IJCTA, 9(10), 2016, pp. 4553-4562
© International Science Press

Optimization Strategies for Product Rating Using User Review Analysis

***Prasanth R Sharma **Yathu E M ***Jayakumar P**

Abstract : *e*-commerce web has evolved as a key source of unstructured data rich in customer opinion. Understanding user reviews and mining opinion is an important research area in computer science which mostly relies on statistical inference methods. A deterministic and efficient algorithm to mine opinion remains far in the future. In this paper, domain specific strategies for optimizing existing product rating methods are discussed. The proposed strategies include a simple method to filter out irrelevant words from extracted nouns, an application of improved apriori algorithm and a new mathematical estimation of the overall rating. These methods can improve the efficiency in terms of running time of algorithms and accuracy of results. An empirical analysis of accuracy and efficiency of our methods is also presented.

Keywords : Product rating; Review analysis; Apriori algorithm; Optimization

1. INTRODUCTION AND BACKGROUND

The popularity of e-commerce sites and online shopping is increasing steadily. An innovative feature of e-commerce is the feedback collection mechanisms in it. Customers can mark their feedback in the form of textual reviews, star ratings, numerical ranking etc. Such 'direct' feedback is considered to be more reliable and genuine than traditional '*word over mouth*' feedback. Reviews help prospective customers to decide whether a product is suitable for them. Thus they serve as an indirect product recommender. They help manufactures know the defects of their products and know the market trends. Using this, manufactures can improve the quality of the product by rectifying defects and can plan marketing strategies. Thus these reviews are important for both customers and manufactures.

As of now, it has been estimated that on an average every product receives 50-100 reviews and if a product is popular then it receives more than 1000 reviews. These reviews are given by several classes of users ranging from lay to learn. An interesting trait observed is that vendors provide detail about a limited number of features which usually receives less trust and experts provide details on numerous features and are highly trusted. Common users give their experience with the product, in the form of reviews. These classes of reviews gives user opinion on product features from various perspectives and can be used to rate the product scientifically.

Product rating engine operates on review dataset by identifying product features and opinion words. Product features and opinion words are likely to be nouns and adjectives respectively. Their extraction is done using Parts of Speech (POS) tagging function of Natural Language Tool Kit (NLTK). Identification of frequent feature set

* Department of Computer Science and Applications, Amrita School of Engineering Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India prasanth.r.sharma@gmail.com

** Department of Computer Science and Applications, Amrita School of Engineering Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India yathu.elamkulam@gmail.com

*** Department of Computer Science and Applications, Amrita School of Engineering Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India jayakumarp@am.amrita.edu

from the extracted nouns is done using an improved apriori algorithm and the orientation of opinion words are identified using a seed list. The presence of a combination of frequent features and opinion words in a sentence makes it a candidate for orientation check. Such an association contributes to the count of positive or negative opinions of the associated feature. The rating of features is done using these values and overall rating is calculated using the rating of individual features.

The paper is organized as follows. In section II, an overview of the associated work is presented. Section III explains methodologies. An implementation of our approach is given in section IV. A discussion on the observations and results are given in section V. Conclusion and future works are given in section VI.

2. RELATED WORKS

V.Y Karkare et.al [1] categorizes the extracted opinion words (adjectives) into five categories, based on their orientation. Category-1 indicates a most negative opinion and category-5 indicate a most positive opinion. They have rated each feature of the product utilizing a weight function and gave a comparison of the features of the products from one brand to another.

Lei Z et.al concentrates on the difficulty of double propagation [2]. In double propagation features are assumed as nouns or noun phrases and opinion words are assumed as adjectives. The advantage is that only a single initial seed opinion lexicon is enough instead of supplementary resources. It runs smoothly on medium size dataset but for large dataset it shows certain limitations. The precision of the method drops with the size of data set because it extracts many nouns/noun phrases which are not features and adjectives which are not opinions.

In [3], Balakrishnan et.al discusses an approach where the orientations of a stream of tweets from the Twitter micro blogging site are identified. They detail preprocessing steps to structure the tweets since they are particularly disordered.

