Project Report

Feature Based Opinion Extraction from Customer Reviews

Shreya Laddha, Siddhesh Pawar
Supervised by: Prof. Gourab Nath

September 9, 2020

Abstract

In this project, we implemented and extended some of the existing works on feature extraction and sentiment analysis for better and more informative summarizing. Our task will help the manufacturer or the seller to gather total sentiment towards a popular feature of the product based on intensity of the opinions expressed by the customers in their reviews. This will not only help the potential customers to get a realistic analysis of the product they are about to buy but also the sellers to get a detailed analysis of their product performance.

1 Introduction

It is a common practice that merchants and manufacturers selling products on the online services such as Amazon ask their customers to review the products and associated services. As e-commerce and these online services are becoming more and more popular, the number of customer reviews that a product receives grows rapidly which gives rise to the need to automating the process of reading the reviews and drawing meaningful summaries from the reviews which would help not only the customers to make decisions on whether to buy the product but also to the manufacturer to know exactly what all things need to be improved in the existing product and which ones to prioritize. The customers in these platforms are highly encouraged to share their feedback about their experiences and the products they have purchased in the form of ratings and reviews. A product may be associated with several features and the customers may have different opinions on each of these features.

In this project, we firstly extract the one word features from a set of reviews based on their frequency of occurrence followed by association rules mining to get a list of two of word features by examining the words that occur frequently. This is followed by sentiment analysis of each and every feature to get an overall sentiment of a particular feature. This summarization task is different from traditional text summarization because we are only interested in the specific features of the product that customers have opinions on and also whether the opinions are positive or

negative. We do not summarize the reviews by selecting or rewriting a subset of the original sentences from the reviews to capture their main points as in the classic text summarization. In this work, we only focus on mining opinion/product features that the reviewers have commented on.

A question that one may ask is "why not ask the merchant or the manufacturer of the product to provide a list of features?" This is a possible approach. However, it has a number of problems: (1) It is hard for a merchant to provide the features because he/she may sell a large number of products. (2) The words used by merchants or the manufacturer may not be the same as those used by common users of the product although they may refer to the same features. This causes problem in identifying what the customers are interested in. Furthermore, customers may comment on the lack of certain features of the product. (3) Customers may comment on some features that the manufacturer has never thought about, i.e., unexpected features. (4) The manufacturer may not want users of its product to know certain weak features. Thus, our first task is to extract the features that the reviewers have commented upon and then mine the opinion related to that feature.

1.1 Literature Review

Our work finds its motivation from [5]. The first task of the pipeline is to extract the product features from the dataset. We draw our motivation of using association rules mining to extract the product features from [8] where the authors have proposed a number of techniques such as compactness pruning, frequent feature extraction, redundancy pruning for mining opinion features from product reviews based on data mining and natural language processing methods such as POS tagging with an objective to produce a feature-based summary of a large number of customer reviews of a product sold online. Similarly, [7] provide a novel algorithm for extracting the opinionated reviews from the data set and summarizing them. The term extraction techniques in [10] use deep-learning methods such as Bi-LSTMs and CNNs to extract subword features and thereby recognize named entities and relation. However, the product features are not named entities and therefore the named entity recognition techniques will not effectively serve the purpose of extracting the product features from the reviews.

The second task is the extraction of sentiment from an opinionated review followed by the classification of opinionated text, e.g. a customer review, as expressing a positive or negative opinion. It is a widely explored topic in academia and industry and has a huge amount of literature related to it. We have used VADER (Valence Aware Dictionary and sEntiment Reasoner) which is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, for classification of sentiment form an opinionated review [9]. We refer to [2] where the authors survey four different approaches: 1)training on a mixture of labeled data from other domains where such data are available 2) training a classifier as above, but limiting the set of features to those observed in the target domain 3) using ensembles of classifiers from domains with available labeled data 4) combining small amounts of labeled data with large amounts of unlabeled data in the target domain to customizing a sentiment classification system to a new target domain in

the absence of large amounts of labeled data and thereby compare results obtained by using each of the four approaches and discuss their advantages, disadvantages and performance. Also, [3] examine factors that make the sentiment classification problem more challenging which we have tried to address using VADER.

