IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue, 7, March 2019 E-ISSN: 2347-2693

Hybrid Recommendation Engine with Web Scraping and Sentiment Analysis

N. Sontakke[1*], Sabarinath S[2], S. Sangamnerkar[3], V. Iyer[4]

[1,2,3,4]Dept. of Computer Engineering, Pune Institute of Computer Technology, Savitribai Phule Pune University, Pune, India

Corresponding Author: nsontakke004@gmail.com

Available online at: www.ijcseonline.org

***Abstract*—**In recent years a lot of data has been generated owing to the exponential increase in internet usage. People are overloaded with information, spanning over multiple domains. This helps people obtain knowledge and come to informed decisions. If we consider purchasing a product as a use case, the buyer can visit multiple websites to find strengths and weaknesses of the product as well as the opinions of other purchasers. To make this process easier and faster for the buyer we propose a robust and scalable hybrid recommendation system which is implemented as a combination of Content Based Recommendation System and Collaborative Filtering Techniques. As a test case, this system has been used to help purchase smartphones based on all features and sentiments of previous purchasers. System gets data using a self-learning web crawler that gathers data from a number of relevant websites irrespective of their different structures. The sentiments of users who purchased the same items previously have also been analyzed to aid the current buyers' decisions. In this paper, we have provided detailed survey and comparison of multiple techniques we reviewed before implementing each module.

***Keywords*—**Artificial intelligence, Decision support systems, Knowledge based systems, Hybrid intelligence systems, Data Mining, Web Mining, Supervised Learning, Recommender systems, Content based retrieval, Information retrieval, Clustering algorithms, Classification algorithms, Natural language processing, Sentiment analysis, Recurrent Neural Networks.

I. INTRODUCTION

Recommender systems are important as they cover a problem-rich research area and because of the many practical applications that help users to deal with information over-load and provide personalized recommendations, content, and services to them. One of these practical applications is the process of recommending smart phones to users. Buyers have to scour multiple websites all giving important information regarding different features of a smart phone. Apart from this there are a tremendous amount of available smart phones to go over each one by one.

Providing custom options for users based on certain features of the smart phones such as RAM size, Operating system, price etc.would reduce customer efforts significantly. E-commerce websites currently recommend products based on the user's recently viewed products, these recommendations are not effective in matching the buyer's current needs. This system can be used to make product specific recommendations by feeding it data relevant to the product. This will be beneficial for the seller as well as the consumer. The system proposed in this paper provides custom options for each user and has the potential to gather data from multiple websites, using a web crawler. The purpose of the crawler includes downloading and indexing the web pages for a search engine for faster retrieval. Another purpose of the crawler is fetching data about a particular topic/product. Crawlers collect information such the URL of the website, the meta tag information, the web page content, the hyperlinks in the webpage and the targets leading from those links, the web page name and any other germane information. [1] [2] Theykeep track of already downloaded links to avoid re-downloading of the same web page again. In this study a domain specific web crawler was used for collecting data about phones from flipkart.com. Various methods used in web crawling like the famous page rank algorithm, Naive Bayes classification for web crawling are discussed.

Additional insight can be gained through analyzing previous buyer sentiments, the crawler is used to gather reviews for each product. Sentiment analysis was conducted on product reviews using Natural Language Processing (NLP) techniques, the Bag of words (BoW) model, Word2Vec model and Long Short Term Memory (LSTM) neural network. [3]

The recommendation system uses various information filtering, textual analysis and more recently, meta-heuristic search techniques to comb through large amounts of data and intelligently find items that match the user's requirements.

A popular technique called collaborative filtering [4] used in recommender systems has been explored. User-item interactivity is taken into consideration and filtering is generally done using neighborhood-based approach. The primary assumption with collaborative filtering is that users with similar tastes will be interested in similar items. User/Item profile are created based on user purchase or browsing history and item reviews and ratings. Along with collaborative filtering, content-based filtering is one of the most widely used techniques. The filtering is done mainly by creating item profiles and by matching these items to similar items that a user was interested in the past.To test the system, we used a variation of Content-based filtering called Case based recommendation that does not rely on user profiling

II. RELATED WORK

Related work has been described in Table 1.

III. METHODOLOGY

In this section, the overall architecture of implemented system is explained along with how each module will work together in the system. Each of the individual modules are then explained in detail.

A. System Architecture

The web scraping module gathers features for a specific smart phone along with reviews left by customers from multiple websites and passes the data to a flask server. The data is stored in a MongoDB database. Data is accessed by the recommendation module as well as the natural language processing module for sentiment analysis.

