Enhancing the Accuracy of Recommender System Using Graph Databases

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By

Ashish Sharma

(801332005)

Under the supervision of:

Dr. Shalini Batra
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Enhancing the Accuracy of Recommender System Using Graph Databases*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Shalini Batra* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Ashish Sharma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Shalini Batra)

Assistant Professor,

CSED

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. S. Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

ACKNOWLEDGEMENT

No volume of words is enough to express my gratitude towards my guide **Dr. Shalini Batra**, Computer Science & Engineering Department, Thapar University, Patiala, who has been very concerned and has aided for all the materials essentials for the preparation of this thesis report. She has helped me to explore this vast topic in an organized manner and provided me all the ideas on how to work towards a research oriented venture.

I am also thankful to **Dr. S. S. Bhatia**, Dean of Academic Affairs, **Dr. Deepak Garg**, Head of Computer Science & Engineering Department and **Mr. Ashutosh Mishra**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of hour and provided with all the help and facilities, which I required, for the completion of my thesis work.

Most importantly, I would like to thank my parents and the almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

ABSTRACT

In recent years size of the data available to user over online media has increased exponentially. Due to this large amount of data, people face problem in viewing all available data due to lack of time. To overcome such type of problem, the recommender system plays an important role. In recent years, the recommender systems are most popular tool and have been used in a many types of applications such as facebook, twitter, youtube.com, Amazon.com etc. Recommender systems broadly fall into two groups: Collaborative and content based recommender systems. The collaborative-filtering methods generate the recommendation to user on the basis of the nearest user which is most similar to them. User-based Collaborative-filtering (CF) which uses the matrix to store the ratings of the user is the most frequently used recommender technique, widely used because of its simplicity and efficient performance. Although it is extensively used, one of its major problems is that its performance decreases when the user-item matrix becomes sparse. One of the proposed solutions is to the usage of combination of graph data base and Locality Sensitive Hashing (LSH). Graph database provide the flexibility to developer to design database without performing any normalization and LSH provides a faster method to find the nearest neighbor for the recommendation to user. The proposed system concludes with comparison of traditional approach with the proposed approach.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Recommender System

Recommender system is the subpart of the information filtering system, which helps users by refining the data and provide useful information to user. When this information comes in the form of recommendation, an information filtering system is called recommender system. Online Recommender system works on the implicit or explicit information provided by the users. Nowadays recommender systems have become a popular tool and have been used in a many types of applications such as facebook, twitter, youtube.com, Amazon.com, MovieLen, Netflix.com, Imdb.com and so on. These systems can provide customized or personalized and non-personalized recommendation to interested user. Following are the snapshots of amazon.com and Imdb.com which show how recommendations take place.

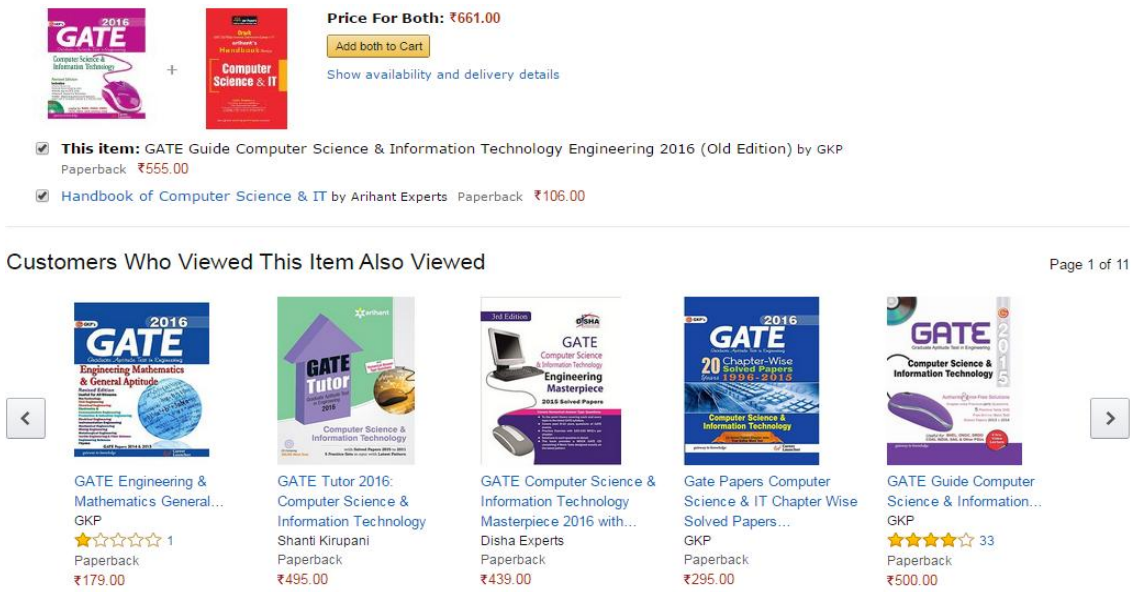

Figure 1.1: Recommendations from amazon.com

Figure 1.2: Recommendations from Imdb.com

In the recent years recommender system earned their importance and are being used in the large amount of applications which includes products, search query, social tagging, movies, news, music, books and articles in general [1, 2, 3].

There are some property of recommender system which decides usefulness and correctness of the recommender system [4]. Some of them are:

- Prediction Accuracy: Prediction accuracy is the most discussed property in the literature of recommender system. It is based on the assumption that systems that produce more accurate prediction will be preferred by the user.
- Coverage: the term coverage means the proportion of the item or the products that the recommender system can recommended to the user. The most useful measurement of coverage is the percentage of all products or items which are suggested to users at the time of experiment.

- Confidence: It can be defined as the Recommender system's trust in its suggestion or recommendation. The confidence level must be high for the automated system which gives recommendation and must tell the level of confidence in each recommendation.
- Trust: It is the reliability or faith developed by the user in the recommender system. If recommender system generate useful and reasonable suggestion to the user then the trust of user on that system will increase otherwise it will decrease.
- Novelty: the Recommender system must generate the novel recommendation to the user i.e. it must be new or should not be the one which the user already seen or rated.
- Diversity: In some cases same type of recommendation is not useful for user. The recommender system must generate the diverse recommendation to the users i.e. it should not be the same type but it must cover multiple types.
- Robustness: It is the ability of the system that how much system is secure against the profile injection attacks. It must generate the accurate recommendation even in the presence of fraud information i.e. system must be capable of handling the attacks.
- Adaptively: It refers to the dynamicity of the recommender system. System must be adaptable to changing environments so that it can give best recommendation to user.
- Scalability: It is measured by performing the experiment with increasing data set. Showing that how the consumption of resources and speed of recommendation behaves as the data set scales up. Recommender system must generate quick result for users and should not slow down when the size of data increases.

RS generate a recommendation list by five major methods [5].These methods are:

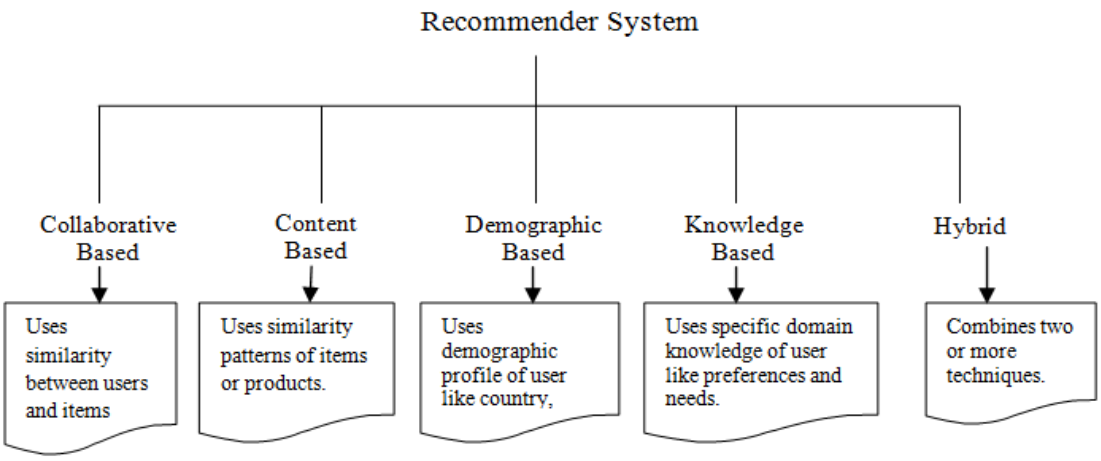

Figure 1.3: Five major methods of recommender system [5]

- Collaboration based: Collaborative filtering generate the recommendation by using the rating information of users in the form of user-item matrix. This system finds the nearest neighbor to the specific user on the basis of ratings and using these users recommendations are generated [6]. Collaborative filtering is the mostly used and famous recommender technique, widely used because of its simplicity and good results.
- Content based: Content-based recommender systems provide recommendation of the similar type of items to those available users that had liked in the past. The general process of the content-based recommender system is that it compares the characteristic of available user profile with the characteristic of items, in order to provide recommendation to the user [7].
- Demographic based: In the demographic based recommender system, socio-demographic attribute such as education, gender, sex, marital status, profession, class, country *etc* are used. This type of system generates recommendation of item based on demographic profile of user [8]
- Knowledge based: In knowledge based recommender system generates the recommendation of item based on the specific domain knowledge about how the feature of particular item meet the needs of user and preference i.e. how the particular item is useful for the user [9].

- Hybrid based: Hybrid based recommender systems merge two or more recommendation approaches to achieve better performance with few limitations of any individual. Generally, collaborative filtering based approach is combined with some other method in an effort to remove the problems [10].