2 Approach

Figure 1 gives the architectural overview of our opinion extraction system. The inputs to the system are the product reviews of all the customers. The output is the summary of the reviews. The system involves three main steps: (1) Review Cleaning; (2) Mining the frequent product features that have been commented on by the customers; (3) Identify the customer opinions and the opinion intensities for each product feature

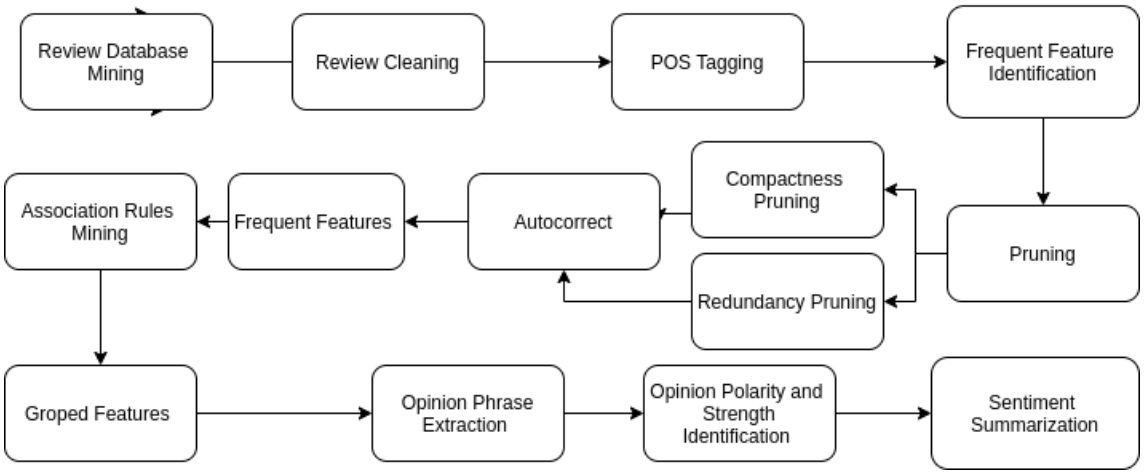

Figure 1: Pipeline For Sentiment extraction and classification

2.1 Review Cleaning

In today's world, emojis have become an important tool for expressing opinions. However, in this work, for the feature extraction part, we neglect any emojis and demojify our review dataset before proceeding further. The dataset also contains a lot of reviews which contain no specific feature and are just simple opinions on the whole product itself. For a review to express any opinion on a feature, it must contain atleast a feature, i.e a noun, and an adjective describing the feature. The minimum word length required is two for a review to be meaningful for our analysis. We remove all the reviews of one word length, as they are mainly an adjective for the whole product. We also remove any two word length review which does not contain (noun+adjective) pair, hence shrinking our review corpus for analysis. All of the review corpus is also turned to lowercase to avoid distinction between words like 'Camera' and 'camera' during POS tagging as described in Section 2.2.1. However, for sentiment analysis we use our original review dataset (with uppercase letters and emojis preserved).

2.2 Frequent Feature Identification

Customers generally write a review about the popular and important features of the product. We denote these features as frequent features. While the other features of the product are either infrequently mentioned or not mentioned in reviews. In our work, we will only mine for the frequent features. Infrequent features are less in numbers and comprise $15 - 20\%$ of the overall features as discussed in [8]. Features can be explicitly or implicitly mentioned in a review. For example, "The camera is amazing but overall, it is not worth spending on the phone." Here, 'camera' is an explicit feature whereas 'value of money' is an implicit feature. We only extract the explicit features as the implicit ones are difficult to mine. Human errors are unavoidable. There is a high possibility that while writing a review, customer might have made a spelling mistake while mentioning the feature. For example, a customer might have written 'camara' and 'batteri' instead of 'camera' and 'battery'. This may lead to a huge loss of information if such incorrectly spelled features are ignored. We have incorporated the spelling correction in our work which takes care of the cases which have not more than one error in the spelling. So, words like 'camara' are treated as 'camera' but the words like 'cemara' are not corrected. The following NLP methods have been used for frequent features identification:

