Journal of Universal Computer Science, vol. 25, no. 10 (2019), 1353-1372
submitted: 21/9/18, accepted: 28/4/19, appeared: 28/10/19 © J.UCS

An Intelligent Data Analytics based Model Driven Recommendation System

Bushra Ramzan
(Department of Computer Science & IT, The Islamia University of Bahawalpur
Bahawalpur, 63100, Pakistan
bushraabaajawa@hotmail.com)

Imran Sarwar Bajwa
(Department of Computer Science & IT, The Islamia University of Bahawalpur
Bahawalpur, 63100, Pakistan
imran.sarwar@iub.edu.pk)

Rafaqut Kazmi
(Department of Computer Science & IT, The Islamia University of Bahawalpur
Bahawalpur, 63100, Pakistan
Rafaqut.kazmi@iub.edu.pk)

Shabana Ramzan
(Department of Computer Science & IT, Govt. Sadiq College Women University
Bahawalpur, 63100, Pakistan
shabana@gscwu.edu.pk)

Abstract: The recommendation systems are getting important due to their significance in decision making, social and economic impact on customers and getting detailed information relevant to a required product or a service. A challenge in getting true recommendations in terms of relevance is the heterogenous nature of data (likes, ratings, reviews, etc.) that a recommendation engine has to cope with. This paper presents an intelligent approach to handle heterogeneous and large-sized data of user reviews and generate true recommendations for the future customers. The proposed approach makes use of Apache Cassandra to efficiently store data (such as customer reviews, feedback of hotel customers) having context properties such as awareness and knowledge of the tourists, personal preferences (such as ratings, likes, etc.) and location of the users. This system consists of three main components: the web front-end, the data storage and the recommendation engine to gain recommendations efficiently. The recommendation engine is relying on Euclidean distance and Collaborative Filtering (CF) to measure similarities in users' review or items' features. Our hotel recommender approach has bifold contribution as it has ability to handle heterogeneous data with the help of big data platform and it also provides accurate and true recommendations.

Keywords: Big data; Hotel recommender, e-Commerce, Customer satisfaction
Categories: J.7, K.3.0, K.3.1, K.8.0

1 Introduction

With the development of new web technologies, the recommender systems (RS) are getting significant attention by the business people as well as customers due to its role in better e-commerce, refined business strategy, improved customer's satisfaction, etc. The success of modern e-commerce systems and online booking and reservations systems heavily relies on the customer's satisfaction and trust. In the recent years, online hotel booking has become one of the primary choices of the hotel customers. A few recommender systems are also developed in the recent past to facilitate the hotel customers to recommend a hotel before he actually makes a booking or a reservation [Liu, et al., 2013]. However, these systems are generic and process only homogenous data, whereas nature of most of the data on web is heterogeneous that is a major bottleneck in the performance of hotel recommendation systems. The heterogeneity of data affects the performance of the RS directly. In this era of competition, complex information causes overload problems which in turn are time consuming and affects the overall performance. Due to the various forms of data (numeric, textual, etc.) in heterogeneous form over the web, the performance of RSs requires more attention towards its improvement. Few recommenders dealing heterogeneous data are now available in market in recent [Zhang, et al. 2012]. Here, heterogenous data mean both qualitative data (text reviews, feedback, etc.) and quantitative data (likes, ratings, stars, etc.). Processing different format of data for data analytics is an open question. Intelligent frameworks are required to process and analyse both type of qualitative and quantitative data. Existing RS frameworks handle one type of data either qualitative or quantitative data. Another challenge is true recommendation by a recommender system to build a customer's trust and improve one's satisfaction level. Here, a true recommendation targets a relevant recommendation with respect to a customer's choices, priorities, budget, etc. Typically, a recommender system banks on the previous information such as customers' reviews of a hotel's services, ratings, etc. [Koren, et al., 2009]. A couple of challenges in achieving true hotel recommendations is processing and analysis of heterogenous web data and intelligent approach that makes recommendations relevant to customer's choice.

Tourism is linked with hotels because tourists always wanted to know about the hotels where they are going to stay in their tour [Mariani, et. al. 2014]. In recent years, booking of hotels through online systems has been increasing rapidly and many websites are working over this domain. A recommendation system (RS) helps customers in effective ways to get information about hotels.

[Hsieh et al. 2017] has discussed that different algorithms and techniques are used in recommender system for considering the aspects of the users' reviews and ratings. R. Burke [2002] has explain that such algorithms are divided into collaborative filtering (CF) and content-based filtering (CBF). The mixer of these algorithms is known as hybrid filtering. In a recommender system with content-based, the user's preferences are presented by their linked points [Lops, et al., 2011]. While, the collaborative filtering is the most extensive recommender technique. Collaborative filtering is a recommended technique [Ekstrand, et al., 2011] and works a class of methods that recommend items to users based on the preferences other users have expressed for those items. It deals with different types of data i.e. hotels, movies, music where the user preferences are changed randomly.