In [4], G Carenini et.al determines the polarity of reviews by Naïve Bayes Classifier. Using Mallet package for Natural Language Processing (NLP) they have identified the candidate sentences and their orientation. Then, a graph was generated by taking the sum of positive and negative reviews. In their approach, a product with greater number of reviews is rated high and they defined a new parameter - *reviews per month* (RPM). For more accurate score they counted the number of "*yes*" for the question "*was this review helpful*" on *flipkart* and the average is calculated.

In [5], K. Dave et.al using association mining extracted features from user reviews. They applied apriori algorithm on the noun/noun phrases sets to produce frequent item sets. They removed non-genuine features by applying redundancy pruning and compactness pruning. In spite of the fact that their methodology was successful in finding frequent features, the drawback of using apriori was the inflation in execution time while managing sizeable databases.

In [6], R. Hemalatha et.al included a new pruning procedure for both non-product and opinion-irrelevant product features. It was an extension of [5]. They determined the subjectivity of a review sentence by gathering a list of positive and negative words from the specialists. At the point opinion-irrelevant features are removed when the frequent features which never or once in a while co-happen with any positive or negative adjectives in the review sentence.

In [7], Mohammed-Al-Maolegi et.al discuses about an improved apriori algorithm for overcoming the problem of large execution time while managing large databases. They improved the apriori algorithm to decrease the time consumed for candidate itemset generation.

3. METHODOLOGY

The major phases in our approach are as follows :

1. Crawling of online shopping site.
2. Extraction of feature words (nouns) and opinion words (adjective) from reviews.
3. Removal of non-feature words.

4. Association mining using Improved Apriori Algorithm.
5. Identifying opinion orientation of review.
6. Product rating calculation.

A. Crawling of online shopping site

Crawling is a way of getting data from the web programmatically and is a method of finding URLs. The web crawler systematically browses the World Wide Web for the required information. The crawler scans the web-pages and identifies the review text. URL patterns are used for removing unwanted web crawl having no information. Retrieving effective content from the web page is a crucial task for the crawler. A web crawler has a seed which is a list of URLs to visit. The crawler visits these ULRs and it recognizes every one of the hyperlinks in the page and adds them to the URL list to visit, called crawler frontier. Those URLs are recursively visited and information is saved. Python library Beautifulsoup4 for scrapping data out of HTML and XML file is used for implementation of the project.

B. Extraction of feature words (nouns) and opinion words (adjecctives) from reviews.

A feature is most likely a noun [2] that describes or gives any knowledge about product based on its specifications, actions, attributes or usage. Features are of two types' relevant features and irrelevant features. Relevant feature is a feature which satisfies the following conditions :

- It should describe some attribute of the product.
- A minimum percentage of reviewers speak about the feature.
- It is not subjective (*e.g. - The phone looks good)*.
- It is relevant to the domain of the product.
- It is quantifiable by some measure (*e.g. - Battery life by standby time*).
- It shouldn't be about any accessories of the product *(e.g.-This phone doesn't come with a screen guard).*

Identification of Nouns and Adjectives

The first step in the process is to obtain the nouns and adjectives from the reviews using NLTK from NLP. The process of tagging each word in the crawled reviews as one of the parts of speech is called as Parts Of Speech Tagging. The POS tagger is used as a preprocessor. The tagger first tokenizes the review sentence. Then it uses several kinds of information like dictionary, lexicon, rules, and so on to tag the words. Dictionaries have categories of a particular word, i.e. a word belongs to more than one category. For example, the word *run* is both noun and verb. Taggers solve such ambiguity using probabilistic information.

C. Removal of non-feature words

The nouns and adjectives tagged by POS tagging may contain irrelevant words that are less likely to be a feature word. Various methods to filter out irrelevant words exist in the literature. An important observation is that the three letter words present in review dataset have no relevance in this domain. The POS tagger tags such words as nouns *(Table I)*. This means there is a high probability that a noun which is less likely to be a feature word is included in the feature set. As a rule, a data set that contains *n* items can possibly generate up to 2^n "1 frequent itemsets, ignoring the invalid set (*i.e.* null set). Because *n* can be very large in numerous practical applications, the search space of itemsets that need to be investigated is exponentially huge. A simple solution is to remove all three letter words from the noun set. This can be done in linear time and can reduce the overall processing time significantly. Considering the exponential complexity even a minor reduction in the size can significantly reduce the practical time requirement. Note that such an optimization is highly domain depended.

