Instructions for Project Report by the Professor

Final project report (10-15 pages) due on 11/30/17

13 pages

1) Introduction: a summary of the problem, previous work, methods, and results

---2 pages

2) Problem description: a detailed description of the problem you try to address

---1 page

3) Methodology: a detailed description of methods used

--- 5 pages (1 page architecture , 2.5 page apriori, 2.5 page naive bayes)

4) Results: a detailed description of your observations from the experiments

--- 2 pages

5) Conclusions and future work: a brief summary of the main contributions of the project and the lessons you learn from the project, as well as a list of some potential future work.

--2 pages

Class Project Final Report
Posted on: Monday, November 13, 2017 10:32:00 AM MST
(0) It will be due on **November 30th, might-night, AZ time**
(1) use the same format as your project proposal;
(2) the length of your final report should be 10-15 pages;
(3) submit a single pdf file. As mentioned before, if you used the third party code, do NOT submit its source codes. Only submit the source codes that you did during this class project which you want to claim for credit;
(4) one submission for each group;
(5) name your files as 'group_#_project_report.pdf';
(6) list each team member name (exactly the same as it is shown in black board, including middle names), ASU id, and user id to avoid the grade recording error. Also, list for each team member (a) the specific contribution (in terms of what s/he did); and (b) and the contribution (in terms of percentage).
(7) check the class project guidance for additional requirements.

For all the writings, including proposal and the final report

• 1 inch margin on each side, on A4 paper

• 12 pt, times new roman for main body

• Single column

CSE 575 – Statistical Machine Learning

Amalgamation of association rule mining and opinion mining to build a smart suggestion engine

Team Members

Name	ASU ID	Email	Percentage
Durga Prasad Kothinti	1211231616	dkothint@asu.edu	16.66
Venugopal Yalla	1211230745	vyalla@asu.edu	16.66
Mounika Vuppula	1211040698	mvuppala@asu.edu	16.66
Neha Pulipati	1211214339	npulipat@asu.edu	16.66
Harsha Sharma	1211203211	hsharm17@asu.edu	16.66
Srinivas Puranam	1211168267	spuranam@asu.edu	16.66

***Abstract* -** Smart suggestion systems continue to remain as the most popular method of interaction between businesses and customers over the Internet. The aim of the project is to build a system that enhances customer experience by aiding the customer in making informed decisions by providing recommendations of related products for a given product. In the first phase of the project, products are recommended to customers by identifying the frequent itemsets based on the market basket data using the Apriori algorithm. The second phase of the project involves providing the customers with a list of popular pros and cons for the suggested products based on the results obtained by mining the opinions(reviews) of the customers for that product. The Naive Bayes and Maximum Entropy algorithms are used to perform opinion mining and the results obtained from the two are compared and performance is analysed.

I. Introduction

A. Summary of the problem

The emergence of e-commerce has opened up new possibilities for customers to purchase products seamlessly without visiting shops and outlets. Websites such as LinkedIn, Amazon, Hulu, Netflix, Facebook, Twitter, Pandora, Goodreads use recommendation systems. Use of a recommendation engine is becoming a standard element of a modern web presence.

It is an easy task to choose from limited number of available alternatives but when the collection becomes large, it is a tedious job to evaluate the features of similar products in terms of quality, cost and other parameters. A recommendation system aims to make the task of choosing the correct product easier for the customer. The main purpose of a recommender system is to provide tools to leverage the data mining process and interests of other customers. Although many commercial recommendation solutions deployed today do an excellent job, we believe we can add more smartness to them to further enhance the customer experience by aiding him/her to make more informed decisions.

A recommendation system shows a list of products which might interest the customer. We can further improve the customer experience by listing the pros and cons of the recommended products instead of just the displaying the recommended products. It increases the chances of revenue generation because the customer gets the chance to evaluate the suggested products in depth at the first glance.

B. Previous work

Many of the largest e-commerce websites are already using recommendation systems and rely on them to help customers make informed decisions. The popularity and dependency on such systems is growing exponentially for the businesses to perform better. A lot of work and effort has been put into making these systems as efficient and customer friendly as possible by understanding and analyzing the needs and interests of the customers on a daily basis. Some of the previous work done in this area is in use by the most famous e-commerce websites like Amazon where user's website usage pattern is being observed and related products are recommended.