The different recommender system methods need different input of users and use different databases for suggesting the recommendation for user [5]. Knowledge based recommender system uses the item knowledge and product database for generating the recommendation. It requires users input or query into the system. The content based recommender system takes the features associated with item as an input and database for generating the recommendation. The collaborative filtering based recommender system takes the rating of user as input and database of ratings given by users for recommending the items. The demographic based recommender system uses socio-demographic information as an input, user's rating and demographic database for predicting the recommendation for user. The recommender system approaches and their input pattern are described in fig 1.4:

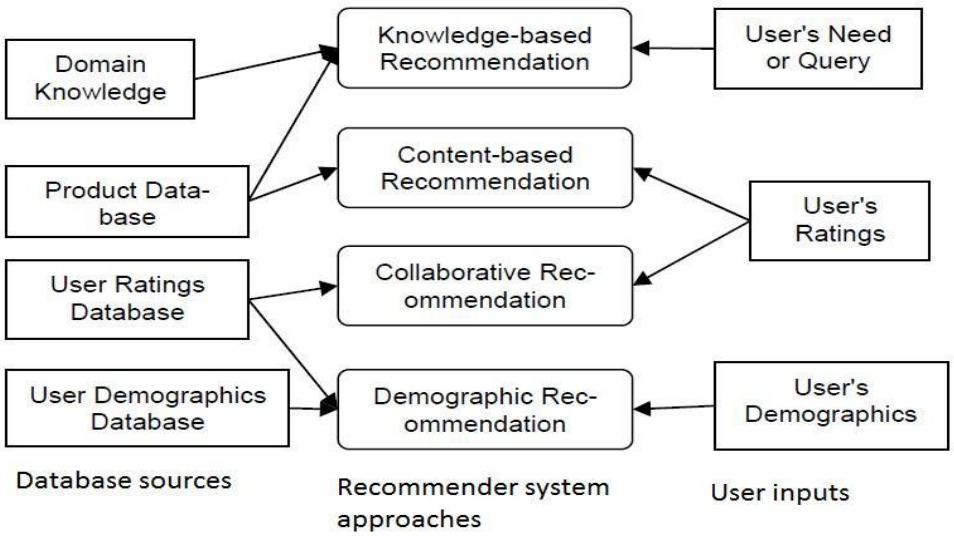

Figure 1.4: Recommender system approaches and Inputs [5]

1.2 Collaborative Filtering (CF) Based Approach

Collaborative filtering based recommendation is a technique of filtering data based on the collaboration of other users. Collaborative filtering uses the user-item matrix in spite of user or item information. Collaborative filtering is the mostly used and famous recommender technique, widely used because of its simplicity and good results. The first recommender system, Tapestry [11] use this term of collaborative filtering, and since then it has been widely accepted. It is based on the fact that if two users *X* and *Y* have rated *n* items similarly, or behave similarly in any environment than they will also rate or behave similarly on other items also. Collaborative filtering is divided into following groups:

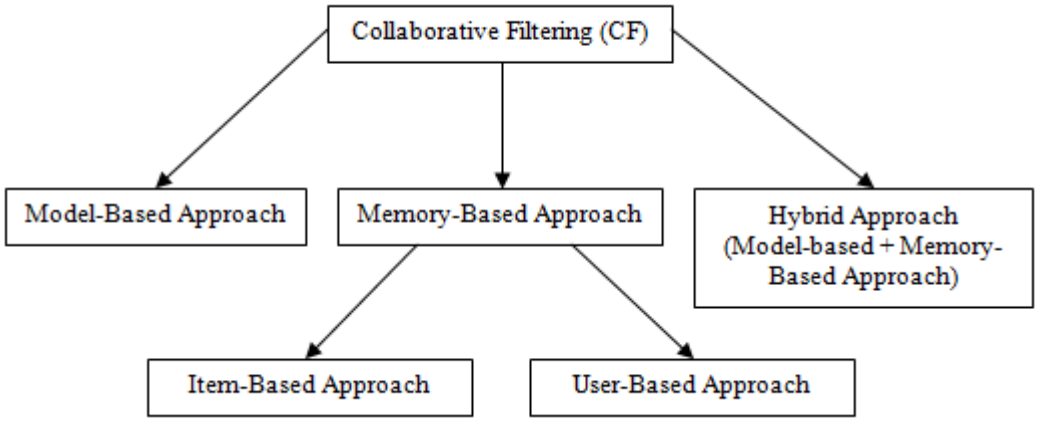

Figure 1.5: Hierarchical model of collaborative filtering

- Memory-based: Memory-based basically memorizes the rating matrix and provide recommendations based on the relations between users and items e.g. neighbour-based methods.
- Model-based: Model-based uses the rating matrix to fit in a parameterized model to give the recommendations.

Memory-based Collaborative filtering can be further divided into two types:

- User-based: In user-based technique the neighbours are calculated and using their information the recommendations are provided.
- Item-based: In item-based technique, the similar items are calculated and predictions are calculated using the slope-one algorithm from the similar items.

Item-based Collaborative filtering technique is good for sparse data. User-based methods work efficiently for low user count while the item-based methods outperform all the memory-based technique for large user-counts. Collaborative filtering suffers from various problems of sparse user-item graph, cold-start problem, shilling attacks, grey-ship problem *etc*.

Other methods of collaborative filtering could be based on implicit feedback or explicit feedback. In the implicit feedback system observe the behavior of user in order to decide rating of the particular item by analyzing that what type of item they liked. But in case of explicit feedback, feedbacks are provided explicitly in form of like or dislike.

1.2.1 Types of Collaborative Filtering (CF)

1.2.1.1 Memory Based

These methods make use of information rating in order to predict the similarity between users or items. This is the well-known former process and has been implemented in variety of application. The most widely used Memory Based method is K-Nearest Neighborhood method. In the memory based approaches, the ratings value of the item ‘i’ for the item ‘i’, given by user 'u', is computed as rating summation of some similar type of users to the item:

$$r_{u,i} = aggr_{u' \in U} r_{u',i} \tag{1}$$

Various aggregation functions used in RS include:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$
$$r_{u,i} = k \sum_{u' \in U} simil(u,u') r_{u',i}$$
$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} simil(u,u')(r_{u',i} - \bar{r_{u'}}) \tag{2}$$

Where ‘k’ represents the normalizing factor and ‘$\bar{r}_u$’ is the average rating of the all the items of the user ‘u’.

The K-Nearest neighbor algorithm computes the similarity between users or items and generates the recommendation on basis of similarity. This similarity between items or users can be calculated through several methods such as Pearson Correlation Coefficient, Cosine Similarity *etc.*

Pearson Correlation Coefficient is calculated as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (3)$$

Where ‘x_i’ is the rating of items and $\bar{\mathbf{x}}$ is the average rating of all items given by user ‘x’.

Cosine Similarity is calculated as:

$$\cos(x,y) = \frac{x \cdot y}{\|x\| \times \|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad \ldots (4)$$

The user-based K-Nearest method identifies the top ‘k’ most similar users. After the ‘k’ most similar users are identified, then based on user-item matrices, set of items are recommended.

The advantage of this approach is that this method is very simple and easy to use; new data can be added easily.

There are some disadvantages of this method:

- It is depend on individual ratings.

- The performance decreases when user-item matrix become sparse, which is very obvious in case of web items.
- Collaborative filtering also suffers from the cold start problem, because fails to provide the recommendation to new users which has not rated any item.

1.2.1.2 Model-Based

Model-based recommender systems involve building a model based on the given information set of ratings. In other words, one who retrieve some information from the information set, and use that as a "model" to make recommendations without having to use the complete information set every time. This approach provides the benefits of both speed and scalability.

Memory-based recommender systems are not always fast and expandable as individual would like, especially in the term of actual systems that generate immediate recommendations on the basis of very large information sets. To accomplish these goals, model-based recommender systems are proposed.

The item-based CF is the example of Model-based filtering, in this similarity are calculated by the adjusted cosine-based similarity formula:

$$s(i,j) = \frac{\sum_{u \in U} (R_{u,i} - \overline{R}_u)(R_{u,j} - \overline{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \overline{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \overline{R}_u)^2}} \quad (5)$$

Although the basic idea behind model-based recommender systems is the same as memory-based recommender system, there are number of methods that one can actually build the model and use it. Some examples are: Bayesian networks and clustering, Linear algebra problem, *k* most similar entities and singular value decomposition, *etc.*

There are several advantages with this paradigm: It handles the sparse data better than memory based ones. Model-based systems are faster than the memory-based systems because, the time required to perform the query the subset of dataset is usually much smaller than that required to query the whole dataset.

The disadvantages with model- based approach are in the costly model building. It is also inflexible because building a model requires time- and resource-consuming process; so it is usually more difficult to add data in the model-based recommender systems, making them inflexible.

1.3 Graph Database Model

The previous section has shown several approaches to generate the recommendation to user. In all approaches there is a common point: it important to relate the one item to other. Actually, in either item-to-item or user-to-user system, the heart of the technique is to find the connection between elements, the items are compared to each other as well as users.

To store such type of connection the graph data base can be used as it has been identified that graphs are very mature to hold such type of data. The relational database or matrix is good for the small data set, but in case of large and dynamic data it is does not work well. The graph database can be suitable for both small as well as large data set, and it can calculate quickly difficult operation on the relationship. The graph database model consists of the nodes and edges (relationship). Both of them can have attribute which are key to represent them.

A graph is a collection of a set of vertices where some pairs of vertices are connected by edges. Both vertices and edges can have attribute which are key to represent them. Graph database is the unstructured, noSql database where the focus is on the nodes and the relationship between them [13]. One of the major advantages of graph data bases is that unlike the relational database there is no special convention to design graph database. Further, graph database provide the flexibility to developer to design database and unlike the relational database there is no need to normalize the database.