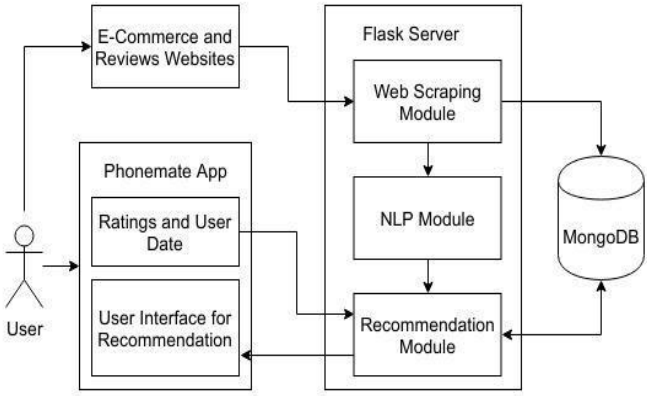

Figure 1. Architecture Diagram

B. Web Crawler Module

This section gives a brief overview of the algorithm used while designing the web crawler and issues we considered during implementation.

1. Challenges that need to be addressed before implementation are:
   - **Network and Bandwidth**: The cost of crawling the web just in terms of network bandwidth requires is more than a 1 million dollars, so it is extremely necessary to utilize the network resources appropriately.
   - **Quality of Service**: The Web crawler aims for a comprehensive coverage of the Web, it should consider that some of the pages not available now but could become available in the future. The time after which a crawler rechecks the web page is dependent on the priority of the web page and the resources available for the crawler.
   - **Missing Relevant pages**: One issue with focused crawlers is that they may miss relevant pages by only crawling pages that are expected to give immediate benefit.
   - **Maintaining Freshness of Data**: HTML pages consist of information that gets updated on daily, weekly or monthly basis. The crawler has to download these pages and updates them into the database to provide up-to-date information to the users.
   - **Relative URLs**: If the links on a web page are to a web page of the same domain then most of the times the hyperlinks are relative. If the crawler fetches such relative URLs then it can lead to invalid pages being accessed so it is necessary to append the relative URL to the base URL that is given in the head section of the web page.
   - **Duplicate Detection**: It is possible that the links that have already been visited are crawled again causing an unnecessary waste of resources. The solution was to maintain a list of links that have been already crawled.
   - **Spider traps**: Web crawlers are also called web spiders, from which the name is derived. Spider traps may be created to "catch" spam bots or other crawlers that waste a website's bandwidth. They may also be created unintentionally by calendars that use dynamic pages with links that continually point to the next day or year so the crawler should 'give up' after a certain period of time.

Table 1. Related Work

Sr. No.	Paper Title	Problem Statement	Objective

1.	Smartphone Recommendation System using Web Data Integration Techniques. Phillip Osial, Kalle Kauranen, Emdad Ahmed, Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada	SmartphoneRecommendation System Using Web Data Integration Techniques	This paper talks about creating a smartphone recommendation system using phone data collected through web-scraping and analyses the attributes of a phone such as price, term, PPI, etc. that matters the most to buyers. It suggests the use of weighted average of attributes of an item for content based filtering and the use of TF-IDF technique for when the item descriptions is textual data.
2	Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions Gediminas Adomavicius and Alexander Tuzhilin	Overview of thefield ofrecommendersystems anddescribes the currentgeneration ofrecommendationmethods	The paper discusses at length, the most Common recommendation system technologies and surveys various techniques for recommender systems.
3	Charu C. Aggarwal. Recommender Systems, The Textbook. Springer International Publishing Switzerland, (2016).	Recommender Systems, The Textbook	A Complete Exhaustive Reference for all types of Recommender Systems

2. Algorithms used to construct the crawler:
   - **Page Rank Algorithm**: The PageRank algorithm out-puts a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. It can be calculated for collections of documents of any size. The PageRank computations require several iterations, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

   - **Naive Bayes**: A naive bayes classifier was used to classify the links according to relevancy. It is mostly used in topic-specific crawlers. The naive bayes classifier includes the three following modules. [5] [6]

   i. **Page analysis module**: Analyzes the downloaded page content in order to extract information to decide on a particular link to follow. Done by checking up HTML content to analyze HTML label tree, deleting useless words and extracting remainder, and transforming the URL into canonical form.

