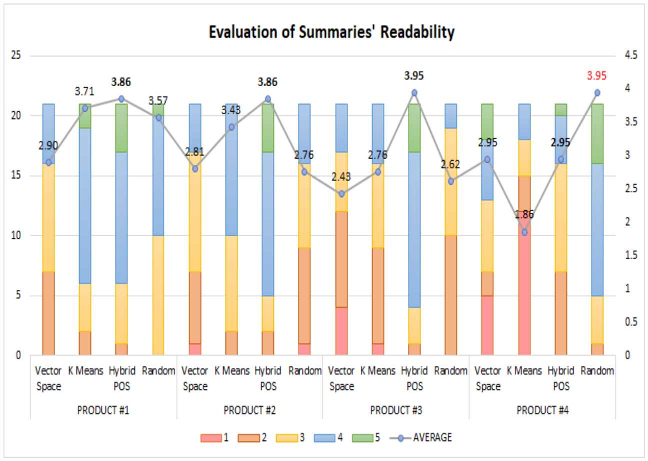

Figure 15: Evaluation results in readability metric

As seen in Figure 15, our Hybrid POS summarizer also generally performed better than the other alternative models in terms of readability. This is again true except for the anomaly case of Product #4. Our team believes the previous high score in conciseness plays a major part in the resulting high readability scores for the random summary for Product #4.

Most of the time, Vector Space's summaries performed the worst in terms of readability. However, for Product #4, its readability looks to be on par with our Hybrid POS model. This is because Vector Space actually generated the shortest summary of *"Why not? I did! What else can i say!? What's more? Get one!"* for Product #4. It was appropriately rated by our reviewers as not concise (Figure 14) since it does not contain any meaningful features about the product. However, this summary is undeniably human readable, therefore garnered its high readability ratings. This is within expectations given the knowledge of the Data Analysis in Section 1. With most sentences of the reviews are of short length, hence for products with many reviews (Product #4 has the most reviews), the TF-IDF of common short expressions will also return a higher score. This further proves the reason why the Vector Space model is not an effective model for review summarization.

Table 3: Overall Average Scores of the 4 Summarizers

Overall Avg	VectorSpace	K Means	**Hybrid POS**	Random
Conciseness	2.58	3.10	**3.61**	3.32
Readability	2.77	2.94	**3.65**	3.23

Finally, Table 3 shows the overall average scores of the summaries of the four products reviews. The results clearly indicate that on average, our Hybrid POS model outperformed all other models in terms of both conciseness and human readability through the algorithm we designed. Refer to Appendix C for the summaries for the three random products generated by our Hybrid POS Summarizer.

In addition to these quantitative results, we also want to analyse the performances of our models qualitatively. Table 4 shows the summary output from each model for Product #2: a phone case. The Vector Space model often extracts sentences that are heavy in details and mention multiple aspects (e.g., colour, design and texture) of a product. Such summary may be difficult to follow as reader may not be aware of the main idea that the summary is trying to put across. The sentences produced by K-Means is shorter and seems more concise as compared to that of Vector Space. However, these sentences do not connect well with each other and appears to be random such that they resembles the sentences produced by the Random model. More specifically, the K-Means summary starts off by talking about the design and and colour of the phone case showing a personal preference over a certain colour. This is followed by a sentence regarding the texture of the phone case. The summary finally ends by mentioning another user's colour preference which is not well aligned with the previous sentence about the product's colour. In contrast, our Hybrid POS model combines the strengths of Vector Space and K-Means to output sentences that are short, concise and rich in essential details. Most importantly, the summary possesses a good structural flow such that it is humanly readable.

3 APPLICATION

In this section, we establish text sentiment classifiers to detect sentiment opinions in a given text. Users are able to input a text review to our application and get the corresponding sentiment polarity which is positive, neutral or negative. We use the traditional logistical regression (LR) method and deep neural network (DNN) based on this Amazon cell phone review.