Table 1. Some reviews containing three letter words which are wrongly identified as product feature.

Review Sentence	*POS Tag*
I used samsung galaxy phone. I technically examined the operating system and not found any bug in the sys.	[(‘I’, ‘PRP’), (‘used’, ‘VBD’), (‘samsung’, ‘NN’), (‘galaxy’, ‘NN’), (‘phone’, ‘NN’), (‘.’, ‘.’), (‘I’,‘PRP’), (‘technically’, ‘RB’), (‘examined’, ‘VBD’), (‘the’, ‘DT’), (‘operating’, ‘NN’), (‘system’, ‘NN’), (‘and’, ‘CC’), (‘not’, ‘RB’), (‘found’, ‘VBN’), (‘any’, ‘DT’), (‘bug’, ‘NN’), (‘in’, ‘IN’), (‘the’, [(‘I’, ‘PRP’), (‘used’, ‘VBD’), (‘samsung’, ‘NN’), (‘galaxy’, ‘NN’), (‘phone’, ‘NN’), (‘.’, ‘.’), (‘I’, ‘PRP’), (‘technically’, ‘RB’), (‘examined’, ‘VBD’), (‘the’, ‘DT’), (‘operating’, ‘NN’), (‘system’, ‘NN’), (‘and’, ‘CC’), (‘not’, ‘RB’), (‘found’, ‘VBN’), (‘any’, ‘DT’), (‘bug’, ‘NN’), (‘in’, ‘IN’), (‘the’, ‘DT’), (‘sys’, ‘NN’), (‘.’, ‘.’)]
phone look best in GOLD color, not too many pre-installed apps, HD video clarity is very clear	[(‘phone’, ‘NN’), (‘look’, ‘NN’), (‘best’, ‘JJS’), (‘in’, ‘IN’), (‘GOLD’, ‘NNP’), (‘color’, ‘NN’), (‘,’, ‘,’), (‘not’, ‘RB’), (‘too’, ‘RB’), (‘many’, ‘JJ’), (‘pre-installed’, ‘JJ’), (‘app’, ‘NN’), (‘,’, ‘,’), (‘HD’, ‘NNP’), (‘video’, ‘NN’), (‘clarity’, ‘NN’), (‘is’, ‘VBZ’), (‘very’, ‘RB’), (‘clear’, ‘JJ’)]

D. Association Mining using Improved Apriori Algorithm

Apriori algorithm is an established algorithm for mining association rules. It follows candidate generation approach rather than pattern growth approach followed in FP-growth. Initially the algorithm obtains all frequent itemsets. At that point it delivers all association rules from frequent itemsets. The frequency of itemsets is characterized by counting their occurrence in transactions.

Identifying Frequent Itemset

Apriori generates candidate itemsets of length $(i + 1)$ based on frequent itemsets of length i. In i^{th} iteration, the algorithm scan database to obtain frequency of i-itemsets that contains only one item by counting each item in database. This is repeated until there is no more i-itemsets.

Association Rule Mining

An association rule is an implication of the form $X \rightarrow Y$, where X and Y are disjoint itemsets. Support and confidence are two parameters used to characterize the quality of an association rule. Support decides how frequently a rule is relevant to a given data set, while confidence decides the conditional probability of Y given X.

$$\text{Support, } s(X \rightarrow Y) = n(XUY)/N$$

$$\text{Confidence, } s(X \rightarrow Y) = n(XUY)/n(X)$$

Where $n(A)$ defines the frequency of A and N is the number of transactions.

Despite being simple, apriori algorithm has certain limitations. In large database the execution time increases [7] because of hold or generating the candidate set. So, an improved version of apriori algorithm by Mohammed Al-et al [7] is used.