   ii. **Character extension**: Characteristic extraction module is responsible for describing the extracted content of pages. Mathematical parameters are used to calculate the priority of URLs that are not visited by classifier.
   The contents of a link context are represented as an n-dimensional vector, which corresponds to each element of a vocabulary word. Link context is the content adjacent to a link in the page and hints the theme of the target page. As an extreme case, the entire page can be viewed as link context of all hyperlinks in it. For each topic, all words, from the positive and negative samples phrases, are put into the vocabulary (T). Each of positive and negative samples expressed by a vector, and each element of a vector corresponds to a word in vocabulary (T).Therefore, a page p is represented as a vector $X_p = w_{1p}, w_{2p}, ..., w_{np}$. Here '$w_{1p}$' denotes the weight of the first word in page (p), and 'n' is the size of the vocabulary

   If 'w_{sp}' is used to denote the weight of word (s) in page (p). A modified TF-IDF algorithm was proposed to calculate the weight of each word, and the formula is as follows:

$$w_{sp} = \underbrace{\left(0.5 + \frac{0.5 \cdot tf_{sp}}{max_{s' \in Tp} tf_{s'p}}\right)}_{TF} \cdot \underbrace{ln\left(\frac{|E|}{df_s}\right)}_{IDF} \quad (1)$$

   iii. **Relevance analysis**: To analyse how relevant the page is to the specific topic we use the bayesian formula, the posterior probability Pr(class=+1—x) calculated by the following formula:

$$Pr(class = +1|x) = \frac{\prod_{i=1}^{n} P(x_i|class=+1)Pr(class=+1)}{P(x)} \quad (2)$$

Assumption of Naive Bayesian classifier is that the property of data x is independent of its distribution. Data (x) is in term of x_1, x_2, ..., x_n

C. Recommendation Engine

The recommendation system is a hybrid approach with Collaborative Filtering and Content Based Filtering methods.

1. Challenges addressed while designing the system:
   - **Cold start problem**: Majority of intelligent algorithms such as Bee Colony Optimization requires some past data about the purchase history of the user for that particular type of item. This is referred to as the cold startproblem, i.e. the system cannot start working and generate any results about users it has not yet stored sufficient information about.
   - **Overspecialization**: Content-based recommenders will always map the user to items that carry similar attributes to each other. This implies that items with vastly different attributes will never be mapped to each other. The significance of this is that the items being recommended may become too specialized and the system may enter a cycle while trying to suggest similar items.

2. Collaborative Filtering System Components:
   - **User Feedback**: In order to obtain accurate recom-mendation via Collaborative Filtering, it is very important to have data regarding user-item relationships to compare two user/item profiles to recommend new items to the target user. This is done in two ways, explicit feedback and implicit feedback.
     i. Implicit feedback involves recorded user-item interactions, like item viewing and purchasing. This is less reliable in nature as interactions can only be interpreted as positive. For example, a user may be disappointed with a purchase, or may have visited a link because of a click bait title.
     ii. Explicit feedback including ratings and reviews that the user may give a certain item. They are an accurate representation of user sentiment and thus, more reliable. However, its biggest disadvantage is the cold start problem.
   - **Data Exploration**: Data has been scraped by us from one of India's largest e-commerce retailers Flipkart. The data for each phone is well structured which makes it easy to search for items with given attribute value in ourdatabase.Sentiment analysis is being applied on each of the reviews and a general pattern of whether thephone has received overall positive reviews or negative reviews is being determined.
   - **Data Preprocessing**: In this step, the data collected from the website is processed so that it can be used for efficient searching of phones by querying.

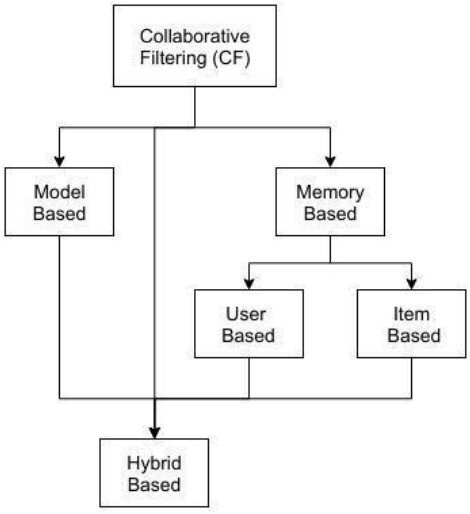

Figure 2. Types of Collaborative Filtering

3. Techniques used for Collaborative Filtering:
   - **Model Based Collaborative Filtering**: Model based methods are predictive models which use machine learning methods. This approach has two stages: a training phase, at the end of which a predictive model is generated, and a prediction phase. This is done using Naive Bayes Classifier.
   - **Memory Based Collaborative Filtering:** Also known as Neighbourhood Based Collaborative Filtering, it generates recommendations by analysing user-item interactions after comparison with its neighbourhood. This can be done using the K Nearest Neighbour Classifier. [7] [8] These neighbourhoods are of two types:
     i. User Based: The assumption with this neighbourhood is that like-minded users will prefer similar items. The basic idea is to determine users, who are similar to the target user A, and recommend ratings for the unobserved ratings of A by computing

weighted averages of the ratings of this peer group.