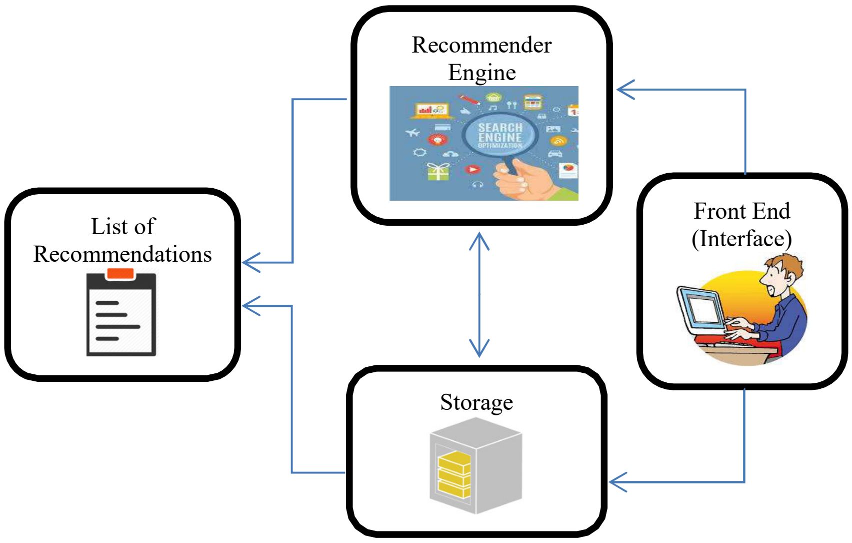


Figure 1: Architecture of a typical Recommender System

The core problem in recommenders is storage and processing of the list of thousands of items like hotels here. To resolve this issue a recommendation system is proposed here which is based on item-based collaborative filtering and Euclidean distance method using NoSQL database such as Apache Cassandra in a fault-tolerant and concurrent manner for the sake of improving the performance and efficiency due to the huge size of data of the hotels and users expeditious reviews [Ibrahim, et al., 2019a]. We have tested the performance gains of using Cassandra in terms of response time of the web application. It also resolves the scalability issue of collaborative filtering Recommenders because it is difficult for a system to generate recommendations in practice if the users and items databases are huge and require massive computation [Ibrahim, et al., 2019b].

The proposed system is suitable to store and retrieve the required information efficiently and compute high speed recommendations within seconds. This paper aimed to make the following main contributions

1. To build a Hotel Recommender based on Item- based Collaborative Filtering and Euclidean Distance algorithm using Cassandra NoSQL database.
2. To optimize performance and overcome data heterogeneity problem in the proposed recommendation system using Cassandra database.
3. To develop an application for hotel recommendation by incorporating hotel information from the linked data of external resource available online.
4. To develop an application with a view of developing hotel recommender by using web services on the base of user’s ratings.

The rest of the paper is organized and structured as follows. Section 2 discusses the related work and Section 3 describes the used methodology. Section 4 provides the details of experiments and results are discussed in Section 5 and paper is concluded in Section 6.

2 Related Work

The initial focus of the paper is to analyze and understand prior research and work done in recommender system concepts such as information filtering, Hotel selection applications, Cassandra as a storage and recommendation service [Kabassi, et al., 2010].

2.1 Recommender Systems

Recommender systems are the systems which help consumers discover items they may like. Fasahte [Fasahte, et al., 2017] discussed that many researchers have conducted several experiments on the Trip advisor dataset. Different filtering technique has been used to predict the unrated items. Hu [Hu, et al., 2016] explains that the data for the analysis containing reviews and ratings is obtained from Trip advisor.

A method called KASR (Keyword Aware Service Recommendation) has been proposed for the big data analysis by Meng [Meng, et al, 2014]. The Hwang [Hwang, et al., 2015] focus on B&BS reviews and hotels to perform a hotel review for the hotel management systems. Hotel reviews were obtained from Tripadvisor.com. In a big data environment, KASR has been implemented in Hadoop and cloud for the sake of improving the efficiency and the scalability in the environment of big data. Ghose [Ghose, et al., 2012] has performed the experiment for this research has done on real-world data sets, these data sets are of 1G, 512 M, 256 M and 128 M. KASR has a significant role in improving the scalability and the accuracy of the service recommender systems. Almeida [Almeida, et al., 2012] has studied the impact of a communication system on reservation system of tourist's hotel is examined.

Lin [Lin, et al., 2015] designed a personalized hotel recommendation by using the text browsing tracking and mining techniques. Rianthong [Rianthong, et al., 2016] developed a useful stochastic programming model to plan the hotels sequence in order to enable the customers to search the hotels at a lower search cost. Sharma [Sharma, et al., 2015] has examined a recommendation system by using a multi-criteria review-based approach based on the user's reviews and preferences. A dataset from a website Booking.com was used for the analysis. Chang [Chang, et al., 2013] hypothesized on the recommendation of the hotel that is based on the surrounding environments. Jannach [Jannach, et al., 2012] presented the recommendation of hotels that is based on the multi-dimensional customer ratings.

A model of the personalized Intelligent Information for the hotel services is examined by the Chawla [Chawala, et al., 2013]. An investigated on the recommendations for improving the news articles through the user clustering is performed [Bouras. et al., 2017]. Word Net-enabled k-means algorithm is used. Chen [Chen, et al., 2018] examined a nonlinear and Fuzzy programming approach for the optimization of the performance of ubiquitous hotel recommendation system. The experiment has been done on hotels. Valcarce [Valcarce, et al., 2015] analyzed the platform of the distributed recommendation for the big data using MySQL Cluster and Cassandra.

2.2 Collaborative Filtering

Collaborative filtering-systems gather user's previous data for the certain items like hotels, books, Articles etc. A hotel recommendation system based on Rankboost algorithm and cluster based collaborative filtering is proposed by Huming [Huming, et al., 2010] to help users to choose a hotel according to their desire. The data for the experiment collected from hoteltravel.com. For the quantitative and qualitative analysis, the 5 points rating scale was used. For selecting the cluster centers there are many indicators for the satisfaction of customer which should be considered. Several researchers have applied collaborative filtering algorithm [Shambour, et al., 2016] for recommendations. Some of them are discussed below. Moreover, in order to calculate the correlation with other users or to do deductions in the feature space both collaborative and content-based filtering depends on knowledge of the user [Cacheda, et al., 2011]. By handling this problem, the recommendation systems can develop users' friendly systems with an easy access in order to the overview information and with no dead-end situations.