1.4 Locality Sensitive Hashing

Localities sensitive hashing belongs to the special class of algorithm called randomize algorithms. This algorithm does not provide the guarantee for an exact match but instead

it provides a higher probability of returning the correct answer or one close to it. Locality sensitivity hashing is used for reducing the dimension of high-dimension data. The locality sensitive hashing hashes the input data so that same data map to the same bucket with high probability. Locality sensitive hashing is differ from the general hashing because its main aim to increase the probability of collision for similar data where as in conventional hashing reduce the probability of collision for similar data as shown in figure 1.6:

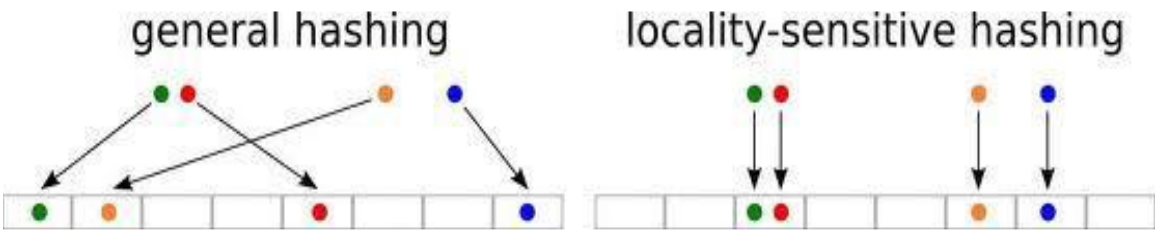

Figure 1.6 Comparison between General hashing and Locality Sensitive Hashing [14]

In the locality sensitive hashing any pair that hash to the same bucket for any of the hashing is to be consider as "Candidate pair" and similarity is checked only on candidate pair rather than checking the similarity for each pair, as it is assumed that dissimilar pair will never fall into same bucket they are never checked. If some dissimilar pairs fall into same bucket then they are termed as "*false positive*". Likewise it is also assumed that similar pair falls into same bucket. Similar pair that does not fall into same bucket is "*false negative*".

In order to use the locality sensitive hashing in deferent application, one needs to develop locality sensitive hashing according to the field defined for that application. A family of min hash function is used with the banding technique to distinguish pair [15]. An effective way to select the hashing is to break the signature matrix 'M' generate though Min hash technique into b-bands each one consisting of r-rows. For each row there is a hash function which takes r-integer and hash them to same bucket, we cannot use the same bucket for each band but we can use same hash function for all bands as shown in fig 1.7

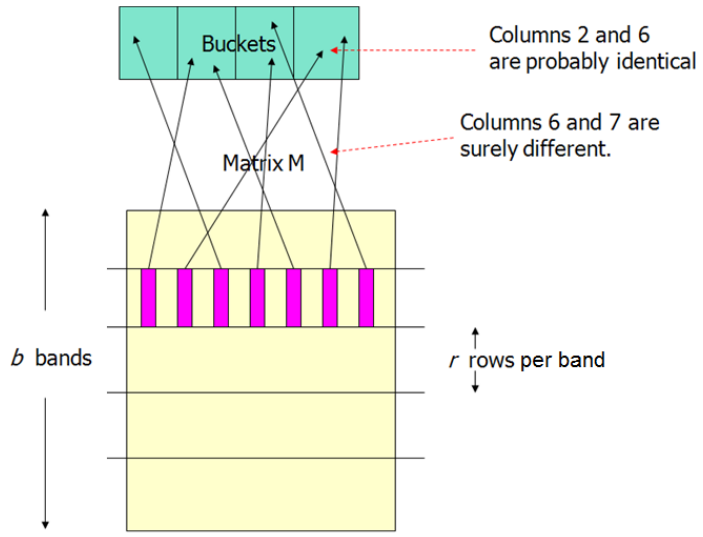

Figure 1: Banding technique of locality sensitive hashing [16]

Let $w_1 \leq w_2$ be a two distances based on to some distance w. A family G of functions is said to be (w_1, w_2, q_1, q_2) - sensitive if for every $g$in G [17]:

- If $d(a, b) \leq w_1$, then probability that $g(a) = g(b)$ is at least q_1.

- If $d(a, b) \leq w_2$then probability that $g(a) \neq g(b)$ is at most $q2$.

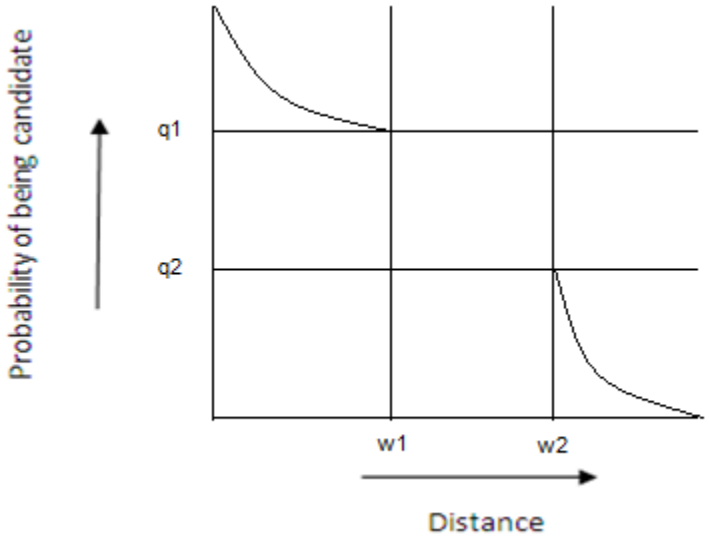

Figure 1.8: Behavior of a (w1, w2, q1, q2)-sensitive function

Figure 1.8 depicts a graph in which x-axis represents distance and y-axis represents the probability of being candidate pairs. If the distance of points is in between w1 and w2 then the probability cannot be defined. w1 and w2 can be close to each other but the problem is that then q1 and q2 will not be distinguishable [16].

A most important theory in the LSH is amplification. Given a (w_1, w_2, q_1, q_2)-sensitive family G, a new family *G'* can be constructed by either AND-construction or OR-construction.

AND-construction of *G'* is defined as follows: Each member of *G'* consists of *t* members of *G* for some fixed *t*. If *g* is in *G'*, and *g* is constructed from the set $\{g_1, g_2, ..., g_t\}$ of members of G, $g(x) = g(y)$ if and only if $g_i(x) = g_i(y)$ for all $i = 1, 2, ..., t$. As members of *G'* are independently chosen from *G*, *G'* is an (w_1, w_2, q_1^t, q_2^t)-sensitive family [16].

OR-construction of *G'* is defined as follows: Each member of *G'* consists of *l* members of *G* for some fixed *l*. If *g* is in *G'*, and *g* is constructed from the set $\{g_1, g_2, ..., g_l\}$ of members of *G*, $g(x) = g(y)$ if and only if $g_i(x) = g_i(y)$ for all $i = 1, 2, ..., l$. Similarly, *G'* is an (w_1, w_2, $1 - (1 - q_1)^l$, $1 - (1 - q_2)^l$)-sensitive family.

1.5 Structure of the thesis

Below is the summary of the rest of the chapters of the thesis:

Chapter 2: Literature Survey. This chapter introduces all the related work and survey of the recommender system and its approaches.

Chapter 3: Problem Statement. In this chapter the gaps in the field of recommender system and methodology of proposed approach is described.

Chapter 4: Proposed Work. Proposed methodology with example is explained in details in this chapter.

Chapter 5: Experimental Analysis. This chapter contains the description of the results and observations of the experimental analysis of proposed system.

Chapter 6: Conclusion and Future Scope. The whole work presented in thesis is summarized in this chapter and it also contains the directions for the future research of the work.

Chapter 2
Literature Survey

2.1 Evolution of the Recommender system

In recent years recommender systems have been used in e-commerce application on the web and they have become important area of research [17] [18]. Almost two decade of research on the Recommender system have led to generation of verity of algorithms and tools for analyzing their performance. In the early 1990s manual collaborative filtering was used to provide solution for dealing with the problem of online information overload. The Tapestry [11] was the first collaborative filtering based recommender system: it allowed the user to query for the item in an information domain based on the others user's opinion or actions.

The oldest version of the recommender system is the "word of mouth". In the past people used this method to buy something and recommender system was built to deal with this "word of mouth" [19]. John, Joseph and Loren developed a project in the area of the recommender system called "Grouplen" from the Minnesota University. Now a days' recommender system has become a separate area of research since the exponential growth in the internet and has become solution to e-commerce system and it is very helpful in gaining higher profit.

There is large number of application on which recommender system proved very useful such as e-commerce, matrimonial, course recommender, planner, pandora, facebook.com, twitter.com, linkedin.com, youtube.com *etc*.

2.2 Collaborative Filtering Approaches

The Collaborative filtering (CF) still one of the most widely and common used method for generating the recommendation to users [20]. Most of the applications like snapdeal.com, flipkart.com amazon.com *etc.* use CF because of its simplicity and good result. According to the Breese *et al.* Collaborative filtering is divided into three categories: Memory-based method, Model-Based method and Hybrid method [21].