ii. Item Based: This involves creating a neighbourhood for items similar to a certain target item, based on ratings and reviews given to them by users. In order to make the rating predictions for target item B by user A, the first step is to determine a set (S) of items that are most similar to target item B. The ratings in item set (S), which are specified by A, are used to predict whether the user A will like item B.

- **Hybrid Model**: The two systems mentioned work best in different scenarios. Model Based handles sparsity and scalability much better than Memory Based, however it needs to train a new model frequently which is a very time-consuming process. Also, Memory Based are much easier to implement and can accommodate new data easily. Thus, a combination of the two approaches can overcome the shortcoming of faced with either of them.

In a weighted hybrid system, the score of a recommended item can be computed from the available results based on availablerecommendation approaches. On the other hand, switching hybrid system uses some standards to switch among recommendation approaches.

4. Algorithms - Collaborative Filtering:
   - **Naive Bayes Classifier:** A Simple Bayesian Classifier can be used in the collaborative filtering problem space. [9] Despite its simplicity, it is shown to be competitive with other complex approaches. In our model, other users correspond to features and the matrix entries correspond to feature values. To determine the most likely class of the target item, the following formula is calculated:

$$Class = \underset{class_j \{Like, Dislike\}}{\arg\max} p(class_j | U_1 = Like, U_3 = Dislike, \cdots, U_n = Like)$$

$$= \underset{class_j \{Like, Dislike\}}{\arg\max} p(class_j) \prod_i p(U_i = class_k | class_j) \quad (3)$$

   - **K Nearest Neighbours**: The basic idea in neighborhood based approach is to use either user-user similarity or item-item similarity to make recommendations from a ratings matrix. The concept of a neighborhood implies that we need to determine either similar users or similar items in order to make predictions. One of the easiest ways to construct a k-NN graph is to calculate the similarities between all of the nodes and extract the most similar nodes for each node. [10]

To calculate similarity metrics, we use Pearson Correlation. When using Pearson Correlation, the similarity is represented on a scale of -1 to +1 where a positive high correlation, a negative value indicates inversely high correlation and zero correlation indicates uncorrelated samples.

- **User Based Pearson Correlation**

$$s(u,v) = \frac{\sum_{iI_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{iI_{uv}} (r_{ui} - \overline{r}_u)^2 \sum_{iI_{uv}} (r_{vi} - \overline{r}_v)^2}} \quad (4)$$

s(u,v) represents the similarity of items i and j.
I_{uv}: set of items rated by both u and v.
r_{ui} and r_{vi}: rating of u and v for item i.
$\bar{r}_u$ and $\bar{r}_v$: mean rating of user u, v over all items

- **Item based Pearson Correlation**

$$s(i,j) = \frac{\sum_{u U_{ij}} (r_{ui} - \overline{r}_i)(r_{ui} - \overline{r}_j)}{\sqrt{\sum_{u U_{ij}} (r_{ui} - \overline{r}_i)^2 \sum_{u U_{ij}} (r_{ui} - \overline{r}_j)^2}} \quad (5)$$

s(u,v) represents the similarity of items i and j.
U_{ij}: set of common users who rated i and j.
r_{ui} and r_{vi}: rating of u and v for item i.
$\bar{r}_i$ and $\bar{r}_j$: mean rating of item i, jacross all users

5. Content Based Filtering:
   In a content based recommendation system, we initially identify a fixed set of at-tributes/features for our set of items. The attributes/features that are present in that item are marked. There are two ways to do this:
   - Create a Boolean vector of attributes for each item and mark the index true if the attribute is present, else mark it false.
   - Create an integer/floating point/string vector of attributes for each item and store the value that the attribute holds for that particular item.

In our smart phone recommendation system, we need to use the second approach as the value of attributes such as price, primary camera, primary memory, RAM, etc. matter while comparing two phones. The vector of attributes represents our item profiles.