Most systems follow a process flow where the customers' data is gathered in order to find similarities between the users of the website in order make apt suggestions[1]. The customer's demographic information, search pattern, purchase history, click stream data is analyzed in order to enhance the customer experience.

Collaborative filtering assumes that the customer's current buying pattern will remain the same in the future; a small group of customers is found and a list of items recommended items is generated based on his previous behaviour[2]. There are two types of collaborative filtering: item-based and user-based. They are implemented according to what the preference of the application should be. Depending on what kind of pattern is more important for an

efficient recommendation process, the suggestions are generated either using the user related information (similarities between users) or item related information (similarities between products).

Association rule mining is another way of building a product suggestion system. Association rules specify the rules about how one event is related with another. It is also a type of clustering that classifies data by relevance[3]. A set of rules is built based on transaction information that convey that when an event A occurs another event B occurs. The probability of the event B occurring when the event A occurs is called the confidence, based on which decisions are made.

C. Methods

At a high level, our system is composed of two modules:

Module I - In the first module, we have implemented one of the association rule analysis algorithms– Apriori, to identify and extract frequent itemsets from the market basket data based on which the initial list of suggestions is built.

Module II - In the second module, we are performing opinion mining on the products review data. We intend to extract the user's opinion specific to relevant features of the products, identify patterns in the opinions corresponding to different features and then identify the most liked and disliked features of the products.

The smart engine shall present this information to the customer along with the ordered recommended products enabling him to not just go with the top recommended product but to make a more informed decision by considering the positive and negative aspects of all the recommended products matching his personal preferences.

D. Results

A smart suggestion engine uses Apriori algorithm to train the machine learner. It uses two configuration parameters *minsup* and *minconf* to generate rules. We perform opinion mining using Naive Bayes algorithm and calculate maximum entropy on the generated rules to list their pros and cons.

It makes the decision making process easier for the customer. Smart suggestion engine is a reusable component and can be easily integrated with any website to provide suggestions with the advantages/ disadvantages of each suggestion. It enhances revenues because it is an effective means of selling more products.

II. Problem Description

Recommendation systems change the way inanimate websites communicate with their customers. Rather than providing a static experience in which users search for and potentially buy products, recommendation engine increases interaction to provide a richer experience.

The underlying principle of recommendation systems is that past interests and preferences are often good indicators of future choices. A recommendation system utilizes various sources of data such as user ratings, browsing behaviour and purchasing trend to infer customer interests.Significant dependencies exist between user and item-centric activity. For instance, a user who is interested in a laptop is more likely to be interested in another laptop, chromebook or tablet rather than in a table. Various categories of items show significant correlations, which can be leveraged to make more accurate recommendations.

Algorithms for recommendation systems are dependent on human preference and ratings which itself is an unstable parameter. Individual's rating scale tends to fluctuate, this is known as calibration conundrum. For instance, if a user's mood is bad today, he might give four stars, but tomorrow he'd give five stars for the same product. Reviews of a product can also be heavily dependent on other factors irrelevant to the quality of the product such as packaging and delivery speed.

Large-scale recommendation systems are prone to the issue of scalability. Many algorithms work well with small datasets but as the data size increases the algorithm becomes either slow or inaccurate which in turn defeats the whole purpose of a recommendation engine.Scalability is not an issue for offline processing scenarios but with real time data a more adaptive and scalable algorithm needs to be deployed.

Smart suggestion engine aims to address these issues by presenting the pros and cons of a recommended product which provides the user with an understanding of why they might find a particular suggestion interesting. For example, a customer buying a mobile phone is likely to buy a case for its protection. Smart engine displays the list of suggested products which includes a mobile case and also includes its pros such as hard and fit. This approach makes it more likely for the customer to act on the recommendation and also helps improve customer loyalty and retention.

Smart suggestion engine is scalable system as it utilizes a database system for storing transactional data. Database access is much faster than storing the transactional data as a list in the memory. Efficient and optimized queries can result in fast retrieval, search and update of millions of records in the database which is not possible in case of file storage.