Algorithm 1 Improved Apriori

```
1: $L_1 = FindFrequent - 1Items(T)$
2: for $k = 2$ to $L_{k-1} \neq \emptyset$ \$ do
3:     //Generate the $C_k$ from $L_{k-1}$
4:     $C_k$ = candidates generated from $L_{k-1}$
5:     //get the item $I_w$ with minimum support in Ck using $L_{k-1}$
6:     x = GetItemMinSup($C_k, L_1$)
7:     // get the target transaction IDs that contain item x
8:     Tgt = GetTransactionID(x)
9:     for each transaction t in Tgt do
10:        Increment the count of all items in $C_k$ that are found in Tgt
11:        $L_k$= items in $C_k \geq$ MinSupport
12:    end for
13: end for
```

Like normal apriori, improved apriori also scan transactions to generate 1-itemset (L_1). Then removes the noun words that do not satisfy the minimum support. For each item in L_1, the algorithm generates an additional id called Transaction ID which gives the transactions (nouns set inreviews in this case) where the item is found. Candidate item set of length 2, C_2 is generated by joining $L_1 * L_1$. In order to find the support count of an association (x, y) where $(x, y \in C_2)$, transaction IDs of $L_1$is used. This eliminates the need of scanning entire transaction database for new support count and thus can save a lot of time. The same is repeated for C_{i+1} until no new frequent itemsets are identified. Improved apriori approach decreases the time consumed by 67.38% [7]. The entire process is depicted in Figure 1:

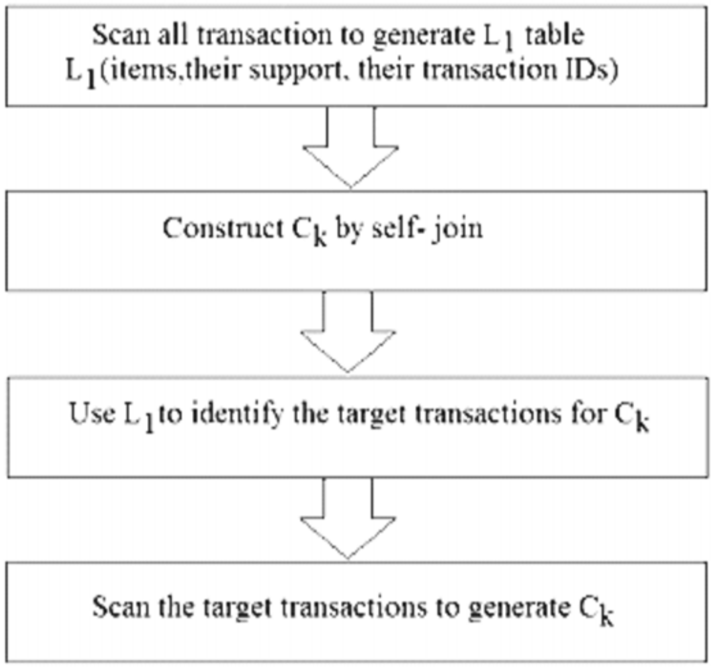

Scan all transaction to generate L1 table
L1(items,their support, their transaction IDs)
Construct Ck by self- join
Use L1 to identify the target transactions for Ck
Scan the target transactions to generate Ck

Fig. 1. C_k generation in improved apriori algorithm.

E.Identifying Opinion Orientation of Review

Once the frequent features are extracted, the review sentence containing them is classiûed as either positive or negative. A seed list of positive and negative words is used to identify the orientation of adjectives. If a word is not present in the seed list, its synonyms are obtained from the *synset* in *wordNet* and a secondary search with the obtained synonyms is performed. Upon a hit the orientation of the original word is defined as that of the synonym and is added to the seed list. If no synonym gives a hit then the word is ignored.

In order to find the orientation of the sentences containing frequent features, first tokenize review sentences using sentence tokenizer in NLTK and thus for each review a list of tokenized sentences is obtained. From this list, only those sentences which contain the frequent features are selected. Then a check for a positive or negative word in these sentences from the seed list is done. The orientation of a sentence is defined by the orientation of these words, unless it contains negations. If negations are present then orientation is reversed.

F. Product Rating Calculation

The *positivity quotient* (P_i) and *negativity quotient* (Q_i) of i^{th} feature f_i is defined respectively as the ratio of positive comments and negative comments obtained to the total number of classified reviews of f_i.

$$P_i = \frac{\text{No: of positive reviews of feature } f_i}{\text{Total no of classified reviews}}$$

$$Q_i = \frac{\text{No: of negative reviews of feature } f_i}{\text{Total no of classified reviews}}$$

The difference between P_i and Q_i gives the individual rating of the feature f_i. A base value of 5 is used for rating every feature and the individual rating is measured on a scale of 0 to 10.

$$\text{Individual Feature Rating} = 5 * (1 + (P_i + Q_i))$$

Finally the overall product rating is defined as the average of the individual feature rating.

$$\text{Overall Product Rating} = \frac{\sum \text{Individual Feature Rating}}{\text{Total no of Frequent Features}}$$

4. IMPLEMENTATION

crawler.py- This python class contains methods used to crawl the *flipkart* website. Beautifulsoup4 package is used to read the contents in HTML and XML file. It automatically search for the hyperlinks for each product id in the content and adds it to the database.

