TACKLING THE PROBLEMS OF DIVERSITY IN RECOMMENDER SYSTEMS

by

MANIKANTA BABU KARANAM

B.E., Osmania University, India, 2008

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:

Major Professor
Dr. William H. Hsu

Abstract

A recommender system is a computational mechanism for information filtering, where users provide recommendations (in the form of ratings or selecting items) as inputs, which the system then aggregates and directs to appropriate recipients. With the advent of web based media and publicity methods, the age where standardized methods of publicity, sales, production and marketing strategies do not. As such, in many markets the users are given a wide range of products and information to choose which product they like, to find a way out of this recommender systems are used in a way similar to the live social scenario, that is a user tries to get reviews from friends before opting for a product in a similar way recommender system tries to be a friend who recommends the options.

Most of the recommender systems currently developed solely accuracy driven, *i.e.*, reducing the Mean Absolute Error (MAE) between the predictions of the recommender system and actual ratings of the user. This leads to various problems for recommender systems such as lack of diversity and freshness. Lack of diversity arises when the recommender system is overly focused on accuracy by recommending a set of items, in which all of the items are too similar to each other, because they are predicted to be liked by the user. Lack of freshness also arises with overly focusing on accuracy but as a limitation on the set of items recommended making it overly predictable.

This thesis work is directed at addressing the issues of diversity, by developing an approach, where a threshold of accuracy (in terms of Mean Absolute Error in prediction) is maintained while trying to diversify the set of item recommendations. Here for the problem of diversity a combination of Attribute-based diversification and user preference based diversification is done. This approach is then evaluated using non-classical methods along with evaluating the base recommender algorithm to prove that diversification is indeed is possible with a mixture of collaborative and content based approach.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my major professor, Dr. William H. Hsu for his guidance, suggestions, comments, patience and encouragement in the completion of the thesis work. I owe him my gratitude towards the completion of my thesis work.

I would also like to thank my Dr. Doina Caragea for guiding me through the initial phases of the thesis and also for commenting and suggesting changes during the course of the thesis work.

I also would like to thank Dr. Gurdip Singh for agreeing to serve in my committee and also, for guiding and suggesting in both his courses and my thesis.

1 Introduction and Background

1.1 Introduction

'It is often necessary to make choices without sufficient experiences of the alternatives. As such, a typical recommender system tries to assist users in this process of choosing alternatives, where the user is still naïve in the current scenario'. [35] A recommender system is a computational mechanism for information filtering, where users provide recommendations (in the form of ratings or selecting items) as inputs, which the system then aggregates and directs to appropriate recipients. With the advent of web based media and publicity methods, in many markets the users are given a wide range of products and information to choose which product they like, to find a way out of this recommender systems are used in a way similar to the live social scenario, that is a user tries to get reviews from friends before opting for a product in a similar way recommender system tries to be a friend who recommends the options [35].

As *Chris Anderson* mentions in his 'A Long Tail', 'we are leaving the age of information and entering the age of recommendation'. In today's world, the age where standardized methods of publicity, production and marketing strategies do not work because of a large section of users who prefer rated, reviewed and recommended products; As such, in many markets the users are given a wide range of products and information to choose which from, to find a way out of this; recommender systems are used in a way similar to the live social scenario, that is a user tries to get reviews from friends before opting for a product in a similar way recommender system tries to be a friend who recommends the options.

The above mentioned process is a sequential process and early recommender systems were thought to be a problem of sequential prediction, but further development and research led to the stage where they are considered to be an iterative decision problem.

It has been more than a decade since the first papers on recommender systems have been published ([35], [18]) and there have been many successful systems implemented in one's daily life. They have seen a great deal of commercial success (for example *Amazon* and *Netflix*). Even then, the area of recommender system has a lot of research problems left to deal, with lots of improvements still needed since; they are still far from perfect, and face tremendous challenges in increasing the overall utility of the recommendations. These challenges are present in all stages of the recommendation process. They begin with the *cold start* problem: given a new user

how do we recommend items to the user without forcing her to give feedback on a starter set of items [47]. Similarly, given a new item introduced into the system, how do we know when (and to whom) we should recommend it?