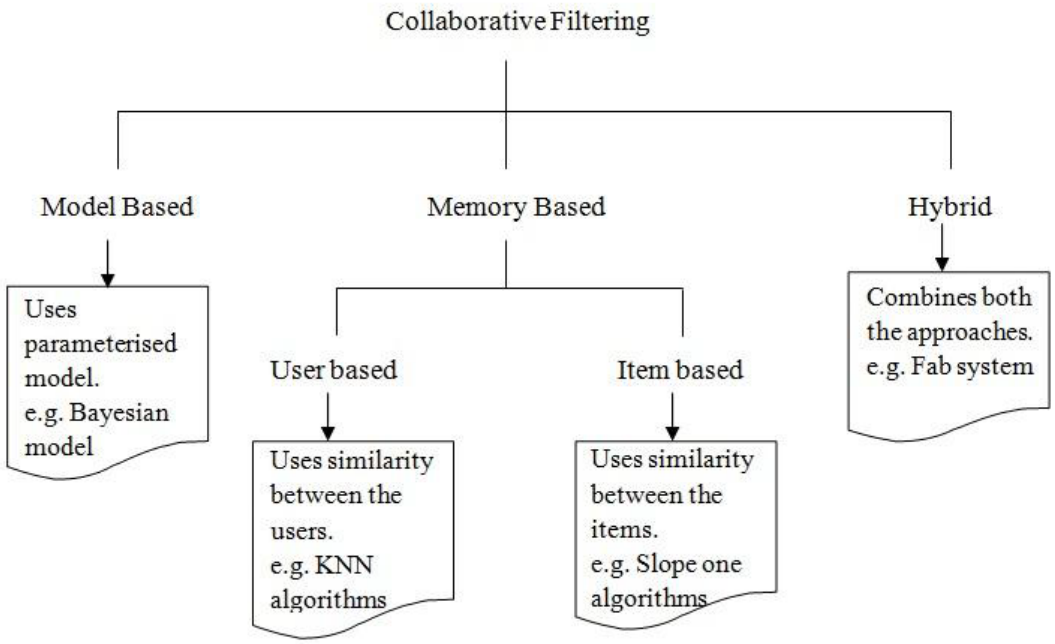

Figure 2.1: Classification of collaborative filtering

Memory-based basically memorizes the rating matrix and provide recommendations based on the relations between users and items. Memory-based methods are mostly used method and Resnick *et al.* first use the concept of the Memory-based model [22]. Breese *et al.* [21] divide memory based algorithm into two sub groups: user based and item based. In the user based similarity of different users is used for finding the neighbors. Sarwar *et al.* [23] used various methods like Euclidean distance, Manhattan distance and Pearson coefficient for find the similarity. K-nearest neighbor is the most widely and common method of the user based system as discussed by Khoshgoftaar and Su [25]. Sarwar *et al.* introduced the item based method and used the similarity of item instead of users [24]. In the item based similarity between items are calculate using the cosine similarity. The most common example of the item based method is the slope one algorithm[26].

There are some drawback of memory based CF:

- The performance of memory based CF decrease if the data become sparse.
- Memory based CF suffer from the cold start problem.
- It is deepened on the human rating.

- Scalability of memory based reduces for large database.

A model based algorithm performs better for sparse data and it is also scalable. It gives recommendation by using the rating matrix to fit parameterize model e.g. Bayesian classifiers [4].

The major shortcomings of model based CF are:

- Model based CF is expensive to build.
- There is trade-off between the recommender accuracy and scalability.
- Model based CF is lack in diversity.

Hybrid based CF [27] uses the combination of both memory based and model based method to generate more efficient algorithm. Fab recommender system is the most common example of hybrid based CF algorithm. According to Khoshgoftaar and Su hybrid based CF also has some limitations. Some of these are:

- It is expansive to implement.
- Increased complexity
- It required lot of information.

Content based recommender systems make use of content of the item and matching of the item is done on the basis of the content only [28]. Pandora.com is the most famous internet radio system use the concept of content based RS. But this class of RS also has some serious issues. Some of them are:

- The content of items is fixed, so the analysis of content is limited.
- It suffers from serendipity because they always recommend similar type of item.
- They are vulnerable to attacks.
- They are suffered from cold start problem.

Demographic based recommender system [8] use the socio- demographic attribute such as education, gender, sex, marital status, profession, class, country *etc*. This type of system generates recommendation of item based on demographic profile of user [8]. Some of the usual examples of demographic based systems are Matrimonial sites, naukri.com and so on.

Demographic based system also has some drawback:

- If the information about the user is less, then in that case it may give bad recommendation.
- This type of system is highly dependent on the information.
- It is highly dependent on user inputs.

Knowledge based recommender system [9] generate the recommendation of item based on the specific domain knowledge about how the feature of particular item meet the needs of user and preference i.e. how the particular item is useful for the user. In these type of system Similarity function calculate that how the user needs match to the recommendation [9]. Entree system, a restaurant recommender system is the example of the knowledge based system.

Few short comings of knowledge based system are:

- Knowledge based RS require large amount of information.
- The accuracy of recommendation is less.
- If users with no or less information, then in that case it cannot predict recommendation for that users.
- Human interaction is must to provide all the detail about them.

Hybrid based recommender systems merge two or more recommendation approaches to achieve better performance with fewer of the limitations of any individual one. Generally, collaborative filtering based approach is combined with some other method in an effort to remove the problems [10]. Netflix.com is the example of the hybrid based recommender system which use the combination of collaborative based filtering and content based method to recommend movies to users.

Drawbacks of the hybrid system are:

- Increased complexity.
- Expensive implementation.

- Context dependent.

2.3 Combining different approaches with collaborative filtering

Some of the serious issues found in collaborative filtering have been overcome but still some of the concerns need to be addressed. According to the R. Burke [9] some of the major problems of RS are sparsity of user item matrix, complexity of matrix, Shilling attacks, cold start problem, accuracy *etc.* Much work has been done to overcome the problems and techniques like Bayesian classifier, K-means, neighbour-based methods, Matrix factorization, locality sensitive hashing have been used to overcome these problems.

2.4 Graph Database

It has been found that relational database have some serious drawback while processing high dimension and dynamic data used in the social network or in the recommender systems (RS) because all the data in the relational database have to represent in the form of table. On the other side, graph database represent data as the collection of nodes and edges.

Mostly graph databases have been used in the field such as chemical informatics, machine learning and bioinformatics [29]. Graph database is the new way to deal with the unstructured web data like data of ratings used in the RS. However, there exist some works that deal with the data storing and data processing for RS.

Betul J. *et al.* proposed how recommender system performs efficiently with graph databases [30]. However, their work did not consider any cutting method for decreasing the complexity of searching which makes the work not appropriate for large volume of graph.

Currently a wide variety of graph databases are available in the market such as Neo4j, Oracle noSql, Orly, Cayley, AvangoDB *etc.* Chad Vicknai *et al.* explained that security is the very important key aspect in the selection of databases for any kind of systems [31].

2.4.1 Neo4j Graph Database

Neo Technology was established in the year 2007. One of the most widely used technologies of the neo tech. is the Neo4j. It is full ACID based open source noSql graph database implemented in the java language. Graph database (GDB) has potential to explore and search highly interconnected social network. GDB can solve the problem of joining of large table in relational database by simple graph traversal. Neo4j is available in two editions: Neo4j enterprise edition and neo4j community edition. Enterprise edition provide clustering across multiple machines. Community edition offering full feature graph but could run one machine only.

The first version 1.0 of the neo4j was released in February, 2007. The next improved version 1.6 of neo4j was released in Jan, 2012, 1.7 in April, and 1.8 in October. In the middle of 2013 neo4j release its newer version 1.9 with auto clustering support. The latest version 2.2.2 of neo4j has been introduced in May, 2015 with full support of Oracle and Java JDK 8.

2.4.2 Cypher Query Language

Cypher is the query language of the Neo4j Database. It is the analytical language that allows developer to read and write on the graph database. One of the main advantage of this language is that it is human readable and easy to understand. The new feature of Neo4j extended support for labels and indexes. Now nodes of the graph can be labeled and its properties can be used as indexes. This feature improves the performance of cypher queries when graph size and nodes increases. It makes queries faster and easy.

2.5 Locality Sensitive Hashing

The nearest neighbor search algorithm perform well for low-dimensional data, but when the dimension of data goes high in that case time and space complexity increases exponentially. P. Indyk and R. Motwani [32] proved that for sufficient high dimension data, all the existing partitioning scheme demotes to linear search. For the nearest neighbor search in recent year several tree-based indexing schemes have been proposed, such as SR-tree [36], Cover-tree [34], KD-tree [38], R-tree [35], Navigating tree [33] etc.

but this indexing scheme performed poor when the data goes high and it has been proven that for high dimension data these scheme show only small improvement over linear search.

Indyk and Motwani proposed Locality Sensitive Hashing (LSH) based on high-dimensional approximate similarity search scheme. LSH is linearly dependent on the dataset size. Instead of reducing the probability of collision, this method uses several hashing function to hash the object to increase the probability of collision for similar objects. Then nearest neighbors can be determined by hashing the object and retrieving those elements which are stored in the same buckets. The disadvantage of LSH is that it is fast and simple only when the points are available in the binary Hamming space.

In [37] an improvement in the running time of the earlier algorithm has been purposed for the L2 norm which results in an efficient Approximate Nearest Neighbor scheme (ANN) for the case Lp norm ($p<1$). It takes $O(\log(n))$ to search exact near neighbor for data having growth boundary conditions. The proposed scheme gives 40 times faster results than kd-tree. In [39] Gan *et al.* proposed an LSH algorithm, Collision Counting LSH (C2LSH) that uses a m-base LSH functions to form dynamic compound hash functions instead of traditionally used static compound hash function. Broder *et al.* [40] designed an LSH scheme based on Jaccard similarity of sets which uses Minhash functions to take different permutations to calculate Jaccard similarity for sets. In [41], Panigrahy *et al.* proposed an LSH scheme which is entropy-based. It minimizes the number of hash tables required, by looking up a number of query offsets in addition to the query itself. In [42] Dasgupta *et al.* proposed a technique to improve the calculation of Euclidean distance of d-dimensional data by using randomized Hadamard transforms. Hua *et al.* in [43] proposed an algorithm LSBF which improves approximate membership query. This algorithm uses Locality sensitive hashing functions instead of uniform and independent hash functions in bloom filter.

Randomize algorithms such as LSH is widely used in nearest neighbor or similarity searching in application like recommender system [44], entity resolution [45] and image searching [46]. Recently, some of the work has been done to make real time RS using LSH [47]. Abbar *et al.* proposed a real time RS for diverse related article [48]. Authors of

[49] proposed the top N matrix factorization recommendation in social stream. Moreover, LSH is used to make efficient news RS [44].

Chapter 3

Problem Statement

User-based Collaborative-filtering (CF) technique is the most frequently used recommender technique, widely used because of its simplicity and efficient performance. With this technique system is capable of finding the user who has the similar taste and based on the similar user, system creates the rated list of recommendation which is known as user-based recommendation.