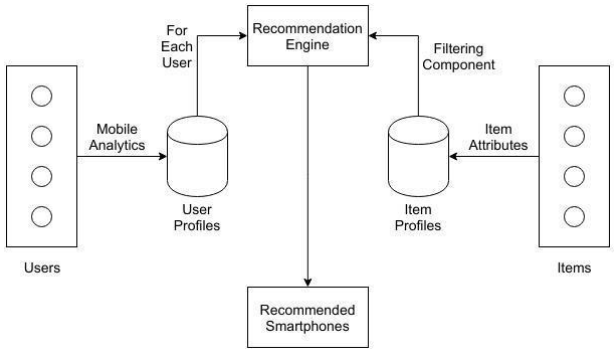

Figure 3. System Diagram for Content Based Filtering

In our smart phone recommendation system, we need to use the second approach as the value of attributes such as price, primary camera, primary memory, RAM, etc. matter while comparing two phones. The vector of attributes represents our item profiles.

Further, a user profile is created for each user that uses our recommendation system. The user profile stores meta data about the preferences of the user. This can be represented again as a vector of attributes/features based on the items that the user likes/dislikes. The existence of a profile for a user is essential for any content-based recommender engine to recommend an item to the user. Finally, the recommendation engine compares the user profile and the item profile to evaluate if the item would be useful for the user according to his preferences or not.

Finally, the recommendation engine compares the user profile and the item profile to evaluate if the item would be useful for the user according to his preferences or not. The recommendation engine is basically an algorithm such as a simple query-based filtering technique or smarter algorithms such as Bee Colony Optimization, Multivariate networks-based similarity matching or even a neural network.

- **Case Based Recommendation System:** A database is created of the set of items based on a set of relevant features (e.g., price, primary camera, RAM). Querying of items is done using a set of constraints that are explicitly taken from the user through a questionnaire. This method does not capture the user data implicitly through analytics but it can be modified to do so. It is mostly preferred in the initial stages when user profile is unavailable. [11] (See Figure 4)
- **Artificial Bee Colony Optimization:** Artificial Bee Colony Optimization is an optimization algorithm that models the behavior of real bee colonies in an evolutionary computing model. [12] Bee colonies have three kinds of bees:
  i. Worker Bees
  ii. Onlooker Bees
  iii. Scout Bees

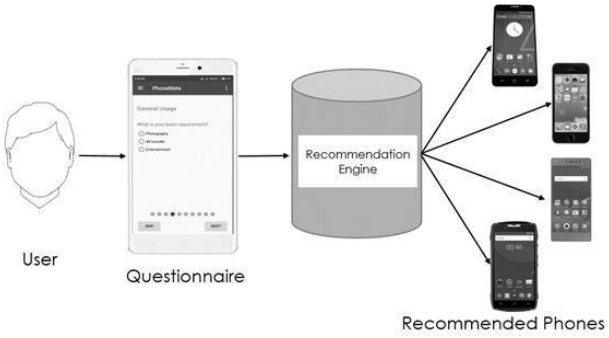

Figure 4. Case Based Recommendation System

The colonies are divided into two halves, where the employed bees are taken to be half of the colony while the onlooker bees are the second half. The number of worker bees is equal to the number of available food sources. The worker bees that visit the previously unknown food sources are the scout bees.

The worker bees collect the nectar from the food sources and play a waggle dance on returning to the hive. This dance communicates to the onlooker bees the location of the food source and the amount of nectar that it contains. The onlooker bees choose a subset of these food sources and search around these food sources to find better food sources. After evaluating the location for better food sources, the onlooker bees return to the hive.

Simultaneously, the scout bees target the unexplored food sources. The search for these unexplored food sources starts by exploring for a random unvisited real nectar source. They collect the nectar from these sources and return to the hive. In the recommender system, artificial bee colony optimization is mainly used in the filtering component.

D. Sentiment Analysis

Sentiment analysis is useful while determining positive or negative responses of the public towards a particular product. [13] [14] In this section we cover the preprocessing steps required to clean the data gathered. The sentiment analysis model was constructed and implemented using the Bag of words (BoW) model, Word2Vec model and Long Short Term Memory (LSTM) neural network.

Text classification involves two main steps. We find a word embedding to convert text into numerical representations and then we fit the representations of text to machine learning or deep learning architectures.

1. **Word embeddings:** A common approach of word embedding is frequency based embedding such as BoW model. It learns a vocabulary list from a given corpus and represents each document based on some counting method of words. (CountVectorizer in sklearn to compute occurrence counting of words)
   For words appearing frequently with little meaningful information about the sentiment so instead of occurrence counting we use TfidfVectorizer to understand impact of these words.
   Another approach of word embedding is prediction based embedding. Word2Vec is a combination of Continuous Bag of Words (CBoW) and skip-gram model. Both shallow neural networks which learn weights for word vector representations.