This thesis focuses on the development of a diversity-boosting component of a recommender system - one that returns a set of items that are not to similar to one another. This component extends and is used in tandem with an algorithmic approach to collaborative recommendation: one that performs partitioning-based clustering of user ratings and applies classification-based prediction.

Information retrieval is a major part of this recommender system since the items considered here are movies and they are described with the use of text documents. To measure similarities between textual documents there have been many information retrieval techniques developed but a relatively new technique, Latent Dirichlet Allocation is used. Collaborative-filtering is one of the most successful techniques used in the construction of a recommender system. This approach requires the system to find a set of users similar to the active user, to accomplish this task portioning-based clustering (K-Means clustering) is used.

1.2.Background

The research in this field of recommender systems has been a long one; the first recommender system [14] was developed in 1992, and they introduced the term collaborative filtering. As such, for almost two decades there has been a significant increase in the techniques used for recommending. Many popular algorithms have been proposed to solve the ever evolving problem of recommending.

In Table 1-2, some of the popular techniques used for developing recommender systems are listed and a brief description of the background, input and the process used for each of the techniques is documented [9]. Also, in Figure 1-1, a graphical representation of the techniques is given.

Terms	Notations
Users: People who use the system to provide reviews or feedback, using which the system tries to recommend to an unknown new user.	u
Set of Users	U
Items: Things a system has to recommend.	i
Set of Items	I

Table 1-1: *Notations used in a recommender system*

Technique	Background	Input	Process
Collaborative	Ratings from **U** of item in **I**.	Ratings from **u** of items in **I**.	Identify users in **U** similar to **u**, and extrapolate from their ratings of **i**.
Content-based	Features of items in **I**.	**u**'s ratings of items in **I**	Generate a classifier that fits **u**'s rating behavior and use it on **i**.
Demographic	Demographic information about **U** and their ratings of items in **I**.	Demographic information about **u**.	Identify users that are demographically similar to **u**, and extrapolate from their ratings of **i**.
Utility-based	Features of items in **I**.	A utility function over items in **I** that describes **u**'s preferences.	Apply the function to the items and determine **i**'s rank.
Knowledge-based	Features of items in **I**. Knowledge of how these items meet a user's needs.	A description of **u**'s needs or interests.	Infer a match between **i** and **u**'s need.

Table 1-2: *Recommendation Techniques.* [9]

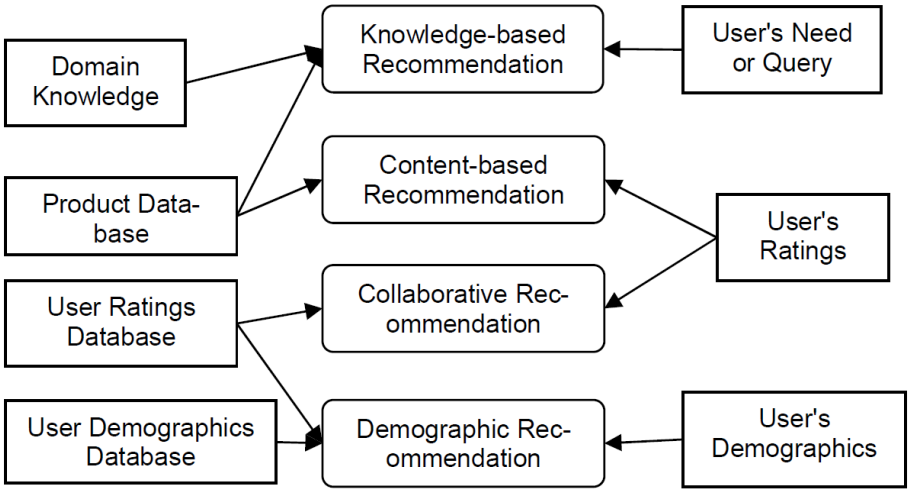

Figure 1-1: *Recommendation Techniques.* [9]

The approaches used in constructing a recommender system are classified into three main categories; they are content-based, collaborative filtering and hybrid approaches [3]. As, these form the major part of the work done in this thesis work, a brief understanding of these approaches is needed.