3.1 Gap Analysis

User-based Collaborative-filtering uses the matrix to store the ratings of the user. Although it is extensively used, one of its major problems is that its performance decreases when the user-item matrix becomes sparse. For the recommendation, User-based CF requires searching of the nearest neighbour. There are many approaches available for the nearest neighbor search but these approaches perform well for low-dimensional data, when the dimension of data goes high in that case time and space complexity increases exponentially. This proposed work provides a novel technique to overcome sparsity and search complexity by the using the combination of graph data base and with locality sensitive hashing (LSH).

3.2 Objective

The main objectives of the thesis are:

- To study, analyze and explore the various techniques of the recommender system.
- To propose a novel scheme for improving the accuracy of user-item rating matrix of a recommender system.
- To test and validate the proposed system and compare the result with the existing system.

3.3 Methodology

Major aim of the work is to reduce the possibility of sparsity and time complexity of searching the nearest neighbor. Methodology followed is:

- Graph Data base is used to reduce the problem of sparsity in user-item rating matrix system so that accuracy can be improved.
- Similarity index of node in the graph is calculated using Jaccard similarity.
- Locality Sensitive Hashing can be used for finding nearest neighbor in the database so that time complexity of searching is reduced.

Chapter 4
Proposed Methodology

4.1 Introduction

In the proposed system, the graph database and locality sensitive hashing (LSH) have been aggregate with the user-based collaborative filtering to provide the accurate recommendation to user. LSH is the method which is widely used for the finding the similar object. The main idea of this method is to select the object and hash in such way that object which are similar are likely to have in the same bucket. Figure 4.1 shown the steps that are followed in the proposed system:

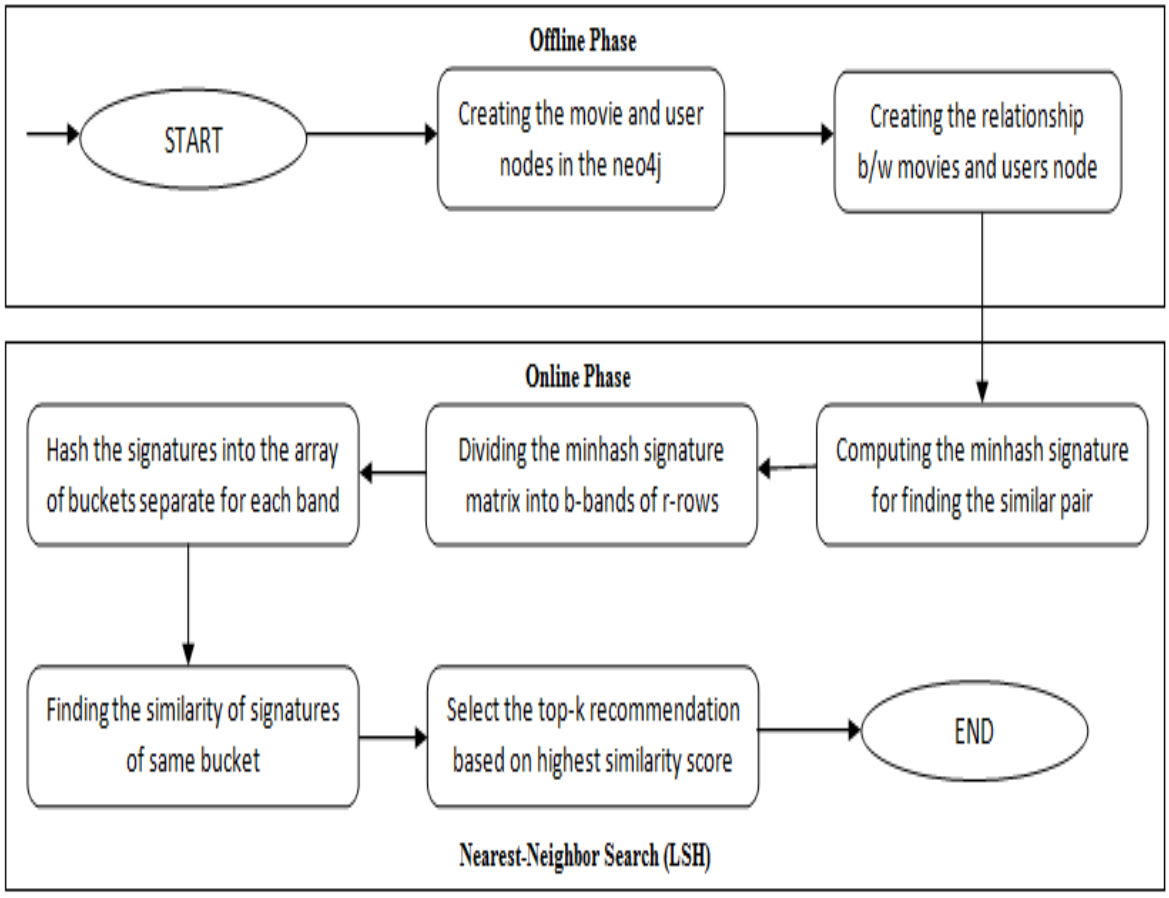

Figure 4.1: Flow chart of the proposed system

4.2 Creation of the Database and Cypher Query

Unlike the relational database there is no special convention to design graph database. Graph database provide the flexibility to developer to design database and unlike the relational database there is no need to normalize the database. In case of movies recommendation system node and the relationship has been created with the property.

Nodes:

The Nodes of the Graph database represent entity.

- The Movie Node
- The User Node

Relationship:

There are two relationships.

- Rate (from user to movies)
- Similarity (from users to users)

Property:

The Movies nodes have following properties:

- Title
- Year of release
- Imdb rating

The User node has only one property:

- Name

The Rate relationship has only one property:

- Rate

Cypher is the query language of the Neo4j Database. It is the analytical language that allows developer to read and write on the graph database. To create multiple movie and user node, one common method is the use of spreadsheet or the *.csv file. One can directly import the data set through *.csv. This is very useful in case of the big data set.

Query for creating the Movie Node:

MERGE (: Movie {Title: 'Undisputed', Release:'2002', Imdb_rate:'7.5'});

Query for creating the User Node:

MERGE (:User{User_id:'john'});

Query for creating a relationship between User and Movie nodes:

MATCH (u1:User), (m1:Movie) WHERE u1.User_Name='john' AND m1.title='Undisputed' CREATE (u1)- [:RATED{rate:7}]->(m1);

Snapshot of the graph data base for the movie recommendation system has been show in the figure:

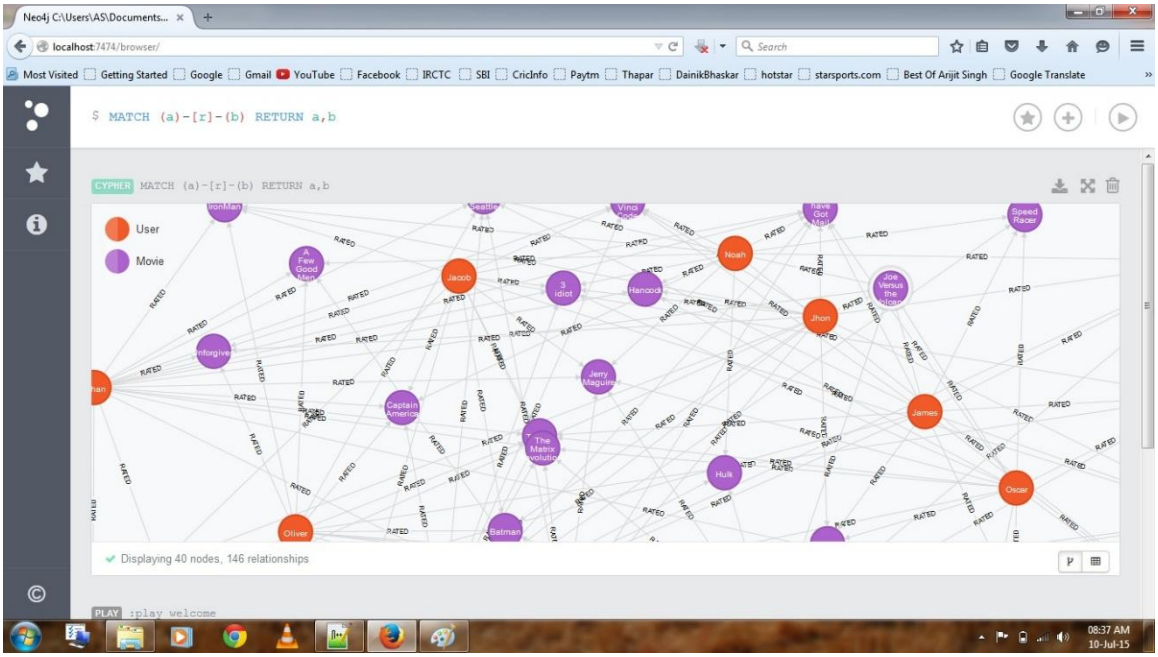

Figure 4.2: Created graph database

4.3 Finding the Minhash Signature

The next step in the proposed system is to find the minhash signature of the user *U* set. Minhash signatures reduce the dimension of data so that they can be compared quickly for similarity. Algorithm 1 is used to compute the minhash signature for finding the similar user.

Algorithm 1 Minhash Signature

Input: Movies nodes 'm_i' and users node 'u_i' in the graph

for all u_i rating movie m_i **do**

Create relationship between movie and user

end for

Output Minhash Signatures

1. Compute $h_1(r)$, $h_2(m)$……$h_n(r)$
2. **for** each row r **do**
3. **for** each column c **do**
4. **if** c has not 1 any row r **then**
5. Do nothing
6. **end if**
7. **else if** c has 1 in row r **then**
8. **for** each hash function h_i **do**
9. **if** $h_i(r)$ is smaller value than M(i, c) **then**
10. M(i, c) = $h_i(r)$
11. **end if**
12. **end for**
13. **end if**
14. **end for**
15. **end for**

Algorithm 1: Minhash Signature

Even though minhash signature can be used to find the similarity of any user, but it may be impossible to find similar user with greatest similarity efficiently. So the next step in the proposed system is to find the similarity of neighbors using LSH.