3 Materials and Method

3.1 Similarity Measures

In the recommendation process, a similarity measure or the distance measure is the most important factor upon which the decision of the recommendations is made. The accuracy and quality of recommender systems are comprehensively based on similarity metrics used. There are many techniques that help to measure similarity like cosine similarity, Tanimoto Coefficient Similarity, and log likelihood ratio etc. One of the measures is Euclidean distance measure which considers the distance between the feature vectors to approximate similarity between users' ratings. To find the distance between two users (x, y), we have used Euclidean distance as a similarity measurement metric in this paper.

3.1.1 Euclidean Distance

Euclidean distance is the metric for the measurement of similarity and dissimilarity between the objects or items. The reason to choose Euclidean Distance metric over other metrics is due to these advantages.

- The most significant advantage of this method is that it does require less computation and is simple to be implemented.
- Euclidean distance transform is a global operation; it involves the most straightforward approach to its construction.
- Euclidean Distance is the most common use of distance. It refers to the distance between points or objects. It is calculated as the root of square differences between co-ordinates of a pair of objects.
- The Euclidean distance tool is used frequently as a stand-alone tool for applications, such as finding the nearest hotel. Alternatively, this tool can

be used when creating a suitability map, when data representing the distance from a certain object is needed. For example, the distance to each hotel or city is identified. This type of information could be extremely useful for planning a tour or trip.

The distance between X, Y two vectors can de define as:

$$E(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2} \quad (1)$$

The equation (1) represents Euclidean distance metric, which calculates the vector distance between the two objects (hotels), represented by the square root of the sum of squared values. For the measurement of data on the same scale Euclidean distance is the only suitable way. This means that for the determination of the size and numbers it can have, it has its own scale. Therefore, for comparisons Euclidean distance can be used for best measure of similarities. Table 1 shows the details of the similarity computations.

Euclidean	User1	User2	User3	User4	User5
Item1	5.0	4.0	3.0	2.0	5.0
Item 2	3.0	3.0	2.0	-	-
Item 3	2.0	2.0	5.0	-	3.0
Distance	0.000	1.118	3.937	2.500	0.500
Similarity to user1	1.000	0.472	0.203	0.286	0.667

Table 1: Item-Based Similarity Computation

This method is working as users is a point in many items. In the Table I, each user has given a rating to each item. Distance‘d’ between 2 such users is calculated first by using formula 1/ (1+d). When users are similar, distance is zero but distance increases as the similarity is less. Distance’d’ indicates the level of similarity between users as shown in the Table 1. Euclidean distance never gives negative value as a similarity and when the value increases it means that they are more similar. This algorithm compares ratings of the items for one item but not for one user to items. Item-based similarity provides better results instead of user-based similarity which usually affected by the taste of the user or mood of the user so it can be changed over time.

$$E(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2} \quad (2)$$

$$E(x,y) = \sqrt{(x_1 - y)^2 + (x_2 - y_2)^2 y_i)^2} \quad (3)$$

The pseudo code to calculate similarities of hotels in the recommended approach are shown in Algorithm 1 and in Algorithm 2.

Algorithm 1. Similarity calculation using Euclidean distance metric

```
1.  function similarity(recommendation_model, id, r_base)      ›where
    recommendation_model=Array  id=Integer  r_base=Array
    Input: Integer Id , Two Arrays recommendation_model and r_base
    Output:  Similarity  numeric value
2.      if r_base = Id then
3.          if recommendation _model = r_base then
4.              r_base_array =recommendation_model [r_base]
5.          else
6.              return 0
7.      else
8.          r_base_array = r_base
9.      end if
10.     if similarity=0 then
11.         retrun 0
12.     end
13.     for similarity until 0 do
14.         sum_of_squares  =sum_of_squares + (recommendation_model[item]-
    r_base_array[item] )^2
15.         return 1÷ (1 +sqrt of sum_of_squares)
16.     end for
17. end function
```

Algorithm 2. Algorithm for comparison.

```
1.  function cmp (a,b)
    Input: Two Integers a and b
    Output: Comparison of a and b
2.      If a = b then
3.          return 0
4.      else if a > b then
5.          return -1
6.      else a < b
7.          return 1
8.      end if
9.  end function
```

Hotel relations and occurrences are filtered for the recommender system can be calculated on the basis of item co-occurrence weight age. This technique is very useful to find the occurrences of hotels using quantitative values. The co-occurrence can be measure using Jacquard coefficient and Simpson coefficient.

$$Ss(A,B) = \frac{Cr(A \cap B)}{\min(Cr(A)\ OR\ Cr(B)} \tag{4}$$

In the recommender system Simpson coefficient is adopted for item A and B. In the above equation (5) the Ss is the Simpson coefficient, the Cr is the count of hotel rating. The hotel rating in the system is predicted by using weighted sum approach [30]. Each rating is computed with the mean approach of the weighted sum of deviation for item-based collaborative filtering shown in equation (6)

$$\int_{z,n}^{SimCF} = r_z + \frac{\sum_{z=1}^{NN\ S\ imCF} SimCF_{z,n} \times (r_z - r_n)}{\sum_{z=1}^{NN\ SimCF} SimCF_{z,n}} \tag{5}$$

Rating of hotel *z* and *n* mean value is denoted by *rz* and *rn* factor. SimCF x, n is representing of collaborative filtering similarity of hotel ratings. In the CF similarity measurement, the nearest neighbor of a hotel is targeted.

