iRECOMMENDER FOR E-COMMERCE

17-035

Dissertation

M.S.D Dharmawardhana

W.W.G.B.P Bandara

A.G.D Ugayanga

K.M.S Bandarage

Bachelor of Science Special (honors) In Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

October 2017

iRECOMMENDER FOR ECOMMERCE

17-035

Dissertation

(Dissertation submitted in partial fulfillment of the requirement for the Degree of Bachelor of Science Special (honors) In Information Technology)

M.S.D Dharmawardhana - IT14048906

W.W.G.B.P Bandara - IT14015472

A.G.D Ugayanga - IT14034350

K.M.S Bandarage –IT12010554

B.Sc. (Special Honors) in Information Technology

Sri Lanka Institute of Information Technology

October - 2017

Declaration

We declare that this is our own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, we hereby grant to Sri Lanka Institute of Information Technology the nonexclusive right to reproduce and distribute our dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)

Signature: Date:

The above candidates have carried out research for the B.Sc. Special (Hons) degree in IT Dissertation under my supervision.

Signature of the supervisor: Date:

ABSTRACT

In e commerce industry product recommendation gives high impact on the number of sales. To make accurate product recommendations on time, we need to have a best preforming, highly accurate and scalable recommendation engine. There are two types of common recommendation engines, Collaborative based filtering engines and Content based filtering engines. Collaborative filtering based on historical data collected from the ecommerce web site and content based filtering based on user's opinions. In here we used social media to extract the user's opinions in order to overcome the cold start problem. And nowadays e commerce owners are struggling to combine above mentioned two approaches to make hybrid highly accurate recommendations. This Dissertation is intended to address the above mentioned problems.

As a solution for above scenario we made a recommendation engine called iRecommender. Which can do collaborative filtering, Content based filtering and combine both recommendation engine approaches to make hybrid recommendations by analyzing the live data streams comes from the social media and e commerce historical datasets. To extract user's opinions from social media sites we uses NLP techniques available with the NLTK package.

Since there are millions of records to process at single second, we used Apache Spark Distributed processing engine on top of the Hadoop (Hadoop Distributed File System) to process the data. As the final knowledgebase we used Neo4j graph database and to achieve the Distributed collaborative filtering we used alternating least squares (ALS) algorithm with apache spark machine learning capabilities. For the real-time data stream processing we used Apache Hadoop map reduce technique.

Through our research iRecommender, we successful gained the user friendly, highly Scalable, highly available comprehensive and cost effective Recommendation engine for efficient, accurate and effective product recommendations.

ACKNOWLEDGEMENT

The work described in this dissertation was carried out as the 4th year research project for the subject Comprehensive Design Analysis Project. The completed final project is the result of combining all the hard work of the group members and the encouragement, support and valuable inputs given by many others. Therefore, it is our duty to express our gratitude to all who gave us the support to complete this major task. We are deeply indebted to our supervisor Ms. Dinuka Wijendra and Lecturers of Sri Lanka Institute of Information Technology whose suggestions, constant encouragement and support in the development of this research, particularly for the many informative and constructive discussions. We are also extremely grateful to Mr. Jayantha Amararachchi, Senior Lecturer who gave and confirmed the permission to carry out this research and for all the encouragement and guidance given.

We also wish to express our honorable gratitude for our families, colleagues and friends for all their help, support, interest and valuable advice. Finally, we would like to thank all others whose names are not listed particularly but have given their support in many ways and encouraged us to make this a success.

Table of Contents

Contents

List of Figures

List of Tables

List of Abbreviations

MSE	Mean-Square Error
RMSE	Root-Mean-Square Error
HDFS	Hadoop Distributed File System
OS	Operating System
LAN	Local Area Network
CSV	Comma Separated Value
TF	Term Frequency
RDD	Resilient Distributed Dataset
FR	Functional Requirement
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit

1. INTRODUCTION

With the evolution of Internet technology people have the trend of using internet services to get their ornaments. When it comes to the marketing, most of the market places are converted as ecommerce platforms where goods and services are offered through the internet. Even though there are number of ecommerce recommendation [1] platforms available, the owners are struggling on the personalized product suggestions for their specific customers. Since emergence of the ecommerce concept most of the university and non-university researchers have followed number of research methodologies with integration of newest technologies available to find out the best recommendation engine for product suggestions. Their goal was to make the product recommendation platform more efficient and more accurate. Even though the most of researches have been involved, they couldn't come up with the best solution for an efficient and accurate solution. This dissertation is written to discuss about what methodologies we have followed to overcome the above mentioned problems and how we achieved the final goals. Specifically, this is intended to address the existing cold start [2] problem and hybrid [3]recommendation concepts with NLP opinion [4] mining concepts.

1.1. Background context

When it comes to recommendation engines there are two types of recommendation engines with five major problems.

Types of recommendation engines

- **Content based filtering recommendation engines**
  Content-based filtering [5] methods are based on a description of the item and the profile of the user's preference.
- **Collaborative based filtering recommendation engines**
  Collaborative filtering [6] is a method of making automatic predictions (filtering) about the interests (Product Ratings) of a user by collecting preferences or taste information from many users (collaborating)

Problems with existing recommendation engines

- **Cold Start Problem** [2]

system cannot draw any useful inferences about a user before gathering some information about them

- **Scalability Problem**
  When number of users increases system couldn't cater all the users as expected

- **Inefficient data cluster management**
  Most of the recommendation engines use apache Hadoop, HP or Oracle cluster solutions to improve recommendation engine processing power. Those techniques are very costly and processing job managements are not efficient.

- **Use only one type of Recommendation approach**
  Most of the recommendation engines use only one type of approach to make recommendations then the results are tending to inaccurate and inappropriate.

- **Database delay**
  Most of the recommendation engines use relational databases. To get a single output from relational databases we have to use table joins. Joins are very in efficient so that will cause to delay the real time recommendations

To address the above problems, we come up with social media data consumed distributed computing solution. In here to do the content based filtering we used user's social media [4] extracted data and for collaborative filtering we used Apache Spark machine learning capabilities with ALS [7]algorithm. As the knowledge base here we used neo4j graph based database [8] since it has no any delay when it comes to the pattern matching.

1.2.Research gap

Although there are several existing recommendation engines are available for product recommendation, those are not addressed the solutions provided by our product iRecommender. Our solution consists with many functionalities and features others do not have. In this research our main target was to find a way to make hybrid recommendations and find a way to overcome the cold start problem

Most of researchers used one recommendation approach to predict the customer tastes. Then we found that the most of the time that test predictions are not feasible and accurately personalized for each specific user, Because of lack of using the recommendation methodologies and technologies. And in here we have found out many of the researchers are continuously trying, to make a solution for overcome the Cold Start [2] problem. researchers suggest deep learning [9], content based recommendation [10], Comparing product specifications etc. And the other serious problem we found the response time of recommendation engine is very high because most of the recommendation engines use relational databases [8]as their knowledge base. Even though those methods and technologies are used to overcome the Cold Start problem, the Cold Start problem is still there in the ecommerce websites. And we found that the personalized recommending accuracy is not best at most of the e commerce web sites. So to address the Cold Start, scalability, availability, high response time and Low personalized recommending accuracy rate problems we developed the product called iRecommender. Following table 1 shows what are the major differences between our product and existing products

Table 1Comparison with existing products

	Netflix	eBay	Amazon	Ali Express	Walmart	iRecommender
Cold Start Problem	✓	✓	✓	✓	✓	X
Database delay	✓	X	X	X	X	X
No scalability	X	X	X	X	X	X
Inefficient cluster management	✓	X	X	X	X	X
Usage of only one type of recommendation technology	✓	✓	✓	✓	✓	X

1.3. Research problem

There are several methods on suggesting products to the users of ecommerce websites. But almost all of them are making their suggestions based on one recommendation engine approach either collaborative based filtering or content based filtering. When new users register with the ecommerce platform the websites don't have any suggestions to the newly registered users. Then the newly registered users get generalized recommendations until they buy something from the e commerce website. As this is an inefficient method, users are not getting the exact product suggestions. The failure of product suggestions on an ecommerce website leads to more issues. Users tend to use ecommerce to save their time. As suggestions are failed, users will have to search again and again for products. Sometimes the users have no idea on which keywords to search. By taking in to consideration above issues, the impression that users have on the website are getting reduced. This will lead to decrement in sales on the site.

1.4. Research objectives

The iRecommender comprehensive recommendation engine mainly focused on seven research objectives and with the completion of the project we hoped to come up with commercially valued product with fulfillment of those objectives.

1. **To establish low cost solution for small to medium scale companies**

   We observed that for the small and medium scale companies running their own recommendation engines with obsoleted technologies with very law recommendation accuracy. Since our product is available to plug and play and very small configurations with any e commerce platform it is very easy to integrate our system with any e commerce platform. And purchasing power of growing companies are limited. They can't spend hundred thousands of dollars on single software on their path. Our optimal objective is serve that small to medium scale business community to improve their business with the help of iRecommender low cost solution.

2. **To Support for both customer and Business perspectives**

   Customer perspective

   • Without any trouble, user could buy the products which he needed most.
   • Time saving
   • Customer doesn't need to have any special technical knowledge to search.

   Business perspective

   • Increase sales of the e-commerce web sites
   • Recognize the customer taste when browsing.
   • Increase loyalty of Customers.

3. **To make User friendliness and Simplicity**

   All the available recommendation engines are very hard to configure with any e commerce platform and that requires reach technical knowledge. Since our product has very small and simple configurations with any e commerce platform that is going to be user friendly and easy to the handle its operations

4. **To provide Hybrid recommendations**

   iRecommender introduce the new concept of hybrid recommendation in order to make high accurate product recommendations that will tend to increase the sales on the site.

5. **To Support for Big data handling with Hadoop and apache spark**

   E commerce sites generate millions of records per second. To process and handle such huge volume of data we need to have high performance processing power. To achieve that we used HDFS (Hadoop Distributed File System) and apache spark distributed computing system.

6. **To Development of powerful data manipulation techniques**

   The main power source of iRecommender is data. We need to properly cleansed data to extract, transform and load into the system. Prior manipulation of big data gives ease of ETL and much efficient way to carry out the prediction
   process.

7. **To Predictive analysis of data with predictive models**

   In order to come up with solutions for problems in collaborative filtering we need to develop best predictive models. There are several approaches like classification, clustering, regression, artificial intelligence, neural networks, association rules and genetic algorithms etc. Among those we chose clustering which is most scientific, widely accepted and best approach for our problem. And here we used alternating least squares (ALS) algorithm for predictive model building which is highly optimized for distributed collaborative filtering

2. METHODOLOGY

This section describes detailed descriptions the approach that we used to made the iRecommender. It includes how we conducted the feasibility study before beginning of the project, requirement gathering phase, design of the architecture, physical implementation and testing of the product. Also what are the tools, technologies and libraries used in order to make iRecommender as better recommendation engine.

As a software development methodology we followed agile development methodology for iRecommender instead of classical waterfall model or iterative waterfall model, because it's had better chance to re-correct the mistakes done in previous phases. When we identified our research problem we conducted a literature survey on the domain to identify major problems, solutions and already available Recommendation engines in the domain. We designed our high level architecture diagram, system diagram, class diagram and use case diagrams for our solution. Here onwards we summarized the deep informative methodology of project iRecommender.

2.1. Feasibility Study

Traditional feasibility studies have been carried out by the team to ensure that the project is technically, operationally and financially feasible. The team explored various recommendation engines such as amazon, Walmart, Netflix and eBay etc. And also several other tools and technologies were explored for the other research areas such as collaborative filtering and content based filtering. Finally, the team came to the conclusion that the project is financially and technically feasible since most of the technologies to be used are free and open source.

To determine the Strengths Weaknesses Opportunities and Threats which impacting our system we performed a basic SWOT analysis regarding commercial value in basically. Table 2 shows our SWOT analysis

Table 2 SWOT analysis

Strengths	Weaknesses
• Can keep product purchase cost below. • Can keep recommendation accuracy high than our competitors. • Introduce new concept of hybrid Recommendation • Can keep the system more scalable	• No partnerships and no strong relationships with our clients. • Lack of features with comparing well-known recommendation engines. • Lack of marketing ability than our competitors.
Opportunities	Threats
• Internet can be used as a marketing tool. • Recommendation engines commercially available in market are very hard to configure with new platform than our tool.	• Negative attitude in clients regarding low cost solutions. • Strong relationship/attitude in clients for our competitors.

Finally, we agreed with this project is worthy and it's very beneficial to carry out within 1 year

2.2. Requirements gathering & analysis

After the feasibility study of the project we moved to requirement gathering and analysis phase. We had successfully conducted several meetings with our project stakeholders and supporters. Then we realized the recommendation engine should be easy to configure with any e commerce platform and should be fast enough to cater hundreds of users at ones. By doing a lot of search we found that our system should be able to handle real-time data streams and millions of records per second. So this data need to handle in very efficient and effective way otherwise system will slow down ultimately.

Simultaneously we tried to analyze the cold start problem then we realized to get another independent data source to solve the cold start problem. we found that most efficient reliable one is using social media data of users to solve this problem. Since the cold start problem happens when recommendation engine doesn't have any historical data about the customer's preferences. And in here we found that to handle the real-time data streams containing user's opinions we need to develop real-time data stream listeners.