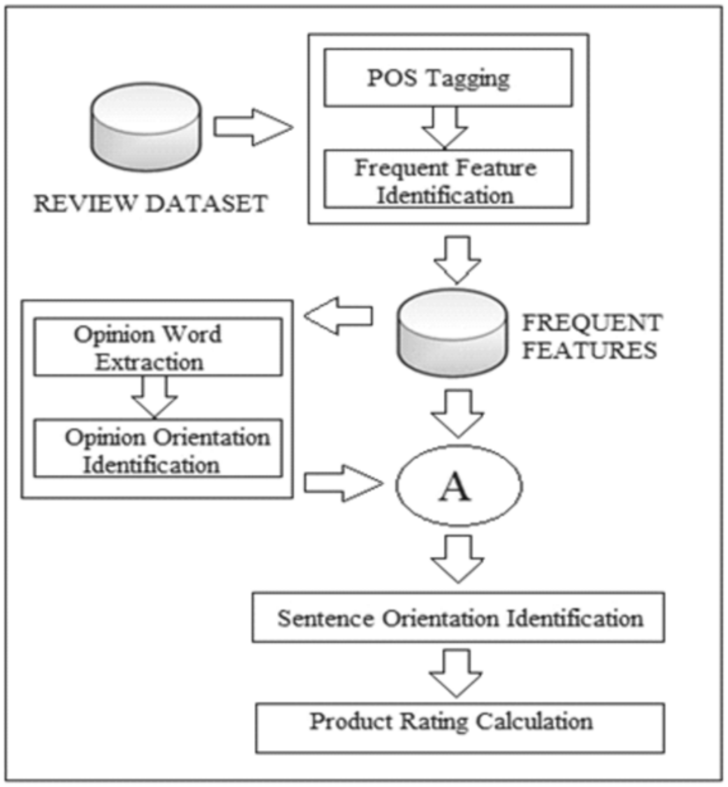

REVIEW DATASET
POS Tagging
Frequent Feature Identification
FREQUENT FEATURES
Opinion Word Extraction
Opinion Orientation Identification
A
Sentence Orientation Identification
Product Rating Calculation

Fig. 2. A general architecture of review engine.

productInfoCrawler.py- This class is used to visit the hyperlinks and the contents are extracted. Then, details regarding each product like product name, review id, review content, price, star rating, and etc. are identified.

extractor.py- In this python class, extraction of relevant feature and opinion words, and reduction of feature set is done.

aprioriComparison.py- In this class the feature set is processed using improved apriori algorithm. It contains an implementation of normal apriori algorithm also. A comparative study between normal apriori and improved apriori is also done using this.

adjectiveOrientationCheck.py- This class checks the orientation of adjectives and its synonyms (if required) using seed lists and *synset*.

sentenceCheck.py- The reviews are tokenized into sentence and find the sentences having a frequent feature and an adjective. The orientation of each sentence is also identified and classified it for each feature.

scoreCalculation.py- This class contains methods for calculating total positive and negative sentence count for each feature and overall product rating.