4.4 LSH for Finding the Nearest Neighbor

The first step is to divide the minhash signature into b-bands of r-rows such that b*r = n and create the hash table Ht_j j = 1, ..., b for each b-bands. Now a hash family HF is chosen. Each hash table belongs to one hash function h_jj = 1, ..., b. By using the hash function h_j initialize the hash table Ht_jj = 1, ..., b. It is natural that several users fall into the same hash table bucket. Proposed Algorithm 2 hash the signature into the array of bucket separate of each bands.

Algorithm 2 Prepossessing using LSH

Input A set of minhash signatures

b (Number of bands)

Output Hash table Ht_j , j = 1, …, b

1. Divide the n minhash into b-band of size r-row such that b*r = n
2. **for** each band j = 1, …, b **do**
3. Create a hash table ht_j /* family of hash function HF = {$h_1(r)$, $h_2(r)$, ..., $h_n(r)$}*/
4. **end for**
5. **for** each j = 1, …, b **do /*** b is the number of band ***/**
6. Initialize hash table ht_j by applying a hash function h_j
7. **end for**
8. **for** each j = 1, …, b **do**
9. **for** each i = 1, …, n **do**
10. Store Signatures into the bucket $B_j(S_i)$ of hash table ht_j
11. **end for**
12. **end for**

Algorithm 2: Prepossessing using LSH

In the user-based collaborative filtering, recommendations to the users are generated from the similarity of most nearest user by measuring the similarity between users. More generally, let r_{um} is the rating given by the user 'u' to movie 'm'. Let *U* – be the set of all user, *M* – the set of all movies, M_u - is the set of all movies rated by user 'u'. M_{uv} - the set

of movies rated by both user ‘u’ and user ‘v’. Usually, User rates relatively small number of movies. $|M_u| << |M|$. User-based collaborative filtering is based on the some similarity measure (*sim(u, v)*) between users which is calculated based on some common rating and unknown rating *r** based on known rating *r*.

Various similarity measures have been used for recommender system including Cosine Similarity, Euclidean distance, Pearson correlation coefficient, Hamming distance *etc*.

Cosine similarity has been used for similarity measure between users. The cosine similarity is a measure of similarity between two vectors. It is the cosine of the angle between two n-dimensional vectors. The result ranges between -1 and 1.

$$Sim(u.\ v) = \frac{\sum_{M_{uv}} r_{um} r_{vm}}{\sqrt{\sum_{M_{uv}} r_{um}^2} \sqrt{\sum_{M_{uv}} r_{vm}^2}} \quad (6)$$

Recommender system using locality sensitive hashing used the nearest neighbor approach. For the user ‘u’ it check the respective hash table Ht_j j = 1, ..., b and select all the user of the same bucket and compute the similarity of the user whose interest is similar to user ‘u’. Proposed Algorithm 3 returns the m nearest neighbor for user ‘u’ from S based on the highest similarity score.

Algorithm 3 m- nearest neighbor using LSH

Input User ‘u’ for which recommendation to be generate

M (number of nearest neighbor)

Access To hash table Ht_j, j = 1, …, b generated by algorithm 2

Output M (or less) nearest neighbor

1. *S* <- ∅ /* Set of Similar Signature*/
2. **for** each j = 1, …, b **do**
3. *S* <- *S*∪**{**Similar signature found in b_j bucket of hash table Ht_j **}**
4. **end for**
5. **for** all signature of the *S* **do**
6. Examine the similarity between signatures
7. **end for**
8. Return m-nearest neighbor of ‘u’ from in *S* based on the highest similarity score

Chapter 5
Implementation and Results

5.1 Results

For the experiment, two types of RS have been used. First one is the recommender system with relational database (RDB) which was implemented by MySql [50], and second one is the recommender system with graph database (GDB) which was implemented by Neo4j [51]. All the experiments were performed on Window 7 enterprise edition, Intel® Core™ i5 processor with the clock speed 2.90 GHz, and 4GB RAM. For the experiment, data set publicly provided by MovieLens, a well known movie recommendation website has been used [52]. Based on this data set recommendation are generated for varying number of movies and users as shown in Table I.

TABLE I: DATA SET

Types of data / Size of data	**Users**	**Movies**	**Ratings**
1000	700	300	10,138
2000	1500	500	55,023
4000	3000	1000	85,503
6000	4000	2000	356,004
8000	5000	3000	756,003

The execution time for searching the item and creating the list of recommendation is measured to evaluate the performance of query for varying datasets. The experiment result for both types of databases is shown in figure 3.

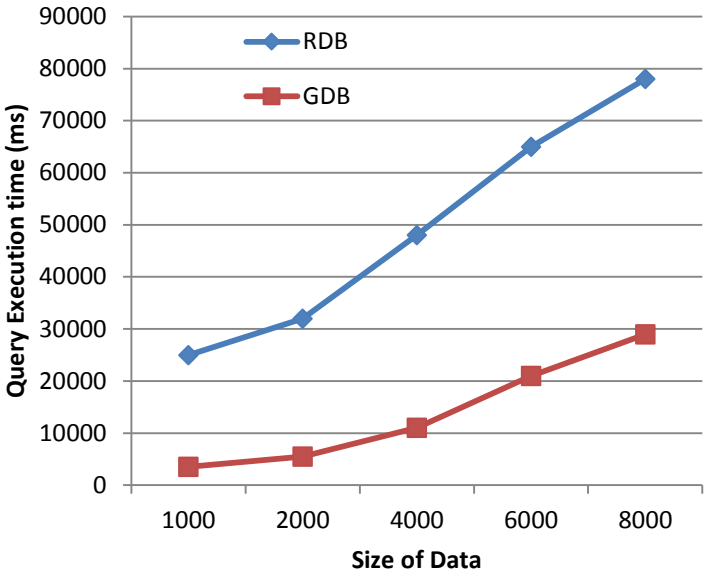

Figure 5.1: Comparison of execution time for various data sizes

It has been rightly said that recommendation is a popular technique used for separating the wheat from the chaff. None can deny the fact that graph databases can scale more naturally to large data sets and to datasets with changing or on-the-fly schemas as compared to relational data bases. It has been experimentally examined that data retrieval is quite fast and proficient when graph data bases are used in recommender system.

5.2 Illustration of implementation steps

The steps follow in the system are:

1. After the creation of Graph database for the movie recommendation the next step is to compute the similarity between users. To find the similarity Jaccard similarity is applied on the relationship between the pair of users of the graph, for movie recommendation purposes this is defined as the intersection of their sets of movie divided by the union of their sets of movie. This results in a score between 0 and 1 representing how “similar” the two users are to each other. Figure 5.2 show the similarity relationship created between users.

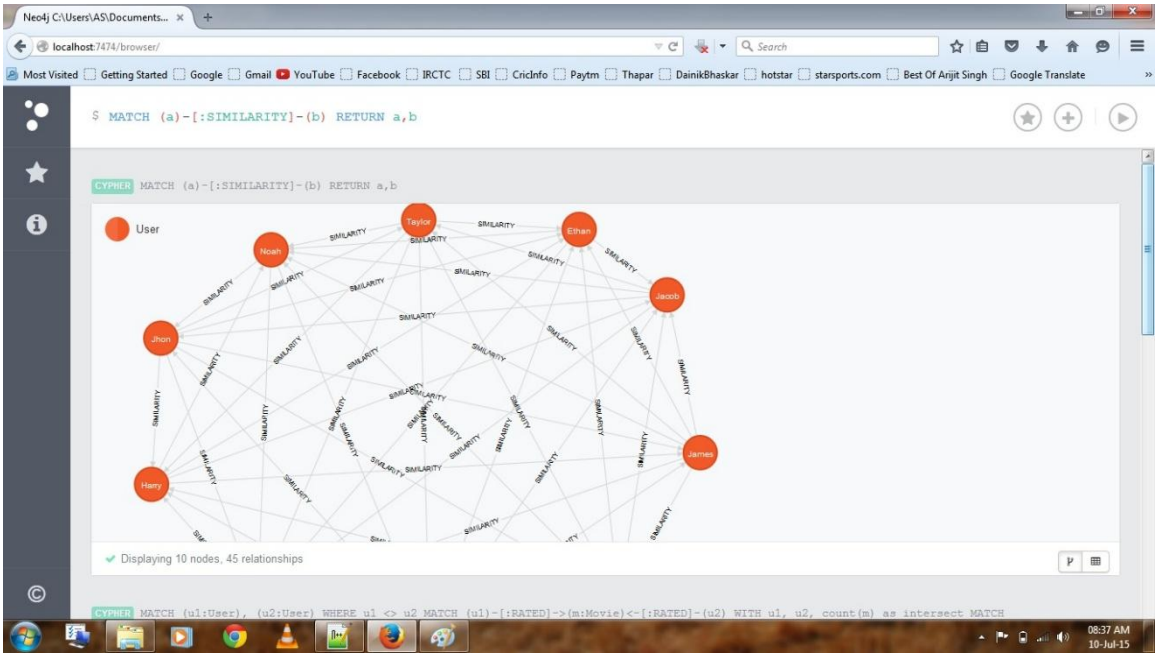

Figure 5.2: Computation of similarity between users

2. Now the next step is to find the nearest neighbor for the specific user by using the similarity value. Figure 5.3 show the most nearest neighbor of the user ‘Ashish’ with the similarity value.