III. Methodology

A. Architecture

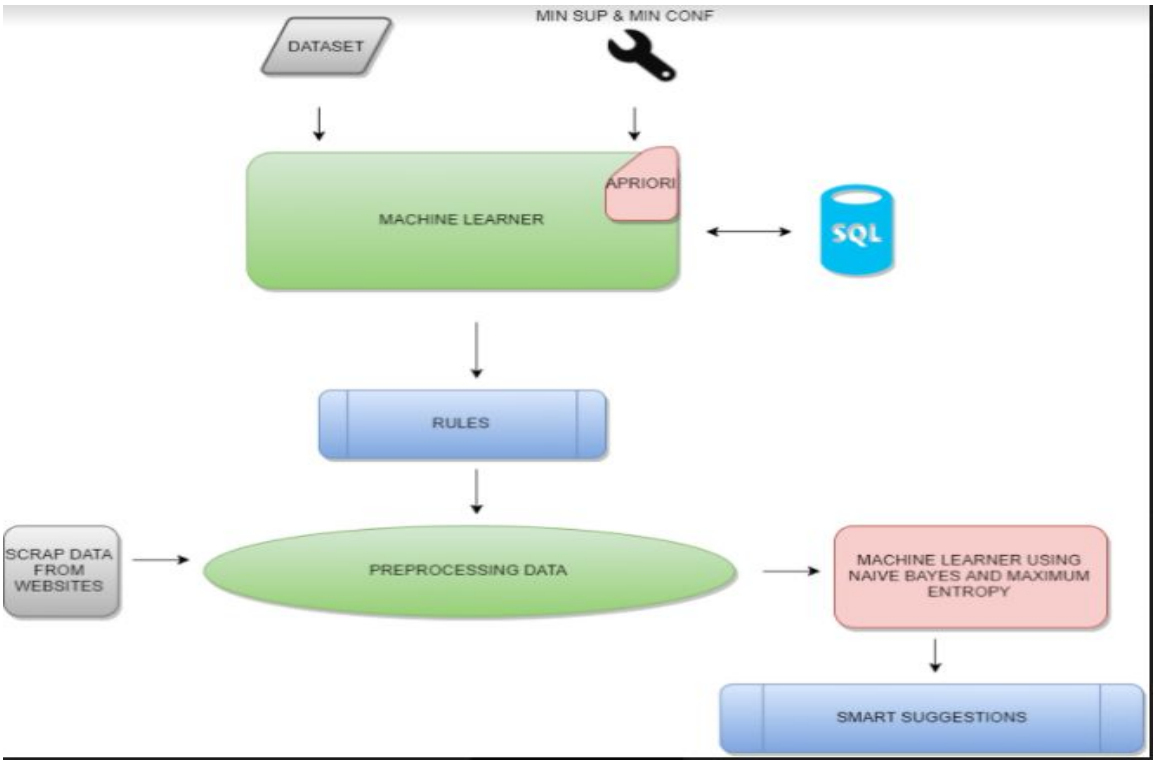


Fig - A detailed architecture diagram for smart suggestion engine

The dataset consists of 75,000 transaction records which contain 49 unique products. The configuration parameters *minsup* and *minconf* are hardcoded as 0.02 and 0.01. The configuration parameters and dataset are input into the machine learner which implements Apriori algorithm.

Dataset dumps all the transactions into a table named *'transactions'* in the MySQL database. The table consists of two columns : id and transaction list. We can load the transactions as a list in memory and then run a double for loop to calculate confidence and support. But, this results in large access time and is not a feasible solution for real time datasets. MySQL database makes the smart suggestion engine scalable for large datasets and improves the performance from 240 ms to 7 ms.

Support is the measure of how frequently a product appears in the dataset. Confidence is the probability of having a product in the transactions which already contain a specified product. After discarding all products whose support is less than *minsup*, we are left with 25 products as 24 products are filtered out. We then filter out the products whose confidence is less than the *minconf.* Then, take the maximum six confidence values for every product and those corresponding products are the recommended products.We dump generated rules into a file which is used by the second half of the algorithm.

Based on the product ids in the rules generated above, the reviews for the suggested products analysed from the product review dataset where all the reviews regarding various features of the product are classified as pros or cons. The model is trained using both Naive Bayes and Maximum Entropy in order to identify a particular feature as a pro or a con. We perform certain text pre-processing tasks on the review data like word extraction and stopword removal in order to make the analysis more efficient. Below is a list of types of negations handled in the implementation.