Algorithm 3. Algorithm Recommends hotel with similarity

```
1.  function recommendhotels(recommendation_model, hotel_id)
    Input: Integer hotel_Id, Array recommendation_model
    Output: Recommendation with similarity in numeric value
2.      for recommendation_model =>value do
3.          If(db_hotel_id != hotel_id) then
4.              Sim = similarity (recommendation_model, db_hotel_id, hotel_id)
5.          end
6.          If (sim>0) then
7.              recs [db_hotel_id] = sim
8.          end
9.      end for
10.     Sort(recs, cmp)
11.         return recs
12. end function
```

Algorithm 4. Algorithm to generate Recommendations

```
1.  function getrecommendations (recommendation_model, r_base)
    Input: Two Arrays recommendation_model and r_base
    Output: Rrecommendations along ratings
2.  Sim = 0
3.      If  r_base = numeric then
4.          r_base_array = recommendation_model[r_base]
5.      else
6.          r_base_array = r_base
7.      for recommendation_modal (value)
8.          If id != r_base then
9.              Sim = similarity (recommendation_model, id, r_base_array)
10.         end
11.         If sim>0
12.         for (recommendation_model [id]
13.             If r_base_array != key then
14.                 If total != key then
15.                     total [key] = 0
16.                     total [key] = total [key] + recommendation_model[id][key] ×
    sim
```

```
17.                  end
18.               If sim_sums != key then
19.                     sim_sums[key] = 0
20.               else sim_sums[key] = sim_sums[key] + sim
21.            end for
22.            for total = key(value)
23.                rank [key] = value÷ sim_sums[key]
24.             end for
25.       end for
26.       sort(rank, cmp())
27.           return ranks
28.   End function
```

The step by step flow chart of the proposed system is shown in Figure 2.

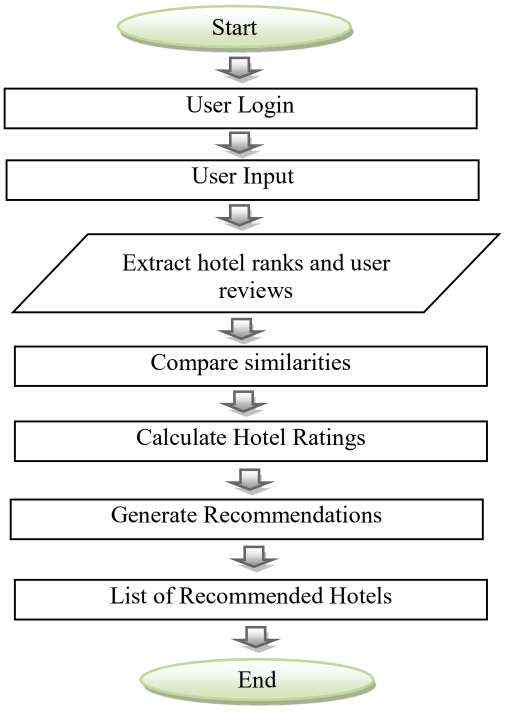


Figure 2: Flowchart of the proposed system

In our proposed application we have used Euclidean distance as a best available similarity measure. The recommendations are obtained from similarity measurement calculations by comparing the hotel rating given by the user with the hotel ratings given by different users already stored in Cassandra. Pseudo code to generate the recommendations is shown in Algorithm 3 and Algorithm 4. The working process of the proposed recommender can be described by the following steps.

- First of all, the active user provides its choice as an input as per given search criteria
- Then the system collects and process the reviews of hotels by previous users stored in the database and compute the average ratings for each
- Then it computes the similarity between the active user and previous users' ratings.
- Then system calculates the personalized rating of each hotel and present a recommendation list of hotels to the active user based on its choice/taste which is the output of the proposed recommender system.

3.2 Implementations details

The proposed system has been tested in the development environment which is based on reliable open source tools such as Cassandra, JSON, PHP, and Java. The computing environment details are given in Table 2.

System Specification	
Hardware	4 Core (TM) i7-3770, 3.40GHz,500GB Disk, 8GB RAM
Software	CentOS 7.0, Apache Cassandra 3.11.1, JDK 1.8/Eclipse, and PHP 7.1
Plug-In	PHP-Cassandra extension

Table 2: Recommender System Computing Environment

3.3 Web Service

The hotel recommender system can be accessed via a web page. First of all, a user can connect through specified URIs and HTTP connections. After that when the user requires any data from the web page, a request is submitted through HTTP method along with the necessary parameters required in a particular method and the requested data is displayed through the web page. The response message of HTTP is converted into the JSON format to keep the uniform data format. The description of used functions along with their query parameters in URIs is shown in Table 3.

The recommender system is accessed using a fixed URI (http://HotelRecommenderSystem/User_request/?method) is used to convert data contents in JSON to deliver the particular information about the hotel. Hotel rating ranges from 1 to 5 scales; however, the application will not display any information on rating 0. The most significant point of interest is that the web services over the recommender are protected by SSL during the data transmissions.

Name	Explanation	Other Query Parameters	Request Method
Search	This method scrutinize data using search criteria (Name/Rating/City/Region to get the list of available hotels data	Searchtitle	GET
getratings	Need previously assigned users ratings before returning the list of recommendations showing percentage.	Ratingsinnum	GET
dispratings	Show the specified hotels with rating	ratingstars	GET
hotel Recom.	provides a list of recommended hotels	Id/name	GET
hoteldetail	Give hotel details, name id and region	Id	GET

Table 3. Description of Methods Applied

3.4 Proposed Recommender System

The architecture of the proposed Recommender system application consists of three main components such as 1) external resources, 2) search parameters, 3) Hotel Recommender App. All the components are elaborated in detail.

3.4.1 External Resources

The available external sources for hotel Reviews are hotel.com, Expedia, TripAdvisor, Booking.com etc. Hotel reviews dataset used in this study is taken from the Hotel website of TripAdvisor. Reviews are divided into different data sets to check the performance of Cassandra and working of our methodology in incremental parts. After the completion of this process, the complete data is stored into Cassandra databases using the developed methodology automatically.

3.4.2 Search Parameters

The recommender system displays the details of all hotels present in the database. Our application displays the list of hotels which are 4333 in total by giving the number of pages below at the main page. When the user clicks any hotel, its complete details are displayed. User can also search any hotel through a keyword search criteria of recommender application. Our recommender searches a hotel using four parameters such as Name/Rating/City/Region and provides Recommendations.