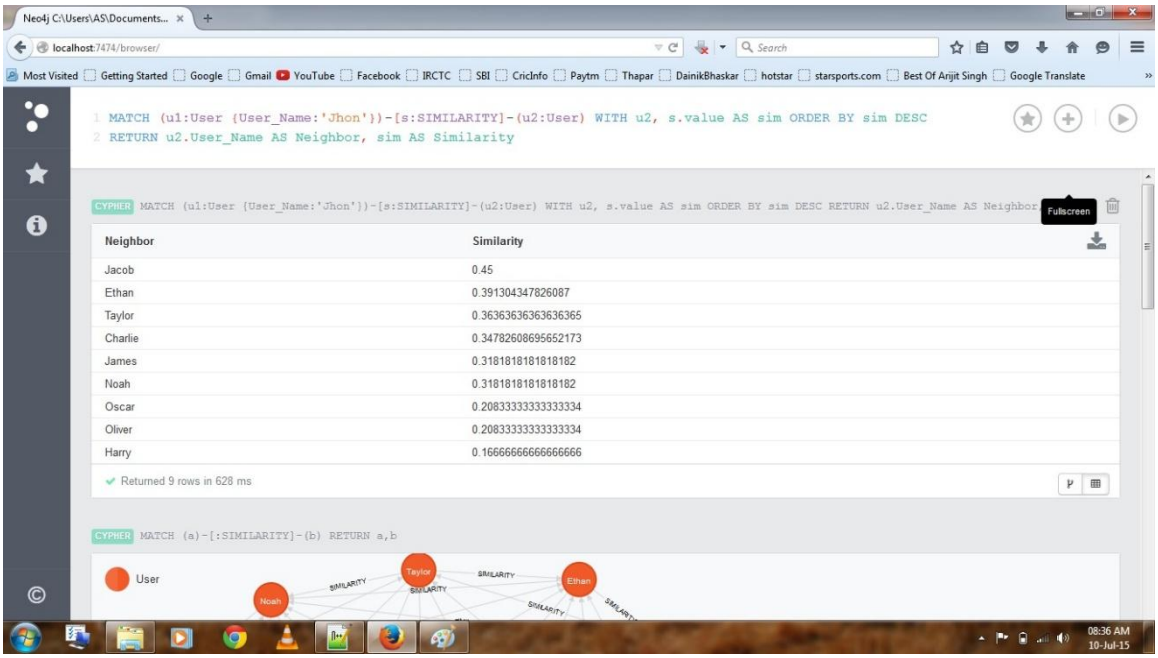

Figure 5.3: Nearest neighbor of specific user

3. In the next step, select the top-k users which are similar to ‘Ashish’. Those movies which are watched by nearest neighbors and that are not seen by ‘Ashish’ will be return in this step.

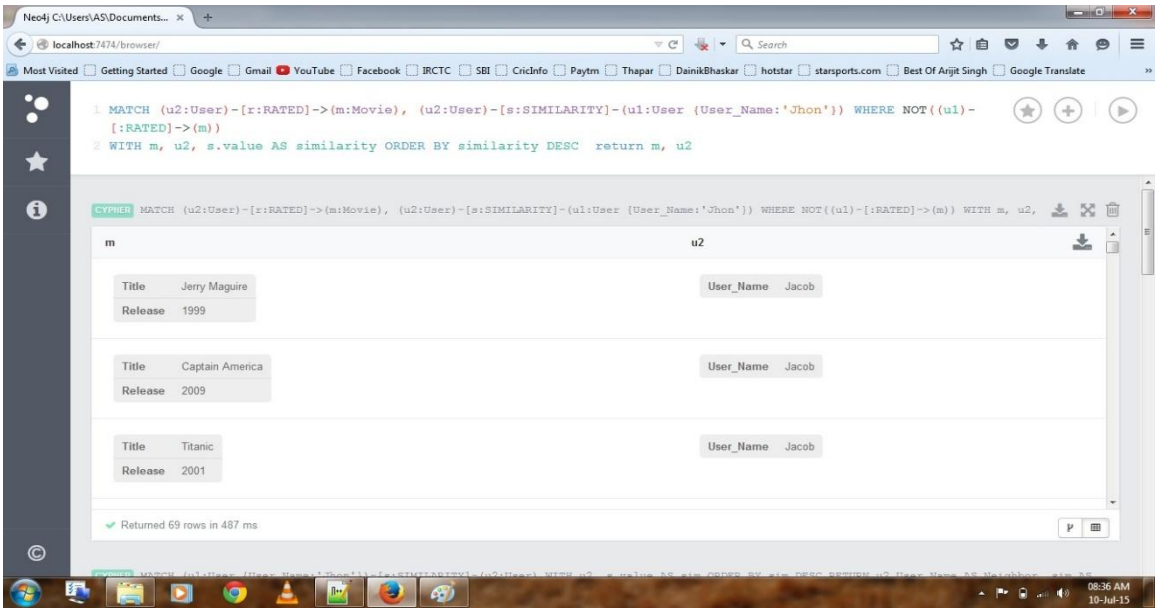

Figure 5.4: Movies which are not seen by ‘Ashish’

4. The above step return movie watched by each user but it will be more useful to create the group of movies.

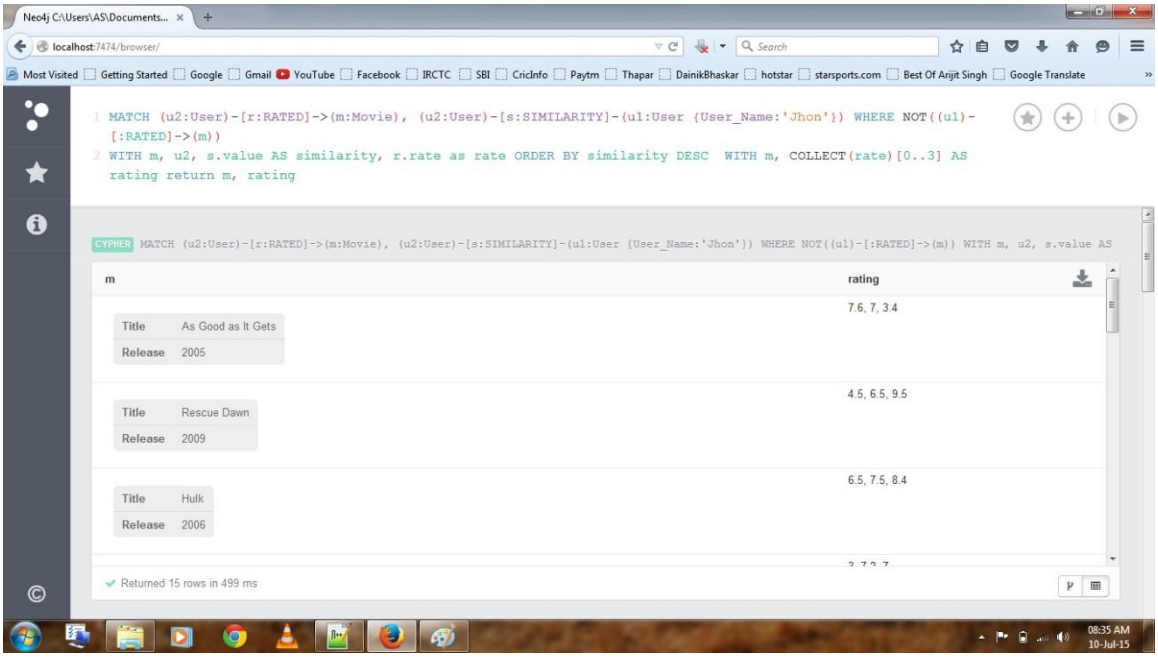

Figure 5.5: Mean ratings of movies given by users

5. Next step determines the mean of all the ratings and arranges mean in descending order. So final movie recommendation for the user ‘Ashish’ is shown in the figure 5.6

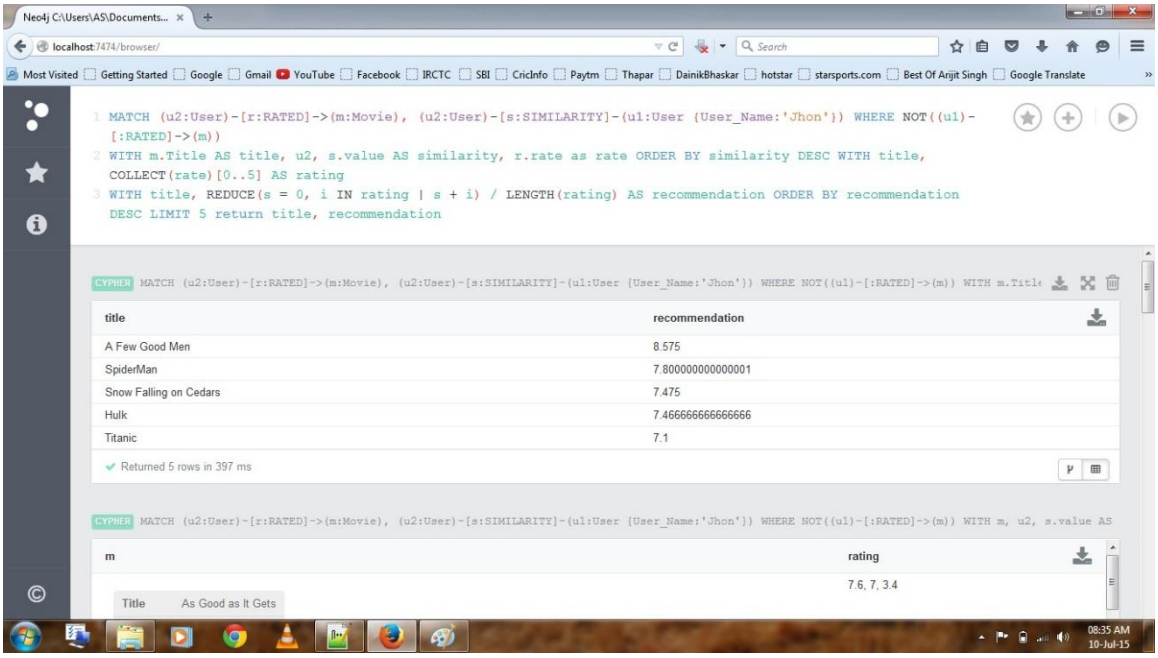

Figure 5.6: Final Movie Recommendation

It looks like that the nearest neighbors liked the movie “hoffa” and rated it with the score 9.5. This movie is most likely to please the user Ashish.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

With the ever increasing e-commerce data, storing and querying massive data is becoming a big challenge for recommender systems. Recent studies on recommender systems indicate that graph data model is more efficient than relational data model for processing complex data. This work proposed a new graph data storage model for the collaborative filtering-based recommender system which includes resolving the issue of sparse data processing by using a probabilistic data structure locality sensitive hashing. The experimental results show that graph database approach is definitely efficient and effective for recommender system.