Next we thought about the collaborative filtering. When we talk about collaborative filtering the predictive model accuracy is the most important factor. So in here we developed real-time live graph in order to monitor the prediction accuracy. For further more details about requirements please go to appendix A

As additional work we looked over some already available recommendation engines and blogs related to the recommendation engine designing, also we created our own dataset and tried some tutorial to identify and get clear picture about the environment what we are actually dealing with. Once we completed with all these points we immediately moved to design phase to create the architecture for our iRecommender system.

2.3. Design

The design phase discusses the structure of the project. The system basic functionalities and its developments are elaborated in this phase. Designing under the proposed system is mainly focused on accuracy of the recommendation system. Since we use the agile SDLC model for develop the system, it will be easy to manage and make changes if there is any problem in the designing or if there is anything to change according to users.

iRecommender consist of four main components which need to implement.

- Social media mining and analyzing component.
- Opinion mining and analyzing component.
- Trend prediction component.
- Web mining and validation component.

2.3.1 System architecture of whole system

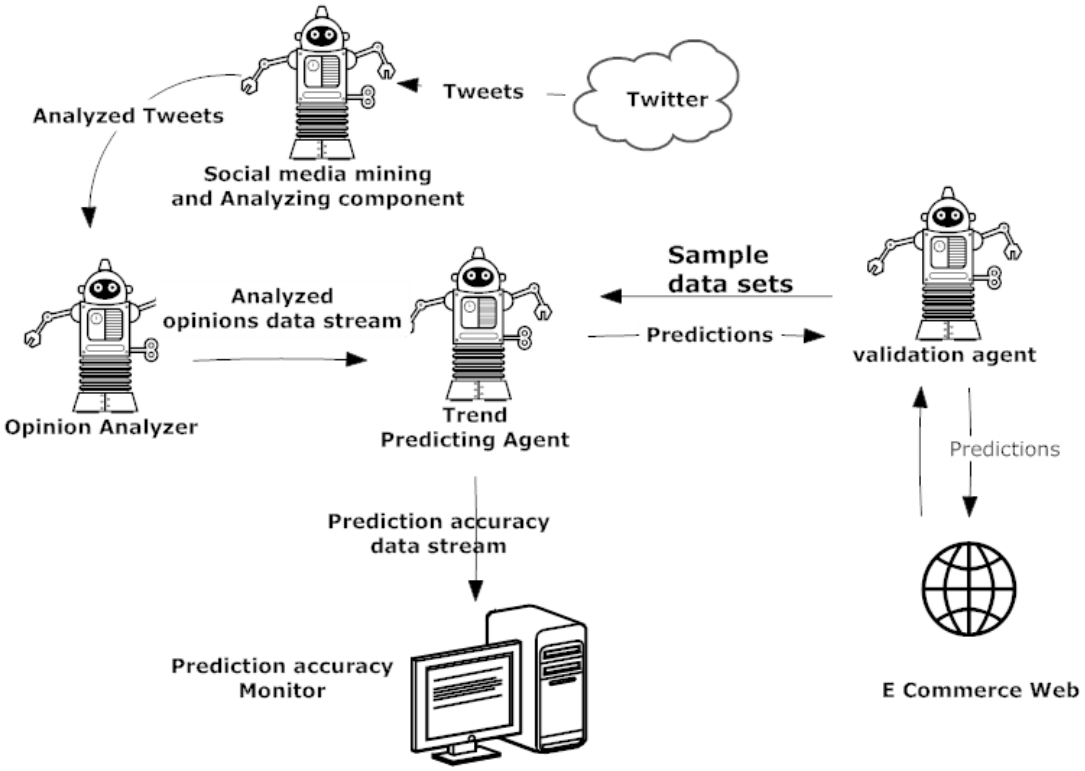


Figure 1 System architecture

2.3.2 Social media mining and analyzing component

This component is developed for collect each user's tweets form the twitter and to do the data cleansing by removing the noise words in order to extract the user's opinions about the products listered on a given ecommerce site.

Social media mining and analyzing agent's responsibilities.

- Analysis each customer's social media
- Remove noise words of tweets
- Identify the products name if customer have mentioned web site's selling products.
- Identify the emotional words which can get customer's opinion about that products.
- Finally Pass the Data to user opinion predictor agent.

Analysis each customer's social media

In the current research we used Twitter and a system for collecting user generated data (i.e. tweets) for a period of time. Twitter has attracted much interest the last few years from

researchers, because Twitter has provided a public API with many resources to developer perspectives to do some research.

Twitter provides the Application Program Interface (API) which allows programmatically accessing and retrieving tweets by a query term. The Twitter API takes a set of parameters related to features such as the language, the format of the results, the published date, the number of tweets and a query and returns the tweets that contain the query and meet the parameters. In our system, only tweets written in English were collected. An application was developed to collect and store the data. The application queried the Twitter. The data were stored in a ".txt" format to facilitate us analyze.

This component analyzing current logged person twitter accounts and retrieved tweets. Those tweets are stored in a file name in user's web ID. It means one user has one file with their tweets. At a time, if there are five users are logged in system then five files are generated. In here those tweets are retrieve as it is. There can be Hash Tag, Links and emoji etc. So, it should be clean to do further process.

Remove noise words of tweets

The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

Sentiment classification over Twitter is usually affected by the noisy nature (abbreviations, irregular forms) of tweets data. A popular procedure to reduce the noise of textual data is to remove noise words by using pre-compiled noise words lists or more sophisticated methods for dynamic noise words identification.

Pre define words List:

{‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’, ‘until’, ‘below’, ‘are’, ‘we’, ‘these’, ‘your’, ‘his’, ‘through’, ‘don’, ‘nor’, ‘me’, ‘were’, ‘her’, ‘more’, ‘himself’, ‘this’, ‘down’, ‘should’, ‘our’, ‘their’, ‘while’, ‘above’, ‘both’, ‘up’, ‘to’, ‘ours’, ‘had’, ‘she’, ‘all’, ‘no’, ‘when’, ‘at’, ‘any’, ‘before’, ‘them’, ‘same’, ‘and’, ‘been’, ‘have’, ‘in’, ‘will’, ‘on’, ‘does’, ‘yourselves’, ‘then’, ‘that’, ‘because’, ‘what’, ‘over’, ‘why’, ‘so’, ‘can’, ‘did’, ‘not’, ‘now’, ‘under’, ‘he’, ‘you’, ‘herself’, ‘has’, ‘just’, ‘where’, ‘too’, ‘only’, ‘myself’, ‘which’, ‘those’, ‘i’, ‘after’, ‘few’, ‘whom’, ‘t’, ‘being’, ‘if’, ‘theirs’, ‘my’, ‘against’, ‘a’, ‘by’, ‘doing’, ‘it’, ‘how’, ‘further’, ‘was’, ‘here’, ‘than’}

Figure 2 predefined words

We would not want these words taking up space in our data, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words.

By referring research papers, we have found few noise words identification method to twitter data from different dataset and observed how removing noise words affected. using pre-compiled lists of noise words negatively impacts the performance of Twitter sentiment classification approaches. On the other hand, the dynamic generation of noise words lists, by removing those infrequent terms appearing only once in the corpus, appears to be the optimal method to maintaining a high classification performance.

As mentioned above, hope to develop an algorithm to identify and remove noise words from the tweets by considering following aspects.

- By using pre define noise words list.
- Removing single letters / punctuation marks and other symbols.
- Removing words with low inverse document frequency.
- Remove Links
- Remove Emoji

In here our retrieved data retrieved in encoded form and also we can’t keep some kind of words which are links, emoji, hash tag, symbol and other unnecessary character. Before put in NLTK process we need to clean those things too.

Example:

- Tweet: This is a sample sentence, showing off the stop words filtration.

- First split the words: outcome is like array

['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', .']

- Then Filter the noise words and avoid them:

['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

[‘a’, ‘the’, ’off’] words are removed and those are not considered further process.

In above example is not affected by any kind of things. Refer following example.

- Tweets: "pain is my old friend 😆😅 #FilmDoctorStranger"

In that case, process need decode data to process, otherwise NLTK algorithm can’t process. So that time we need to text parsing. We can’t decode this because it affects [“U’ Pain”] to avoid this it has implemented a function.

```
regex_str = [
    emoticons_str,
    r'<[^>]+>',  # HTML tags
    r'(?:@[\w_]+)',  # @-mentions
    r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)",  # hash-tags
    r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&amp;+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+',  # URLs

    r'(?:(?:\d+,?)+(?:\.?\d+)?)',  # numbers
    r"(?:[a-z][a-z'\-_]+[a-z])",  # words with - and '
    r'(?:[\w_]+)',  # other words
    r'(?:\S)'  # anything else
]
```

Figure 3 splitting example

using above regex string it filter those kind of words and it identify the special character. Then it no need to decode text. So it works normal way.

Identify the products name and emotional words

In this function have to identify the products names by looking tweets words which are selling e-commerce site. So that we have to develop products dictionary by using e-commerce web site's database data. Then After we should identify which are the products customer have mentioned by using lexical analyze.

Other thing is, we should have to identify the emotional words. To do that, we will use algorithm is the emotional dictionary of the "Linguistic Inquiry and Word Count" (LIWC) software (Pennebaker J. and R., 2001). LIWC contains a broad dictionary list combined with emotional categories for each lemma that were assigned by human.

Note about the dictionaries.

A dictionary is no more than a list of words that share a category. For example, you can have a dictionary for positive expressions, and another one for stop words.

The design of the dictionaries highly depends on the concrete topic where you want to perform the opinion mining. Mining products opinions is quite different than mining food opinions. Not only the positive/negative expressions could be different but the context vocabulary is also quite distinct.

Defining a structure for the text

Before writing code, there is an important decision to make. Our code will have to interact with text, splitting, tagging, and extracting information from it.

This is a key decision because it will determine our algorithms in some ways. We should decide if we want to differentiate sentences inside a paragraph. We could define a sentence as a list of tokens. But what is a token? a string? a more complex structure? Note that we will want to assign tags to our token. Should we only allow one tag per-token or unlimited ones?

Infinite options here. We could choose a very simple structure, for example, defining the text simply as a list of words. Or we could define a more elaborated structure carrying every possible attribute of a processed text (word lemmas, word forms, multiple tagging, inflections...)

As usual, a compromise between these two extremes can be a good way to go.

- Each text is a list of sentences
- Each sentence is a list of tokens
- Each token is a tuple of three elements: a word form (the exact word that appeared in the text), a word lemma (a generalized version of the word), and a list of associated tags

Preprocessing the Text

Once we have decided the structural shape of your processed text, we can start writing some code to read, and pre-process this text. With pre-process I mean some common first steps in NLP such as: Tokenize, Split into sentences, and POS.

We can perform a basic text preprocessing, where the input is the text as a string and the output is a collection of sentences, each of which is again a collection of tokens.

By the moment, our tokens are quite simple. Since we are using NLTK, and it does not lemmatize words, our forms and lemmas will be always identical. At this point of the process, the only tag associated to each word is its own POS Tag provided by NLTK.

Example: "What can I say about this place. The staff of the restaurant is nice and the eggplant is not bad."

Spitted words:

[['What', 'can', 'I', 'say', 'about', 'this', 'place', '.'], ['The', 'staff', 'of', 'the', 'restaurant', 'is', 'nice', 'and', 'eggplant', 'is', 'not', 'bad', '.']]

POS tagged words:

[[('What', 'What', ['WP']), ('can', 'can', ['MD']), ('I', 'I', ['PRP']), ('say', 'say', ['VB']), ('about', 'about', ['IN']), ('this', 'this', ['DT']), ('place', 'place', ['NN']), ('.', '.', ['.'])], [('The', 'The', ['DT']), ('staff', 'staff', ['NN']), ('of', 'of', ['IN']), ('the', 'the', ['DT']), ('restaurant', 'restaurant', ['NN']), ('is', 'is', ['VBZ']), ('nice', 'nice', ['JJ']), ('and', 'and', ['CC']), ('eggplant', 'eggplant', ['NN']), ('is', 'is', ['VBZ']), ('not', 'not', ['RB']), ('bad', 'bad', ['JJ']),

Then after we have to identify the words which are need us. So we use pre define dictionary. As follows.

```
iPhone: [product]
soap: [product]
cream: [product]
phoneCover: [product]
```

Figure 4 Snap shot of dictionary

Then we tagging the words with dictionary. Our output as follow words are tagged with dictionary.

[(''iPhone', 'iPhone', ['product', 'NN']), (‘soap’, ‘soap’, ['product', 'NN'])] likewise those words can identify

2.3.3 Opinion mining and analyzing component

Who is User Opinion predicting agent

This is the one of major parts of the iRecommender System which is analyzing the user’s feelings/emotions and extract the opinion as Positive, Negative or Neutral. These user feelings/emotions are extracted and stored in a csv file by the Social media mining and analyzing agent. These data will be input to the User Opinion predicting agent. Using Natural Language Processing (NLP) techniques, develop a User Opinion predicting Algorithm. This Algorithm will be output the Negativity, Positivity or Neutrality. As well as, rating of that opinion will also extract. These set of data store inside a csv file according to each user. Each user has different csv file.