1.2.1 Content-based Approach

Content-based recommendation can be termed as an outgrowth and continuation of information filtering research [4]. In a content-based system, the objects of interest are defined by their associated features. For example, text recommendation systems like the newsgroup filtering system NewsWeeder [26] uses the words of their texts as features. A content-based recommender learns a profile of the user's interests based on the features of items the user has rated; and can be termed as 'item-to-item correlation' [39]. The type of user profile derived by a content-based recommender depends on the learning method employed. Decision trees, neural nets, and vector-based representations have all been used. As in the collaborative case, content-based user profiles are long-term models and updated as more evidence about user preferences is observed. Content-based approach is one where the recommendation is done solely based on the similarity between the items and the active users' preferences rather than considering the information which the other users provide. Here similarity is computed based on item attributes using appropriate distance measures. Each item is represented by a feature vector or an attribute

profile. The features in general might hold nominal values (topics, classes) or numeric values representing the defining aspects of the item. A variety of distance measures can be used over the feature vectors to calculate the similarity between two items. Some of the commonly used distance measures would be cosine similarity, TF-IDF, *etc.*[37]

1.2.2 Collaborative Filtering

Collaborative filtering (CF) is a process of filtering information using various techniques by the use of collaboration between many agents. In recommender systems, this can be termed an approach of filtering information (regarding the reviews, ratings, feedback of items by various users) [38]. 'Collaborative recommender systems compute item propositions by analyzing community data and similarities between users or items' [49]. The basic usage is done in an iterative process, in step one a set of similar users to the active (current) user are found and the information representing these users (user profiles) is considered. Finding the similarity between users can be done in many ways like considering the patterns of ratings, patterns of reviews or feedbacks and techniques like distance measures can be used to find the similarity between the users. After the initial set of similar users is found the information pertaining to those users is used to obtain the predictions on the items for the active user. To summarize this it can be said that the like-minded users have same preferences and capturing these preferences can help in alleviating the problem of recommender system. Many different techniques have been proposed for collaborative recommendation, including the most original methods [36], Bayesian learning [44], latent semantic indexing [22] and clustering.

Two types of algorithms for collaborative filtering have been studied: memory based and model-based.

1.2.2.1 Memory-Based

This mechanism uses user rating data to compute similarity between users or items, which is used for making recommendations [7]. It is easy to implement and is effective. The basic idea is to compute the active user's predicted vote of an item as a weighted average of votes by other similar users or *K* nearest neighbors (KNN) [45]. Two commonly used memory-based algorithms are the Pearson Correlation Coefficient (PCC) algorithm [36] and the Vector Space Similarity (VSS) algorithm [7]. These two approaches differ in the computation of similarity. The PCC algorithm generally achieves higher performance than vector-space

similarity method. The advantages with this approach are the ability to explain the results, which is an important aspect of recommendation systems. It is easy to create and use. New data can be added easily and incrementally. It need not consider the content of the items being recommended. The mechanism scales well with co-rated items. These approaches focused on utilizing the existing rating of a training user as the features. However, the memory-based method suffers from two fundamental problems: data sparseness and inability to scale up. Data sparseness refers to the difficulty that most users rate only a small number of items and hence a very sparse user-item rating matrix is available. As a result, the accuracy of the method is often quite poor. As for computational scalability, algorithms based on memory-based approaches often cannot cope well with the large numbers of users and items ([45], [7]).