2.2.1 Parts-of-Speech (POS) Tagging:

POS tagging is a task of labelling each word in a sentence with its appropriate part of speech. It is a common NLP technique. POS tagging is used to extract all nouns present in each data point (review text). The product features are mostly nouns or noun phrases. It is tempting to consider all nouns as features which however should not be done as not all nouns are features. We have used spaCy which is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython [6]. By tokenizing each and every word in review text and then thereby using POS tagger of spaCy, we filter out a list of nouns (both common and proper nouns) to be processed further.

2.2.2 Association Rules Mining

After extracting a list of nouns for every review, we need to find the nouns which can possibly be features. The frequently mentioned nouns and their combinations have a higher probability of being a meaningful feature and we can extract them using frequent item-set mining. We apply Apriori Algorithm as mentioned in [1] for discovering all significant association rules between items in a large database of transactions. We extract the nouns and noun combinations of length 2, having the minimum support value (frequency of occurrence) as specified by the user. By default, the threshold support value is taken to be 0.01. These generated frequent itemsets are called candidate features since not all of them are meaningful. For example - 'awesome' and 'battery camera', two frequent itemsets obtained after Apriori algorithm, are not meaningful features and techniques need to be applied to remove such unwanted features from the list.

2.2.3 Feature Pruning

All the features are tagged as nouns, but not all nouns are features. After obtaining a list of frequent nouns, we need to remove all those nouns or noun phrases which are meaningless as features. For this task, we use two types of pruning methods suggested in [1] as follows:

Compactness Pruning
This pruning technique is used to extract meaningful features from the list of candidate features of two words length. A feature containing two words in a sentence is called compact if the word distance between them is less than 3 and the words are not separated by any conjunction and punctuation. Here we have modified the the definition of compactness and the pruning procedure given in [1]. We may have many sentences which contain more than one features. For every sentence, we iterate through the list of candidate features. The total occurrences of the two words of the candidate feature in a sentence together and the number of sentences where these are found to be compact is then tabulated. We define the m_support as follows:

$$m\_support = \frac{\text{Total Sentences where compact}}{\text{Total occurrences together}} \tag{1}$$

m_support lies between 0 and 1. There will be some cases, where the denominator will be zero, in these cases we do not define m_support, and the candidate features are discarded. The candidate features having m_support greater than the user defined threshold value of this m_support will be considered as meaningful features. Therefore, proper tuning of the threshold value is required for good results.

Redundancy Pruning
This pruning technique is used to extract meaningful features from the list of candidate features of one word length. A one word feature is said to be meaningful if it occurs in quite a sufficient amount of sentences alone, without any other superset of that feature in it. A superset of feature is defined as the set of two word meaningful features (after compactness pruning) containing this one word feature. For example, let 'camera' be the feature under analysis. Some other features of the phone contain the word camera as well, e.g. front camera, back camera, rear camera, camera quality. All these two word features comprise the superset of the feature 'camera'. We define p_support as follows:

$$p\_support = \text{Number of sentences in which feature occurs alone} \tag{2}$$

Or in other words, p_support = Count of sentences in which no two word feature from the super set occurs together with the one word feature. For every sentence, we iterate over the list of one word candidate features. The total occurrences of the feature and the p_support are then tabulated for every candidate feature. We observe that a lot of meaningless features like 'amazon' and 'app' have very high p_support values and the total occurrences match their p_support since no superset of these features exists. The candidate features having p_support greater than the user defined threshold value of this p_support and whose total occurrences does not equal its p_support will be considered as meaningful one word features. The second condition helps us to get rid of many meaningless features although there is

a tradeoff of losing a few features. Proper tuning of the threshold value is required for good results.