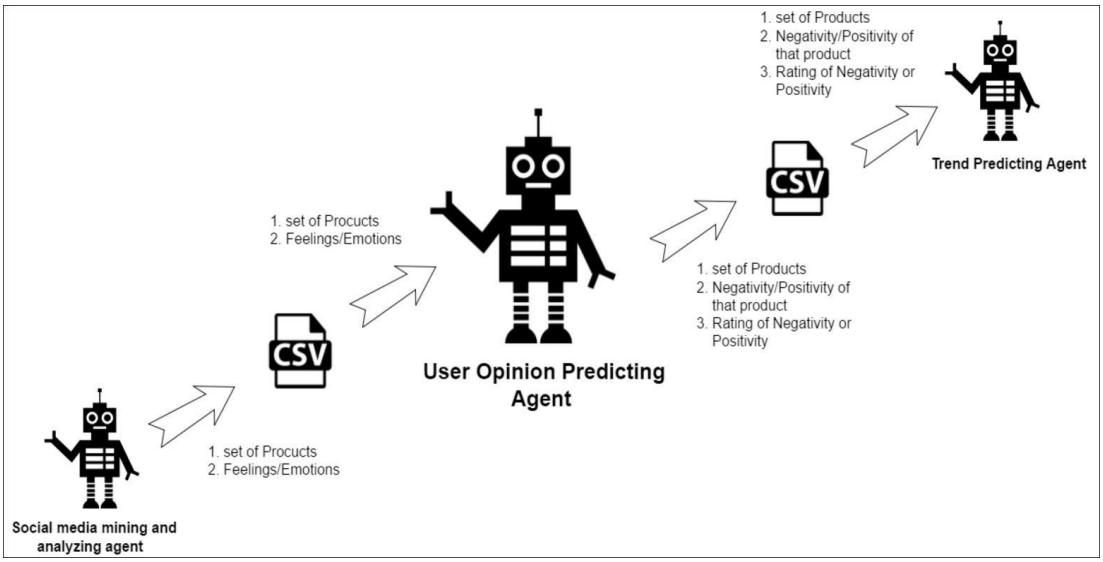


Figure 5 Work flow of opinion predicting agent

What is an Opinion

The simple meaning of the Opinion is "a personal belief or judgment that is not founded on proof or certainty".

But, “the fact that an opinion has been widely held is no evidence whatever that it is not utterly absurd”.

Word of mouth is powerful though.

What is an Opinion to a Machine

It is a "quintuple", an object made up of 5 different things

$$(o_j, f_{jk}, so_{ijkl}, h_i, t_l)$$

Equation 1 quintuple

Oj - The thing in question (i.e. product) / Target entity
Fjk - A feature/aspect of oj
Soijkl - The sentiment value of the opinion of the opinion holder hi on feature fjk of object oj at time tl. soijkl is Positive, Negative or Neutral
Hi - Opinion holder
Ti - The time when Opinion is expressed

These 5 elements have to be identified by the machine
(defined by Bing Liu in the NLP handbook)

The Main Process of opinion miner

The main Objective of this section goes to extract the Negativity or Positivity of the feelings (Key words) which are taken from the .txt file given by the Social Media Mining and Analyzing Agent. This Main Approach is divided in to 3 major sub-parts.

- Get set of data.
- Execute User Opinion Predict Algorithm for Opinion Mining.
- Store the output result.

Get set of data

In the first step of this section get the set of the words from the .csv file which are given from the early stage. These data should read and input to the Algorithm which is create from this section.

Execute User Opinion Predictor Algorithm for Opinion Mining

In this step, System will analyze the extracted Key words (Feelings) and determine the Positivity or Negativity of these Key words. To do that, develop User Opinion Predictor Algorithm.

User Opinion Predictor Algorithm

This can be done by using Sentiment Analysis. Basic Sentiment Analysis algorithms use Natural Language Processing (NLP) to classify words as positive, neutral, or negative. Keyword spotting is the simplest technique leveraged by sentiment analysis algorithms.

Keyword spotting is the simplest technique leveraged by sentiment analysis algorithms. Input data is scanned for obviously positive and negative words like 'happy', 'sad', 'terrible', and 'great'. Algorithms vary in the way they score the words to decide whether they indicate overall positive or negative sentiment. Different algorithms have different libraries of words and phrases which they score as positive, negative, and neutral.

After getting extracted data set which are stored in .csv file, we are ready to execute the sentiment analysis algorithm on each Key words (feelings/emotions). Then, we will calculate an average score for all the Key words (feelings/emotions) separately.

Using this User Opinion Predictor Algorithm, we give the Key words as input to this Algorithm. Output of this will be the Negative, Positive or Neutral.

Store the output result

The output of the User Opinion Predictor Algorithm, should be stored inside a .csv file. This is stored as combination of product and related Opinion on this Product which is analyzed and computed by Algorithm.

2.3.4 Trend prediction component

The Trend Predicting Agent is a software system, which is developed to make hybrid recommendations for each customer on a given e-commerce platform. It is going to find out relationships between product to product and user to user using ALS algorithm and going to provide efficient, accurate, and quick personalized recommendations for each customer of the ecommerce platform according the found out tremendous insights from mentioned relationships. As feeds(inputs) for this system I used analyzed social media data and e commerce historical data.

Below figure shows work flow of the trend predicting component

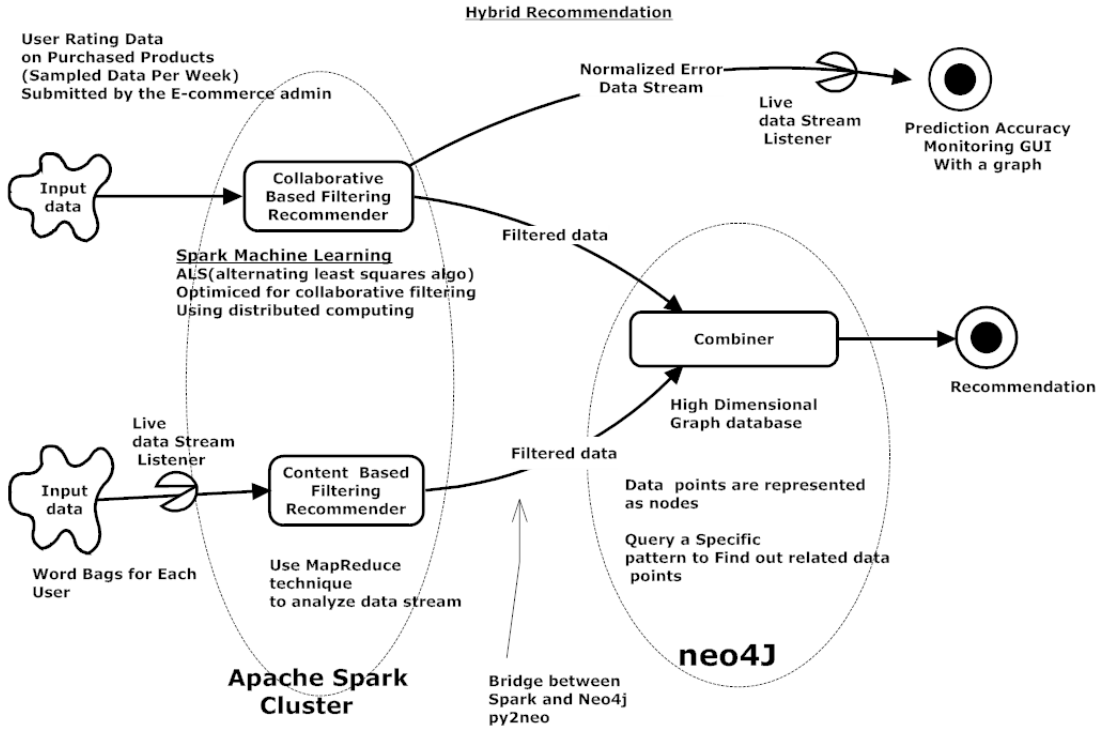


Figure 6 Work flow of the Trend Predicting component

System architecture of Trend predicting component

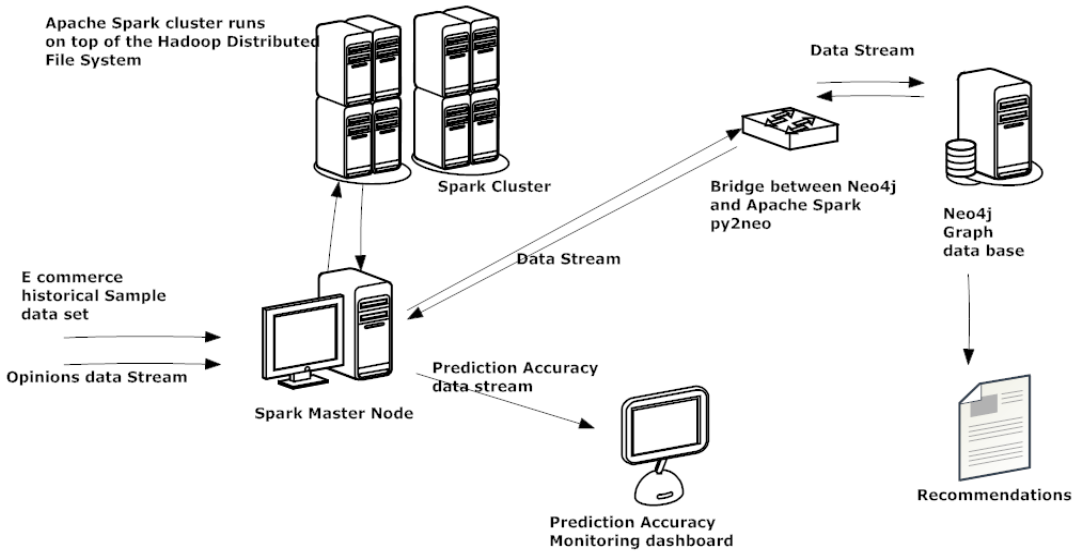


Figure 7 System architecture of Trend predicting component

As shown in the diagram two data streams input to the apache spark master node Then those data separates and distribute to apache spark worker node cluster in order to do the data analysis. After the data analyzation all the data records are collected to the master node and all the data pulled in to the graph in order to do the hybrid pattern matching.

While the master node doing its data analysis and predictive model building a data stream send to the predictive model accuracy monitoring dash board.

Apache spark cluster management architecture

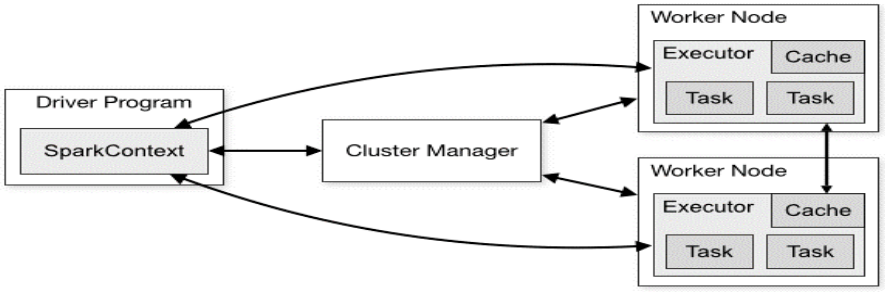


Figure 8 Spark cluster management

The above figure shows how to manage and communicate the data analyzation jobs in apache spark worker nodes and master node.

Design content based filtering system

When the User's social media data analyzed into the positivity and negativity and pass to the Apache spark cluster, first of all it is doing the data cleansing task. So at the data cleaning all the unwanted characters will be removed and make all the words to lowercase and Extract the positive dataset, then cluster the data accordingly item (product) wise using Map Reduce [11] technique and then using a Term Frequency(TF-IDF) it is going to find out most suitable (Highest frequent product name) product recommendations for specific customers of the ecommerce. After finding recommendations the dataset feed in to a graph for further analysis.