3.4.3 Hotel Recommender Application

In order to get ratings information of the desired hotels the Hotel Recommendation system is designed in PHP and Java code with templates of Smarty. Users can access this web application from any platform by using web Browsers. The Hotel recommender is designed to provide hotel recommendations based on the ratings given by other users which matched the user rating stored in Cassandra belongs to dataset.

Hotel recommender interface enables the user to access hotel on the basis of review rating, location, region or title by setting his/her preferences. The developed application is user friendly and simple to use. It can be accessed with low internet bandwidth as well.

4 Experiments and Results

Apache Cassandra is used in this research because it is a powerful Big Data supporting NoSQL database. The dataset used in this study is taken from the hotel website of trip advisor. The static metadata includes hotel Records and customer profiles which are used in experiments for numerical simulation tests in Cassandra. This dataset consists of 878561 reviews and ratings from 4333 hotels (1.4GB) that are crawled in from Trip Advisor. JSON is used to make it easier to read the data.

4.1 System Overview

In the hotel recommender Application, every search parameter has been interfaced so that users can query according to their choice based on search parameters The application is supportive to any web browser and the user can access it online, it uses the environment of development which is based on reliable open source tools such as Cassandra, JSON, PHP, and Java. The computing environment is shown in Table 4. The application accepts Rank on the basis of ratings/rank stored in NoSQL Cassandra database given by previous users' data contained in used NoSQL dataset, after similarity calculation and filtration, the list of recommended hotels.

4.2 Evaluation Metrics

Once the recommender has been proposed and developed, it is essential to demonstrate the accuracy of our proposition. The system should demonstrate that the recommendations given by the proposed system are acknowledged by the targeted users. By considering the end goal to realize whether the proposed Recommendation application is compelling and effective to resolve scalability by the framework, the three exemplary accuracy measurements tools are utilized as assessment measurements.

Higher F-Score represents better quality of recommendations. Alongside these standard measures, we will moreover utilize an uncommon assessment convention to evaluate the results of the incremental update procedure. We have divided the data into data chunks (of 20%, 40%, etc.) and performed the calculations on these different chunks of the data and computed Precision, Recall and the F-measure. At that point, we incrementally included the ratings of the remaining of the dataset into the training set in steps and every time again made repeated measurements of accuracy measure Precision, Recall, and the F-measure as shown in Table 4.

Data update using Euclidean similarity	F-measure	Recall	Precision
20%	0.818	0.775	0.861
40%	0.788	0.732	0.856
60%-	0.756	0.701	0.821
80%	0.707	0.653	0.773
100%	0.660	0.611	0.721
Avg. ratio	0.745	0.694	0.806

Table 4: Evaluation Metrics

Figure 3 shows the result of recommendations under the Euclidean similarity mechanisms. The recommender system supports computations over a system which is already explained before where complete computing environment is also mentioned. The trip advisor dataset is used for experiments to evaluate how large number of user's ratings would influence the recommendation accuracy.

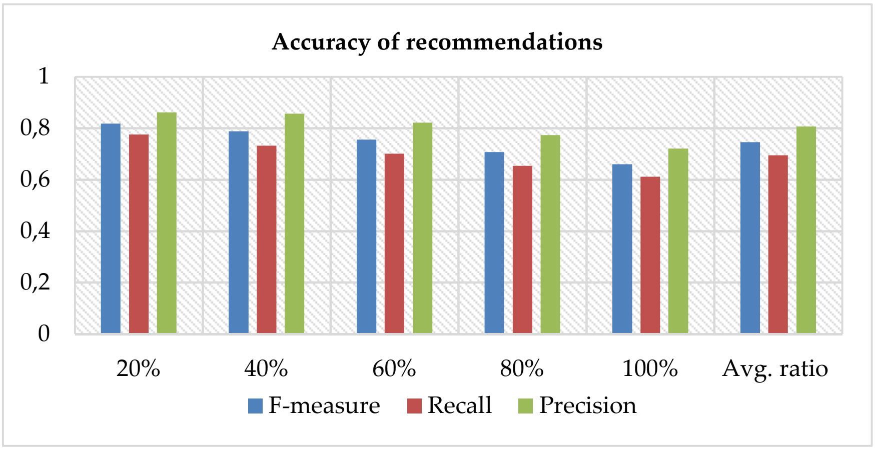


Figure 3: Performance metrics with Incremental chunks of Dataset

4.3 Data Storage and performance evaluation

To analyze the performance of the developed application in respect to time we have taken query execution timestamp as the performance measure of the system. To test the system Step by step experiments are performed by periodically insertion of rating data records initially started from 5000 data records, ranging between from 5000 to 800000 data ratings. Testing is performed on two different databases such as Cassandra and MongoDB to determine the query execution time spent on incremental chunks of the dataset. We have used MongoDB in the testing-phase but it is observed that when query executed with MongoDB, the system got halted and crashed as the data computation is performed with the proposed recommender. Cassandra at the same time performed smoothly which is the reason we have implemented Cassandra in our proposed approach.

There were 5000 number of records that have been analyzed by the recommender system initially for testing (see Figure 4), the query execution time of Cassandra was 20 milliseconds and in Mongo DB it was 45 milliseconds. Similarly, by incrementally adding the records, the total records were reached to 800000 and the corresponding execution time is shown graphically in Figure 4. Total query execution time in milliseconds was 1265(1.265 seconds) in Cassandra and 2462 milliseconds (2.462 seconds) in MongoDB. The comparison of results in testing phase shows that the Cassandra NoSQL database is efficient while MongoDB efficiency is weaker in comparison to Cassandra.