3.1 Data Pre-processing

Sentiment scores in the raw dataset range from one (most negative) to five (most positive). We reset the original five categories to three classes including negative, neutral as well as positive. We create three word cloud figures [18,19,20], as displayed in Appendix A, for these three sentiment categories. The distribution of three classes is illustrated in Figure 16. Data with positive labels is more than 7 times that of negative or neutral labels. This imbalance problem poses a challenge to our work. If the score is greater than three, we think it is a positive review. Otherwise, it's negative. If its score is exactly equal to three, it's neutral. Before training, we need to split the original dataset to training set and validation set. We train the model on the training set and utilize validation set to select the best model and avoid overfitting. In total, there are 190,919 samples in the cell phone review corpus. We split the data set 70/30 by using the function train_test_split() in Keras [2], which tries to keep the sentiment distribution in training and validation dataset as same as the original one. The training set holds 133,643 items, and the remaining 57,276 samples are in the validation set.

Table 4: Summary outputs from different models for a phone case product. The summary output from our Hybrid POS model scores the highest in terms of conciseness and readability.

Vector Space: It has a cool 3d effect but at the cost of the paint rubbing off. I have many as i love to change them around. Thanks the case is not white, its more of a silver. I was worried about this order because the picture on the description page kept changing to a less-desireable green/orange cover. The idea of the design is a sweet idea but it wears off as the paint is over the case and not under a sealant or anything. The "rubberized" cover feels a little greasy (like it was armor-all'd), but the design is gorgeous and a little 3d looking!
K-Means: I love this case! The idea of the design is a sweet idea but it wears off as the paint is over the case and not under a sealant or anything. I was worried about this order because the picture on the description page kept changing to a less-desireable green/orange cover. The "rubberized" cover feels a little greasy (like it was armor-all'd), but the design is gorgeous and a little 3d looking! I have many as i love to change them around. Thanks the case is not white, its more of a silver.
Hybrid POS: I love this case! I have many as i love. Another pretty phone case that I really love. The "rubberized" cover feels a little greasy (like it was armor-all'd), but the design is gorgeous and a little 3d looking! Overall, would recommend.
Random: I have many as i love to change them around. Thanks the case is not white, its more of a silver. It was an excellent buy! The "rubberized" cover feels a little greasy (like it was armor-all'd), but the design is gorgeous and a little 3d looking! And i love the way the case feels to the touch because of the rubber. I was worried about this order because the picture on the description page kept changing to a less-desireable green/orange cover.

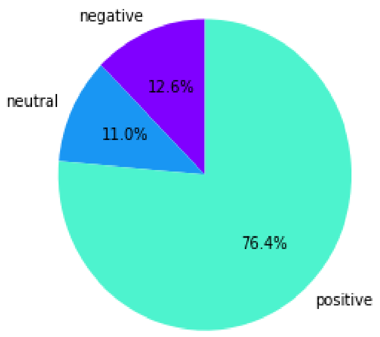

Figure 16: Distribution of positive, neutral and negative sentiment polarities.

In LR and DNN models, we keep stop words, removing several punctuation marks and convert uppercase letters to lowercase letters. Because there are many negative Words in public stop words lists, which are important in sentiment analysis. DNN models accept input sentence with fixed length. Because of this, we do padding or cutoff, and we set the fix length to 200.

3.2 Logistical Regression

The logistical regression model gives the probability that a review belonging to each sentiment class. For feature extraction, we employ TF-IDF as input to the LR model.

Firstly, we introduce CountVectorizer in the Scikit-learn [9] toolkit to encode the training set to a matrix of token counts. We tried to remove stop words with help of CountVectorizer. Next, we exploit TfidfTransformer to compute TF-IDF value by fitting and transforming the previous count matrix. Then, we train a logistic regression model according to the given training data. Finally, we evaluate this LR model on the validation set and analyze all features.

```
countVector = CountVectorizer(stop_words='english')
X_train_counts = countVector.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_test_counts = countVector.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
logreg =
↪  LogisticRegression(C=1e5,multi_class='multinomial',max_iter=300)
logreg_result = logreg.fit(X_train_tfidf, Y_train)
prediction['removing_stop_words'] = logreg.predict(X_test_tfidf)
```