Map Reduce Example

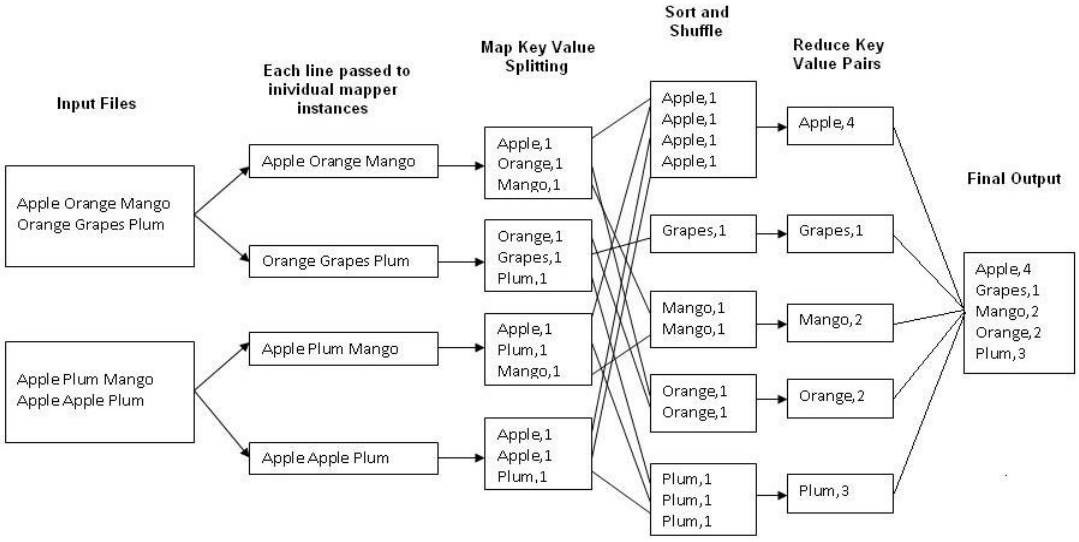


Figure 9 Map Reduce Example

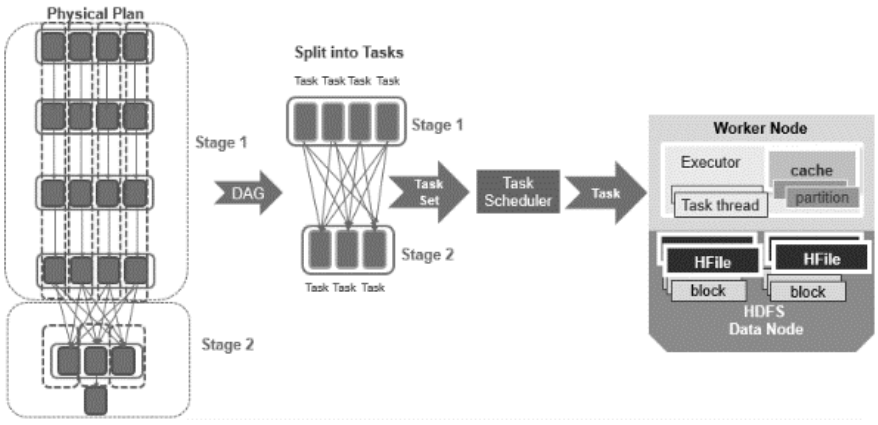


Figure 10 Spark map reduce Work flow

Design collaborative based filtering system

To develop collaborative based filtering system, we need to have historical dataset from the e commerce web site. And to make updated and most accurate predictions we need to have most recent datasets. So in here e commerce website admins supposed to submit daily or weekly wise historical (product ratings given by users) data set to the apache spark master node. Then historical dataset automatically feed it to the machine learning algorithm called Alternating Least Squares (ALS) algorithm [12] [13]. Then the ALS approximates the sparse user item "rating" matrix of dimension K as the product of two dense matrices User and Item factor matrices of size U×K and I×K (see figure 7 below). The factor matrices represent hidden features which the algorithm tries to discover. One matrix tries to describe the latent or hidden features of each user, and other one tries to describe latent properties of each item. ALS is an iterative algorithm. In each iteration, the algorithm alternatively fixes one factor matrix and solves for the other, and this process continues until it converges.

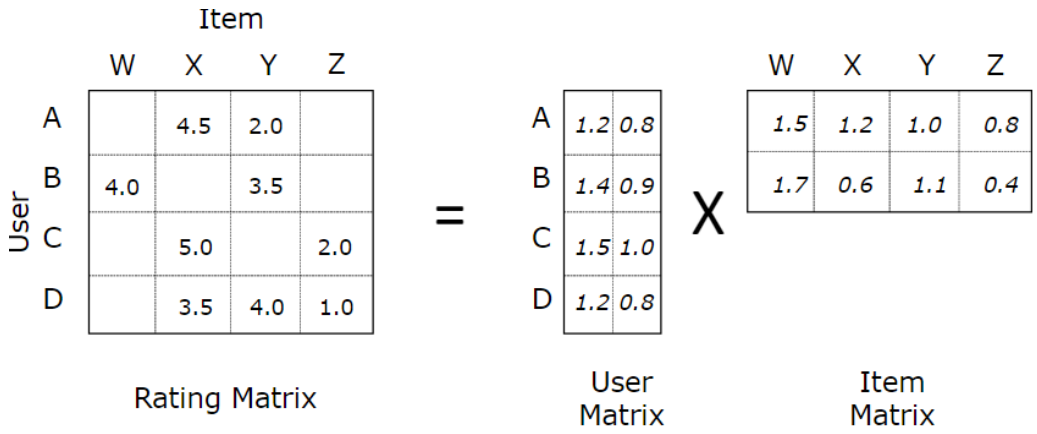


Figure 11 ALS algorithm example

Then finally ALS algorithm predict recommendations based on item-to item similarity. And updates the graph database. In here the basic idea is people who liked similar items in the past will like similar items in the future.

Following figure shows how the ALS algorithm build its predictive models in order to make recommendations

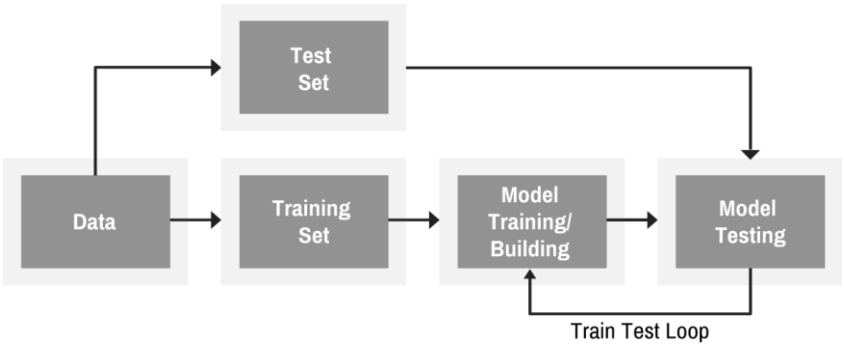


Figure 12 ALS algorithm Predictive model building cycle

Design the recommendations combiner

In here the graph concept is used to combine both collaborative filtering and content based filtering. Then the recommendation filters feed all the data points in to a high dimensional graph and then we could search the graph for specific patterns. Neo4j graph database used to do this task.

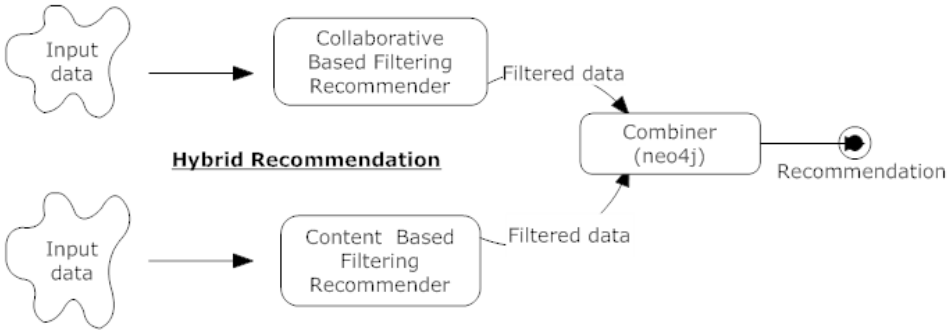


Figure 13 work flow of hybrid recommendation

Neo4j is highly optimized graph database for pattern [14]matching. When we pull data points in to graph, all the data points represented as nodes and all the relationships represented as vertices. Then we can query the database for search specific data patterns. I have shown below in figure 11 expected data points pattern [15]to make hybrid [3] recommendations

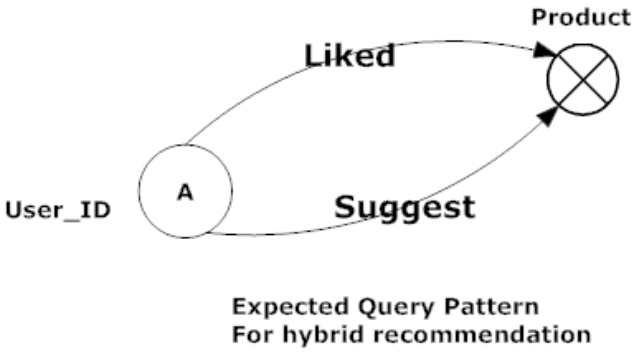


Figure 14 expected hybrid query pattern

In here from user_Id to product "Liked" relationship is generated using content based filtering system. And the "Suggest" relationship is generated from the collaborative filtering [16] system. Using cypher query we find out which product satisfies the both relationships.

2.3.5 Customer behavior analyzer agent and Validation Agent

Who is User Customer Behavior Analyzer?

Customer behavior analyzer is the component which tracks the user activities on the e commerce website. This component is triggered at the point of login of the customer and important data which are required for the iRecommender's validation agent are collected through the actions performed by the user.

Who is Validating Agent

This is the component of the iRecommender system which is responsible for the validation of the prediction done by Predicting agent. The validating agent tracks and analyses the user activities that were collected by the Customer Behavior Analyzer. The analyzed data are exposed to web mining techniques to validate the product suggestion done by the predicting agent.

How does Customer Behavior Analyzer work?

The customer behavior analyzer is responsible on tracking the user activities on the website. Once the user is logged in to the website, the analyzer is triggered and all the activities of the customer are recorded in a log file. The Log files in different web servers maintain different types of information [5]. The basic information present in the log file are

- User name: In some web sites the user identification is made by getting the user profile and allows them to access the web site by using a user name and password. In this kind of access, the user is being identified uniquely so that the revisit of the User can also be identified
- Visiting Path: The path taken by the user while visiting the web site. This may be by using the URL directly or by clicking on a link or through a search engine.
- Path Traversed: This identifies the path taken by the user with in the web site using the various links. Time stamp: The time spent by the user in each web page while surfing through the web site. This is identified as the session.
- Page last visited: The page that was visited by the user before he or she leaves the web site.
- Success rate: The success rate of the web site can be determined by the number of downloads made and the number copying activity under gone by the user. If any purchase of things or software made, this would also add up the success rate.
- User Agent: This is nothing but the browser from where the user sends the request to the web server. It's just a string describing the type and version of browser software being used.
- URL: The resource accessed by the user. It may be an HTML page, a CGI program, or a script.
- Request type: The method used for information transfer is noted. The methods like GET, POST.
- In addition to the information available on the log files, the customer behavior analyzer optimizes the website to record every click that the user performs on products, purchases done by the customer and the additional items that were visited by the user [6]. These collected data are exposed to Web Mining techniques to extract important data for the validation agent [7].

Overview of Web Mining

Web mining is the process of extracting information from World Wide Web through the conventional practices of the data mining. There are three types of Web Mining approaches as Web Structure mining, Web Content mining and Web Usage mining. On the process of Validation Agent (Web Mining) and Customer behavior analyzer development. We are using the Web Usage Mining approach [7] [5].

What is Web Usage Mining?

In the web usage mining process as described on the Figure 1, data mining techniques are applied on the collected data from web analyzer by analyzing the trends and patterns of users performed on the e commerce website [7].

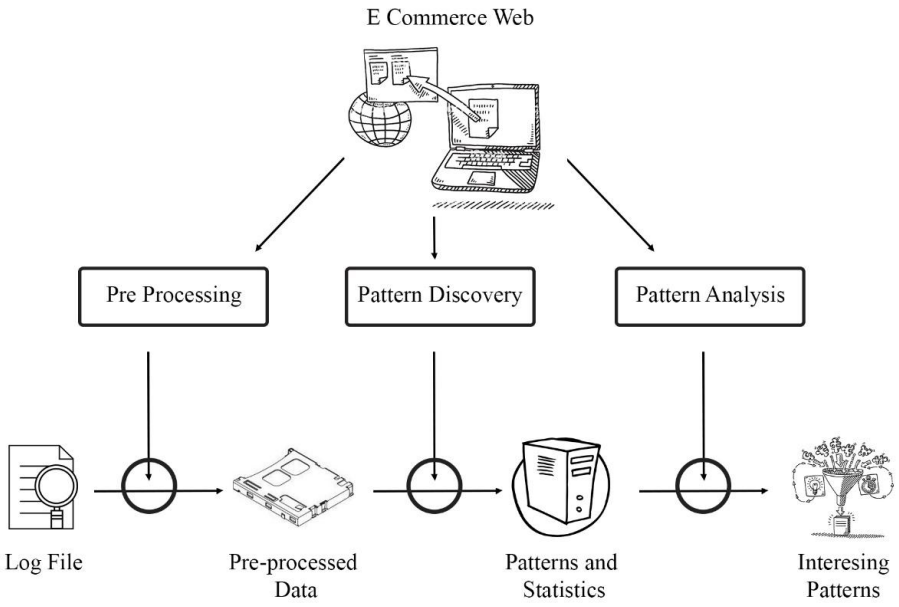


Figure 15 : Web Usage Mining

The web mining process undergoes three main process as Pre-Processing, Pattern discovery and Pattern analysis [6].

Pre-Processing

The data on the log file are not directly analyzable. The Log file contains unwanted data that must be cleaned. During the preprocessing step. These unwanted data are identified through algorithms and they were removed to minimize the log file obtained.

Pattern Discovery

The minimized log file that contains formatted data are exposed to the step of pattern discovery. Using data mining techniques these data are analyzed to identify useful information for the validation Agent. The patterns are clustered under details like Session Id and User ID identified from the log file.

Pattern Analysis

During Pattern Discovery process, both important information that were received from the Predicting agent and the Pattern discovery are used. By introducing suitable algorithms and exposing the collected data, the validating agent validates the suggestions made by iRecommender.

How does Validation Agent Work?