2. **Fitting data to algorithms:** Once we have numerical representations of text data, we are ready to fit the feature vectors to supervised learning algorithms, such as Multinomial Naive Bayes, Logistic regression and Random Forest.
   LSTM's are useful in text mining problems since it involves dependencies in the sentences that can be captured in the memory. Simple LSTM has embedding layer as input layer, LSTM layer as hidden analyser and a dense layer as output layer.
   LSTM with Word2Vec embedding will give more accuracy by taking into account semantic similarity of words.

3. **Algorithms and techniques:**
   - **Data Preprocessing**: Flipkart data has ratings from one to five and the data with positive sentiment is from rating 4 or 5 and data with negative sentiment is rating 1 or 2. Data with rating 3 is ignored as it's neutral and not useful.
     It involves removing html tags, conversion to lowercase, removal of non characters and removal of stopwords, punctuation and unnecessary ordering information. Alphabetized words are converted to lowercase and misspelled words are replaced. Data is converted to 1D vector or 2D tensor.
   - **Creating Vocabularies:**
     i. **CountVectorizer** provides a simple way to tokenize a collection of text documents, build a vocabulary of known words encode new documents using that vocabulary. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document because these vectors will contain a lot of zeros, we call them sparse. The encoded vectors can then be used directly with a machine learning algorithm.

     ii. **TF-IDF Vectorizer:**An alternative to calculating word frequencies is term frequency - inverse document.Term Frequency: This summarizes how often a given word appears within a document.Inverse Document Frequency: This downscales words that appear a lot across documents.TF-IDF are word frequency scores that try to highlight words that are more interesting, e.g. frequent in a document but not across documents. The TfidfVectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.TF-ID vectorizer performs better than the count vectorizer a detailed overview is given in the results section.

     iii. **Bag of Words Model:** A bag of words model is a simple and flexible way of extracting features from text, to use in models in machine learning. It is only concerned with the text present not the ordering of data. BoW can be implemented using TF-IDF and CountVectorizer, we will compare the results. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves a vocabulary of known words and measuring presence of known words. [15]

     iv. **Word-2-Vec Model:** Word2Vec attempts to understand meaning and semantic relationships among words. It works in a way similar to deep approaches, such as recurrent neural nets or deep neural nets, but is computationally more efficient.Word2Vec does not need labels in order to create meaningful representations. This is useful, since most data in the real world is unlabeled. Words with similar meanings appear in clusters, spaced such that some word relationships, e.g.: analogies, can be reproduced using vector math. The famous example is that, with highly trained word vectors, "king - man + woman = queen." The module uses the python gensimpackage to provide an excellent Word2Vec implementation. [16]

v. **Simple LSTM:** Deep learning for sentiment analysis LSTMs make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies the previous cell state, previous hidden state, input at the current time step. LSTMs do not manipulate the entire information but rather modify it, they are able to forget and remember things selectively. So, the basic process is to prepare data to 2D tensor in keras and train a simple LSTM (Layers: Embedding, LSTM and dense) [17]

vi. **Final Module**: As per the detailed analysis presented in the resultssection. After comparing all the methods by using them on gathered data the module finally uses a LSTM trained with Word-2-Vec feature extractor which in turn will be trained using TF-IDF vectorizer vocabulary.

IV. RESULTS AND DISCUSSION

A. Web Crawling and Scraping

1. **Extraction of phone features:** Data of more than 1600 phones was extracted by crawling flipkart.com. The seed URL was the link to the page that appears after searching "Best phones" on flipkart.com. The data consisted of 140+ phone features, some of the features being specific to certain phone. The data consisted of about 650+ smartphones and remaining feature phones. The data extracted was stored in a csv file. Requests, Pandas, BeautifulSoup, Threading etc. python modules were used.

2. **Extraction of reviews:** Extraction of comments was done through a different method. Selenium web driver was used. Approximately 90 comments per phone can be fetched for more than 500 phones.

B. Sentiment Analysis Metrics

1. **Evaluation Metrics:**
   a. Accuracy Score: Finds the fraction of correct predictions and it's easier to implement on a binary classification problem.
   b. Confusion Matrix: Matrix summarizing true positives, true negatives, false positives, false negatives.
   c. Comparison: The following table compares the evaluation metrics of the different techniques used.