Table 1. List of Negations

Negation Class	Negations
Syntactic	no, not, rather, couldn't, wasn't, didn't, wouldn't, shouldn't, weren't, don't, doesn't, haven't, hasn't, won't, wont, hadn't, never, none, nobody, nothing, neither, nor, nowhere, isn't, can't, cannot, mustn't, mightn't, shan't, without, needn't,
Diminisher	hardly, less, little, rarely, scarcely, seldom
Morphological	*Prefixes*: de-, dis-, il-, im-, in-, ir-, mis-, non-, un- , *Suffix*: -less

In Naive Bayes, the *prior* and *likelihood* for each word are calculated to predict the mood of the sentence and classify it as a pro or a con. In Maximum Entropy the maximum *entropy* for each word or weights of all the unique words is calculated. The results from both these implementations are later compared and analysed.

B. Module I - Apriori Algorithm

Apriori is an algorithm for discovering frequent itemsets in transaction databases. It was proposed by Agrawal & Srikant in 1994.The frequent itemsets identified by Apriori are used to determine association rules which highlight general trends in the database.

Let $I = \{i_1, i_2, i_3, ..., i_n\}$ be a set of n attributes called items and $D = \{t_1, t_2, ..., t_n\}$ be the set of transactions, known as the database.Every transaction, t_i in D has a unique transaction ID, and it consists of a subset of itemsets in I.

A rule can be defined as an implication, X ⇒ Y where X and Y are subsets of $I(X, Y \subseteq I)$, and they have no element in common, i.e., $X \cap Y$. X and Y are the antecedent and the consequent of the rule, respectively.

Support is the measure of how frequently an itemset appears in the dataset. A low support rule is uninteresting because customers seldom buy those items together.

The rule X ⇒ Y holds with support s if s% of transactions in D contain X ∪ Y. Rules that have a s greater than a user-specified support is said to have minimum support.

Confidence is the measure of how often an association rule is found to be true. The higher the confidence the more likely is it for Y to be present in transactions that contain

X. Highly correlated association rules are interesting from business perspective for making decisions.

The rule X ⇒ Y holds with confidence c if c% of the transactions in D that contain X also contain Y. Rules that have a c greater than a user-specified confidence is said to have minimum confidence.

$$\text{Support, } s(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

$$\text{Confidence, } c(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Given a set of transactions, Apriori algorithm aims to generate all association rules that have support and confidence greater than the user-specified minimum support (*minsup*) and minimum confidence (*minconf*).

The advantage of Apriori algorithm is that it finds all the itemsets that meet the minimum support and confidence criteria. Also, it is this significantly faster than the naive method.

Steps involved in implementing Apriori algorithm

Step 1: Calculate the support for each product in the dataset. We have total of 49 products in the database.

Step 2: Discard all products whose support is less than the minimum support value.

− We have 49 products in the dataset. After Step 2 we are left with 25 products as 24 products are filtered out.

```
# Max product Id is 49 in the sample data set
for each_transaction in range(1, 50):
    args = ("%_" + str(each_transaction) + "_%")
    cursor_instance.execute(
        '''SELECT * FROM TRANSACTIONS_DATABASE.TRANSACTIONS_TABLE WHERE TRANSACTION_LIST LIKE %s''', args)
    each_transaction_count = cursor_instance.fetchall()
    support_each_transaction = len(each_transaction_count) / total_num_transactions
    if support_each_transaction >= support:
        support_dict[each_transaction] = support_each_transaction
```

Fig 2 – Calculating support for each product and filtering out products whose support is less than the minimum support

Step 3: Calculate the confidence for association rule X → Y if both X and Y contain a single product.

− We calculate the confidence of each product with the remaining 24 products.

Step 4: Filter out the products whose confidence is less than the minimum confidence value.

```
support_list = list(support_dict.keys())
for i in range(len(support_list)):
    final_rules = {}
    for j in range(len(support_list)):
        if i == j:
            continue
        else:
            args = ("% " + str(support_list[i]) + " %", "% " + str(support_list[j]) + " %")
            cursor_instance.execute(
                '''SELECT * FROM TRANSACTIONS_DATABASE.TRANSACTIONS_TABLE WHERE TRANSACTION_LIST IN
                (SELECT TRANSACTION_LIST FROM TRANSACTIONS_DATABASE.TRANSACTIONS_TABLE
                 where TRANSACTION_LIST LIKE %s) and (TRANSACTION_LIST LIKE %s)''', args)
            each_transaction_count = cursor_instance.fetchall()
            confidence_each_transaction = len(each_transaction_count) / (
                support_dict[support_list[i]] * total_num_transactions)
            if confidence_each_transaction >= confidence:
                final_rules[support_list[i], support_list[j]] = confidence_each_transaction
    final_confidence_list = sorted(((v, k) for k, v in final_rules.iteritems()), reverse=True)[0:6]
```

Fig 3 – Calculating confidence and filtering out products whose confidence is less than the minimum confidence value

Step 5: Take the maximum six confidence values for every product and those corresponding products are the recommended products.