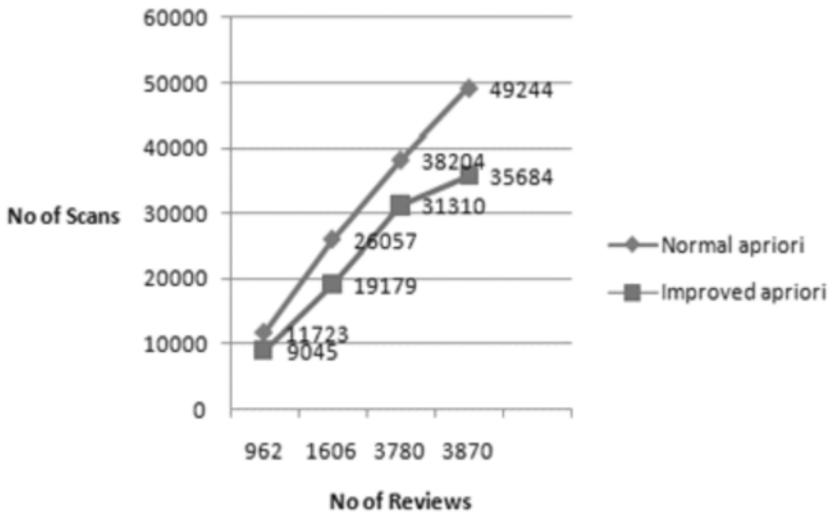
5. RESULT AND ANALYSIS

The following section discusses about observations and results.

For the product Apple iPhone5 we crawled 962 reviews from the *flipkart* website. After POS Tagging 162 three letter words is tagged as a noun and are removed. The feature set is given as the input to normal apriori and to the improved apriori. The result shows that total number of scans and the time consuming for frequent feature set generation got reduced (Table II, Figure III). For normal apriori number of scans is 11723 and for improved apriori it decreased to 9045.

Table 3. Comparison of the performance of improved apriori and normal apriori methods after removing three letter words.

Product Name	*Total no. of reviews*	*Normal Apriori*		*Improved Apriori*	
		No of scans	*Time Taken (s)*	*No of scans*	*Time taken (s)*
Apple iPhone5	962	11723	1.681	9045	1.069
Lenovo A6000 Plus	3780	38204	5.874	31310	4.731
Asus Zenfone2	1606	26057	3.004	19179	2.208
Lenovo K3 Note	3870	49244	9.018	35684	6.178



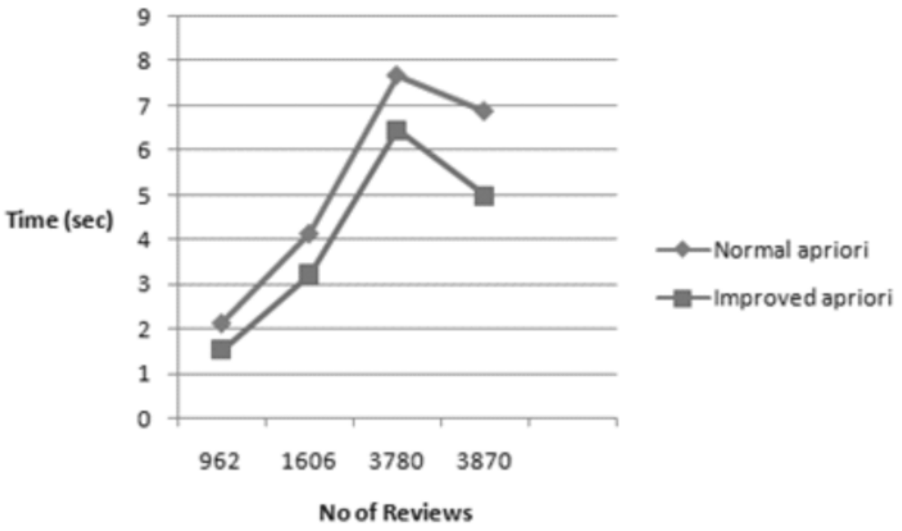
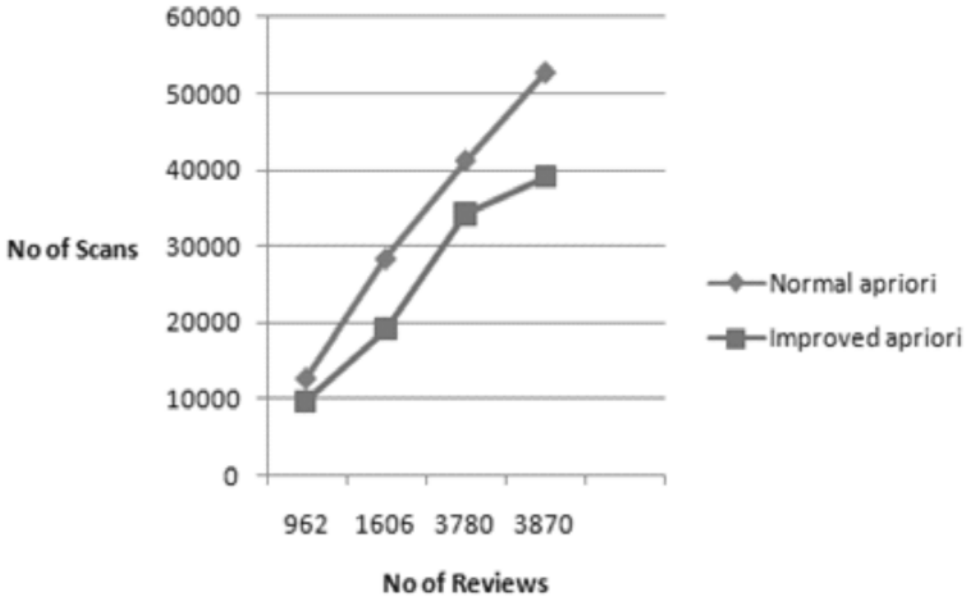