1.2.2.2 Model-Based

In contrast, model-based algorithms build models that can explain the records of historic ratings well and predict the ratings for an active user using estimated models [23]. Two popular model-based algorithms are the clustering for collaborative filtering ([24], [44]) and the aspect models [20]. *Clustering techniques* work by identifying groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Some clustering techniques represent each user with partial participation in several clusters. The prediction is then an average across the clusters, weighted by the degree of participation. The *aspect model* [20] is a probabilistic latent-space model, which considers individual preferences as a convex combination of preference factors. The latent class variable is associated with each observation pair of a user and an item. The aspect model assumes that users and items are independent from each other given the latent class variable. There are many model based CF algorithms. These include Bayesian Networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, Multiple Multiplicative Factor, Latent Dirichlet allocation, markov decision process based models. This approach has a more holistic goal to uncover latent factors that explain observed ratings.

It handles the sparse data better than memory based ones. This helps with scalability with large data sets. It improves the prediction performance. It gives an intuitive rationale for the recommendations. The disadvantage with this approach is the expensive model building. One needs to have a tradeoff between prediction performance and scalability. One can lose useful

information due to reduction models. A number of models have difficulty explaining the predictions [45].

1.2.3 Hybrid Approach

The term *hybrid recommender system* is used here to describe any recommender system that combines multiple recommendation techniques together to produce its output. There is no reason why several different techniques of the same type could not be hybridized, for example, two different content-based recommenders could work together, and a number of projects have investigated this type of hybrid, [5] which uses both naive Bayes and kNN classifiers in its news recommendations, is just one example.

The earlier survey of hybrids [9] identified seven different types of hybrid approaches:

i. *Weighted:* The score of different recommendation components are combined numerically.
ii. *Switching:* The system chooses among recommendation components and applies the selected one.
iii. *Mixed:* Recommendations from different recommenders are presented together.
iv. *Feature Combination:* Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
v. *Feature Augmentation:* One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
vi. *Cascade:* Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
vii. *Meta-level:* One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

1.2.4 Open Research Problems

In this section some of the open research problems are discussed, these are the major problems which have been documented from the literature review ([2], [47], [28], [32], [40], [43], [31]), which was done for the background information and also in understanding the state-of-the-art recommender systems. The major reasoning behind such open problems is in most cases the recommender algorithms are accuracy driven and this can in turn hurt the recommender system itself [28].

1.2.4.1 Overconcentration / diversity issues

Recommender systems in most cases are overly focused on accuracy and this scenario is called as overconcentration [2]. Focusing solely on accuracy drives the recommender system to recommended items which have high predicted rating, which can raise the scenario where the items might be similar to each other in terms item-attributes. This scenario can limit the recommendation item space. An example given by the author (Amer-Yahia, Lakshmanan, Vassilvitskii, & Yu, 2009), in a movie recommender system if a user X has rated in favor of movies in fantasy genre previously, then an accuracy driven recommendation system, tries to be sure to recommend movies from fantasy genre which user X might enjoy but the system doesn't take risks to predict if the user X might try some movie from other genres. The result of never taking risks might make the recommender system homogenous and boring; this can result in user X losing interest in the system itself. 'The goal of *recommendation diversification* is to identify a list of items that are dissimilar with each other, but nonetheless relevant to the user's interests' [47].

1.2.4.2 Accuracy / freshness / relativity

A different but subtle problem which is not addressed usually is that of freshness [2]. Lack of freshness also arises with overly focusing on accuracy but as a limitation on the set of items recommended making it overly predictable. In (Amer-Yahia, Lakshmanan, Vassilvitskii, & Yu, 2009) the author suggests that even if the set of items recommended are diverse, it still might lack freshness, to support this claim a simple example is given, a system that only recommends blockbuster movies from every genre. While such a system would likely have high accuracy scores (these items are blockbusters for a reason), and the set of recommended items is diverse (since recommendations come from many different genres), such a system would rarely be useful.

