Sentiment Analysis and Feature Extraction on Phone Product Reviews

Course project for CSE 6240: Web Search and Text Mining, Spring 2020

Jiatuan Luo
jluo324@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Shuyu Ding
sding64@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Qihang Zhang
qzhang333@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

KEYWORDS

Text mining, sentiment analysis, natural language processing, feature extraction

ACM Reference Format:

Jiatuan Luo, Shuyu Ding, and Qihang Zhang. 2020. Sentiment Analysis and Feature Extraction on Phone Product Reviews Course project for CSE 6240: Web Search and Text Mining, Spring 2020. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

Customer reviews not only help buyers to make better purchase decision but also enable companies to improve their products. In this project, we plan to train an LSTM network on the *Amazon Reviews: Unlocked Mobile Phone from Kaggle* [9] data set to classify the correct customer sentiment from given product reviews. Besides, we plan to perform characteristic extraction on the given reviews and summarize the customers' opinions on those characteristics and give business insights to the companies. The three baselines we used to compare with our LSTM model are Random Forest [2], K-Nearest Neighbors (KNN), and Naive Bayes[12]. Our LSTM model outperformed the baselines with the highest accuracy (0.877), precision (0.895) and F1 (0.874) scores and the tied highest recall (0.852) with KNN. We also successfully extracted features for the top 7 phone brands with most reviews and summarized customers' opinions on those features for each brand. The summaries we provide can give good business insights to phone manufacturers.

2 INTRODUCTION & MOTIVATION

An important advantage of online shopping is that we as customers can see other people's opinions on a product. We often find ourselves relying on a product's rating and comments to come to a purchase decision. From the companies' perspective, these opinions are valuable in informing them what could be the main reasons that the consumers are dissatisfied with their products and why. It also helps them understand the strengths and weaknesses of their products and to get an idea of the general taste of the public in order to continuously improve their products.

For our project, we will perform sentiment analysis on large amounts of product comments for mobile phones to extract core information from them. We are interested in whether customers are satisfied or dissatisfied with a product's different features, and how the sentiment manifests in a comment. We will extract the features which the reviewers are discussing in the comment, and their sentiment towards these features. This will help us understand the aspects in which the customers are not satisfied with a product. And lastly, we will train a deep learning based model to predict labels for new comments. Knowing the customers' collective opinions on different features of a product will help e-commerce companies understand their customer base, develop new marketing strategies, and tailor their next product to a wider range of consumers, which can lead to increase in both sales and the company's reputation.

Conference'17, July 2017, Washington, DC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 LITERATURE REVIEW

Before our project, several research papers have already studied the problem of document feature extraction. In this section, we are going to briefly introduce a number of related works to our project.

Wei and Gulla(2010) proposed a HL-SOT(Hierarchical Learning (HL) process with a defined Sentiment Ontology Tree (SOT)) approach to label product's attributes in product reviews. This method could detect the review contents related to other products automatically and labeling it to the corresponding product. This method also could distinguish specific sub attributes belonging to one main attribute regardless of domain knowledge[10].

Lu and Zhai(2008) also proposed a semi-supervised learning method regarding automatically interpret opinions/reviews. They treated the experts reviews retrieved from CNET as a guidance and use PLSA(Probabilistic latent semantic analysis) model with Maximum A Posterior (MAP) estimation to separate ordinary review alignments of expert opinions and remaining reviews that cannot be well-aligned with the expert review. For opinions aligned to an expert review, they also used a similar model to separate those which are similar to the review and which are supplementary for this review. This method lifts the restriction that domain knowledge requirement to create labels and generate an easily digested opinion summary[6]. Key drawback of this methodology is it could not be applied for products missing professional reviews.

Zhai et cl.(2011) proposed a semi-supervised method, using limited lexical similarity to find some labels and then used EM algorithm based on naive Bayesian classification to assign a class to each unlabeled expression[11].

