Association data mining in Sentiment Analysis

Mr. Rajendra Gawali¹, Ms. Prajacta Lobo²

¹Professor, Computer Department, Lokmanya Tilak College of Engineering, Mumbai University, India
²Lokmanya Tilak College of Engineering, Mumbai University, India

Abstract - Prior to making a purchase, an online shopper typically browses through several similar products of different brands before reaching a final decision. This will help buyer to make better decisions based on reviews provided by previous customer. Reviews can be positive, negative or neutral. These review data available online in multiple formats with huge volume, thus identifying this sentiment is very important. Analysing this data manually is very time consuming as well as erroneous. Our focus here is extracting efficient or relevant words from review data, sentiment polarity classification, learning and comparing algorithm in finding frequent itemset.

Keywords - Sentiment analysis, association data mining, frequent itemset, FP-Tree

I. INTRODUCTION

The general problem of feature identification of reviews and then highlights the specific instance of the problem that we aim to solve. Let us first give the definition of some primary concepts.

Definition 1: product feature
Product features refer to all the components, qualities or physical characteristics of a product such as size, color, weight, speed, etc.

Definition 2: opinion sentence
An opinion sentence is a sentence that consists of at least one product feature and its corresponding opinion word.

Definition 3: explicit and implicit feature
An explicit feature is a feature of a product which is directly talked about in review sentence. An implicit feature is a feature that is not explicitly mentioned in the sentence and it can be implied. The following sentence shows a negative opinion on a cellphone:

"It is not easy to carry."
“Weight” is an implicit feature of the cellphone which is implied from the sentence.

Definition 4: frequent and infrequent feature
A feature f is frequent if it appears in majority of the review sentences. f is called infrequent if it is only appeared in a few number of reviews.

After putting all these definitions together we go through with general problem of identifying features in the reviews. Most current researches focus on discovering explicit product features. Generally, the current approaches are either supervised or unsupervised.

Although, supervised approaches sound to be more accurate [11], but they need training set that is generated by the human. This approach is effective when the documents are not too away in terms of the subjectivity. This means that if we have two datasets, each of which focuses on a particular topic, the training set for them should be different as well. Let us consider the case we are dealing with opinion orientation of the sentences in a movie and a product review dataset. Normally, opinion words used to express one’s feeling about a movie is different from the situation they are talking about the quality of a product. In a movie dataset some words may carry a negative orientation while the same word in a product review dataset can deliver positive orientation. The same problem may be occurred while dealing with a dataset consisting of reviews on a number of products. Usually, feature words used by the reviewers are varied across different types of product as the components of each product may be unique. So accumulating a set of terms as the training data may bring about running into trouble.

A common unsupervised approach that has proposed by many researchers is based on association mining technique. Focusing on the nouns or noun phrases it is supposed that those nouns that are frequently occurred in the review dataset are most likely to be considered as product features. After identifying nouns they ran an association miner which is based on Apriori algorithm to find frequent itemsets that are likely to be frequent features. This method is simple and efficient and gives reasonable results. However, this technique has some major shortcomings. Apriori algorithm tests combination of the items without considering of the items ordering. For instance, the words “dvd” and “player” may be occurred in 14 transactions (sentences) as “player dvd” while 87 transactions contain “dvd player”. The algorithm cannot recognize the difference between the two situations and it returns only one possible combination such as “palyer dvd” with totally 101 occurrences. However, depending on the chosen threshold, the item “player dvd” may be considered as an infrequent item and it is not expected to be listed here. Moreover, in case that there exist a large number of frequent patterns, Apriori have to take many scans of large databases and generate huge number of candidates which reduces the performance of the system. Our work focuses on handling the above problems with the previous work by applying a more efficient frequent pattern mining algorithm.

ISSN: 2231-2803 http://www.ijcttjournal.org Page 83
E-commerce, communication, and opinion sharing. It has several blogs devoted to diverse topics like finance, politics, travel, education, sports, entertainment, news, history, environment, and so forth, on which people frequently express their opinions in natural language. Mining through these terabytes of user review data is a challenging knowledge engineering task. Recent years researchers have proposed approaches for mining user expressed opinions from several domains such as movie reviews, political debates, restaurant food reviews, and product reviews and so forth. Our focus in this paper is efficient feature extraction, sentiment polarity classification, learning and comparing algorithm in finding frequent itemset from online product reviews dataset.