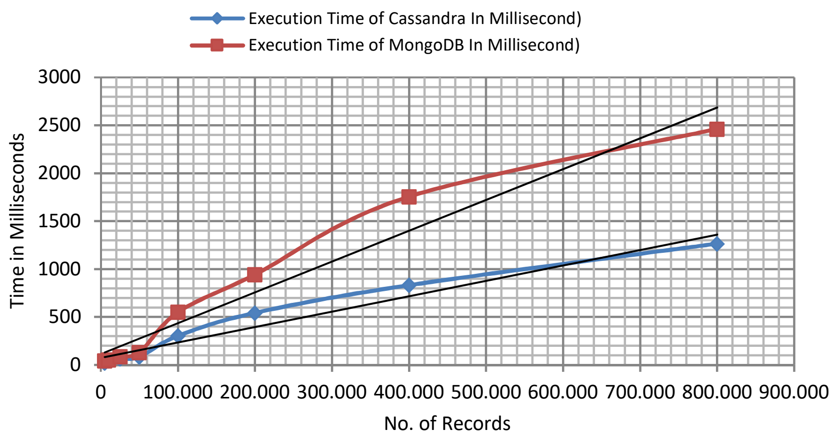


Figure 4: Comparison of Cassandra and MongoDB Performance

Cassandra with the same system has optimized the performance of the designed application of hotel recommender system so we have implemented Cassandra in the proposed framework to improve performance by reducing processing time.

4.4 Discussion

We have gathered a group of 12 users of different age groups and provided them the chance to use our designed recommender system as well as other available recommenders i-e. yatra, trivago, and tripAdvisor to get recommendation for hotels of their choices. These participants or users have been asked to give express their satisfaction level after performing searches over each of these recommenders. They have performed random searches and their satisfaction level is recorded in a table in three categories i.e. less satisfied(LS), satisfied(S), highly satisfied(HS) against three parameters i-e time efficient(T), relevance of results with query(R), cost effective(C) as shown in Table 6.

Satisfaction level of the participants' shows that after using the proposed recommender the opinions of the users are changed most of them has given best category to the proposed recommender system. Performance outcomes generated in terms of response time performed over Big data are recorded in Table 5.

Table 6 shows the performance and response time (processing Time which includes fetch time, load time and search time) obtained after experiments to depict the performance of the system graphically in Figure 5 as shown below.

User Profession	Age	trivago			Yatra			tripadvisor			Proposed Recommender		
		T	**R**	**C**	**T**	**R**	**C**	**T**	**R**	**C**	**T**	**R**	**C**
business	35	LS	LS	S	LS	LS	LS	S	LS	LS	S	HS	HS
student	21	S	S	LS	LS	LS	LS	LS	S	LS	HS	HS	LS
doctor	40	LS	LS	LS	S	LS	LS	LS	LS	LS	HS	S	HS
teacher	33	LS	LS	LS	LS	LS	S	S	LS	S	HS	S	HS
student	17	HS	S	S	LS	S	LS	LS	S	S	LS	HS	HS
business	52	LS	LS	LS	LS	LS	HS	S	LS	S	S	HS	S
business	26	S	LS	LS	LS	LS	LS	LS	S	LS	S	HS	HS
employee	37	LS	S	HS	LS	G	LS	LS	LS	LS	S	HS	HS
student	19	LS	S	LS	S	S	LS	S	LS	S	HS	LS	LS
Teacher	44	LS	LS	S	LS	LS	S	S	S	S	HS	LS	S
teacher	50	LS	LS	LS	LS	LS	S	S	LS	HS	S	S	HS
doctor	30	S	LS	LS	LS	LS	LS	LS	S	LS	S	S	HS

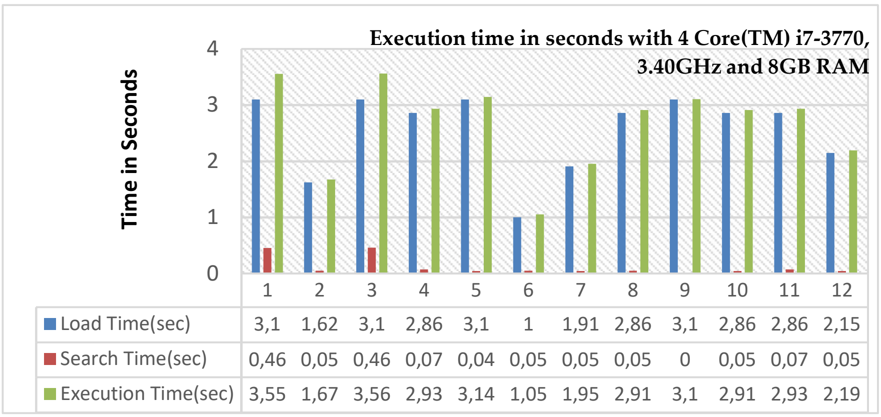
Table 5: Participants Opinions

No.	Load Time	Search Time	Execution Time
1	3.0994	0.0455	3.5544
2	1.6212	0.0500	1.6712
3	3.0994	0.0461	3.5606
4	2.8610	0.0738	2.9348
5	3.0994	0.0447	3.1441
6	1.0013	0.0488	1.0501
7	1.9073	0.4583	1.9531
8	2.8610	0.0469	2.9079
9	3.0994	0.0024	3.1018
10	2.8610	0.0459	2.9069
11	2.8610	0.0727	2.9337
12	2.1457	0.0461	2.1918

Table 6: Queries Response Time

When the user logs into the designed recommender application and demand any recommendation after providing their request data in the search criteria fields according to their choice, the system performs computation using item-based Collaborative Filtering and Euclidean distance algorithm over the ratings stored in

Cassandra to collect the required recommendations. The system has shown promising results by time reduction in the processing time. The system also calculates the time stamps as loading time, searching time and execution time.