3.3 Deep Neural Network

In order to improve performance, we construct a Bi-LSTM model followed by attention, max pooling and average pooling mechanisms. The model architecture is illustrated in Figure 17.

```
class bilstm_attn(torch.nn.Module):
    def __init__(self, batch_size, output_size, hidden_size,
    ↪  vocab_size, embed_dim, weights, bidirectional, dropout,
                 use_cuda, attention_size, sequence_length):
        super(bilstm_attn, self).__init__()

        self.lookup_table = nn.Embedding(self.vocab_size,
        ↪  self.embed_dim)
        self.lookup_table.weight = nn.Parameter(weights,
        ↪  requires_grad=False)
        self.layer_size = 1
        self.lstm = nn.LSTM(self.embed_dim,
                            self.hidden_size,
                            self.layer_size,
                            bidirectional=self.bidirectional)

        if self.bidirectional:
            self.layer_size = self.layer_size * 2
        else:
            self.layer_size = self.layer_size

        if self.use_cuda:
            self.w_omega = Variable(torch.zeros(self.hidden_size *
            ↪  self.layer_size, self.attention_size).cuda())
            self.u_omega =
            ↪  Variable(torch.zeros(self.attention_size).cuda())

        self.label = nn.Linear(hidden_size * self.layer_size * 3,
        ↪  output_size)
        self.dropout = nn.Dropout(dropout)
        self.max_pool = nn.AdaptiveMaxPool1d(1)
        self.avg_pool = nn.AdaptiveAvgPool1d(1)
        self.softmax = nn.Softmax(dim=1)
        self.init_weights()
```

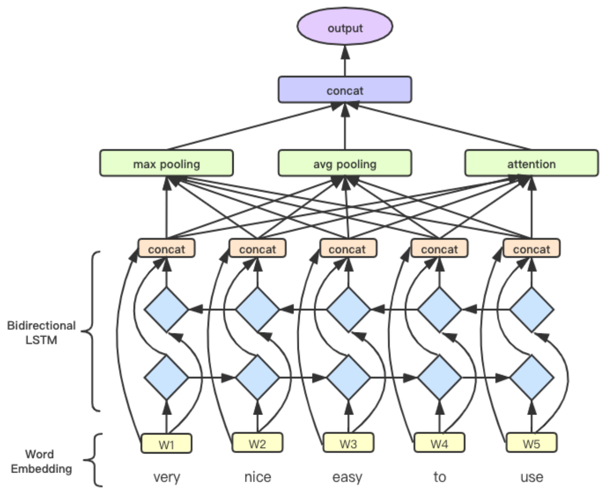

Figure 17: BiLSTM Attention based model.

For the embedding layer, we introduce a 300-dimensional pre-trained GloVe word embeddings[2]. We tried to learn word embeddings based on training set. However, the experiment shows that this model overfits soon. The raw sentence is $s = (w_1, \ldots, w_t, \ldots, w_n)$,$1 \leq t \leq T$. Since, in the Section 3.1, we've mentioned that the fixed length is 200, so that $T = 200$. After passing through the word embedding layer, the sentence representation is $s = (v_1, \ldots, v_t, \ldots, v_T)$, $v_t \in \mathbb{R}$, $d = 300$. v_t is the word vector of word w_t. Then, the forward (left to right) and backward (right to left) LSTM layers encode the sentence from two directions respectively, computing both directional features:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(\overrightarrow{h_{t-1}}, v_t), \overleftarrow{h_t} = \overleftarrow{LSTM}(\overleftarrow{h_{t-1}}, v_t) \tag{4}$$

The final output of the t^{th} word is displayed as follows:

$$h_t = [\overrightarrow{h_t} || \overleftarrow{h_t}], t \in [1, T] \tag{5}$$

Next, we employ attention, max pooling and average pooling mechanisms on these concatenated context features. After directly concatenating these three different features, we apply a linear layer to compute the label that the input sample belong to.

3.4 Results

Experiments are conducted on the Amazon cellphone review dataset. After preprocessing, this dataset contains three labels (positive, negative, neutral). There are 133,643 samples for training and 57,276 items for validation. We exploit accuracy and F1-score to evaluate each model.

For logistical regression, we implement two variants with tiny difference including removing stop words and containing these words. The results are displayed in Table 5.