Zhang et cl.(2012) introduced a system, Weakness Finder, to find product weakness of the products from Chinese products reviews. They utilized similarity of the words based on Chinese WordNet-Hownet to group explicit features from the dataset and combined association rules with collocation selection method (PMI $*$ frequency) to find implicit features. They also applied sentence level sentiment analysis method, calculating polarity of each sentence to identify its orientation[13]. However, the features grouping method based on similarity cannot handle domain synonyms issue and the method to find implicit features may not work in English environment.

4 DATA DESCRIPTION

4.1 Data Preparation

The dataset we used for this project is named *Amazon Reviews: Unlocked Mobile Phone from Kaggle* (https://www.kaggle.com/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones). The raw dataset contains more than 400,000 instances with six features, which are **Product Name**, **Brand Name**, **Price**, **Rating**, **Review**, and **Review Votes (How many people found this review useful).**

The whole dataset contains 413,840 data entries. We first decided to remove all entries that did not leave reviews, and the total number dropped to 413,778. Since only about one-third of the data in **Review Votes** is non-zero (135,397), we determined this column is not useful for our project and dropped this column. 5,933 entries of the whole data have missing values in **Price**, and we imputed the missing values with the mean of the whole data. In the data preparation stage, in order to make the original data suitable for sentiment analysis, we first added a new column named **Label** that contains the true attitude of each review. The **Label** column is transformed from the **Rating** column, where a rating above 3-star is considered as a positive attitude, a 3-star rating is considered as neutral, and a rating below 3-star is a negative attitude.

After we have obtained the ground truth, we then cleaned up the reviews to make them suitable for sentiment analysis. To clean up the reviews, we performed HTML tag removal, non-letter characters, and stop-word removal. Furthermore, we also manually set up a list of synonyms and standardized those words (For example, *picture* and *photo* are synonyms and *picture* is replaced by *photo*). After that, we used NLTK tokenizer to split each review into sentences, and trained a word2vec model to get the word embeddings. Last but not least, we clustered the word embeddings into 10 groups using K-Means, and used the result to design our matrix for sentiment classification.

The objective of this project is to train a good model that is able to classify the true sentiment of a customer regarding mobile phone products and get deeper business insight by looking further into customers' preference on the most significant phone characteristics and help the company prepare for their product iterations. This data set not only contains a sufficient number of product reviews (400k) for us to not only train the model but also split a part for validation. We also have enough reviews that can support us to do characteristic extractions and provide business insights.

4.2 Raw Data Statistics

After completing data preprocessing, we explored raw data statistics. Here is a brief summary of the important properties of our dataset.

- The total number of reviews is 413,778 and the vocabulary size is 60,689 after data cleaning.
- The average number of sentences per text is around 2.958.
- The average number of tokens per sentence is 13.92.
- After using fuzzy-wuzzy to solve partial data entry inconsistent issues, there are 292 different cell phone brand names.
- 72,337 (17.5%) are 1-star ratings, 24,724 (6.0%) are 2-star ratings, 31,763 (7.7%) are 3-star ratings, 61,374 (14.8%) are 4-star ratings, and 223,580 (54%) are 5-star ratings.

Table 1 shows the summary statistics of the price, the number of sentence per review, and the number of tokens per sentence.

	Price	Sentence Per Review	Token Per Sentence
Mean	226.867	2.958	13.921
Std. Dev.	273.019	4.257	14.446
Min	1.730	1.000	0.000
Median	144.710	2.000	10.000
Max	2598.000	222.000	742.000

Table 1: Summary statistics of the price, the number of sentence per review, and the number of tokens per sentence.