The main difficulty in analyzing online users’ reviews is that they are in the form of natural language. While natural language processing is inherently difficult; analyzing online unstructured textual reviews is even more difficult. Some of the major problems with processing unstructured text are dealing with spelling mistakes, in correct punctuation, use of non-dictionary words or slang terms, and undefined abbreviations. Often opinion is expressed in terms of partial phrases rather than complete grammatically correct sentences. So, the task of summarizing noisy, unstructured online reviews demands extensive Pre-processing [1].

The objective of this paper is to analyze customer reviews submitted on different product. Now a day’s huge amount of data and information are available for everyone on the internet or in printed form. This data can be stored in many different kinds of databases and information repositories. We have conducted an experiment study on this data to find frequent itemset. For this we have used FP-Growth and FIN algorithm, by making variation in Apriori algorithm [2-6]. It improves performance over Apriori for lower cardinality and it does not follow generation of candidate-and-test method. It also reduces the scanning of database and needs only two scanning of database. Also we have conducted a comparative study between these two algorithms for finding product sentiment using frequent itemset.

II. PROPOSED SYSTEM

The proposed system has been implemented in Java. The architectural overview of this system is given in fig.1 and each component is detailed subsequently. Here we implement Nodeset, a more efficient data structure, for mining frequent itemset. Nodeset requires only pre-order (or post-order code) of each node which makes it save half of the memory compared with N-list and Node-lists. Based on nodeset we present efficient algorithm called FIN to mining frequent itemsets. For evaluating performance of FIN we have conduct experiments to compare it with FP-Growth algorithm.

The major parts of this implementation are:
1. Analysis of review text with more accurate recommendations for products
2. Text to transaction processing
3. Finding frequent itemset from this transaction using FP-Growth algorithm
4. Finding frequent itemset from this transaction using FIN algorithm

Comparing result of both these algorithms in terms of memory usage and execution time taken.

![Fig. 1 System Architecture](image)

1.1.

1.2. Select Algorithm

1.2.1. FP-Growth Algorithm

FP-Growth works in a divide and conquers way. This is efficient and scalable method to complete set of frequent patterns. It allows frequent itemset discovery without candidate itemset generation. It requires two scans on the database. FP-Growth computes a list of frequent items sorted by frequency in descending order (F-List) during its database scan. In its second scan, the database is compressed into a FP-tree. Then FP-Growth starts to mine the FP-tree for each item whose support is larger than $\xi$ by recursively building its conditional FP-tree. The algorithm performs mining recursively on FP-tree. The problem of finding frequent itemsets is
converted to searching and constructing trees recursively [7-9, 12].

Algorithm 1: FP-tree construction [10]

The algorithmic program works as follows:

1. Scan the transaction database once, as among the Apriori algorithmic program, to seek out all the frequent items and their Support.
2. Sort the frequent items in descending order of their Support.
3. Initially, begin making the FP-tree with a root “null”.
4. Get the primary transaction from the transaction database. Takeaway all non-frequent items and list the remaining items in line with the order among the sorted frequent items.
5. Use the transaction to construct the primary branch of the tree with each node corresponding to a frequent item and showing that item’s frequency that’s one for the primary transaction.
6. Get the next transaction from the transaction database. Takeaway all non-frequent items and list the remaining items in line with the order among the sorted frequent items.
7. Insert the transaction within the tree using any common prefix that may appear. Increase the item counts.
8. Continue with Step 6 until all transactions among the database are processed.

Algorithm 2: FP-Growth [12]

The implementation of FP-Growth is divided into three steps.

1. Database is read and the count of items is found. According to the minimum support threshold, frequent items are selected and sorted.
2. Initialization of the FP-tree is done. From the frequent items a node list is created which will be connected to nodes of the tree. After initialization the database is read again. This time, if an item in a transaction is selected as frequent then it is added to the tree structure.
3. Beginning from the least frequent item, a frequent pattern finder procedure is called recursively. The support count of the patterns are found and displayed if they are frequent.