C. Module II - Naive Bayes and Maximum Entropy for opinion mining

The second phase of the project is to analyse the product review data in order to present the best of pros and cons of the products suggested using the Apriori algorithm. To begin with, a lot of data pre-processing had to be done in order to achieve efficient results.

Tokenization - breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of **tokenization**, some characters like punctuation marks are discarded.

Combine conjugate words - **negation** identification and detecting its scope within a sentence (text) are necessary in finding out the sentiments from a piece of text.

Removal of stop words - Insignificant words in a sentence like is, the, are, etc. are removed in order to shift the focus of the analysis only on the words that contribute to the mood of the sentence.

Next, the data is divided into train and test sets in order to perform the analysis using both Naive Bayes and Maximum Entropy.

Naive Bayes assumes that each data point/feature is independent and is given by -

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

Naive Bayes calculates the probability that classification y is correct given the features x1x1, x2x2, and so on equals the probability of y times the product of each x feature given y, divided by the probability of the x features.

Steps involved in implementing Naïve Bayes algorithm

Step 1 : Find unique words from reviews

Step 2 : Divide training dataset into two classes

Step 3 : Find #occurrence of each word in two classes

Step 4 : Calculate prior

Step 5 : Calculate likelihood for each word using step3

Step 6 : Using Step 4 and Step 5, for each review in test data calculate probability of review for a given class. Assign the review to the class which has highest probability.

The Maximum Entropy classifier is a probabilistic classifier which belongs to the class of exponential models. Unlike the Naive Bayes classifier that we discussed above, the Max Entropy does not assume that the features are conditionally independent of each other. The algorithm is based on the Principle of Maximum Entropy and from all the models that fit our training data, selects the one which has the largest entropy. The Max Entropy classifier can be used to solve a large variety of text classification problems such as language detection, topic classification, sentiment analysis and more.

Entropy is given by:

$$Entropy(t) = -\sum_{j} p(j|t) \log p(j|t)$$

Steps involved in implementing Maximum Entropy algorithm

Step 1 : Find unique words from reviews

Step 2 : Divide training dataset into two classes.

Step 3 : Find #occurrence of each word in two classes.

Step 4 : Using training data, find the weights of all the unique words.

Step 5 : Find the sum of all words in the test data multiplied by weight

Step 6 : If value is >= 0, classify as 1, else 0.

IV. Results

A. Rule Generation from Machine Learner implementing Apriori Algorithm

The dataset consists of 75,000 transactional records which consists of 49 distinct products. After discarding all products whose support is less than *minsup (0.02)*, we are left with 25 products as 24 products are filtered out. We then filter out the products whose confidence is less than the *minconf (0.01)*. Then, take the products which top six confidence values for every product and those corresponding products are the recommended products.The rules are sorted according to confidence values in descending order. For instance, in the figure below, 1 -> 4 3 2 5 7 9 implies that products with Ids 4,3,2,5,7 and 9 will be recommended when product with id 1 is being considered by the customer. We dump generated rules into a file which is used by the second half of the algorithm.