4.3 Data Analysis

First, we are interested in whether the proportion of positive ratings is different among some major brands. In terms of major brands, we selected the top 10 most frequently appeared brands in our dataset. From the figure below, we can see that among the ten brands, nine out of ten brands have above sixty percent of positive ratings and show clear differences between the proportion of positive attitude and negative attitude, but **CNPGD**. We would like to know how the company manages to operate as a popular phone brand even though the company has more dissatisfied customers. It turns out that the reason why the company is popular is because the price range of **CNPGD**'s phone ($29.99 - $99.99) is much lower than the mean and median price of the dataset. This observation leads us to our second data analysis approach, which is whether the phone price is correlated with the ratings. Figure 1 shows the proportion of positive/neutral/negative reviews of the top 10 most popular phone brands.

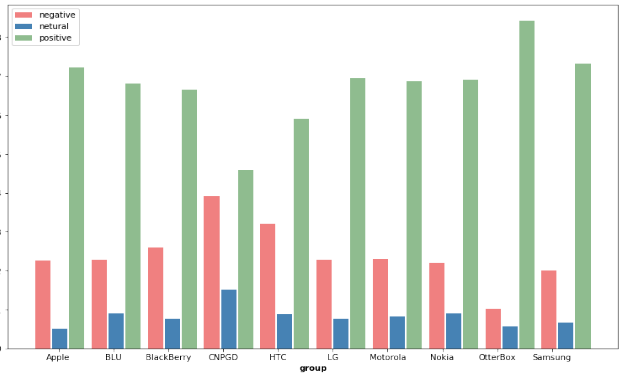

Figure 1: Proportion of positive/neutral/negative reviews of the top 10 most popular phone brands

Second, we looked into whether there is a correlation between the price of the phone and the sentiment of the customer feedback. The correlation coefficient between the price and class label is around 0.096. From the score we cannot conclude that there is a correlation between phone price and customer's opinion. Furthermore, we conducted ANOVA to see whether there is a true mean difference in phone price among the three classes. The F-statistics is 939.36 and the p-value is 0.0. From the below table, we see that the difference among the mean price of three opinion groups is statistically significant, where the mean phone price of positive reviews is bigger than that of negative reviews by $55.13, and bigger than that of neutral reviews by $41.92. On the other hand, the mean price of neutral review is lower than that of negative reviews by $13.21. Figure 2 shows the pairwise Tukey's HSD results.

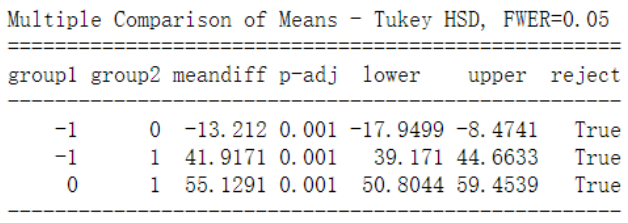
```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj  lower    upper  reject
----------------------------------------------------
    -1      0  -13.212 0.001 -17.9499 -8.4741   True
    -1      1  41.9171 0.001   39.171 44.6633   True
     0      1  55.1291 0.001  50.8044 59.4539   True
----------------------------------------------------
```

Figure 2: The Tukey's HSD results regarding the mean price difference among the three label classes

Third, we were also interested to see whether a customer's opinion is correlated with the length of the review. Same as the previous analysis, we looked at the correlation coefficient and performed ANOVA. The score of the correlation coefficient is around -0.28, the F-statistics is 6890.85 and the p-value is 0.0. From the table below, we see that the mean length of positive reviews is statistically significantly shorter than that of negative and neutral reviews by 5.76 sentences and 5.86 sentences, while the difference between mean lengths of negative and neutral reviews are not statistically significant. Figure 3 shows the pairwise Tukey's HSD results.