2.2.4 Spelling correction:

Once we have got a list of meaningful features from pruning, we apply spelling correction to the corpus to replace misspelled words with correct words. For 'camera' may be misspelled incorrectly as 'cemera' in any review (and may lead to the algorithm ignoring the review), so we form a dictionary and keep a track of such words and replace those words with the correct words during opinion extraction.

Levenshtein distance is calculated for every noun extracted from POS tagging with every feature in the list obtained after pruning. Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. Max distance taken is varied from 1 to 3, and it is observed that the best results are obtained for max distance as 1, otherwise a lot of wrong cases come up in our list. If the first letter of both the noun taken and feature matches, then only the distance is calculated. This reduces many wrong cases to come up like brick and price. However, some wrong cases still exist. For example - 'qualify' is treated as 'quality' and 'like' is treated as 'life'. There is a tradeoff between some wrong information and loss of lot of information. In this work, we ignore any wrong information considering very low probability of having much influence on our opinion scores.

For every word a dictionary is made whose key are the nouns and the values are from the feature list, which are grammatically correct. This dictionary lets us keep a track of all the words and their correct forms without changing our actual review corpus. So any opinion associated with the misspelled word will be treated as opinion associated with the correct meaningful feature.

2.3 Opinion Mining

2.3.1 Opinion extraction

Our first task is to mine the opinion words corresponding to the feature in the sentence. Opinions can be explicitly and implicitly mentioned in a sentence. For example - " The phone is amazing but it is too big in size", this statement contains an explicit opinion about the battery but an implicit opinion about 'size of the phone'. We will focus on explicit opinions in our work as implicit opinions are difficult to mine. Opinionated phrases are of following types - Adj+Adj pair, Adv/Neg+Adj pair, Adv/Neg+verb pair and verb+Adj pair. An opinion phrase can either occur before the feature or after the feature. For example - "the phone has good battery." and "The camera is amazing." contains the opinion before and after the feature respectively. In this work, we assume that a customer expresses a single opinion about a particular feature in a sentence in his/her review, which is the usual case. In [7] nearest adjective to a product feature is considered as the opinion word for that particular feature. This is supported by the previous work on subjectivity [4]. For one word feature, we extract the nearest opinionated phrase to it from any side

in the sentence. For two word feature, the nearest opinionated phrase from any side of any of the two words in the sentence is extracted by the help of POS tagger of spaCy. Next, sentiment scores are obtained based on the intensity of emotions in opinions.

2.3.2 Scoring the Sentiments

We use VADER (Valence Aware Dictionary and sEntiment Reasoning) [9] for polarity extraction and scoring the sentiments. VADER is applied to the phrases obtained from the reviews before the cleaning (lowercasing and demojification) so as to retain the emojis and the capitalized words which are treated differently by VADER. It outputs 4 polarity scores namely positive, negative, neutral and compound. A summation of the all the positive, negative and compound polarity scores of all opinion phrases corresponding to a specific feature is done to finally get total positive, negative and compound polarity scores associated with that feature. The total positive and total negative polarity scores were taken to get the final sentiment score of a particular feature as follows:

$$\text{Sentiment Score}(\%) = \frac{\text{Total Positive Polarity}}{\text{Total Positive Polarity + Total Negative Polarity}} * 100 \quad (3)$$

This final score tells us the percentage positive opinion towards the specific feature.

VADER works well with opinionated phrases having adjective in them. However, for some cases where the opinion is conveyed through verb+adverb pair or a negation+verb pair, VADER fails. For example - VADER assigns a zero negative and a zero positive score for "The camera is not working properly." But this statement clearly depicts a negative opinion about 'camera'. So for the cases where we do not have any adjective in the opinionated phrase and VADER assigns it a 0 negative and 0 positive polarity score inspite of the phrase containing a negation word, we assign the phrase a 0.25 negative score, 0 positive score and a -0.25 compound score. There exists a dictionary of all negation words in VADER, which is used here. The value of 0.25 is taken arbitrarily as it does signify that the opinion is in negative but also does not make a very large impact on the overall score of the feature. This improvement to VADER technique is suggested by us in our work.