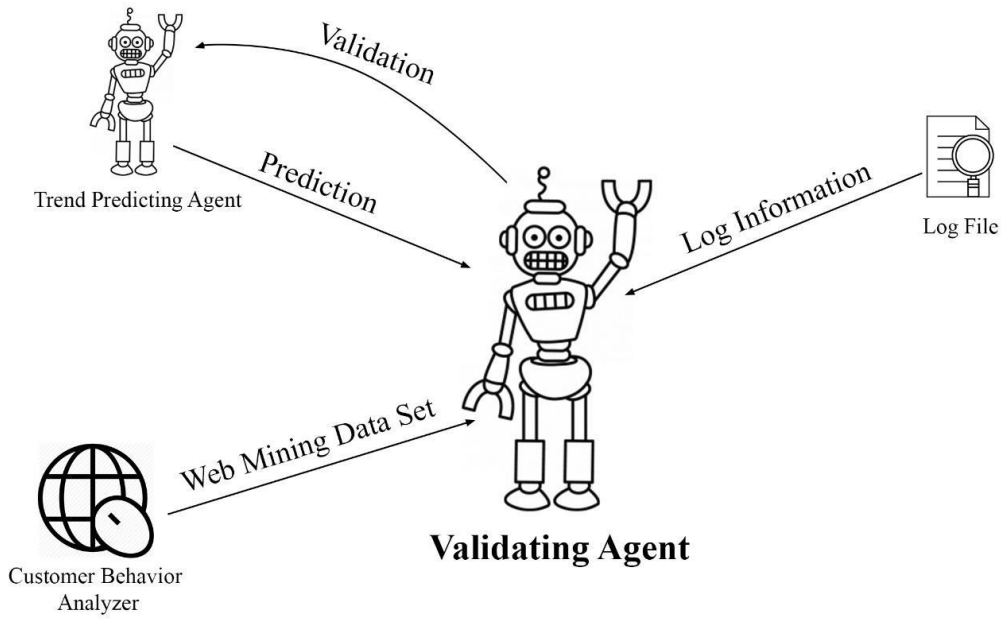


Figure 16 : Validation Agent process

As Figure 2 describes, the validation agent which joins with Customer Behavior Analyzer is responsible on the existence of iRecommender. Inputs for validation agent are taken from the Predicting agent and the customer behavior analyzer. For new users, the prediction is done by analyzing the tweets. Once the prediction is done and the customer is actively using the website, the validation agent Talley's the activities of customer with the predictions done by iRecommender. The validation agent keeps its connection with the predicting agent and provide the information that are necessary to generate the future product suggestions [7,6].

2.4. Implementation

In the design phase we constructed high level architecture, system architecture and work flow of the system. After the design phase we moved to implementation phase according to the agile Software Development Life Cycle (SDLC).

2.4.1 Social media mining and analyzing component

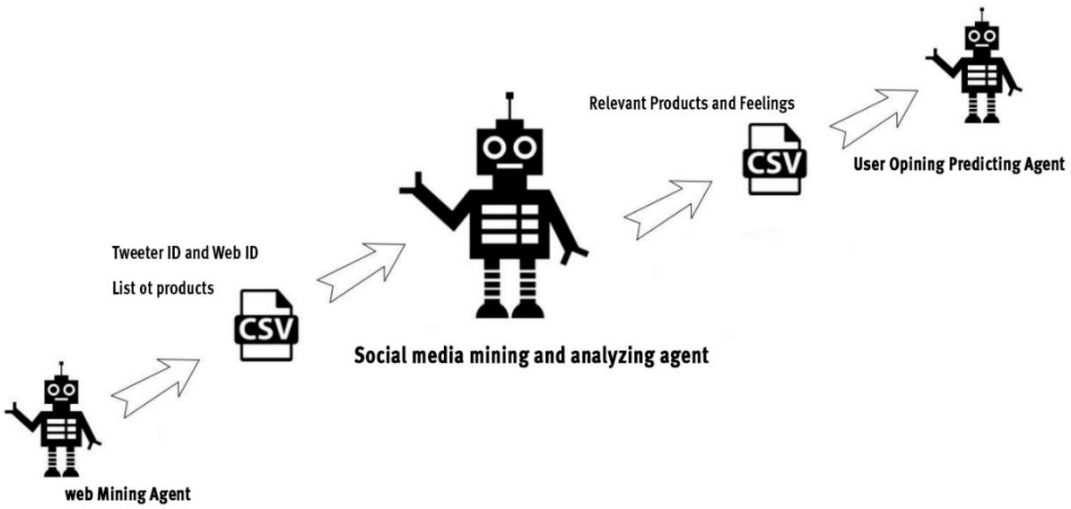


Figure 17Work flow social media mining agent

This component is implemented through the log file. Which is content user's web ID and user's tweeter ID. Component identifying the both IDs and it retrieve data and write data which is retrieved from twitter and write data to file with the name of web ID.

Log file is generated from Web Mining Agent, when the current log users web ID and twitter ID written it. In here we consider only current log users. Because it can be affect to performance of system.

And also social media Mining agent should have to generate products dictionary which can be identify their current products. Web mining agent generate products list file. Web mining agent retrieve their products from e-commerce website's database and write those data to csv file. Then Social Media Mining Agent read the file and generate its own dictionary file.

After social media mining agent process it results put out to the Opining mining agent. As a text file. This text file's named in web ID. Because of opining mining agent should have to identify relevant users result to give accurate output.

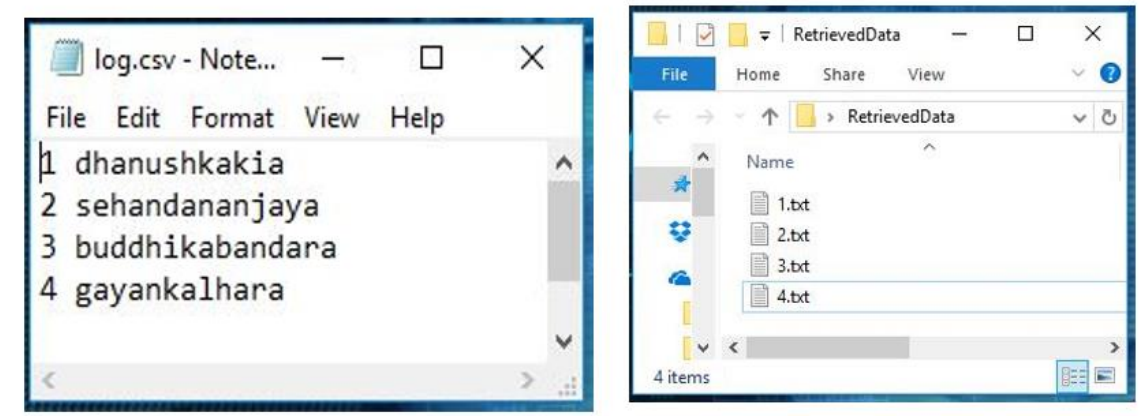


Figure 18 Twitter Id and extracted tweets

2.4.2 Opinion mining and analyzing component

Social Media Mining and Analyzing Agent generate some outputs and these outputs become input to User Opinion Predicting Agent. Social Media Mining and Analyzing Agent generate inputs and these inputs are stored as ".txt" extension files. Input file structure is given below.

```
car nice dont
flower beautiful not
phone damn noanyadjective
vehicle awesome noanyadjective
plain great not
table helpful noanyadjective
```

Figure 19: Input file structure from Social Media Analyzing Agent

These outputs are categorizing according raw wise. In each raw, there 3 phrases delimited by "space". First one is the product which extracted from twits posting by logged user. Second one is the User Feeling/Emotion for the product. Third and last one is the Adjective. Adjective means, sometimes users putting some incremental, detrimental or invert type word phrases like 'don't', 'more', 'not', 'lack of', 'little', 'very'. There are no any adjectives, I used "noanyadjective" key word to identify it clearly.

These inputs are read by User Opinion Predicting Agent and analyzing these and generate the outputs as Negative, Positive or Neutral. These are stored inside a ".csv" file. These became inputs to the Trend Predicting Agent. Output file structure is mention below

car	nice	negative
flower	beautiful	negative
phone	damn	negative
vehicle	awesome	positive
plain	great	negative
table	helpful	positive

Figure 20: Output file structure to Trend Predicting Agent.

2.4.3 Trend prediction component

Trend prediction system can be divided into 4 sections in the workflow diagram.

- Collaborative filtering system
- Content based filtering system
- Recommendations combiner
- Accuracy monitoring dashboard

Collaborative filtering system

Python is our main primary language for this project since it highly supports to machine learning and NLP stuff. The collaborative filtering system is implemented using python language and the collection of apache spark driver programs. In here anaconda package manager used to manage the different python packages.

The goal of collaborative filtering is make recommendations based on user rating and item to item similarity. To measure item to item similarity it is need to have personalized machine learning models. ALS algorithm used to do that task. Using built personalized predictive models it is able to make the recommendations to the users based on item to item similarity. finally, all the predicted data points are pull in to the graph with "suggest "relationship. Check figure 12 for graphical representation of the relationship.

Snapshot of graph database

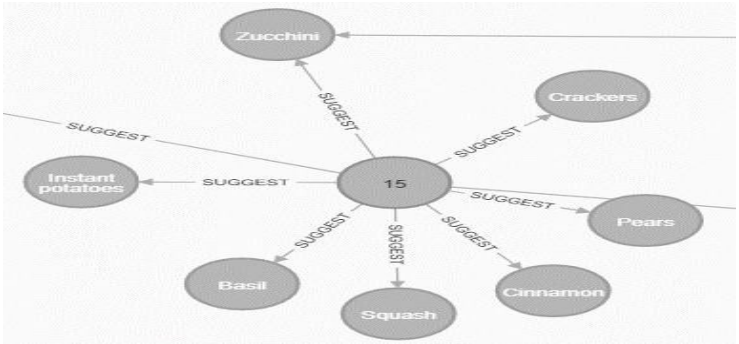


Figure 21 Snapshot of graph database

When building personalized predictive models, model accuracy is the most important factor to make highly accurate recommendations. To ensure the accuracy of predictive models in here used Mean Squared Error. This error should be fall between zero and two. If the error goes beyond two, the collaborative filtering system is going to drop the built model and going to build a new model by increasing the number of model training times. In same time collaborative filtering algorithm sends live data stream to error monitoring dash board.

To calculate the Mean Squared Error and RMSE used following equations and code snippet.

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - p_i)^2}$$

Equation 2 Root Mean Squared Error

Where d_i = r[1][0] ←test data set user ratings

P_i = r[1][1]←Predicted user ratings

```
test_input = testData.map(lambda x:(x[0],x[1]))
pred_test = model.predictAll(test_input)#make the prediction for all the test dataset
test_reorg = testData.map(lambda x:((x[0],x[1]), x[2]))#original data set
pred_reorg = pred_test.map(lambda x:((x[0],x[1]), x[2]))#predicted data set
test_pred = test_reorg.join(pred_reorg)#join both data sets
test_MSE = test_pred.map(lambda r: (r[1][0] - r[1][1])**2).mean()#calculate measn squared error
test_RMSE = sqrt(test_MSE)#get the square root for normanlize
```

Figure 22 code snippet used to calculate RMSE

Sample DataStream of Root Mean Squared Error

Table 3 Sample DataStream of Root Mean Squared Error

Number of predictions made	RMSE
1	0.664
2	0.693
3	1.664
4	0.074
5	0.860

Content based filtering system

Content based filtering system make recommendations based on user opinion data collected through analyzing the user's social media data. In here a live data stream listener listening for the new opinion data sets and when it receives new data sets it is going to analyze using map reduce technique. And after getting map reduce result set

it pulls the highest frequent ten recommendations to the graph database with "Likes" relationship. Check figure 13 for graphical representation of the relationship.

Snapshot of graph data base

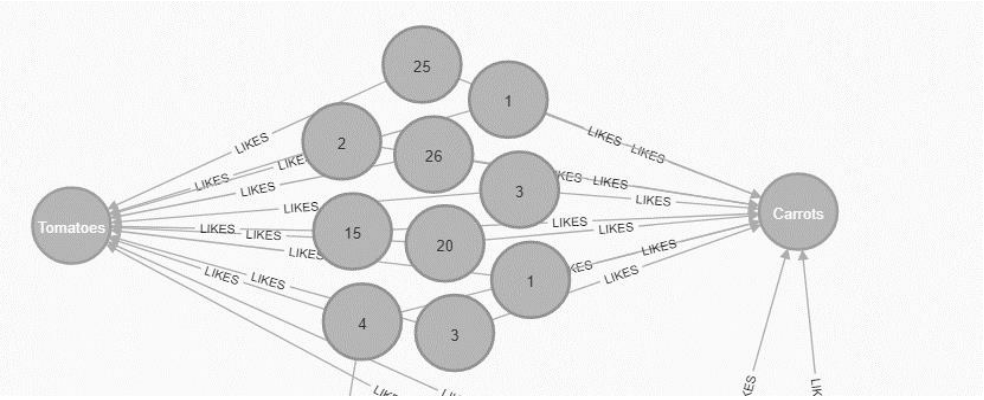


Figure 23 Snapshot of graph data base

Recommendations combiner

By representing all the data points in a multi-dimensional graph, using following method I filter out ten hybrid recommendations for requested user Id. If there is no enough data to make hybrid recommendations that gap will be filled by providing the content based filtering recommendations. And final result will be pass to the e commerce web through flat files.

```
def patternFinder(userID):
    var_user=str(userID)
    hybridTuples=0
    query = "match(n:User)-[:SUGGEST]->(i:Item)<-[:LIKES]-(n:User) WHERE n.name='"+var_user+"' return i.name as Hybrid_item
        limit 10 "
    hybridData = graph.run(query)
    fileName=''+var_user+'Recommendartions.csv'
    with open(fileName, 'wb') as csv_file:
        writer = csv.writer(csv_file, delimiter=',')
        for d in hybridData:
            print(d)
            writer.writerow(d)
            hybridTuples=hybridTuples+1

    if(hybridTuples!=10):
        limitation=10-hybridTuples
        query = "match(i:Item)<-[:LIKES]-(n:User) WHERE n.name='"+var_user+ "' and not ((i:Item)<-[:SUGGEST]-(n:User) )
            return i.name as Liked_item limit "+str(limitation)+""
        likedData = graph.run(query)

        with open(fileName, 'ab') as csv_file:
            writer = csv.writer(csv_file, delimiter=',')
            for p in likedData:
                print(p)
                writer.writerow(p)
```

The final outcome of Trend predicting component is set of product(item) names for given user id of e commerce platform as shown below

```
user Id=10000
(Hybrid_item: iphone6)
(Hybrid_item: samasungG2)
(Hybrid_item: hpLaptopi5)
(Hybrid_item: iMac)
(Hybrid_item: samasungG3)
(Liked_item: Iphone5s)
(Liked_item: iMacbook)
(Liked_item: LenovoS3)
```

Figure 24 final outcome of the trend predicting agent

Accuracy monitoring dashboard

In here live data stream listener captures the MSE data stream send by the collaborative filtering system and feed in to python "matplotlib [17]" library to generate the real-time graph and the graph will be displayed by using python "Tkinter [18]" library. According to the data visualization e commerce site owner can get idea about quality of the data set. If the quality of the dataset provided by e commerce admin MSE goes up. Then he wants to submit another dataset to the Apache spark master node.