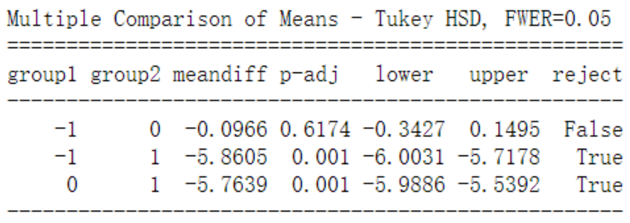
```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj   lower   upper  reject
----------------------------------------------------
    -1      0  -0.0966 0.6174 -0.3427  0.1495  False
    -1      1  -5.8605  0.001 -6.0031 -5.7178   True
     0      1  -5.7639  0.001 -5.9886 -5.5392   True
----------------------------------------------------
```

Figure 3: The Tukey's HSD results regarding the mean sentence per review difference among the three label classes

Fourth, we explored into the pros and cons of relabeling the ratings into three classes. After relabeling, 284,954 (68.8%) reviews are positive, 31,763 (7.7%) reviews are neutral, and 97,061 (23.5%) reviews are negative. One main reason we did the transform is because we wanted to expand the class size of the non-positive class. However, after the transformation, we have obtained a larger negative class size but, unfortunately, a larger positive class size as well. We will see how this form of labeling works in the classification steps and might apply specific techniques to solve the imbalance issue.

Finally, we looked at the top 10 most frequent words in positive and negative reviews. We filtered all the reviews by setting a minimum length of three words per review. Surprisingly, we observed many overlapped common words in both groups. By further looking into the reviews, we realized that many reviewers either used sarcasm or negation words (such as "not good") when they made negative comments but "not" is considered as a stop word and has been removed in the data cleaning step. In the classification step, we will see how the results turn out and might consider not removing negation words for negative comments if it improves the classification results. Table 2 shows the top 10 most frequent words from both positive and negative reviews.

Word	Positive Review	Negative Sentence
phone	32833	99236
this	18079	52620
with	13138	38260
that	11011	32011
great	8579	25267
good	8407	24732
have	8131	23580
very	7455	22422
like	4608	13543
screen	4606	13764

Table 2: Top 10 most frequent words from both positive and negative reviews

5 EXPERIMENT SETTING AND BASELINES

5.1 Experiment setting

We randomly sampled eighty percent of the data set as the training set and the rest twenty percent left as the validation set. For baselines, grid-search cross validation method was performed with k = 3 (number of folds). For our proposed model, we manually tuned hyper parameters. To evaluate the classification results, we decided to use accuracy score, precision, recall, and F1 score. All experiments were run on the CPU.

5.2 Baseline Description

To obtain a baseline for our experiment, we adopted word2vec to vectorize our text corpus to create the design matrix [7]. Then we selected three machine learning models that are inherently suitable for multi-class classification, which are Random Forest [2], K-Nearest Neighbors (KNN), and Naive Bayes classifier [12] implemented with scikit learn [8]. And for all three models we performed hyperparameter tuning and cross validation. The reason we used word2vec to generate our design matrix over other models such as bag-of-word, or term frequency inverse document frequency models is because it consistently produces the best result in terms of accuracy according to our tests and experiments. And the machine learning models we selected are among the most commonly used models applied to multi-class classification problems, which we believe should be sufficient to serve as our baseline.

The helper code and packages we relied on to produce our results include Python's NLTK, gensim, and scikit-learn libraries.

The hyperparameters we optimized include K, the number of neighbors for KNN, and max_depth, min_samples_split, min_samples _leaf for Random Forest. We were only able to look at a limited number of hyperparameters due to constraints of in time and computing power. However, we will perform a more extensive search for optimum hyperparameters in the next step. We did not try to optimize the parameters for the Naive Bayes model.

5.3 Baseline Result

We applied Grid Search CV method to perform cross validation and hyperparameter tuning, the best parameters returned are:

- K = 5 (number of nearest neighbors)
- min_samples_leaf = 1
- max_depth = 15
- min_samples_split = 2.

The four metrics we used to measure model quality - accuracy, precision, recall, and the F1 score (all weighted) are shown below in Table 3.