1.2.4.3 Handling non-static data

Sometimes the data of items or user profiles might be changing; the items may have been upgraded or they might get more information, this is one problem where most recommender systems are trained for a set of static item set, and recommend using those, but cannot handle new data continuously [43]. Dealing with changing user preferences can be a big head-ache for a recommender system which keeps track of user profiles. User preferences for products are

drifting over time. Product perception and popularity are constantly changing as new selection emerges. Similarly, customer inclinations are evolving, leading them to ever redefine their taste. Thus, modeling temporal dynamics should be a key when designing recommender systems or general customer preference models [25].

1.2.4.4 Learning method for new users

For a new user, to recommend items can be a challenge and trying to understand the user can lead to many solutions but trying to find the best fit for the user is hard. One difficult, though common, problem for a recommender system is the *cold-start* problem ([40], [31]), where recommendations are required for items that no one (in our data set) has yet rated. Collaborative filtering cannot help in a cold-start setting, since no user preference information is available to form any basis for recommendations. However, content information can help bridge the gap from existing items to new items, by inferring similarities among them. Choosing the features can be a very difficult one. Since all users are not understood on the same basis.

1.2.4.5 Sparse data

Although every system has ‘super’ raters, that is people who rate thousands (and in some cases tens of thousands) of items, the number of such people is low. In fact, most people give feedback on only a handful of items, resulting in a large, but very sparse dataset. The data also exhibits the long tail phenomenon, with the majority of items getting just a handful of ratings. However, as recent studies have shown, understanding the tail is crucial—while no item in the tail is rated by too many people, the vast majority of people rate at least a few items in the tail [32].

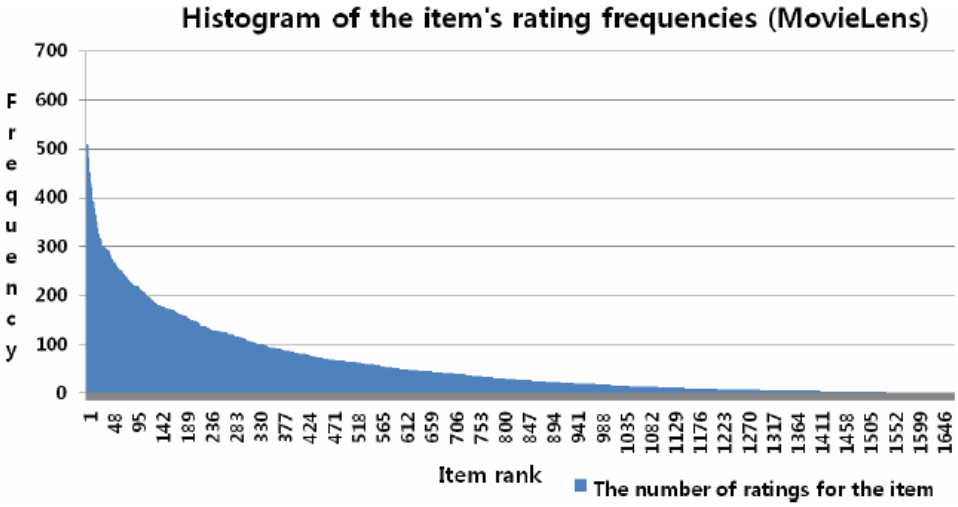

Figure 1-2: *Long tail in recommender systems* **[32]**

1.3 Thesis Goals

Regardless of the underlying recommendation strategy, item-based or user-based, one of the key concerns in producing recommendations is *over-concentration*, which results in returning items that are too homogeneous. Traditional solutions rely on post-processing returned items to identify those which differ in their attribute values (*e.g.*, genre and actors for movies). Such approaches are not always applicable when intrinsic attributes are not available.

In this thesis, I consider the problem of diversifying the set of item recommendations. This thesis presents an approach to diversify the recommended item set that is based on the prediction combinations of content-based approach and collaborative filtering approach. Although the problem of diversity and freshness seem to be very closely related they are very independent of each other. The problem of freshness is also handled partly. Therefore the goals of the thesis work are stated below:

i. Develop a recommender algorithm which can produce accurate predictions. Accuracy is measured in terms of Mean Absolute Error (MAE); on a range of 1 to 5 accuracy greater than 70% (*i.e.*, MAE less than 30% (1.5)) is considered to be reasonable.