1.2.2. FIN Algorithm

FIN uses novel data structure called Nodeset, for mining frequent itemsets. Different from recently used data structures called Node-list and N-list, Nodesets require only pre-order (or post-order code) of each node without the requirement of both pre-order and post-order. This causes that Nodesets consume less memory and are easy to be constructed.

FIN [11-12] directly discovers frequent itemsets in a search tree called set-enumeration tree. For avoiding repetitive search it also adopts a pruning strategy names promotion, which is similar to Children-Parent Equivalence pruning to greatly reduce the search space.


Input: A transaction database DB and a minimum support n.
Output: A POC-tree and F1 (the set of frequent 1-itemsets).

1. [Frequent 1-itemsets Generation] According to n, scan DB once to find F1, the set of frequent 1-itemsets (frequent items), and their supports. Sort F1 in support descending order as L1, which is the list of ordered frequent items. Note that, if the supports of some frequent items are equal, the orders can be assigned arbitrarily.

2. [POC-tree Construction] The following procedure of construction POC-tree is the same as that of constructing a FP-tree (Han, Pei, & Yin,2000).
   - Create the root of a POC-tree, Tr, and label it as “null***
   - For each transaction Trans in DB do the following.
     - Select the frequent items in Trans and sort out them according to the order of F1.
     - Let the sorted frequent-item list in Trans be [p | P], where p is the first element and P is the remaining list.
     - Call insert tree ([p | P], Tr).
     - The function insert tree([p | P], Tr) is performed as follows.
       - If Tr has a child N such that N.item-name = p.item-name, then increase N’s count by 1;
       - else create a new node N, with its count initialized to 1, and add it to Tr’s children-list.
     - If P is nonempty, call insert tree(P, N) recursively.

3. [Pre-code Generation] Scan the POC-tree to generate the pre-order of each node by the pre-order traversal.


Output: F, the set of all frequent itemsets.
1. F £;
2. Call Algorithm 3 to construct the POC-tree and find F1, the set of all frequent 1-itemset;
3. F2 £;
Input: A transaction database DB and a minimum support n.
4. Scan the POC-tree by the pre-order traversal do
5. N currently visiting Node;
6. i y the item registered in N;
7. For each ancestor of N, Na, do
8. ix the item registered in Na;
9. If ix|y F2, then
10. ix|y.supportix|y.support + N.account;
11. Else
12. ix|y.supportN.account;
13. F2 F2[ |ix|y];
14. Endif
15. Endfor
16. For each itemset, P, in F2 do
17. If P.support n [DB], then
18. F2 F2{P};
19. Else
20. P. Nodeset £;
21. Endif
22. Endfor
23. Scan the POC-tree by the pre-order traversal do
24. Nd currently visiting Node;
25. iy the item registered in Nd;
26. For each ancestor of Nd, Nda, do
27. ix the item registered in Nda;
28. If ix|y F2, then
30. Endif
31. Endfor
32. F F1;
33. For each frequent itemset, isit, in F2 do
34. Create the root of a tree, Rst, and label it by isit;
35. Constructing_Pattern_Tree(Rst, {1 | 2 F1, i
36. Endfor
37. Return F;

1.3. Select input data

Primary source of data is Amazon [13], this dataset contains product reviews and metadata, including 143.7 million reviews spanning May 1996 - July 2014. Out of these huge data we obtain cell phone and its Accessories review data, from which we obtain approximately 1000 reviews. Product in this site has large number of reviews. To obtain this data, we started with a list of asin like strings (Amazon product identifiers) obtained from the Internet Archive. Sample review is as shown below. This large data file can be open using Log Expert tool. This tool downloaded from website [14]. This dataset is a superset of existing publicly-available Amazon datasets. Out of above fields we used reviewText (text of the review) as input field in our analysis.

```
{ "reviewerID": "A2SUAM1J3GNN3B", "asin": "0000013714", "reviewerName": "J. McDonald", "helpful": [2, 3], "reviewText": "I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. The is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!", "overall": 5.0,
```

1.4. Text to Transaction Processing with Data cleansing

This involves cleaning the extracted data before the analysis is performed. Here we are using custom logic to keep only relevant words in review before converting into transactions. Usually this involves identifying and eliminating non textual content from the textual dataset, and any information that can reveal the identities of reviewers including: reviewer name, reviewer location, review date etc.