Model	Accuracy	Precision	Recall	F1
Random Forest	0.815	0.838	0.815	0.822
KNN	0.862	0.846	0.852	0.847
Naive Bayes	0.674	0.628	0.674	0.610

Table 3: Accuracy, precision, recall, and F1 score (all weighted) of the three baseline models

5.4 Baseline Discussion

According to our result, KNN yielded the best model across all three metrics, and Naive Bayes produced the worst model. We also took a dive into the confusion matrices and discovered that the models were particularly good at identifying positive comments, but did not so well at identifying neutral comments. This finding is consistent across all three models. We suspected that this is very likely due to the imbalance of the dataset, since we have significantly more positive entries than the other two classes. We also found that neutral comments were difficult to classify for all three models. Interestingly, a larger number of neutral comments were misclassified as positive than negative. From this we extrapolated that comments that gave three star ratings tend to either lean toward positive sentiment or negative sentiment, with a larger number leaning toward the positive. In other words, more comments with three star ratings are more similar to positive comments than negative comments. We will dive more deeply into these findings and account for such problems in the following steps of our project.

6 METHODS

In this section, we are going to describe how we built the LSTM model for sentiment classification and how we extracted product features from the reviews and determined the customer's opinion of the feature.

6.1 Long Short Term Memory (LSTM) Model for Sentiment Classification

In this paragraph, we are going to describe how we preprocessed the cleaned text data to make the product reviews valid inputs for our LSTM model. The reason that we decided to try the neural network approach is that deep learning as a powerful machine learning technique has been shown that can produce state-of-the-art prediction results by learning multiple layers of features of the data. Among the various choices of network structures, LSTM has shown robust performance against the notorious exploding/vanishing gradients problem and satisfying predicting power for sentiment analysis problems. Thus, our group decided to build our own LSTM model and see whether the evaluation scores can be increased by this deep learning approach. With the cleaned product reviews, we first created a new data set with only the reviews and the corresponding labels (1 as positive, 0 as neutral, and -1 as negative). After dropping entries with missing values, we split the data into training set (80%) and validation set (20%). The training set was fit into a word tokenizer with the maximum number of words to keep was 50,000. To be more specific, each of the top 50,000 most frequent words occurred in the training set was assigned with an index. After the tokenizer was fit, the review entries in both training set and the validation set were turned into number sequences that each word was replaced by the corresponding index. Although product reviews have been turned into numerical sequences, the length of the input variables must be a fixed number which turns out padding technique is necessary for our problem. The maximum review length was set to a hundred as the summary statistics show that around eighty percent of the reviews are shorter than a hundred. With the input variables ready, we implemented an LSTM model with Keras library in Python [3]. Our LSTM model contains one embedding layer, two LSTM layers, and a final output layer. Here are the hyperparameters that we tuned during our trainings:

Hyperparameter	Tuning Range
Embedding Vector Length	[32,128]
Batch Size	[256,1024]
Units Per Layer	[32,128]
Drop Out	[0.1-0.4]
Recurrent Drop Out	[0.1-0.4]
Number of Epochs	[5-10]

Table 4: Hyperparameters tuned and the tuning range during training

6.2 Feature Extraction on Product Reviews

In this paragraph, we are going to introduce how we performed feature extraction from the product reviews and how we calculated the sentiment score of each extracted feature. As the original data contains more than 400,000 text reviews, our group has decided to group the original data by product brand, and then perform the algorithm on each product brand to reduce the running time. After we have chosen a specific brand, the number of reviews is dramatically decreased (Samsung has the most reviews and the number of reviews is 65,728). Next, for each review, we performed POS tagging technique with the NLTK package [1] to mark up every word in each review to a corresponding part of a speech tag. After every word has been tagged, we further cleaned up the reviews by removing stop words. The reason to remove stop words after POS tagging is because the sentence structure will be modified after stop word removal and wrong tags may be assigned to existing words. With all the words having been tagged, we created a dictionary to store the word-tag pairs and the number of occurrence that this word has occurred in the corpus. Words that with various tags were not considered for future steps. To extract the features, we iterate through the dictionary and the feature will be extracted if the feature is tagged as noun or noun phrases, and the number of occurrence is more than the threshold term we have set (0.01*total number of reviews in the corpus). In order to increase the accuracy of our feature extraction, we manually created two lists of words, one includes features that might be missed by the algorithm, and another list of words to exclude even though the word meets the criteria. To calculate the opinion score, for each feature, we further extract the nearest opinion words (at most 2 words) within a certain distance (at most 5 words away). Then, the polarity of the adjective was obtained from WordNet [4]. To improve accuracy, we negated the polarity if we detected negation, and manually created lists of words to be considered and to be excluded for both the positive and negative sentiments. The formula to calculate the sentiment score is shown in Equation 1