Fig. 3. Graphical representation of comparison of the performance of improved apriori and normal apriori methods after removing three letter words.

Table 3. Comparison of the performance of improved apriori and normal apriori methods without removing three letter words.

ProductName	*Total no. of reviews*	*Normal Apriori*		*Improved Apriori*	
		No of scans	*Time Taken(s)*	*No of scans*	*Time Taken(s)*
Apple iPhone5	962	12808	2.118	9804	1.531
Lenovo A6000 Plus	3780	41195	7.688	34301	6.453
Asus Zenfone2	1606	28488	4.131	19217	3.203
Lenovo K3 Note	3870	52658	6.884	39098	4.981



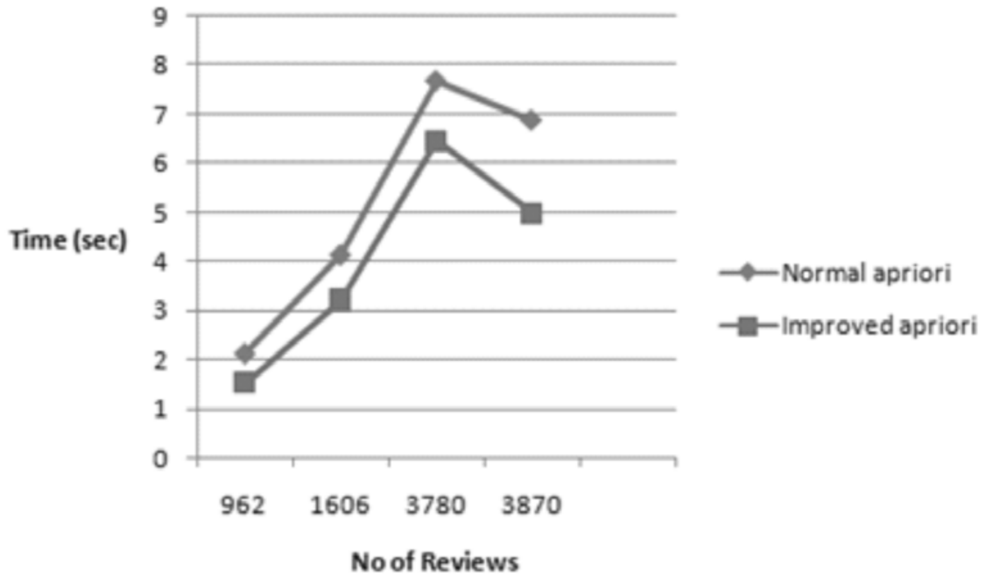


Fig. 4. Graphical representation of comparison of the performance of improved apriori and normal apriori methods without removing three letter words.

Another important analysis performed is about the impact of three letter word in rating calculation. Presence of three letter words affects the overall product score. A positive opinion or negative opinion for such incorrect key word can affect the positivity or negativity of the overall opinion. Table IV shows the feature rating of certain mobile phones with and without three letter words in the feature set. Similarly, Table V shows the overall rating calculated using the individual rating in both cases. The variance in the values(Figure III and IV) shows the impact of three letter words. The rating obtained without three letter words tend to be more realistic.