LR models are able to achieve more than 79% accuracy. On the F1-score, both of them are less than 60%. About whether keeping or deleting stop words, the results shows that in sentiment analysis task it's better to keep stop words. We keep this setting in the follow-up experiments. The results demonstrate that both accuracy and F1-score are enhanced by around 2 points.

[2]https://nlp.stanford.edu/projects/glove/

Regarding the deep neural network, we use the 300-dimensional pre-trained GloVe word vectors to initialize the word embedding layer. The optimizer is Adam[5] with a learning rate of 0.001. The batch size is 32 and the hidden size is 128. The drop out is 0.5. The sequence length and attention size are both 200. The loss function is cross entropy.

Due to the imbalance problem in sentiment analysis, we solve this by adding weights to the loss function. By doing this, we want to train our model to focus on negative and neutral samples. If we don't take any measures to solve this imbalance problem, the model tends to label every sample as positive. In this case, although model reaches high scores on our criteria, it can't achieve our purpose to classify correctly. We changed the loss weight of each label according to each label's proportion in training set. In the experiments, we set the weights [1.0, 6.07, 6.88] to positive, negative and neutral categories respectively. The results are shown in Table 6.

BiLSTM models perform better than traditional LR models. On accuracy, the balanced model improved by 5.34 point than the n-grams LR model. For the F1-score, the weighted DNN model reaches 67.03%. Compared with the best LR model of 59.50% and the balanced DNN of 63.96%, it's a huge improvement. Some outputs of test samples are illustrated in Table 7.

LR models	Accuracy	Precision	Recall	F1-score
removing stop words	79.24	58.61	55.92	57.03
containing stop words	**80.11**	**60.54**	**58.69**	**59.50**

Table 5: Logistical Regression results.

	Accuracy	Precision	Recall	F1-score
balanced loss	**85.45**	**72.56**	61.55	63.96
weighted loss	80.00	64.54	**71.89**	**67.03**

Table 6: Deep Neural Network results.

4 CONTRIBUTIONS

This section will state the respective contributions of each member for this project. The team split the Data Analysis task into four subtasks for each of the members to analyze and report their findings. Popular Products and Frequent Reviewers in Section 1.1 was done by Tan Mengxuan. Sentence Segmentation in Section 1.2 was done by He Linzi. Tokenization and Stemming in Section 1.3 was done by Ong Jia Hui. POS Tagging in Section 1.4 is done by Irene Ng Yu Si. The Review Summarizer task in Section 2 was jointly done by Tan Mengxuan and Ong Jia Hui, while the Sentiment Analysis Application in Section 3 was done by He Linzi.

Table 7: Sentiment classification outputs of several test samples.

Review Text	Ground Truth	LR Model	Balanced Model	Weighted Model
I have not used it for data transfer but it sucks for charging	Negative	Negative	Positive	Negative
the case does not cover the sides of the phone but I like the bling on the back very much	Neutral	Neutral	Positive	Neutral
I liked this product but found that it would not fit my son's phone as his is an iphone 5s If I had not purchased a different case for my phone I would have kept this for myself	Positive	Neutral	Neutral	Neutral

REFERENCES

[1] Benjamin Bengfort, Rebecca Bilbro, Nathan Danielsen, Larry Gray, Kristen McIntyre, Prema Roman, Zijie Poh, et al. 2018. *Yellowbrick.* https://doi.org/10.5281/zenodo.1206264

[2] François Chollet et al. 2015. Keras. https://keras.io.

[3] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017). To appear.

[4] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC).* 31–39.

[5] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980 cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[6] Yang Liu. 2019. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318* (2019).

[7] Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

[8] MIT. 2015. *14.123 S15 Microeconomic Theory III, Stochastic Dominance Lecture Notes.* https://ocw.mit.edu/courses/economics/14-123-microeconomic-theory-iii-spring-2015/lecture-notes-and-slides/MIT14_123S15_Chap4.pdf

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[10] Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management* 40, 6 (2004), 919–938.

[11] Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 95–104.

[12] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. *arXiv preprint arXiv:1807.02305* (2018).

Appendix A SENTIMENT DATA VISUALIZATION

For data visualization, we generated the word cloud for each sentiment label, as shown in Figure 18, Figure 19 and Figure 20 respectively. We tried to use review text at first, but the they are too long to extract key words.