$$\text{Score}_i = \sum_j \frac{p_j}{d(w_{ij}, o_j)}, \tag{1}$$

where o_j and p_j are the j^{th} adjective and its corresponding polarity. function d measures the distance between the adjective and the characteristic. Score_i is then calculated by the weighted sum of the polarity scores as an adjective that is closer to the characteristic weights more. The feature extraction part of our project was inspired and modified based on [5].

7 EXPERIMENTS

7.1 LSTM Sentiment Classification

As mentioned in the previous section, we randomly sampled 80% of the data as the training set and the rest 20% percent as the validation set. Hyperparameters in Table 4 were tuned during trainings for the best result. The evaluation metrics are the same as the ones that we used to evaluate our baseline results which are accuracy, precision, recall, and F1. Table 6 shows the evaluation scores of our LSTM model along with the baseline results, and Table 5 shows the hyperparameter values that yield the best result.

Hyperparameter	Value
Embedding Vector Length	100
Batch Size	512
Units Per Layer	64
Drop Out	0.4
Recurrent Drop Out	0.4
Number of Epochs	5

Table 5: Hyperparameter values that generate best result

7.2 Comparative Analysis

As we can see from Table 6, comparing the evaluation metrics side by side with those of the baseline models, our LSTM model out performed the baseline models with the highest accuracy, precision, and F1 scores, and tied highest recall with KNN. This result proves that our LSTM model is able to produce significantly better predictions than classic machine learning approaches.

Model	Accuracy	Precision	Recall	F1
LSTM	**0.877**	**0.895**	**0.852**	**0.874**
Random Forest	0.815	0.838	0.815	0.822
KNN	0.862	0.846	**0.852**	0.847
Naive Bayes	0.674	0.628	0.674	0.610

Table 6: Accuracy, precision, recall, and F1 score (all weighted) of the baselines and the LSTM model

7.3 Feature Summary by Phone Brand

With the given dataset, we summarized the feature extracted from product reviews for top 7 phone brands with most reviews. For each feature mentioned in the review, if the calculated sentiment score is above zero, we counted as a positive feedback, while a score below zero was counted as a negative feedback. Then, the total number of positive and negative opinions for each feature are plotted together as a side-by-side bar plot to give straightforward comparisons. Due to the page limit, we only display the summaries of Samsung and LG in Figure 4 and 5 for illustration purpose, and the rest of the summaries can be found in our GitHub repository.