Table 2. Comparison of results obtained

Sr. No.	Model	Trained on one percent of data	Trained on ten percent of data
1.	Bag of Words: CountVectorizer + Multinomial Naive Bayes	88.35	91.34
2.	Bag of Words: TfidVectorizer + Logistic regression	89.32	93.10
3.	Word2Vec Embedding + Random Forest	84.79	93.10
4.	Simple LSTM	90.61	94.14
5.	LSTM + Word2Vec	88.35	94.40

C. Recommendation Engine

An android application alongside a REST API in Flask were developed. The smartphone data was collected from Flipkart using web scraping techniques. The data available on Flipkart was already structured with every phone having over 40 attributes. However, the data was not completely ideal for our needs. Preliminary data cleaning activities as mentioned in this seminar had to be performed. Further, critical attribute values such as the RAM or Operating System are missing in the dataset. This implies that these phones cannot be compared and hence may not be suggested to the user. Due to a constraint of time and processing power, we selected the following 7 attributes for filtering using content based filtering:

i. Operating System
ii. Price
iii. RAM Capacity
iv. Secondary Memory
v. Primary Camera Quality
vi. Display Screen Size
vii. Weight

Due to the lack of any kind of previous user analytics data, we implemented Case Based Filtering by taking recommendations from the user every time a user requests for smartphone recommendations. The use of memory-based methods will yield much better results. In order to make use of Artificial Bee Colony Optimization for recommendation, two changes need to be made to this implementation:

i. Smartphone item profiles need to be created specified in this seminar using the value-based vector representation

ii. User profiles need to be generated and each user profile must contain textual metadata about the user's likes and dislikes using either implicit feedback such as Mobile analytics or explicit feedback using questionnaires, ratings or textual feedback.

The use of MongoDB as a database was chosen because of the convenience in representing unstructured and unclean data. Further, its strong querying API makes development time shorter and the task of generating our Flask API easier.

D. Application User Interface

Figures 5, 6, 7 are screenshots from the android application.

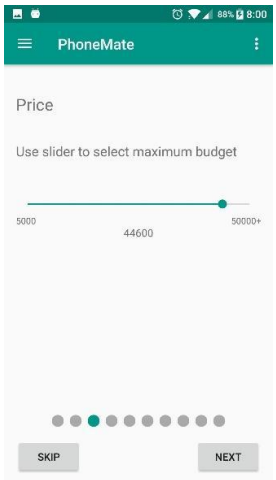

Figure 5. App Screenshot: Questionnaire

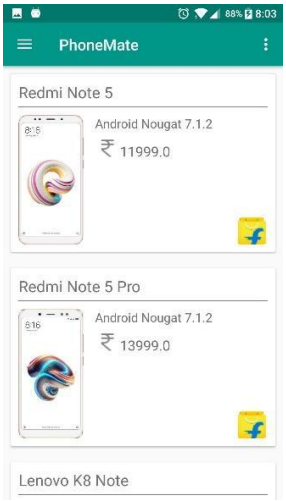

Figure 6. App Screenshot: Recommendations

Figure 7. App Screenshot: Device Details Screen

V. CONCLUSION AND FUTURE SCOPE

Our main aim with the system was to explore and create a recommendation system that could aid the user in making an informed decision and the use case chosen to test the system was smart phone recommendation. The current system is designed around recommendations for smart phones, however, it can be easily scaled for other products available in E Commerce websites.

A domain specific web crawler was implemented to collect data of smart phones along with their customer feedback. Up votes down votes, title and the context of the review have been fetched. Future work will include implementing a crawler with intelligence which would be more useful and generic.

To provide some additional knowledge and perspective on the product, sentiment analysis was done on the gathered reviews with the help of an LSTM model trained on word-2-vec and TF-IDF chosen after detailed comparison.

As we can see from the results, the recommendationof smart phones according to the user requirement is done satisfactorily. The category of basic usage in the questionnaire has been added to ease the process of collecting user preferences from the user. By selecting any option other than "Custom", the user's recommendation will be chosen automatically by the system.

Although results are satisfactory there is scope for improvement in this problem:

i. The use of case based recommendation can be replaced by a memory based technique such as Artificial Bee Colony Optimization or Multi-attribute Network technique.

ii. The dataset can be improved so that some of the features/attributes of a smartphone are not blank.

iii. Data of additional phones can be collected from otherwebsites such as Amazon and GSMArena.

iv. Creation of user profiles using Mobile data analytics can be done so as to make use of any memory-based technique.

ACKNOWLEDGMENT

The authors would like to thank Dr. S. S. Sonawane, Associate Professor, Pune Institute of Computer Technology, for her guidance and motivation

REFERENCES

[1] P. Srinivasan,F. Menczer, G. Pant"*A General Evaluation Framework for Topical Crawlers*", **2004**.