To support the claim that 70% accuracy cutoff is indeed a reasonable threshold, a simple experiment was done. The data set which is used to develop the recommender system, is used to evaluate the changes with accuracy. The intuition is that in a worst case scenario, with changing accuracy levels the items which are relevant can become irrelevant and irrelevant items can become relevant items. Suppose on a rating scale of 1 to 5, with 1 being the worst and 5 being best item for the user, then with changing accuracy the recommender system can term the items indifferently.

The result of experiment is given below, with accuracy to loss in relevancy graph. From the graph it can be inferred that until 70% accuracy the drop in relevancy is uniform after that there is drastic drop in relevancy. Hence this supports the claim of 70% being a reasonable accuracy cutoff for the recommender system.

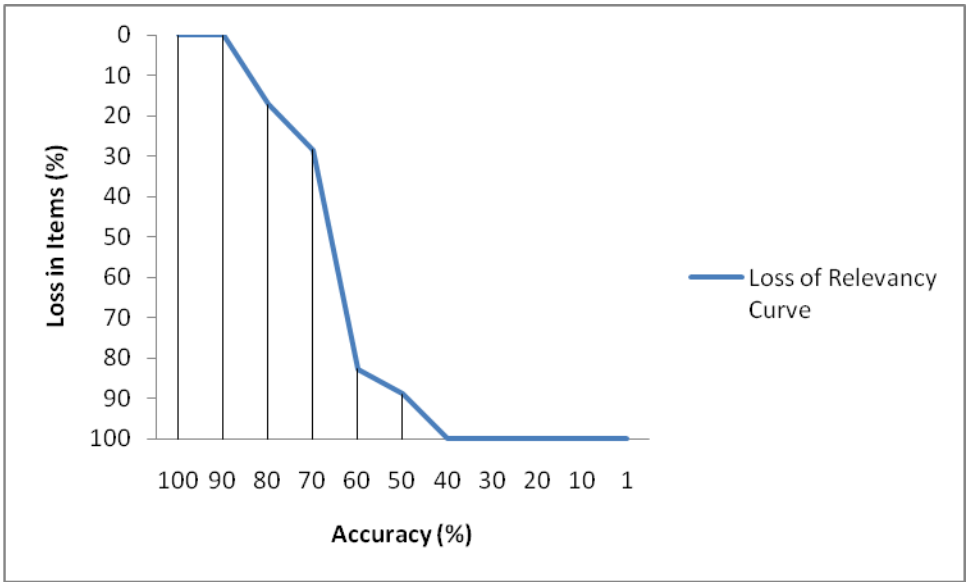

Figure 1-3: *Loss of Relevancy: Accuracy (%) to Loss in Items (%) for validation of thesis goal of 70% threshold cutoff.*

ii. Prove that the generated recommender algorithm when compared to other hybrid recommender algorithms performs reasonably well.

iii. Diversify the set of item recommendations, without major loss of relevance. Prove that once a threshold of the accuracy is reached diversity can be increased further without any major loss of relevance.

Therefore, this thesis asserts that once a base recommender algorithm is generated, one can diversify the set of item recommendations on various metrics with holding an acceptable level of relevance. This thesis is directed at recommender systems which are developed holding only the accuracy metrics as the only goal of recommending. The essence of the thesis work is to understand the diversification process and advantages of diversifying using a hybrid approach rather than an attribute driven diversification or explanation based diversification or random diversification.

1.3.1 High-level Overview

In the thesis work, I develop a recommender system for movies. The data set used consists of 943 users, who combined to rate a total of 1682 movies. The total numbers of ratings are around 100,000. For most of the systems, developed using this dataset the MAE (Mean Absolute Error) and using different base recommendation algorithms [46], is around 1.0 to 1.2 on a scale of 5. So the goal is to develop a recommender system, which tries to maintain a MAE of 1.2 maximum thresholds cut off and tackle the issues of diversity and freshness.