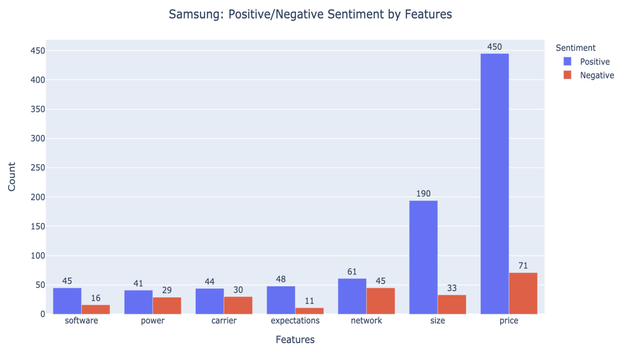

Figure 4: Feature Summary of Samsung

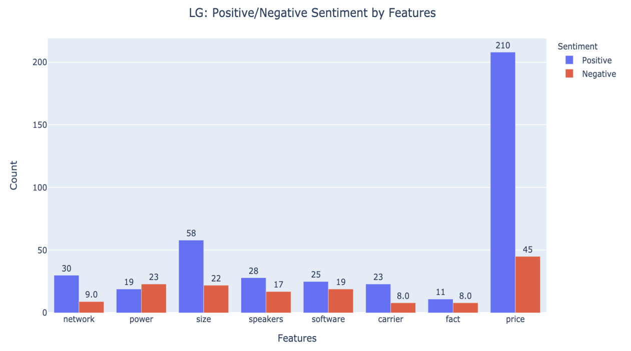

Figure 5: Feature Summary of LG

From the two figures, we can see that the extracted features seem to be reasonable and correct. Looking at Samsung's summary, we could see that all extracted features have more positive feedback than negative ones, while customers are most satisfied with the price and the size. In the LG case, although customers are also pleased with the price, we also observed that more customer have negative opinions on the power. This can be an improvement LG can make to increase the quality of its product and thus gain more customers in the market.

8 CONCLUSION

Our feature extraction algorithm was able to successfully extract features from the comments, such as power, price, size. However, for phone brand with less reviews, there still exists features that are wrongly extracted (e.g. "fact" was extracted as a feature for LG which does not make sense). Furthermore, one limitation of our work is that we did not have an effective way to evaluate our feature extraction results. In order to do so, it requires manually going through each comment and examining whether the algorithm's output matches the ground truth. This requires hours and hours of tedious work, but it is necessary to ensure that our algorithm produces reliable results. We were able to randomly check a few comments, and the results were respectable. For the LSTM model, one limitation of our experiment is that we tuned the hyperparameters manually and ran the experiment on CPU which caused the computational time very long.

The next step for this project would be to systematically assess the quality of our feature extraction model. For the LSTM model, if we are able to run the experiment on GPU and perform large scale tuning or even create deeper model structure, I believe that our model's predicting power can be still increased. In conclusion, we were able to improve model quality with our deep learning LSTM approach. The feature extraction and sentiment analysis we performed could be applied to a wide range of industries, from goods and services to e-commerce. The fact we can extract the subjects from a comment and analyze the sentiment toward each of these what we call features will allow companies to get more detailed information on how consumers react to a product's each feature. Our findings are able to indicates the areas that could be improved upon by phone manufacturers.

9 CONTRIBUTION

All the team members contributed equally to this project.

REFERENCES

[1] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".
[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[3] François Chollet et al. 2015. Keras. https://keras.io.
[4] Christiane Fellbaum. 2012. WordNet. *The encyclopedia of applied linguistics* (2012).
[5] Irecasens. 2017. irecasens/nlp_amazon_reviews. https://github.com/irecasens/nlp_amazon_reviews
[6] Yue Lu and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *Proceedings of the 17th international conference on World Wide Web*. 121–130.
[7] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 142–150.
[8] 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. *the Journal of machine Learning research* 12 (2011), 2825–2830.
[9] PromptCloud. 2017. Amazon Reviews: Unlocked Mobile Phones. https://www.kaggle.com/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones#Amazon_Unlocked_Mobile.csv
[10] Wei Wei and Jon Atle Gulla. 2010. Sentiment learning on product reviews via sentiment ontology tree. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 404–413.
[11] Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011. Clustering product features for opinion mining. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 347–354.
[12] Lin Zhang, Kun Hua, Honggang Wang, Guanqun Qian, and Li Zhang. 2014. Sentiment analysis on reviews of mobile users. *Procedia Computer Science* 34 (2014), 458–465.
[13] Wenhao Zhang, Hua Xu, and Wei Wan. 2012. Weakness Finder: Find product weakness from Chinese reviews by using aspects based sentiment analysis. *Expert Systems with Applications* 39, 11 (2012), 10283–10291.