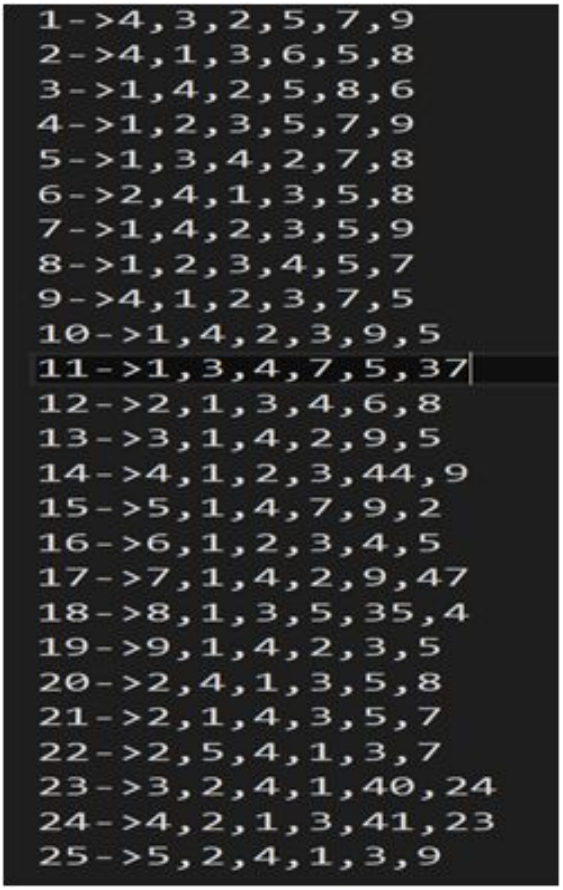
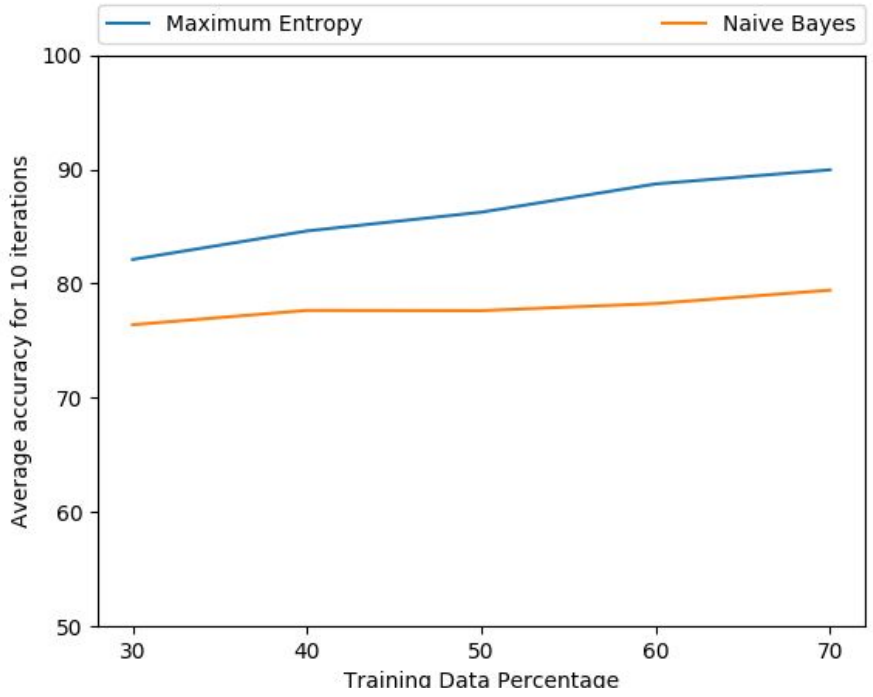


Fig - File consisting of the rules generated by the apriori algorithm

B. Results and Performance measure of Naive Bayes vs. Maximum Entropy

Due to the minimum assumptions that the Maximum Entropy classifier makes, we regularly use it when we don't know anything about the prior distributions and when it is unsafe to make any such assumptions. Moreover, Maximum Entropy classifier is used when we can't assume the conditional independence of the features. This is particularly true in Text Classification problems where our features are usually words which obviously are not independent. The Maximum Entropy algorithm requires more time to train comparing to Naive Bayes, primarily due to the optimization problem that needs to be solved in order to estimate the parameters of the model. Nevertheless, it provides more accurate results after computing these parameters, the method provides robust results and it is competitive in terms of CPU and memory consumption.



The results of classification performed by the algorithm on each product would look like:

```
Performing sentiment analysis on "OtterBox - Defender Series Case for Samsung Galaxy S8 - Black" reviews
Fit is  CON
Material is  PRO
Protection is  PRO
Screen Protection is  CON

Process finished with exit code 0
```

```
Performing sentiment analysis on "Tech21 - Evo Check Case for Samsung Galaxy S8 - Smokey/black" reviews
Button use is  CON
Fit is  PRO
Protection is  PRO
Screen Protection is  CON

Process finished with exit code 0
```

```
Performing sentiment analysis on "ZAGG - InvisibleShield HD Screen Protector for Samsung Galaxy S8+ - Clear" reviews
Dirty is  CON
Ease of use is  PRO
Peeling is  CON
Protection is  PRO

Process finished with exit code 0
```

C. Integration of smart suggestion engine in an ecommerce website

Fig - An example of integration of smart suggestion engine with the an eCommerce website.