3 Experiments

A brief overview of experiments done after applying our proposed method is as follows:

1. Fetching and cleaning the dataset

   Experiments: Downloaded review corpus from Amazon Shark. The dataset of all stars are combined to get a bigger dataset of around 5000 entries. Another dataset of 1427 data points also obtained. Both contain customer reviews of a mobile of a popular brand. Both datasets contained several columns of which only the review text column was considered for the analysis. Preprocessing pipeline is applied to the datasets.

Observation: The data can be sorted using a number of stars and the title of the review can be used as one of the features for sentiment evaluation.

Challenges: Converting everything to lowercase may affect the sentiment analysis so the actual text is kept as well and is to be used during sentiment scoring.

2. Processing the nouns extracted

Experiments: During POS tagging, lemmatization is done on the list for converting plural nouns to singular nouns. Proper nouns, common nouns, noun phrases and compound nouns are extracted as well.

Observation: Noun phrases and compound nouns give us features like 'touch response' but these will be automatically extracted on applying association rules mining, so these are not explored further. There may be some loss of features if we were to consider only one of common nouns and proper nouns. Thus to avoid this loss, both common nouns and proper nouns are extracted collectively as nouns.

Challenges: There exist grammatical errors in the list of nouns extracted, so a method needs to be devised to take care of them.

3. Spelling correction of the nouns

Experiment: Using nltk.corpus.words, only those nouns which are grammatically correct are considered to be taken forward. On comparing the use of Levenshtein distance and cosine similarity, Levenshtein distance is a better option for comparing words. After getting a grammatically correct list, we convert the incorrect words to the correct word if levenshtein distance is less than 3.

Observations: Words like helio, con, pic, pro made it to the correct nouns list. Words like smartphone, redmi, realme, hotspot, touchscreen didn't make it to the list.

Challenges: A lot of wrong observations lead to discarding this method of correcting the spelling of misspelled words. This problem is solved by association rules mining, as it is likely to extract only the correct nouns. The spelling correction needs to be done before sentiment analysis so that we don't lose a lot of information.

4. Association rules mining and frequent feature mining

Experiment: The apriori algorithm is used to get frequent itemsets from the review corpus with $min_{lift} = 1$ and $min_{support} = 0.01$. The results are analysed at different support values.

Observations: All extracted features are grammatically correct. Lot of unwanted features like amazon, app show up when support is 0.01. Some important features like battery backup is lost when support = 0.02.

Challenges: Features like 'amazon', 'app', 'battery camera' need to be removed using some techniques. Features like 'fingerprint', 'finger print' are treated differently. The value of support is dependent on the dataset and hence will be a hyperparameter to be tuned by the user.

5. Feature Pruning

   Experiment: Compactness pruning is applied to remove the non-compact two word features like 'battery camera'. Redundancy pruning is done to remove meaningless one word feature.

   Observations: List obtained after pruning contains only grammatically correct nouns. The values of m_support and p_support are dependent on dataset. The final feature list contain all grammatically correct meaningful features.

   Challenges: The values of the thresholds for both pruning methods is highly dependent on the review corpus, hence these will be hyperparameters for the user to tune. Not all features can be extracted fully by doing pruning. Some important features are lost as a tradeoff to better results.

6. Correcting the nouns

   Experiment: Spelling correction is applied to every noun extracted from POS tagging using the list obtained after pruning.

   Observations: For maximum distance 1, most of the matches come out to be correct. For maximum distance 2 and 3, and also the case where the first letters are not matched, the wrong information is too much and cases like 'buyer' and 'budget' start to match, hence these cases are not considered further. Matching the first words reduces a lot of incorrect cases.

   Challenges: For maximum distance 1, some cases are lost like 'bettry' and 'battery' and some wrong cases still occur like 'like' and 'life'.