6.2 Summary of Contribution

The proposed system put forward solution to most prevalent problem of the recommender system and can provides the accurate result to the user. This system prepare a hybrid combination of graph database and probabilistic data structure with user based collaborative filtering which tries to remove sparsity problem and reduce the searching complexity which single system is unable to deal with. This improvement can be useful in improving the recommender systems in various areas of application and help in generating more accurate results which definitely provide ease to the user in choosing the items.

6.3 Future Scope

One most important dimension which needs a thorough thought is how to include multiple dimensions in the recommender system. Although multi dimension input will increase the complexity but recommendations will definitely be refined by usage of multiple dimensions. Further need is to find how this multi dimensionality can be provided in graph databases without decreasing the search accuracy.

References

[1] M. P. O. Mahony and B. Smyth, “A Recommender System for On-line Course Enrolment: An Initial Study,” pp. 133–136.

[2] J. Bobadilla, F. Serradilla, and a. Hernando, “Collaborative filtering adapted to recommender systems of e-learning,” *Knowledge-Based Syst.*, vol. 22, no. 4, pp. 261–265, May 2009.

[3] J. Ben Schafer, J. Konstan, and J. Riedl, “Recommender Systems in E-Commerce,” pp. 158–166, 1999 in *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158-166. ACM, 1999.

[4] F Ricci, L Rokach, B Shapira, “Introduction to recommender systems handbook”, Springer US, 2011.

[5] R. Burke, “Hybrid Web Recommender Systems,” pp. 377–408, Springer Berlin Heidelberg, 2007.

[6] J. Lee, M. Sun, and G. Lebanon, “A comparative study of collaborative filtering algorithms,” *arXiv Prepr. arXiv1205.3193*, pp. 1–27, 2012

[7] Francesco Ricci, Lior Rokach and Bracha Shapira, “Content-based recommender systems: state of the art and trends, recommender systems handbook”, Springer, 2011, pp. 73-105

[8] M.J. Pazzani, "A framework for collaborative, content-based and demographic filtering." in *Artificial Intelligence Review* 13, no. 5-6, pp. 393- 408, 1999.

[9] R. Burke. "Knowledge-based recommender systems," in *Encyclopedia of library and information systems* 69, no. Supplement 32, pp. 175-186, 2000.

[10] Robin Burke, **“**Hybrid Recommender Systems: Survey and Experiments”, in *User Modeling and User-Adapted Interaction*, November 2002, Volume 12, Issue 4, pp 331-370

[11] Goldberg D, Nichols D, and Oki M B. Using collaborative filtering to weave an information Tapestry, Dec 1992, vol. 35, Issue 12, pp 61-70.

[12] Adomavicius, G Tuzhilin, A. (June 2005). "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible

Extensions". in *IEEE Transactions on Knowledge and Data Engineering 17* (6): 734–749.

[13] The Neo4j Manual, "Powering recommendations with a graph database", *http://neo4j.com/wp-recommendations/*.

[14] "LSH and General hashing" [online] available at http://cybertron.cg.tuberlin.de/pdci08/imageflight/nn_search.html, last accessed on 15th April, 2015.

[15] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions", Comm. ACM, pp. 117-122, 2008.

[16] A. Rajaraman, J. Ullman, "Mining of Massive Datasets", Cambridge University Press, December 30, 2011.

[17] O'Mahony, Michael P., and Barry Smyth. "A recommender system for on-line course enrolment: an initial study." in *Proceedings of the 2007 ACM conference on Recommender systems*, pp. 133-136. ACM, 2007.

[18] Schafer, J. Ben, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. in *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158-166. ACM, 1999.

[19] U. Shardanandand and P. Maes, "Social information filtering: algorithms for automating "word of mouth," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co., pp. 210-217, 1995.

[20] Changrui Yu, Yan Luo and Kecheng Liu. "A Multi-attribute collaborative filtering recommendation algorithm based on improved group decision-making," Volume 426, 2014, pp 320-330.

[21] Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43-52. Morgan Kaufmann Publishers Inc., 1998

[22] Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: an open architecture for collaborative filtering of netnews," in

Proceedings of the1994 ACM conference on Computer supported cooperative work, pp. 175-186. ACM, 1994

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study,*"* No. TR-00-043. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[24] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithmsm," In *Proceedings of the 10th international conference on World Wide Web*, ACM, pp. 285-295, 2001.

[25] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Adv. Artif. Intell.*, vol. 2009, no. Section 3, pp. 1–19, 2009

[26] D. Lemire and A. Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering." In *SDM*, vol. 5, pp. 1-5. 2005.

[27] Y. Li, L. Lu, and L. Xuefeng, "A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in ECommerce." in *Expert Systems with Applications* 28, no. 1, pp. 67-77, 2005.

[28] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends." in *Recommender Systems Handbook*, Springer US, pp. 73-105, 2011.

[29] R. Giugno and D. Shasha, "GraphGrep: A fast and universal method for querying graphs",2002, pp. 112 – 115

[30] Batul J. Mirza, Benjamin J. Keller, and Naren Ramakrishnan, "Studying recommendation algorithms by graph analysis", ACM-2003, Volume 20, Issue 2, pp 131-160

[31] Chad Vicknai, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen and Dawn Wilkins, "A comparison of a graph database and a relational database", ACMSE '10, April 15-17, 2010, Oxford, MS, USA.

[32] P. Indyk and R. Motwani, "Approximate nearest neighbor: towards removing the curse of dimensionality", Proc. Symposium on Theory of Computing, 1998.

[33] R. Krauthgamer, and J. Lee, "Navigating nets: simple algorithms for proximity search", in *Proc. of 15th annual ACM-SIAM symposium on Discrete algorithms, Philadelphia,* PA, USA, pp. 798-807, 2004.

[34] A. Beygelzimer, S. Kakade, and J. Langford, “Cover trees for nearest neighbours”, ICML, ACM, New York, USA, pp. 97-104, 2006.

[35] A. Guttman, “R-trees: A dynamic index structure for spatial searching”, SIGMOD, pp.47-57, 1984.

[36] N. Katayama, and S. Satoh, “The sr-tree: an index structure for high-dimensional nearest neighbor queries”, SIGMOD, pp. 369-380, 1997.

[37] M. Datar et al., “Locality-sensitive hashing scheme based on p-stable distributions”, in *Proc. ACM Symposium on Computational Geometry,* 2004.

[38] J.L. Bentley, “Multidimensional binary search trees used for associative searching”, Comm. ACM, pp. 509-517, 1975.

[39] G. Junhao et al., “Locality-Sensitive Hashing Scheme Based on Dynamic Collision Counting”, ACM, 2012.

[40] A. Broder et al., "Min-wise independent permutations", in *Proc. Theory of computing, ACM Symposium,* New York, USA, pp. 327-336, 1998.

[41] R. Panigrahy, “Entropy-based nearest neighbor algorithm in high dimensions”, in *Proc.ACM-SIAM Symposium on Discrete Algorithms,* ACM, New York, USA, pp. 1186-1195, 2006.

[42] A. Dasgupta, R. Kumar, and T. Sarlós, “Fast Locality-Sensitive Hashing”, ACM conference, New York, USA, pp. 1073-1081, 2011.

[43] Y. Hua et al., "Locality-Sensitive Bloom Filter for Approximate Membership Query", in *IEEE Transactions on Computers*, vol.61, no.6, pp. 817-830, June 2012.

[44] L. Li, D. Wang, T. Li, D. Knox and B. Padmanabhan, “Scene: a scalable two-stage personalized news recommender system”, *in* ‘SIGIR-2011’,pp. 125–134.

[45] H. Liang, Y. Wang, P. Christen and R. Gayler, “Noise-tolerant approximate blocking for dynamic real-time entity resolution”, *in* ‘PAKDD-2014’,pp. 449–460.

[46] W. Dong, Z. Wang, W. Josephson, M. Charikar and K.Li,“Modeling lsh for performance tuning”, *in* ‘CIKM-2008’, ACM, pp. 669–678.

[47] B. Chandramouli, J. Levandoski, A. Eldawy, and M. Mokbel, “Streamrec: A real-time recommender System”, *in* ‘*SIGMOD-2011’,* pp. 1243–1246.

[48] S. Abbar, S. Amer-Yahia, P.Indyk, and S. Mahabadi, "Real-time recommendation of diverse related articles", *in* '*WWW-2103'*, pp. 1–12.

[49] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W, Nejdl, "Real-time top-n recommendation in social streams", *in* '*RecSys-2012'*, pp. 59–66.

[50] MySQL, http://www.mysql.com/.

[51] Neo4j, http://www.neo4j.com/.

[52] MovieLens, https://movielens.org/

List of Publications and Video Link

- **Publications**

  Accepted

A. Sharma and S. Batra, "**Enhancing the Accuracy of Movie Recommendation System Based on Probabilistic Data structure and Graph Database"** In 5th International Conference on Advances in Computing and Communications (ICACC-2015) **IEEE**, 2015.

- **Video Link**

http://youtu.be/gKRNkGogqpU