In that case, there's not much difference between the three word cloud figures because they contain too many same words. These three word cloud figures are generated from the summary fields. In positive review summaries, there are lots of positive descriptions like 'love it', 'five starts' and 'works great'. In contrast, regrading to negative reviews, users tend to describe like 'doesn't work', 'waste money' and 'junk'.

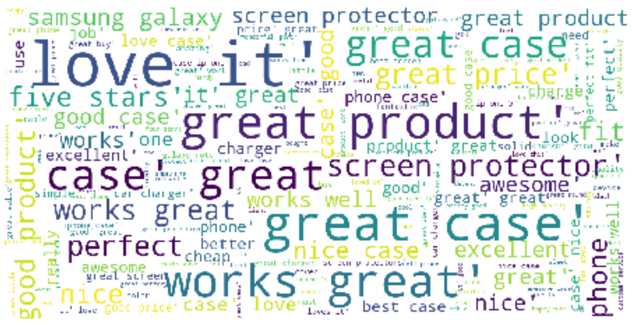

Figure 18: Positive review word cloud.

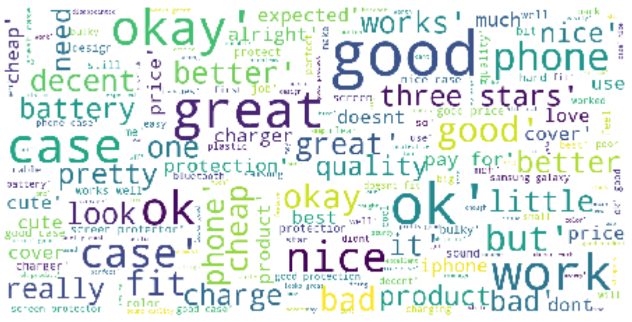

Figure 19: Neutral review word cloud.

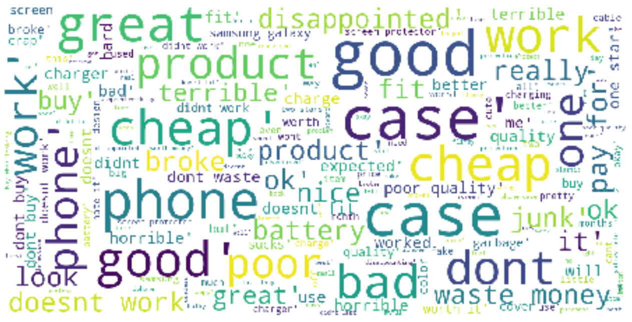

Figure 20: Negative review word cloud.

Appendix B LOGISTICAL REGRESSION FEATURES VISUALIZATION

Several features sorted by their coefficients are displayed in Table 8.

Table 8: Logistical Regression features.

	feature	coefficient
62417	hesitate	-17.741346
33666	compliments	-16.890980
64170	hugs	-16.203985
44422	drove	-14.435323
76553	loves	-14.419846
...	...	...
73304	laughable	15.108552
115159	slowest	15.282027
104011	rebooting	15.340524
10866	abysmal	15.398426
66217	inconsistent	15.876259

Appendix C HYBRID POS SUMMARIZER OUTPUT

Table 9: Summaries generated by Hybrid POS Summarizer.

Random Product #1: Very good charger. Most chargers would have broken by now. It is a genuine blackberry charger. The folding blades make this more portable than the chargers that come with some blackberry models that have adapters for different countries. Overall, for the price, i don't think you can ask for much more than that!
Random Product #2: I love this case! I have many as i love Another pretty phone case that i really love. The "rubberized" cover feels a little greasy (like it was armor-all'd), but the design is gorgeous and a little 3d looking! Overall, would recommend.
Random Product #3: The unit works boosting weak cell phone signals. This will boost multiple phones or 3g ipads. Sometimes we only have one or two bars and it boosts my iphone to full bars and 3g. I have used wilson equipment since the analog days - on an offroad trip years ago to the middle of nevada with no service, i was able to stop and make a call using the wilson amplifier and a high gain directional antenna to a cell site 83 miles away in Bishop, CA. Overall, it's highly recommended.