	1	2	3	4	5	6	7	8	9	10	11	12
🟦 Load Time(sec)	3,1	1,62	3,1	2,86	3,1	1	1,91	2,86	3,1	2,86	2,86	2,15
🟥 Search Time(sec)	0,46	0,05	0,46	0,07	0,04	0,05	0,05	0,05	0	0,05	0,07	0,05
🟩 Execution Time(sec)	3,55	1,67	3,56	2,93	3,14	1,05	1,95	2,91	3,1	2,91	2,93	2,19

Figure 5: System response time

The sum of loading time and searching time together is called as execution time and is also presented graphically in Figure 5. The Average performance of the system is calculated and is graphically depicted in the Figure 6.

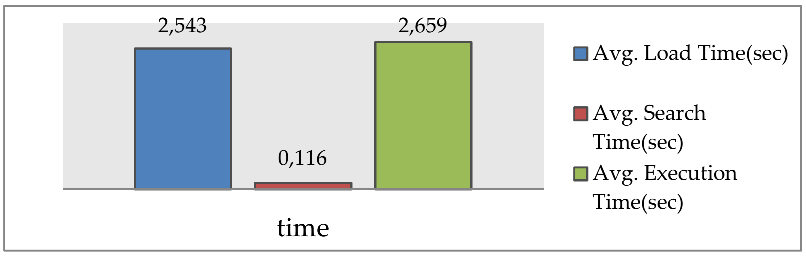


Figure 6: Average response time

The comparison of fetch time in comparison to the load time and search time is graphically represented in the Figure 7, where three different curves of processing time comparison in Cassandra based Hotel Recommender app are drawn. The system calculates the time stamp of loading of data from system to Cassandra database called as fetch time, from Cassandra to memory (RAM) called as load time and also the search time of display results meeting search criteria. The fetch time of the hotel recommender system is slightly high as compared to load and search time (see Table 7 and Figure 7) because the complete dataset in the database is fetched at once so first time when data is fetched into the database it takes more time. The data is fetched at once, so it takes time to fetch the data when any query is processed in the system by the user, the load time and search time will be reduced in comparison to fetch time.

Records in Database	Fetch Time (Sec)	Load Time (Sec)	Search Time (Sec)
5,000	2.48	.87	0.012
50,000	5.78	1.03	0.035
100,000	7.88	2.13	0.063
200,000	9.96	3.76	0.098
400,000	17.85	4.68	0.126
800,000	21.43	9.31	0.131

Table 7: Fetch, Load, Search Time of Hotel Recommender System

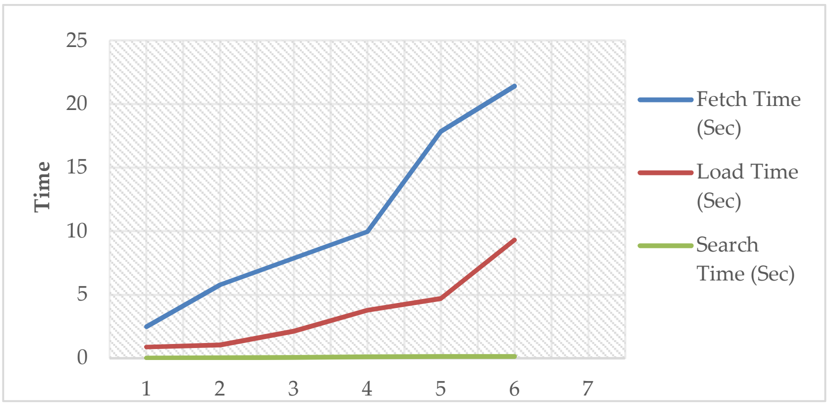


Figure 7: Processing Time comparison

The proposed study also addresses social impact in terms of success of e-commerce, consumer's behavior, customer satisfaction, etc. The social impact can be created by achieving true recommendations for customers in terms of relevance and accuracy of recommendations. The true recommendations will help in improving the customer's satisfaction level and that will play role in success of e-commerce applications that will ultimately affect the business strategy and planning.

6 Conclusion and Future Work

In this paper a hotel recommender application is developed by using an item-based Collaborative filtering algorithm along with Cassandra storage to guarantee the high response time and speed of recommendation generation. Users' ratings are stored in Cassandra which is able to handle big data without failure. For efficient and accurate recommendations, Euclidean distance as an effective similarity measure is used to generate quality recommendations. The proposed hotel recommendation system is a beneficial tool that recommends hotels to users according to their choices using ratings of the dataset. We have tested the performance gains of using Cassandra in terms of response time of the web application to generate recommendations. It has shown significant performance to help reduce the response time by improving performance efficiency and scalability while dealing the large amount of data stored in the database, searching over 800,000 records in 1.265 seconds without failure. The F measure ratio has resulted in 0.74 approximately 74% which demonstrate that the proposed Recommender approach provides promising results in terms of time and performance improvement which help consumers to get recommendations which are related to their tastes and it will impact their future behavior to use recommenders.

In future, we will study methods and techniques which will allow recommender system to automatically use updated reviews and ratings online from the websites dynamically to give fresh recommendations. However, in operation, several new techniques should also be adopted by the operating websites in order to find out if the users have welcomed the resultant hotel recommendations. This can be done, for example, by integrating web cookies in order to capture user navigational traces and behaviors, and by obtaining users' feedbacks on new recommended items.

References

[Almeida, et al., 2012] N. M. Almeida, J. A. Silva, J. Mendes, and P. Oom do Valle, "The effects of marketing communication on the tourist's hotel reservation process," Anatolia, vol. 23, no. 2, pp. 234-250, 2012.

[Bouras. et al., 2017] C. Bouras and V. Tsogkas, "Improving news articles recommendations via user clustering," International Journal of Machine Learning and Cybernetics, vol. 8, no. 1, pp. 223-237, 2017.

[Burke, et al., 2002] R. Burke, "Hybrid recommender systems: Survey and experiments," User modeling and user-adapted interaction, vol. 12, no. 4, pp. 331-370, 2002.