We have prepared relevant word dictionary to compare this input text data. This word dictionary dataset has been downloaded from website [15-16]. This data is converted into input transaction file format as a text file. An item is represented by a positive integer. A transaction is a line in the text file. In each line (transaction), items are separated by a single space. It is assumed that all items within a same transaction (line) are sorted according to a total order (e.g. ascending order) and that no item can appear twice within the same line.

1.5. Transaction to word mapping

After data cleansing is done, all the words are assigned a transaction id and number of times such word occurs in given review. Number of occurrences defines the frequency of such word.

Let’s say "Good" word is found in a review, so during text to transaction processing, it is given a number ‘4’, now an entry will be made into a dictionary as <4,"GOOD">.

When Good word is found multiple times, e.g. 8 times we can say transaction 4 occurs 8 times in given reviews. Hence, in a different collection, this transaction will be represented as <4, 8> = >{transactionId, Frequency}. Hence while
mapping Frequency back to word we can say word "GOOD" with transactionId 4 occurs 8 times.

1.6. Run selected algorithm

The output file format is also defined as a text file, where each line represents a frequent itemset. On each line, the items of the itemset are first listed. Each item is represented by an integer and it is followed by a single space. After, all the items, the keyword "SUPP:" appears, which is followed by an integer indicating the support of the itemset, expressed as a number of transactions. Example 5 supp: 2971, it indicates the frequent itemset consisting of the item 5 and it indicates that this itemset has a support of 2971 transactions.

1.7. Algorithm run statistics

Both the program output statistics as displayed below fig. 2 and fig. 3 were found with minimum support taken as 0.5% and with the help of transaction to word mapping dataset. This algorithm output statistics data file is stored in dataset.

1.8. Comparisons of algorithms

The results of both these algorithms are stored in database. Comparison is based on memory usage V/S time taken by both algorithms to execute. This has been displayed below fig. 4, and graphical representation shown in fig. 5.
We have dealt with a challenging association rule mining problem of finding frequent itemsets using proposed FP-growth and FIN method by making variation in Apriori. The objective is to provide a sentiment of a large number of customer reviews of a product sold online. We believe that this problem will become increasingly important as more people are buying and expressing their opinions on the Web. Analyzing these reviews is not only useful to common shoppers, but also crucial to product manufacturers. The method, described here is very simple and efficient one. This is successfully tested for large data, downloaded from Amazon. We have computed performance comparison by comparing both algorithms. The experimental result shows that FIN is more efficient in terms of memory consumption but taking more execution time compared to FP-growth. Whereas both algorithms improves performance over Apriori for lower cardinality and it does not follow generation of candidate-and-test method. It also reduces the scanning of database and needs only two scanning of database.

III. CONCLUSIONS

The REFERENCES


[3] Lili Zhao and Chunping Li, “Ontology Based Opinion Mining for Movie Reviews”, Ministry of Education School of Software, Tsinghua University, Beijing, China, Knowledge Science, Engineering and Management, Volume 5914 of the series Lecture Notes in Computer Science pp 204-214


[7] Haiping Zhang, Zhengang Yu, Ming Xu, Yueling Shi “Feature-level Sentiment Analysis for Chinese Product Reviews” Dept. of Computer Science Hangzhou Dianzi University Hangzhou, China, 310018, ©2011 IEEE

[8] Haiping Zhang, Zhengang Yu, Ming Xu, Yueling Shi “A MEMs-based Labeling Approach to Punctuation Correction in Chinese Opinionated Text” School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China, p2013/ICA2802.


[16] Jiawei Han and MichelineKamber, “Frequent Item set Mining Methods”, Data Mining – Concepts and Techniques. Chapter 5.2.