To handle the issue of diversity, the approach is to understand the items, and trying to come up with features which represent these items, and next try to sort them out into sets. As, here the items are movies we can try to sort them out into sets in two ways. One possible solution is to take the pre-defined genres and sort the movies by the values of the genres. Another solution is to represent the movies in terms of feature vectors and then trying to sort them into sets based on the distance between them. While, the first solution might result in movies being in multiple sets and they might be given a higher weight for a specific user while recommending genres. The drawback of second solution is that selecting the features can become a problem. In Chapter 5, I discuss the solution more in detail. The issue of freshness is dealt in somewhat a random way here; predictability comes when already known items are seen. Because of this, I treat each item equally and in the later stages by the use of pattern seen in user profile to those IMDB data **Error! Reference source not found.**, I put weights to parameters which affect the predicted rating of the movie. These approaches are discussed clearly and in detail in Chapter 4.

2 Related Research Work

In this Chapter I discuss some of the base recommender algorithm approaches mentioned in Chapter 1. In section 2.1, some of the related work and systems developed using content-based, collaborative filtering and hybrid approaches is documented. Also, a comparison of those approaches is also given. In section 2.2, the research part of this system 'Diversity in recommender system' is discussed with the help some already existing approaches and limitations and advantages of those approaches are done.

2.1 Base Recommender Algorithm

Some of the related work done in developing base recommender algorithms is documented in this section.

2.1.1 Content-based Approaches

As mentioned in the previous Chapter, content-based Approaches are one of the majorly used techniques in developing a recommender algorithm. There have been many systems which have been developed using this approach. As mentioned in the previous Chapter content-based approach tries to identify the item-to-item correlation between items rated/reviewed by the active user to those of unseen items, thus inferring predictions. Even with the essence of such an inference used in content-based recommender approach, there have been many improvements / techniques used to increase such efficiency or accuracy of such inference.

For example, the approach of 'Knowledge Infusion: the process of providing a system with the background knowledge which allows a deeper understanding of the items it deals with.' this deals with finding out models and relationships between information gathered from various sources about the items [42]. Here, although the base technique is to find items which might be liked by the user, this can be done with a better understanding of the items. As items are described by their attribute profiles, this approach does give a richer attribute profiles, which helps in understanding the items better. Clearly, however, the drawback of such an approach is the source of information itself and the validity of the sources this might turn bad in case of web-based applications. Also, building models on the large scale of information is very expensive, although this might increase accuracy the process of model building is an expensive process.

Another technique used to increase the efficiency of a content-based approach is to use feature weighting in the representation of items [11], here changing the feature weights using a collaborative filtering approach is done, with this richer feature profiles of items specific to the active user are generated. Although from the results, it can be concluded that this approach outperforms traditional content-based approaches, but the most common problem with content-based approach is not solved, that is of uniquely identifying the items. The author does not propose a solution to solve the problem of identifying features but tries to understand the identified features better, and this approach works well for a well defined data set, but can perform poorly if the features are wrongly identified.

Likewise in most systems using a content-based approach some common problems seem to appear. Some of the advantages and disadvantages of content-based approaches are discussed here.

Some of the advantages of this approach are the system can actually recommend previously unrated items. In this approach all the items are considered to be equally weighted. Thus this will not only consider popular items but also consider unrated and unpopular items. Another advantage of content-based approach is the ability to recommend items to users with unique interests and to provide explanations for its recommendations. Since it does not consider user similarities, it only goes on to give items similar to the ones which the user likes, so any kind of user can be recommended.

Some of the limitations of content-based approach are that of problem of finding features which can uniquely identify the items is very difficult indeed. In most cases when the items are very similar, then deciding on the number of features needed to identify each one uniquely can cause quite a head-ache. Also, the recommending approach does not learn from the user, if the user has contrasting opinions on two similar items, the system can lead to recommending unrelated items.