[2] Ziyan Zhou and Muntasir Mashuq, "*Web Content Extraction Through Machine Learning*", Stanford University.

[3] Amit Gupte, Sourabh Joshi, Pratik Gadgul, Akshay Kadam, "*Comparative Study of Classification Algorithms used in Sentiment Analysis*", **2014**.

[4] Yun Y., Hooshyar D., Jo J., Lim H., "*Developing a hybrid collaborative filtering recommendation system with opinion mining on purchase review.*", **2017**.

[5] Wenxian Wang, Xingshu Chen, Yongbin Zou, "*A Focused Crawler Based on Naive Bayes Classifier*", Third International Symposium on Intelligent Information Technology and Security Informatics, **2010**.

[6] Duygu Taylan, Mitat Poyraz, Selim Akyokuş, Murat Can Ganiz, "*Intelligent Focused Crawler: Learning which Links to Crawl*", IEEE, **2011**.

[7] Youngki Park, Sungchan Park, Sang-goo Lee, "*Fast Collaborative Filtering with a k- Nearest Neighbor Graph*", Journal of Machine Learning Research, **2014.**

[8] Claudio Adrian Levinas., "*An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals*", **2014**.

[9] Koji Miyahara†, Michael J. Pazzani, "*Improvement of Collaborative Filtering with the Simple Bayesian Classifier*", **2013.**

[10] J. Herlocker, J. A. Konstan, J. Riedl,"*An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms*", Information Retrieval, Vol 5, Issue 4, **2002**.

[11] Smyth B, "*Case-Based Recommendation*", The Adaptive Web. Lecture Notes in Computer Science, Vol. 4321**,** Springer, Berlin, **2007**.

[12] Dima S. Mahmoud, Robert I. John, "*Enhanced Content-based Filtering Algorithm using Artificial Bee Colony Optimisation*", SAI Intelligent Systems Conference, London, **2015.**

[13] Shivaprasad T K, J. Shetty, "*Sentiment Analysis of Product Reviews:A Review*", **2017.**

[14] Suiki Li, "*Sentiment analysis on Amazon Unlocked mobile phones*", **2017.**

[15] Doaa Mohey, El-Din, "*Enhancement Bag-of-Words Model for Solving the Challenges of Sentiment Analysis*", **2016.**

[16] Xin Rong, "*word2vec Parameter Learning Explained*", **2016.**

[17] H. Palangi, Li Deng, Y. Shen, J. Gao, Xiaod, "*Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval*", **2016.**

APPENDIX

A. *Abbreviations*

1D/2D	One/Two Dimensional
API	Application Program Interface
BoW	Bag of Words
CBoW	Continuous Bag of Words
CSV	Comma Separated Values
HTML	Hyper Text Markup Language
kNN	k Nearest Neighbours
LSTM	Long Short Term Memory neural network
NLP	Natural Language Processing
RAM	Random Access Memory
REST	REpresentational State Transfer
TF-IDF	Term Frequency Inverse Document Frequency
URL	Uniform Resource Locator

Authors Profile

Ms. Neha Sontakke is pursuing Bachelor of Engineering in Computer Science from the Pune Institute of Computer Technology since 2015. Her current projects include work in the Natural Language and Image Processing domains. She recently co-authored and presented a paper on Personal Information Identification with Natural Language Processing for GDPR compliance, at the IEEE International Cloud Computing Conference (2018), in collaboration with IBM. She hopes to continue working in the research field and is interested in Artificial Intelligence and it's applications in various domains from operating systems to game development.

*Mr.Sabarinath S.*is pursuing Bachelor of Engineering in Computer Science from the Pune Institute of Computer Technology since 2015. He was a member of IEEE PICT Pune Chapter.His projects include Land Ownership Management using Blockchain,Using BLE Beacons for indoor naviagtion, Augmented Reality based Social Network and various other Web and Mobility based solutions. His interests lie in Artificial Intelligence, Game Development, AR, VR and Blockchain.

Mr.ShubhamSangamnerkar is a final year undergraduate student at Pune institute of Computer Technology, majoring in Computer Engineering since 2015. He has keen interest in Algorithms, Discrete Optimization, Complexity Theory, Machine Learning, etc. His current projects include land deed security using blockchain, AI based framework for auto play of games.

Mr. Vishva Iyer is pursuing Bachelor of Engineering in Computer Science from the Pune Institute of Computer Technology since 2015. He was a member of the IEEE PICT Pune Chapter. He has done a number of projects in web-based systems as well as mobile applications. He is highly motivated about machine learning, neural networks and data analysis and is currently pursuing research projects in the same.