Screen capture of graph

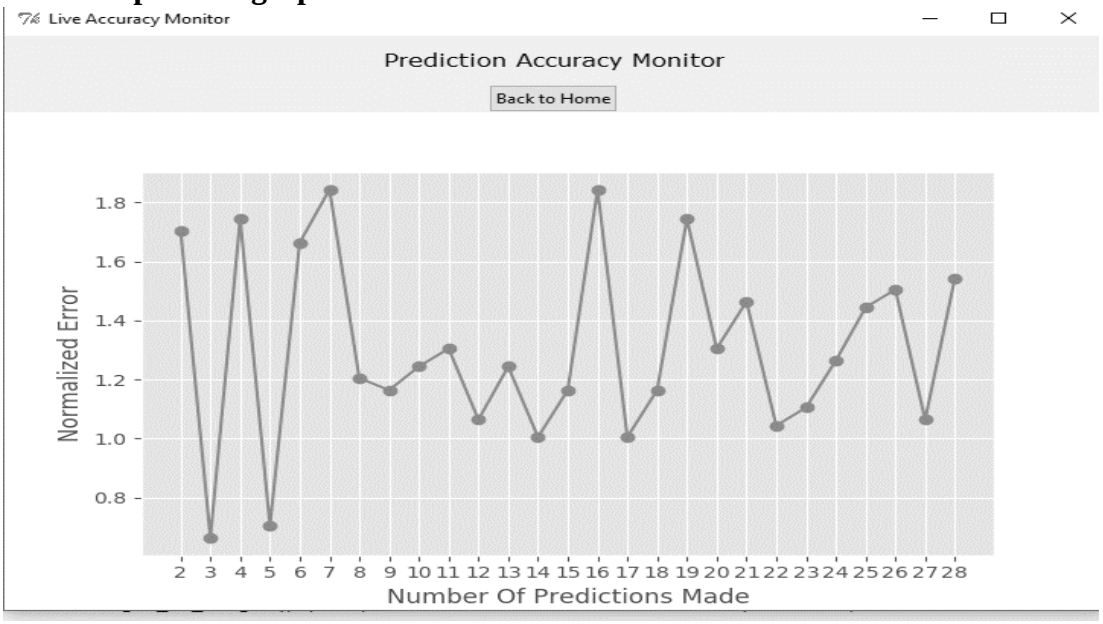


Figure 25 Accuracy Graph demo

Tools, Technologies and libraries used

Tools:

- Anaconda: - for python package management
- I python notebook as cording ground
- JDK-Hadoop runs on top of the jdk

Libraries:

- Py2neo: - bridge between Apace spark and neo4j database
- PySpark:- bridge between python and apache spark
- Matplotlib:- for data visualization
- Tkinter:- for design python GUI graph
- Numpy – for algorithms and array creation

Tecnologies:

- Neo4j: - graph database
- Apache Hadoop: - distributed file system
- Apache Spark:-Distributed computing system
- Cypher :-for query the graph
- Map Reduce: for text analyzation

2.4.4 Customer behavior analyzer agent and Validation Agent

Web mining data collecting process

The process begins with the gathering of data. Once a customer is registered or logged in to the system, the iRecommender Process is starting. The ecommerce website is fully integrated to gather each visit of the customer in a separate .csv file

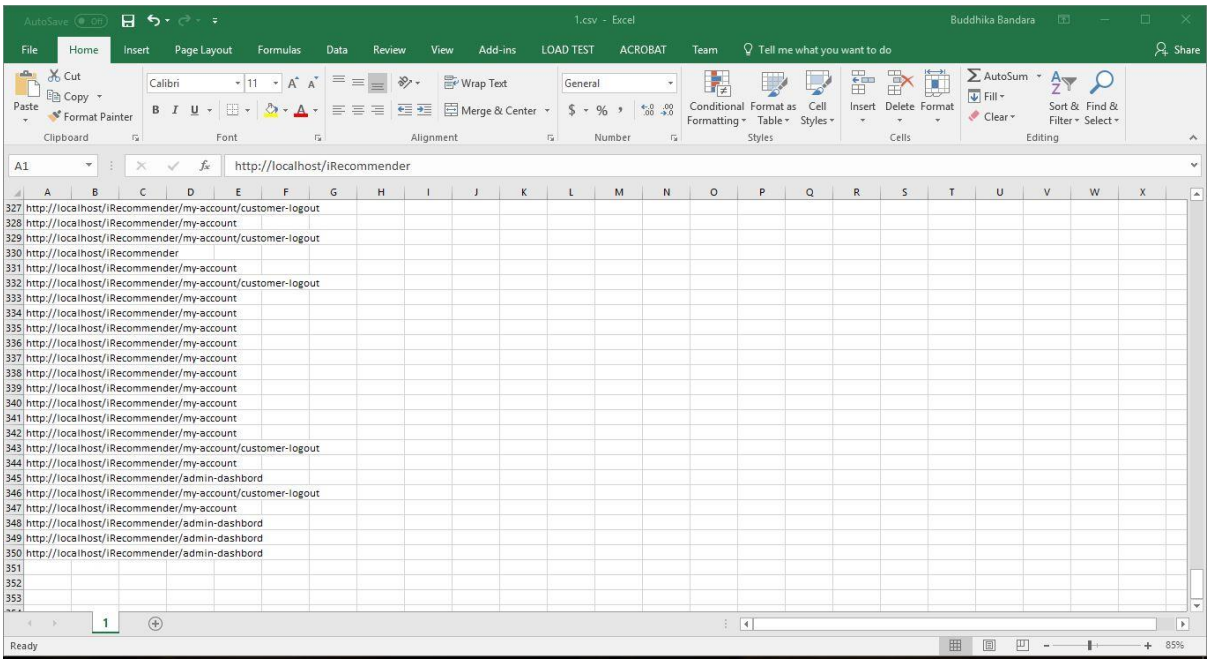
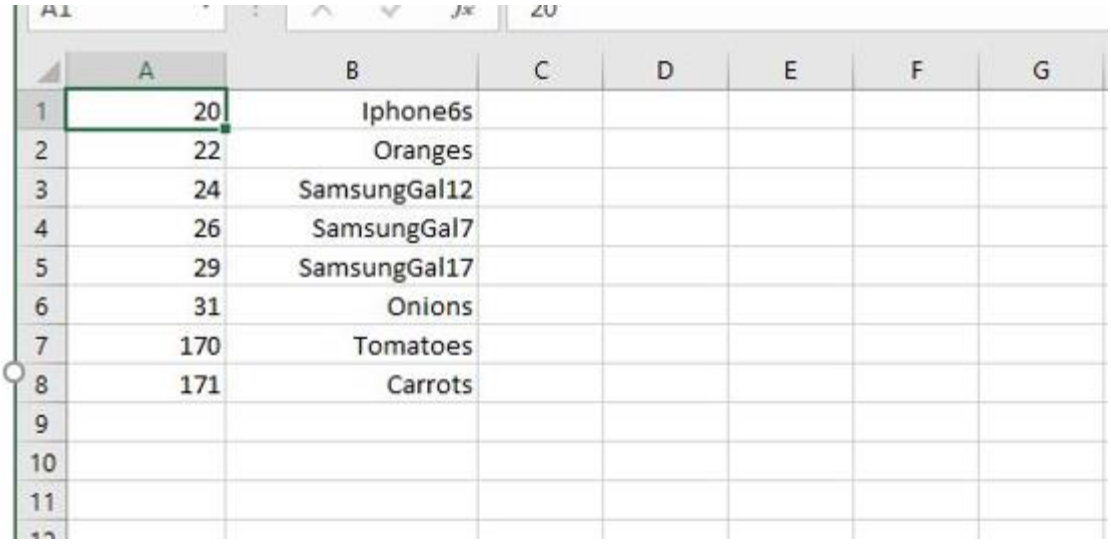


Figure 26 : Customer Log File

As shown in the figure 3. Each click is recorded in a separate row of a ".csv" file which is named with the User ID of the ecommerce website. This renaming helps to identify the logs which belong to each user.

Generating Product List

For several components of the iRecommender system the product list of the ecommerce website is needed. This list is generated to have a boundary defined to make suggestions. The components of the iRecommender system is making suggestions based on this product list. This process makes sure that all the suggestions made are existing in the ecommerce website and varies for other sites.



	A	B	C	D	E	F	G
1	20	Iphone6s					
2	22	Oranges					
3	24	SamsungGal12					
4	26	SamsungGal7					
5	29	SamsungGal17					
6	31	Onions					
7	170	Tomatoes					
8	171	Carrots					
9							
10							
11							

Figure 27 : Product List

As shown in Figure 4 An csv file containing all the available products in the ecommerce website is generated.

Generating Logged User

The generation of logged user triggers the iRecommender process. Once the user is logged, the logged user ID is generated. Then the social media mining and analyzing agent is triggered to analyze the tweets that belong to the logged user. Once the tweet analyzation is completed the User opinion predicting agent triggers its process to gather analyze data of the specific user who logged recently. Then the process is carried out until the suggestion is displayed back to the customer by the validating agent.

Generation Purchase History

Generation of purchase history mainly supports the validation process. Once a user purchased a product, it's possible to use the purchase history in order to validate the suggestion made by iRecommender. This generation process is also a responsibility if customer behavior analyzer. Once the purchase history is generated, the data are exposed to machine learning algorithms to tally with the suggestions made by iRecommender. If the user has purchased a product that is suggested by iRecommender, that suggestion is marked as a valid suggestion.

Updating the iRecommender Suggestions

Updating the suggestions based on the customer behavior is one of the most important process of iRecommender system. The continuity of iRecommender is mainly depend on this process. Once a user has purchased an item, that user is no longer in favor on purchasing the same item. But as the tweets are analyzed, the suggestions give the same product until the tweet count passes the specific products. To overcome this scenario, the suggestions are again validated and updated using the purchases done by the customers. Once the customer purchased an item, iRecommender will keep the item tracked and update the suggestions without displaying the same product again.

2.5. Testing

After the implementation phase we need to test the system against user requirements. Since there are different testing strategies like unit testing, component testing, integration testing, system testing, acceptance testing, performance testing and stress testing etc. we need to design test cases for our units and tester can check whether the all individual units, components and whole system is working fine.

2.5.1 Unit tests for social media mining and analyzing component

Table 4 Test case 01

Test case ID	TC 1
Test case description	Collect tweets for each customer
Input data	Twitter ID Log.csv
Pre – Conditions	Internet connection should be stablished Correct twitter ID should be given
Post Conditions	none
Expected out put	Set of files with Retrieved tweets

Table 5 Test Case 02

Test case ID	TC 2
Test case description	Analyze and remove noise words from collected tweets
Input data	Twitter ID Log.csv Collected tweets
Pre – Conditions	Collected tweets should be in place
Post Conditions	none
Expected out put	Noise filtered word segments

2.5.2 Unit tests for opinion mining and analyzing component

Table 6 Test case 03

Test Case ID	TC3
Test case description	Get output csv file with analyzed data using input text file.
Input data	• Product name • Extracted user feeling / emotion • Adjective
Pre-condition	Input file should be generated according to the logged user.
Post condition	Output csv file should be generated after executing the Algorithm.
Expected output	• Product name • User feeling / emotion • Analyzed Negativity, Positivity or Neutrality according to the user feeling / emotion.