In the above figure we can visually see the utility of integrating smart suggestion engine with an ecommerce website. Samsung S8 is the current product the customer is interested in. The smart suggestion engine utilizes association rule mining and opinion mining to recommend products and list their corresponding pros and cons.

V. Conclusions and Future Work

A. Lessons Learnt

While implementing the Apriori algorithm, we learnt the importance of effectively using databases for storing the transactions. We learnt that trying to perform all calculations without employing database is not only inefficient but makes the product sub-optimal and complex. Our initial implementation without using databases took 240ms to generate the result set. But, this solution is not feasible for real time datasets which contain millions and millions of records. So, we decided to use MySQL database to make the smart suggestion engine scalable for large datasets and this design choice decreased the access time to 7 ms.

Also, we learnt how to arrive at an optimal *minsup* and *minconf* value. When *minsup* and *minconf* is higher we will find less frequent itemsets and rules and the algorithm runtime is faster since most of the transaction items are eliminated in the early stages. Setting these parameters depends on how many rules we want to extract. We learnt that setting our configuration parameters to - 'minconf as 0.01 and minsup as 0.02' gave us the optimal results.

While implementing the opinion mining part of the project, it was observed that efficient data preprocessing is necessary for the algorithm to provide more accurate results. The subtleties in the sentences where an expression like 'not bad' actually means 'good', had to be taken care of to make correct predictions regarding the mood of the statement. Handling negations, removal of stopwords played an important role in the performance of the algorithms.

B. Potential Future Work

Cold start problem for new products will be a challenge for smart suggestion engine. If the smart suggestion engine does not have enough information in the database to form reliable recommendations the entire purpose of the smart engine is defeated. If information is not collected to form recommendations, the items will never be shown to any users, so information will never be collected. The smart suggestion needs to arbitrarily initialize new products to gather real time preferences.

Ratings are a valuable sign of his preferences. However, implicit signals of customer's interest, such as what items he clicks on, how long he reads them, which items are added to a wishlist or shopping cart, etc. should also be considered while building the transactional database.

Another potential area of future work is interpreting negative choices in addition to positive choices. The information contained in a user not choosing to click on a particular

recommendation should also be taken into account.For instance, for a laptop with i3 processor if the smart suggestion engine recommends chromebook, wireless mouse and laptop with i7 processor. If the user clicks on chromebook and laptop with i7 processor but not the wireless mouse, this information should also be considered for future recommendations.

C. Team Members and Contributions

Durga Prasad Kothinti (1211231616), Harsha Sharma (1211203211) and Srinivas Puranam(1211168267) were responsible for the first module of the smart suggestion engine which generates the association rules which are used by the second module to list the pros and cons of the recommended products. Implementing Apriori algorithm involved writing the code for the algorithm, deciding on an appropriate value for *minconf* and *minsup*, optimizing dataset by storing in a database instead of in memory as a list, designing database schema and writing SQL queries for retrieval, search and update.

Mounika Vuppala (1211040698), Venugopal Yella (1211040698), Neha Pulipati (1211214339) were responsible for the second phase of the project that dealt with opinion mining. This phase involved data collection by web crawling, implementing from scratch the Naive Bayes and Maximum Entropy algorithms and comparing the results obtained from the two.

References

[1] Min-Jeong Kang. Recommendation System for Make a Purchase to Application in Smart-Phone User. Hongik University Master's Thesis; 2011.

[2] Korea Communications Agency. Elements according to the step by utilizing big data technology Propulsion Trends and Implications.
Information Communication Technology Issues & Prospects No.10, 2013.

[3] R. Agrawal, T. Imielinski and A. Swami. Mining Association Rules between Sets of Items in Large Database. Proceedings of ACM SIGMOD on Management of Data, PP. 207-216, 1993.

[4] García, Enrique, et al. "An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering." *User Modeling and User-Adapted Interaction* 19.1-2 (2009): 99-132.

[5] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994.

[6] Farooq, Umar, et al. "Negation Handling in Sentiment Analysis at Sentence Level." *JCP* 12.5 (2017): 470-478.

[7] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up: sentiment classification using machine learning techniques." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002.

[8] Riedl, John. "Research challenges in recommender systems." *Tutorial sessions Recommender Systems Conference ACM RecSys*. 2009.