7. Sentiment Analysis

   Experiment: Passing all the sentences through opinion mining pipeline (here the sentences are taken from the review before preprocessing techniques were applied as the cleaning may lead to reduction of intensity of words that is sometimes is expressed by virtue of big bold fonts,emojis, etc. However, the dictionary generated after spelling correction is used to relate the opinions associated to the incorrect word to the correct feature.)

   Results: VADER mostly takes into account the effects of capitalization and emoji along with the adjectives that contributes towards the sentiments. Sentiment scores depict the amount of positive polarity towards a particular feature.

   Challenges: The nearest phrase assumption may get violated at places. VADER is ineffective at times and has many shortcomings not all of which have been taken care of.

4 Conclusions

In this project, we propose a pipeline to extract meaningful features from the customer reviews and get a sentiment score for each feature which is a measure of positive polarity towards that feature. To achieve this objective, we divide our pipeline into three major sections - cleaning, feature extraction and opinion

Feature	Positive Polarity	Negative Polarity	Sentiment Score (%)
Camera quality	43.59	8.6	83.52
Battery backup	20.62	1.48	93.29
Price	43.57	3	93.54
Camera	111.5	22.322	83.32

Table 1: A glimpse of final result of the project

mining. Cleaning the dataset involves cleaning the reviews and removing all useless data. Feature extraction pipeline generates a list of meaningful features. Finally, opinions towards features are extracted and sentiment scores are assigned to each feature in the opinion mining pipiline. We have evaluated this method on a review dataset of smartphone of a popular brand and the results were meaningful.

5 Future Work

Future works includes some modifications in the pipeline such as incorporating the stars by the customers as a measure of the sentiment of the features. After doing a survey of a good amount of products, a threshold score can be decided as a good score, below which the feature is considered to be under-performing, and such cases can be highlighted to the customers for much better insights of a product. Similarly, a threshold value can also be set above which the features are said to be performing excellently and par expectations. Two or more similar product features can also be compared on the basis of the sentiment scores obtained using our method.

Our approach uses VADER for scoring the sentiments. Although, VADER works with emojis and slangs effectively, there are many shortcomings of VADER. For example, *"This film should be brilliant. It sounds like a great plot. However, it can't hold up" or "I hate the Spice Girls. ... Why I saw this movie is a long story, but I did, and one would think I'd despise every minute of it. But... Okay, I'm really ashamed of it, but I enjoyed it. I mean, I admit it's a really awful movie. The plot is such a mess that it's terrible but I loved it."*. In these examples, a human would easily detect the true sentiment of the review. VADER classifier would presumably find these instances difficult, since there are many words indicative of the opposite sentiment to that of the entire review. It seems that some form of discourse analysis is necessary or at least some way of determining the focus of each sentence, so that one can decide when the author is talking about the film itself. Furthermore, it seems likely that this thwarted-expectations rhetorical device will appear in many types of reviews devoted to expressing an overall opinion about some feature. Hence, we believe that an important next step is the identification of features indicating whether sentences are on-topic (which is a kind of co-reference problem).

References

[1] R. Agrawal, Rakesh Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large DataBases*, 2000.

[2] A. Aue and M. Gamon, "Customizing sentiment classifiers to new domains: a case study," in *Proceedings of RANLP, 2005.*

[3] L. L. B. Pang and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques,," in *Proceedings of the EMNLP.*, 2002.

[4] R. Bruce and Wiebe, "Recognizing subjectivity: A case study of manual tagging," in *Natural Language Engineering*, 2000.

[5] R. N. Gourab Nath, Randeep Ghosh, "Cluster analysis of customer reviews: Summarizing customer reviews to help manufacturers identify customer satisfaction level," 2017.

[6] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017.

[7] B. Hu, Minqing Liu, "Mining and summerizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining*, KDD -2004.

[8] M. Hu and B. Liu, "Mining opinion features in customer reviews," in *Proceedings of AAAI, 2004.*, ser. GW'09, 2004.

[9] E. Hutto, C.J. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text." in *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, 2014.

[10] P. Murthy, V. Bhattacharyya, "A deep learning solution to named entity recognition,"," 2018.