2.5.3 Unit tests for trend prediction component

Table 7 Test case 0 4

Test case ID	TC 4
Test case description	Make collaborative filtering recommendation and update the graph
Input data	• Historical product ratings data set • User id
Pre-conditions	• Spark engine should be up and running • Historical dataset should be submitted to the spark master node • Neo4j database should be up and running
Post condition	none
Expected output	Newly predicted data points should be plotted in the neo4j graph

Table 8 Test case 05

Test case ID	TC 5
Test case description	Make content based filtering recommendation and update the graph
Input data	• Opinion mining data stream • User id
Pre-conditions	• Spark engine should be up and running • Data Stream listener should be up and running • Neo4j database should be up and running
Post condition	none
Expected output	Newly predicted data points should be plotted in the neo4j graph

Table 9 Test case 06

Test case ID	TC 6
Test case description	Make content based filtering recommendation and update the graph
Input data	• Opinion mining data stream • User id
Pre-conditions	• Spark engine should be up and running • Data Stream listener should be up and running • Neo4j database should be up and running
Post condition	none
Expected output	Newly predicted data points should be plotted in the neo4j graph

Table 10 Test case 07

Test case ID	TC7
Test case description	Make hybrid recommendation and results write to a flat file
Input data	User id
Pre-conditions	• Spark engine should be up and running • Neo4j database should be up and running
Post conditions	none
Expected output	Generated flat file should be there with ten recommended items

Table 11 Test case 08

Test case ID	TC8
Test case description	Plot real-time MSE data stream in accuracy monitoring dash board
Input data	MSE data stream
Pre-conditions	• Spark engine should be up and running • MSE data stream listener should be up and running
Post conditions	none
Expected output	Accuracy graph should be displayed in the dash board

2.5.4 Unit tests for web mining and validation component

Table 12 Test case 09

Test Case ID	TC9
Test Case Description	Get User Behavior Log File in to a csv in each user visit on the website.
Input Data	• User Clicks • User Navigated URLs • Users purchase history
Pre-Condition	User must be a registered user and should be logged in.
Post Condition	• Output csv should be generated once the user is logged. • Output csv should append its data while user navigates on the website.
Expected Output	• Each and every URL that the user has visited over the browsing of ecommerce website.

Table 13 test case 10

Test Case ID	TC10
Test Case Description	Get Product List from Website
Pre-Condition	Administrator must be logged in
Post Condition	• Output csv should be generated once admin trigger the process. • Output csv should contain the product list available.

Expected Output	A list of products available in the ecommerce website in to a csv file

Table 14 Test case 11

Test Case ID	TC11
Test Case Description	Get the logged user Id in to a txt file once the user logged in.
Pre-Condition	User must be a registered user and should be logged in.
Post Condition	• Output txt should be generated once the user is logged. • Output csv should contain the logged users web ID.
Expected Output	• An txt file that contains the Logged User web ID.

2.6.Results and Research findings

Since our project has four components with different research objectives research findings and results are described using following four sub captors

2.6.1 Social media mining and analyzing component

<need to complete>

2.6.2 Opinion mining and analyzing component

While developing this User Opinion Predicting Agent, I have to develop an algorithm to extract the Positivity, Negativity or Neutrality. Up to some level, I Could to do that using this developed algorithm.

Without using databases for my component, we can speedup our system and can avoid DB crashes and Recovers. Through reading and writing .txt and .csv files to / from our main directories, it is more manageable and more efficient.

Considering our whole System, iRecommender is targeting the small and medium businesses and this system configuration with any e commerce platform is super easy than other existing recommendation engines. That makes our system more flexible.

Results and Discussion of Opinion mining and analyzing component

User Opinion Predicting Agent of iRecommender System is using an Algorithm to extract the Positivity, Negativity or Neutrality and this User Opinion Predicting Algorithm used NLTK Library to develop this component which utilizing Natural Language Processing Techniques.

Final outcome of this Component is Positivity, Negativity or Neutrality of user feelings / emotions which is very important to the Trend Predicting Agent to do further analyzing. These outputs are given to Trend Predicting Agent through a ".csv" file which named by generated user ID.

In real time, iRecommender tracks who is logged in to the e-commerce site, get his / her most recent 100 twits, analyze these twits and finally gives the product suggestions according to his / her likeliness.

2.6.3 Trend prediction component.

iRecommender has its unique functionality that combines both collaborative and content based filtering in order to do the hybrid recommendation. Even though there are number of recommendation engines available they couldn't be able to combine both collaborative and content based recommendation techniques in scalable and accurate manner. But we did it. and we found that the apache spark engine execution plans should be tuned in manual way in order

to get the best performance and the most valuable thing we found is how to establish the connections between apache spark distributed execution engine and neo4j graph database engine. And the other thing is we used ALS algorithm to make collaborative filtering which make super-fast the system than other traditional systems and make more accurate the results.

iRecommender is targeting the small and medium businesses. and this system configuration with any e commerce platform is super easy than other existing recommendation engines. That makes our system more flexible.

Results & discussion of trend predicting component

iRecommender Trend predicting component uses newest technology available to make the recommendations in high accuracy. Content based filter uses map reduce concept and Collaborative filter uses machine learning with ALS algorithm. And all the executions are done in parallel way in a distributed computing environment. In here to combine both content based and collaborative based filtering I used graph representation method. Which I did as an experiment and that was my individual research problem. In here I have to say my experiment was very successful and the accuracy is high.

The final outcome of Trend predicting component is set of product(item) names for given user id of e commerce platform as shown below

```
user Id=10000
(Hybrid_item: iphone6)
(Hybrid_item: samasungG2)
(Hybrid_item: hpLaptopi5)
(Hybrid_item: iMac)
(Hybrid_item: samasungG3)
(Liked_item: Iphone5s)
(Liked_item: iMacbook)
(Liked_item: LenovoS3)
```

Figure 28 Final output of the trend predicting component

When it talks about execution speed, and the time gap between two recommendations It is totally depends on the Apache spark execution plan and number of worker nodes involved in

the execution and apache spark execution plan should be manually optimized. Following figure shows one iteration cycle execution plan of the iRecommender

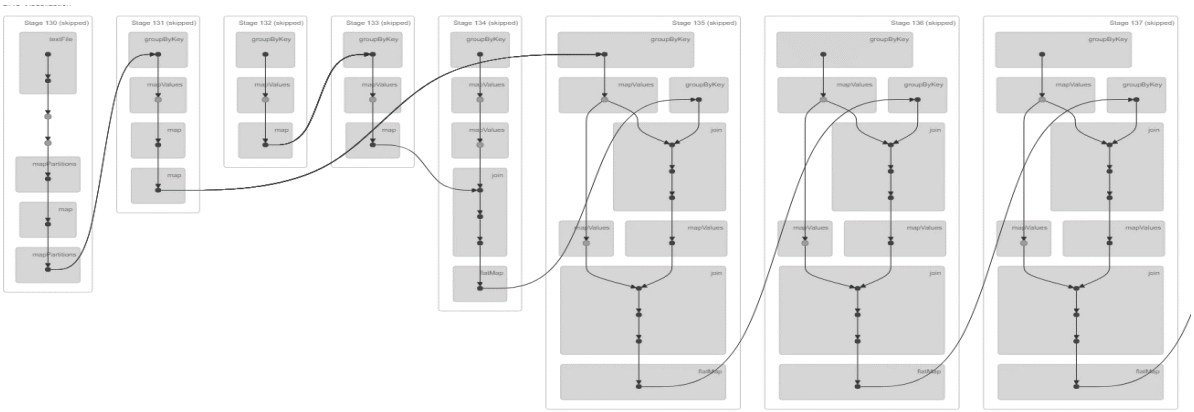


Figure 29Apache Spark execution plan

When it talks about efficiency of predictive model building using ALS algorithm it I observed that with a single node (master node) it takes little bit longer time than the other algorithms. But when it comes to production environment ALS algorithm is the best selection for distributed predictive model building. Following figure show time taken to build some predictive models with ALS algorithm.

first at MatrixFactorizationModel.scala:68	2017/10/04 02:33:10	46 ms	1/1 (16 skipped)
count at ALS.scala:278	2017/10/04 02:33:10	40 ms	1/1 (15 skipped)
count at ALS.scala:277	2017/10/04 02:33:09	1 s	14/14 (3 skipped)
count at ALS.scala:702	2017/10/04 02:33:08	97 ms	2/2 (1 skipped)
count at ALS.scala:694	2017/10/04 02:33:07	0.9 s	3/3
runJob at PythonRDD.scala:441	2017/10/04 02:33:04	1 s	1/1
count at <ipython-input-2-420526096dca>:122	2017/10/04 02:33:01	2 s	1/1

Figure 30 Efficiency of the ALS algorithm

The next important part which I developed is live accuracy monitoring graph. Which shows mean squared error. Basically this function is developed for e commerce admin.

Admin can get idea about the historical data set quality by viewing this live graph. If MSE goes up that tells the historical dataset quality is poor at that times admin needs to submit a another dataset to the apache spark cluster. Following figure shows the snapshot of the graph.

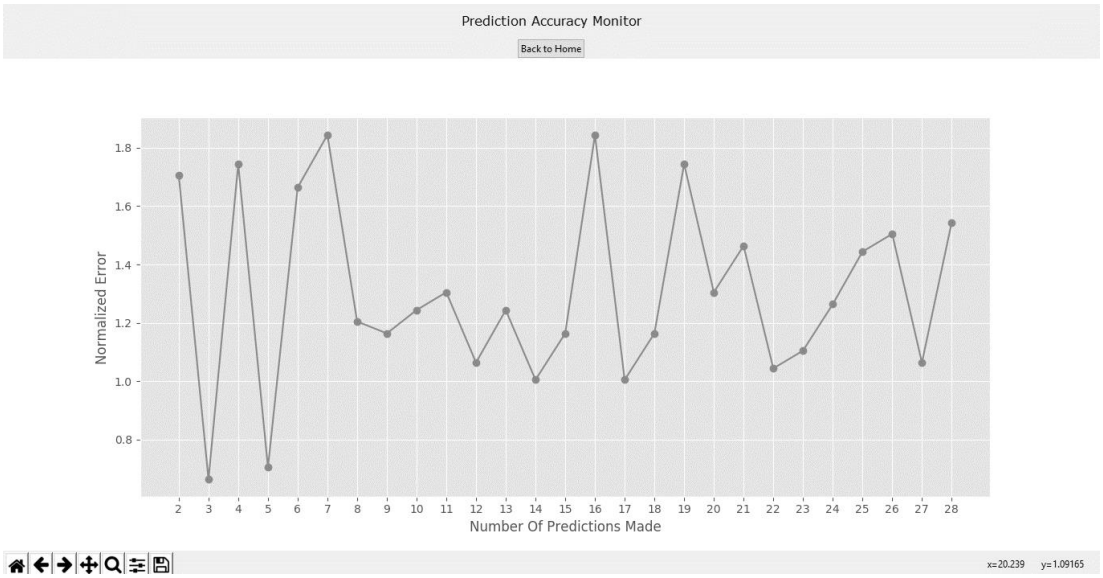


Figure 31 Accuracy monitoring dashboard

2.6.4 Web mining and validation component.

Customer Behavior Analyzing Agent

In the process of developing the Customer behavior analyzing agent, it is revealed that there are more factors that can be considered in a product suggestion system than in a common system in use. The customer behavior analyzing agent is developed to fulfill the task of gathering information from the ecommerce users to support the main system of iRecommender process. Rather than using purchase history of a customer for his future suggestions, Customer behavior analyzing agent gathers user browsing data on the ecommerce website and analyze them to obtain the flavor of the customer. The analyzed data are then used for several proposes inside the system.

These analyzed data are used by several components in the system. Each product that were visited by users are specifically used to make the future suggestions. Once the social media analyzing agent and the trend predicting agent complete there processes these data makes the system eligible to filter the products that belongs to the specific e commerce website where

the user is performing. If not, there will be no limitation for the suggestions and the system will be unable to capture what products should be suggested in respective to the specific ecommerce website

Validating Agent

In the process of developing the Validating agent, it is revealed that there is no validation is taking place in content based suggestion systems or collaborative suggestion systems. This leads to a major drawback of wasting customer's valuable time by making inappropriate product suggestions. With the implementation of validating agent, it is revealed that the information gathered by the customer behavior analyzer can be effectively used in order to validate and update the product suggestions made by iRecommender.

The analyzer gathers information about the purchased products. In the process of validation, this dataset is taken in to consideration. There for the system has gain the eligibility of updating the suggestions. Once a product is purchased, the user will no longer need suggestions for the same products. But the social media data are producing the same suggestions until the user's posts pass the time considered. So, using customer behavior analyzer's purchase history data the suggestion list is validated on each time the user logged in to the system. This validation process is directly connected with the process of trend predicting agent. With the recent tweet analyzation of the user, the suggestions made by the Trend Predicting agent is getting updated. So, the updated products are again taken in to consideration in the validation process. This will lead the iRecommender process to an infinite loop and the product suggestions are made with high accuracy.

Results and Discussion

Customer behavior analyzer of the iRecommender system is using Web mining techniques and machine learning techniques in order to extract and analyze the inputs for the validation process by examining the user behavior in the ecommerce website. The final outcome of the Customer Behavior analyzer is set of products that the user is interested in and another set of products that the user has purchased.

Validating agent of the iRecommender system is using Machine learning algorithms to validate the products suggested by the iRecommender with the user's actions on the ecommerce website. The final outcome of the Validating agent is an updated list of products suggestions for each specific user.

During the process if web mining and machine learning when developing the Customer behavior analyzer, the analyzer gathers log information from each user that logs in to the website. This log information contains lots if data that are filtered and removed in the analyzing process. The decision had to be taken in choosing the storing method of the gathered data and analyzed data. There were two options available in storing user log information.