[Cacheda, et al., 2011] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," ACM Transactions on the Web (TWEB), vol. 5, no. 1, p. 2, 2011.

[Chang, et al., 2013] Z. Chang, M. S. Arefin, and Y. Morimoto, "Hotel recommendation based on surrounding environments," in Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on, 2013, pp. 330-336: IEEE.

[Chawala, et al., 2013] N. Chawla and M. Wiboonrat, "The model of personalized intelligent information for budget hotel services," in Informatics and Applications (ICIA), 2013 Second International Conference on, 2013, pp. 247-251: IEEE.

[Chen, et al., 2018] T. Chen and Y. H. Chuang, "Fuzzy and nonlinear programming approach for optimizing the performance of ubiquitous hotel recommendation," Journal of Ambient Intelligence and Humanized Computing, vol. 9, no. 2, pp. 275-284, 2018.

[Ekstrand, et al., 2011] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," Foundations and Trends® in Human–Computer Interaction, vol. 4, no. 2, pp. 81-173, 2011.

[Fasahte, et al., 2017] U. Fasahte, D. Gambhir, M. Merulingkar, and A. M. P. A. Pokhare, "Hotel Recommendation System," Imperial Journal of Interdisciplinary Research, vol. 3, no. 11, 2017.

[Ghose, et al., 2012] A. Ghose, P. G. Ipeirotis, and B. Li, "Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content," Marketing Science, vol. 31, no. 3, pp. 493-520, 2012.

[Hasieh, et al., 2017] M.-Y. Hsieh, W.-K. Chou, and K.-C. Li, "Building a mobile movie recommendation service by user rating and APP usage with linked data on Hadoop," Multimedia Tools and Applications, vol. 76, no. 3, pp. 3383-3401, 2017.

[Hu, et al., 2016] Y.-H. Hu, P.-J. Lee, K. Chen, J. M. Tarn, and D.-V. Dang, "Hotel Recommendation System based on Review and Context Information: a Collaborative filtering Appro," in PACIS, 2016, p. 221.

[Huming, et al., 2010] G. Huming and L. Weili, "A hotel recommendation system based on collaborative filtering and rankboost algorithm," in Multimedia and Information Technology (MMIT), 2010 Second International Conference on, 2010, vol. 1, pp. 317-320: IEEE.

[Hwang, et al., 2015] S.-Y. Hwang, C.-Y. Lai, S. Chang, and J.-J. Jiang, "The identification of noteworthy hotel reviews for hotel management," Pacific Asia Journal of the Association for Information Systems, vol. 6, no. 4, 2015.

[Ibrahim, et al., 2019a] M. Ibrahim, I.S. Bajwa, R.U. Amin, and B. Kasi, “A Neural Network-Inspired Approach for Improved and True Movie Recommendations,” Computational Intelligence and Neuroscience, vol. 2019, Article ID 4589060, 19 pages, 2019

[Ibrahim, et al., 2019b] M. Ibrahim, I.S. Bajwa. "Design and Application of a Multi-Variant Expert System Using Apache Hadoop Framework". Sustainability 2018, 10, 4280.

[Jannach, et al., 2012] D. Jannach, F. Gedikli, Z. Karakaya, and O. Juwig, Recommending hotels based on multi-dimensional customer ratings. na, 2012.

[Kabassi, et al., 2010] K. Kabassi, "Personalizing recommendations for tourists," Telematics and Informatics, vol. 27, no. 1, pp. 51-66, 2010.

[Koren, et al., 2009] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, 2009.

[Lin, et al., 2015] K.-P. Lin, C.-Y. Lai, P.-C. Chen, and S.-Y. Hwang, "Personalized hotel recommendation using text mining and mobile browsing tracking," in Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on, 2015, pp. 191-196:

[Liu, et al., 2013] H. Liu, J. He, T. Wang, W. Song, and X. Du, "Combining user preferences and user opinions for accurate recommendation," Electronic Commerce Research and Applications, vol. 12, no. 1, pp. 14-23, 2013.

[Lops, et al., 2011] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in Recommender systems handbook: Springer, 2011, pp. 73-105.

[Mariani, et. al. 2014] M. M. Mariani, D. Buhalis, C. Longhi, and O. Vitouladiti, "Managing change in tourism destinations: Key issues and current trends," Journal of Destination Marketing & Management, vol. 2, no. 4, pp. 269-272, 2014.

[Meng, et al, 2014] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: A Keyword-Aware Service Recommendation method on MapReduce for big data applications," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 12, pp. 3221-3231, 2014.

[Rianthong, et al., 2016] N. Rianthong, A. Dumrongsiri, and Y. Kohda, "Improving the multidimensional sequencing of hotel rooms on an online travel agency web site," Electronic Commerce Research and Applications, vol. 17, pp. 74-86, 2016.

[Sharma, et al., 2015] Y. Sharma, J. Bhatt, and R. Magon, "A Multi-criteria Review-Based Hotel Recommendation System," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, 2015, pp. 687-691: IEEE.

[Shambour, et al., 2016] Q. Shambour, M. a. Hourani, and S. Fraihat, "An item-based multi-criteria collaborative filtering algorithm for personalized recommender systems," International Journal of Advanced Computer Science and Applications, vol. 7, no. 8, pp. 275-279, 2016.

[Valcarce, et al., 2015] D. Valcarce, J. Parapar, and A. Barreiro, "A Distributed Recommendation Platform for Big Data," J. UCS, vol. 21, no. 13, pp. 1810-1829, 2015.

[Zhang, et al. 2012] J. J. Zhang and Z. Mao, "Image of all hotel scales on travel blogs: Its impact on customer loyalty," Journal of Hospitality Marketing & Management, vol. 21, no. 2, pp. 113-131, 2012.