Table 4. Feature score of certain mobile phones with and without three letter word in the feature set.

Product Name	*Noun Feature Set*	*Three Letter Nouns*	*Frequent Features with 3 letter words and rating*	*Frequent features without 3 lette rwords and rating*
Apple iPhone5	2301	162	service– 6.72 iphone- 4.97 price- 5.10 camera- 5.79 apple- 5.22 day - 5.17 quality- 5.61 battery- 3.67	service– 6.72 iphone- 4.97 price- 5.10 camera- 5.79 apple- 5.22 quality- 5.61 battery- 3.67
Asus Zenfone2	3762	253	performance- 5.58 screen- 4.32 backup- 6.06 issue- 1.94 battery- 5.41 range- 6.85 camera- 4.49 day- 4.43 quality- 5.06 display- 5.13 price- 6.17	performance- 5.58 screen- 4.32 backup- 6.06 issue- 1.94 battery- 5.41 range- 6.85 camera- 4.49 quality- 5.06 display- 5.13 price- 6.17

Table 5. Product rating with and without three letter nouns

Product Name	*Noun Feature Set*	*Three Letter Nouns*	*Product Rating with 3 letter words*	*Product Rating without 3 letter words*
Apple iPhone5	2301	162	5.27	5.31
Asus Zenfone2	3762	253	5.04	5.11

6. CONCLUSION AND FUTURE WORK

In this paper a discussion on optimization methods for rating products through review analysis is presented. A method for removing irrelevant nouns and an application of improved apriori algorithm are the main optimizations strategies experimented. A mathematical estimation of rating is also implemented and verified. We implemented the methods using python framework and the results from empirical analysis shows an improved performance of the overall process.

In future we like to develop methods for calculating overall rating of a product in total market by analyzing reviews of the product from multiple websites. Other problems need to be addressed are tagging words which are not present in the dictionary and tagging of words from different language. Clustering the related object features can provide more concise review summary.

7. ACKNOWLEGMENT

We would like to express our sincere gratitude to the faculty members of Department of Computer Science and Applications of Amrita Vishwa Vidyapeetham, Amritapuri campus, for providing help and guidance.

Our sincere thanks to Dr. M R Kaimal, Chairman, Department of Computer Science,Amrita Vishwa Vidyapeetham, Amritapuri campus, for his prompt support for the success for completion of this work. We thank our anonymous reviewers for their valuable suggestions.

8. REFERENCES

1. Mrs. Vrushali Yogesh Karkare, Dr. Sunil R Gupta, *"Product Evaluation using Mining and Rating Opinions of Product Features"*, International Conference on Electronic Systems, Signal Processing and Computing Technologies, 2014.
2. Lei Z, Bing L, Suk H L, Eamonn O'Brien-Strain. *"Extracting and Ranking Product Feature in Opinion Documents"*, In the Proceedings of COLING '10, the 23rd International Conference on Computational Linguistics, Pages 1462-1470.
3. Balakrishnan Gokulakrishnan, Pavalanathan Priyanthan, Thiruchittampalam Ragavan, Nadarajah Prasath, A Shehan Perera, *"Opinion Mining and Sentiment Analysis on a Twitter Data Stream"*, The International Conference on Advances in ICT for Emerging Regions – ICTer.
.4. G Carenini, R T Ng and E Zwart, *"Extracting Knowledge from Evaluative Text"*, K-CAP '05 Proceedings of the 3rd international conference on Knowledge capture Pages 11-18.
5. K Dave, S Lawrence and D M Pennock, *"Mining the peanut gallery: opinion extraction and semantic classification of product reviews"*, Proceedings of the 12th international conference on World Wide, New York, 2003.
6. R Hemalatha, A Krishnan and R Hemamathi, *"Mining Frequent Item Sets More Efficiently Using ITL Mining"*, 3rd International CALIBER, Ahmedabad, 2005.
7. Mohammed-Al-Maolegi , Bassam Arkok,*"An Improved Apriori Algorithm for Association Rules"*, International Journal on Natural Language Computing (IJNLC) Vol. 3, No.1, February 2014.
8. Praveen Kumar et al.*"A Comparative Study on Sentiment Analysis and Opinion Mining"*, International Journal of Engineering and Technology (IJET), Vol. 8, Issue 2, 2016.