- Storing data in a database
- Using text based storing procedure

In the development process, the customer behavior analyzer is tested under both available options. Using a database to store and analyze the log information leads the system to take a longer processing time while analyzing and it also heavily increase the usage of the system processor due to the high count of reads and writes that occur in storing log information in each customer activity. But when compared to the database, using a text based procedure is much faster and the system can save the unwanted traffic caused on the processor. The final decision was using a text based procedure in storing user log details.

Once the text based procedure is selected, security issue has to be addressed because a separate file for each user is created inside the server. The privacy of each user must be protected and the website must make sure that the customers can trust the process of website. In order to achieve the security, iRecommender uses the security precautions offered by the hosting service provider. Each and every detail collected and used by iRecommender is fully secured and protected inside the server of the ecommerce website.

3. SUMMARY OF EACH STUDENT'S CONTRIBUTION

M.S.D Dharmawardhana	• Developed Opinion mining and analyzing component.
W.W.G.B.P Bandara	• Developed Web mining and validation component.
A.G.D Ugayanga	• Developed Social media mining and analyzing component.
K.M.S Bandarage	• Developed Trend prediction component.

4. CONCLUSION

As an undergraduate research team, iRecommender recommendation engine is developed with new capabilities. Successful completion of iRecommender project improved our knowledge in areas of Data Mining, Web Mining, Machine Learning, Distributed Computing, NoSQL Graph Databases, Natural Language Processing (NLP) and Social Media Analyzing. Unlike other recommendation engines iRecommender is a low-cost product with flexibility and high accuracy. The main target of this product is to support medium and small e commerce platforms. The system iRecommender has successfully achieved the challenge of combining both collaborative filtering and content based filtering approaches to make hybrid recommendations using a graph based technique.

In SDLC, agile development method is used since mistakes done in previous phases can be corrected in the latter phases. As iRecommender consists with four major components and the output of each component becomes an input to another, the outputs and the inputs were pre-defined in beginning of the project. The pre-defined outputs and the inputs were the milestones that should be achieved by each component of the iRecommender system. Each and every milestone that were assigned to each component was successfully met at the end of the development phase and finally the four major components were implemented together by finalizing the main system of iRecommender.

As the development team, the iRecommender team expects that the research iRecommender and its findings and the results will be an aid and provide guidance on future researches and implementation of any real-time recommendation system who is willing to achieve the performance, accuracy and reliability level that can be expected with the combination of tools, technologies and techniques considered in this research.

5. References

[1] W. pedia, "Recommender_system," wikipedia, 26 02 2017. [Online]. Available: https://en.wikipedia.org/wiki/Recommender_system. [Accessed 2017 02 2017].

[2] I. Analytics, "cold-start-problem," Infinite Analytics, 18 01 2015. [Online]. Available: http://infiniteanalytics.com/classical-cold-start-problem/. [Accessed 17 02 2017].

[3] Y.-Y. Shih, "Hybrid Recommendation Approaches," *IEEE*, vol. V1, pp. 1530-1605, 2005.

[4] S. A. A. Hridoy, "Localized twitter opinion mining using sentiment analysis," 22 Oct 2015. [Online]. Available: https://decisionanalyticsjournal.springeropen.com/articles/10.1186/s40165-015-0016-4.. [Accessed 20 Feb 2017].

[5] P. Lops, M. d. Gemmis and G. Semeraro, "Content-based Recommender Systems: State of," *Springer Science+Business Media, LLC 2011* , p. 33, 2011.

[6] R. Sharma, D. Gopalani and Y. Meena, "Collaborative filtering-based recommender system: Approaches and research challenges," *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad*, pp. 1-6, 2017.

[7] park.apache.org, "park.apache.org," park.apache.org, 16 03 2014. [Online]. Available: https://spark.apache.org/docs/2.1.0/mllib-collaborative-filtering.html. [Accessed 28 03 2017].

[8] "five-signs-to-give-up-relational-database," www.neo4j.com, 27 07 2015. [Online]. Available: https://neo4j.com/blog/five-signs-to-give-up-relational-database/. [Accessed 18 03 2017].

[9] C. Gulcehre, 03 01 2006. [Online]. Available: http://deeplearning.net/. [Accessed 20 03 2017].

[10] Weihong, He and Yi, "An E-commerce recommender system based on content-based filtering," *Wuhan University Journal of Natural Sciences*, pp. 1091-1096, 2006.

[11] J. Jędrzejowicz, J. Neumann, P. Synowczyk and M. Zakrzewska, "Applying Map-Reduce to imbalanced data classification," *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Gdynia, Poland*, pp. 29-33, 2017.

[12] G. K. J. K. a. J. R. B. Sarwar, "Analysis of Recommendation Algorithms for E-Commerce," in *ACM (Minneapolis, Minnesota.)*, 2000.

[13] B. W. Chen, W. S. Rho and Y. Gu, "Supervised Collaborative Filtering Based on Ridge Alternating Least Squares and Iterative Projection Pursuit," *in IEEE*, vol. 5, pp. 6600-6607, 2017.

[14] S. Raschka, "Predictive modeling, supervised machine learning, and pattern classification," 25 Aug 2014 . [Online]. Available: http://sebastianraschka.com/Articles/2014_intro_supervised_learning.html. [Accessed 16 Feb 2017].

[15] Y. Liang and P. Zhao, "Similarity Search in Graph Databases: A Multi-Layered Indexing Approach," *2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA*, pp. 783-794, 2017.

[16] J. Wei, J. He, K. Chen, Y. Zhou and Z. Tang, "Collaborative Filtering and Deep Learning Based," *DASC-PICom-DataCom-CyberSciTec.2016*, p. 4, 2016.

[17] https://matplotlib.org/, "https://matplotlib.org/," https://matplotlib.org/, [Online]. Available: https://matplotlib.org/. [Accessed 17 06 2017].

[18] https://wiki.python.org/moin/TkInter, "TkInter," [Online]. Available: https://wiki.python.org/moin/TkInter. [Accessed 02 06 2017].

[19] J. Yuan, W. Shalaby, M. Korayem, D. Lin, K. AlJadda and J. Luo, "Solving Cold-Start Problem in Large-scale Recommendation Engines:," *2016 IEEE*, p. 10, 2016.

[20] V. Govindasamy and H. Balaji, "Social opinion mining and concise rendition," *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pp. 641-645, 2016.

[21] A. Goel, J. Gautam and S. Kumar, "sentiment analysis of tweets using Naive Bayes," *2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun*, pp. 257-261, 2016.

[22] B. Liu, Sentiment Analysis and Subjectivity, Chicago: N. Indurkhya and F. J. Damerau, 2010.

[23] A. K. Pandey and D. S. Rajpoot, "Resolving Cold Start problem in recommendation system using demographic approach," *2016 International Conference on Signal Processing and Communication (ICSC), Noida*, pp. 213-218, 2016.

[24] S. Liu, Y. Dong and J. Chai, "Research of personalized news recommendation system based on hybrid collaborative filtering algorithm," *016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu*, pp. 865-869, 2016.

[25] F. Ren and Y. Wu, "Predicting User-Topic Opinions in Twitter with Social and Topical Context," *in IEEE Transactions on Affective Computing*, Vols. vol. 4, no. 4, pp. 412-424, Oct.-Dec. 2013.

[26] A. M. Sharif and V. V. Raghavan, "Link prediction based hybrid recommendation system using user-page preference graphs," *2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK*, pp. 1147-1154, 2017.

[27] X. Su, T. M. Khoshgoftaar and R. Greiner, "Imputed Neighborhood Based Collaborative Filtering," *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, NSW*, pp. 633-639, 2008.

[28] H. Cho and M. S. Yoon, "Improving sentiment classification through distinct word selection," *017 10th International Conference on Human System Interactions (HSI), Ulsan, South Korea*, pp. 202-205, 2017.

[29] A. Wijayanto and E. Winarko, "Implementation of multi-criteria collaborative filtering on cluster using Apache Spark," *016 2nd International Conference on Science and Technology-Computer (ICST), Yogyakarta*, pp. 177-181, 2016.

[30] V. M. a. D. N. L. J. Grace, "ANALYSIS OF WEB LOGS AND WEB USER IN WEB MINING," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 03, no. 1, p. 12, January 2011,.

[31] Y. Dou, H. Yang and X. Deng, "A Survey of Collaborative Filtering Algorithms for Social Recommender Systems," *2016 12th International Conference on Semantics, Knowledge and Grids (SKG), Beijing*, pp. 40-46, 2016.

[32] S. Mandal and G. S, "A Lexicon-based text classification model to analyse and predict sentiments from online reviews," *016 International Conference on Computer, Electrical & Communication Engineering (ICCECE), Kolkata, India*, pp. 1-7, 2016.

[33] T. Kawabe, Y. Yamamoto, S. Tsuruta and R. Knauf, "A dictionary-based sentiment classification method considering subject-predicate relation," *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest*, pp. 004217-004222, 2016.

Appendix A: Functional requirements – Trend prediction component

The functional requirements define the main functionality of the component

1.FR1

Use case fields	**Use case description**
Use case 1:-	Read unstructured dataset from csv files
Preconditions	1.System should run without any failures. 2.CSV files with user opinions should be there in the reading directory.(written by the opinion predictor)
Primary actor	Trend predicting agent
Goal	Load the data set in to RDDs
Main success scenario	1.Find out correct CSV file using CSV file ID 2.Read the CSV file 3.Store the read data in to RDD
Extensions	If the required CSV file not found wait until it appears

2.FR2

Use case fields	**Use case description**

Use case 2:-	Data cluster in to Similar product wise
Preconditions	1.System should run without any failures. 2.System should have loaded the data in to the RDDs
Primary actor	Trend predicting agent
Goal	Categorize (cluster) the loaded product names in to similar groups
Main success scenario	1.Read the data from RDD 2.categorize the product in to similar groups 3.write new results to new RDDs
Extensions	If the required data set(RDD) is empty, wait for the data appears

3.FR3

Use case fields	**Use case description**
Use case 3:-	Data cluster in to Positivity Negativity wise
Preconditions	1.System should run without any failures. 2.System should have loaded the data in to the RDDs
Primary actor	Trend predicting agent
Goal	Categorize (cluster) the loaded product names in to similar groups in terms of positivity and negativity wise
Main success scenario	1.Read the data from RDDs 2.categorize the product in to similar groups in terms of positivity and negativity wise 3.write new results to new RDDs
Extensions	If the required data set(RDD) is empty, wait for the data appears

4.FR4

Use case fields	**Use case description**
Use case 4:-	Find 10 products with highest positivity frequency
Preconditions	1.System should run without any failures.

	2.System should have loaded the data in to the RDDs
Primary actor	Trend predicting agent
Goal	Predict the initial suggestions for the customers of the e commerce
Main success scenario	1.Read the data from RDDs 2.count the products versus positive frequency 3.determine 10 products with highest positive frequency 4.upadae the neo4j database with new recommendations under analyzed user id
Extensions	If the required data set(RDD) is empty, wait for the data appears

5.FR5

Use case fields	**Use case description**
Use case 5:-	Get the Product ratings and web mining data
Preconditions	1.System should run without any failures. 2.e commerce platform should have written the data to MySQL database.
Primary actor	Trend predicting agent
Goal	Load the required data to RDDs
Main success scenario	1.Read the data from MYSQL DB 2.Store the read data to RDD
Extensions	If the required data not available, wait for the data appears(Product Ratings etc.)

6.FR6

Use case fields	**Use case description**
Use case 6:-	Calculate the Product ranking in order to do the collaborative filtering
Preconditions	1.System should run without any failures.

	2.Product ratings and user ids should have loaded in to RDDs
Primary actor	Trend predicting agent
Goal	Find the similar item groups using customers buying behavior
Main success scenario	1.Read the data from RDD 2.Data (product ratings and items with user ids) push to a machine learning algorithm 3.Get the similar items according to buying behavior. 4.similar item groups store in neo4j knowledge base
Extensions	If the required data not available, wait for the data appears in RDDs

7.FR7

Use case fields	**Use case description**
Use case 7:-	Get the 10 similar items for each user as product recommendation
Preconditions	1.System should run without any failures. 2.Neo4j knowledge base should have updated
Primary actor	Trend predicting agent
Goal	Recommend the personalized products for each user
Main success scenario	1.Catch the recommendations requests from Validating agent 2.Read the data from neo4j 3.write product suggestions to CSV file with user id
Extensions	If the required data not available reply null value to recommendation request

Appendix B: Functional requirements – Opinion mining and analyzing component

Use Case 01	User Login
Pre-conditions	User should be registered to the EC web site
Successful end condition	Redirect to the EC web home interface
Actors	User of EC web site
Main success scenario	1. Enter username 2. Enter passwords 3. Click login button
Extensions	1a. If the Username is invalid, system will prompt an error message and user should enter correct username again. 2a. If the Password is invalid, system will prompt an error message and user should enter correct password again.

Use Case 02	User Registration
Pre-conditions	User should have a computer with an internet connection
Successful end condition	Redirect to the EC web registration interface
Actors	User of EC web site
Main success scenario	1. Enter valid details in required field 2. Click login button
Extensions	1a. If one of required details is invalid, system will prompt an error message and user should enter correct details again.

Use Case 03	Analyze feelings/emotions and extract opinion
Pre-conditions	Feelings/emotions should get from the csv file
Successful end condition	Successfully extract the user opinion from feelings/emotions
Actors	User of EC web site
Main success scenario	1. Pass the feelings/emotions to the Algorithm 2. Analyze 3. Return extracted opinion

Appendix C: Detailed high level diagram