2.1.2 Collaborative Filtering

As mentioned in the previous Chapter Collaborative Filtering Approaches are one of the most commonly used techniques in developing a recommender algorithm. There have been many systems which have been developed using this approach. As mentioned in the previous Chapter Collaborative Filtering approach tries to identify the like-minded users to the active users, by

using either a memory based approach (section 1.2.2.1) or model based approach (section 1.2.2.2) and from the like-minded users the intuition is to infer predictions on unrated items for the active user. Even with the essence of such an inference in collaborative filtering approach, there have been many improvements / techniques used to increase efficiency of such inference.

Identifying the users' relevance opinion is one of the approaches [10]. The intuition is not just finding like-minded users from the user database but to understand the like-minded user itself. This is done to weight the relevance of the like-minded user to the active user. Thus constructing such a model to understand the user relevance opinion is shown to increase the efficiency of pure collaborative filtering approach. This kind of approach can lead to very accurate predictions, but the approach doesn't guarantee any accuracy when the active user is unique or some unique items which are unrated or not popular might be recommended.

The above mentioned disadvantages are the most common in most systems using pure collaborative filtering approaches, like another system [49] in which collaborative filtering is used in two ways, one to directly infer predictions from the set of similar users to the active user and another is to generate rules from the similar users using domain knowledge and user preferences; these rules are used to infer more accurate predictions. This approach actually solves one of the open problems 'cold-start' thanks to pre-processing done during the inception of the system, but then again it still doesn't handle the general problem in collaborative filtering that of handling unique users and unpopular items.

Thus, from most of the research the advantages and disadvantages of collaborative filtering approaches are documented here in short. Some of the main advantages of collaborative filtering approaches are the mean squared error on the prediction on an item and to those of actual ratings can be considerably reduced. It can actually give an accurate estimate on the items reviewed by like-minded users, thus reducing the set of items to recommend considerably; also the computational complexities are lesser than those of content-based approaches.

There are many known open problems, which are seen with the use of collaborative filtering approach, although these approaches might give recommendations, in most cases the items which the active user is looking for, but in some cases they might have the following problems:- The problem of cold-start can be give a real head-ache, since to find similarity between active user and set of other users, the active-user doesn't have any information regarding him, even if the cold-start is solved by initially giving a set of most popular items,

there might be a possibility that the active user might be very unique and the knowledge base might not have a user who is similar to the current user. Also, if the ratings of user for all the items is sparse, *i.e.*, if the user-item matrix of ratings is constructed and if the matrix is sparse, then the similarity measure is going to weigh down significantly, since the measure of similarity can be less the threshold which might actually signify dissimilar users. Also, in most cases the items which are not popular are avoided and thus reducing the chance of a user having a broader scope of items available, this might actually have a negative cascading effect on the purpose of recommender system.

2.1.3 Hybrid Approaches

The term *hybrid recommender system* is used here to describe any recommender system that combines multiple recommendation techniques together to produce its output. There is no reason why several different techniques of the same type could not be hybridized, for example, two different content-based recommenders could work together, and a number of projects have investigated this type of hybrid [5], which uses both naive Bayes and kNN classifiers in its news recommendations, is just one example.

The main purpose of developing hybrid approaches is to overcome the problem which is seen using either the pure content-based approach or pure collaborative filtering approach. The limitations of a hybrid system completely depends the type of hybrid it is and the base recommender algorithm itself and it does not have common limitations like the previous approaches.

As mentioned in the previous Chapter, a hybrid recommender system can be a combination of any (at least two) recommender approaches, even if they are of the same type.

For example, in the previous section 2.1.2, the system [49] uses a combination of two collaborative filtering approaches in a weighted hybrid recommender system. Also, in the section 2.1.1, the system [42] uses a weighted combination of multiple content-based approaches to predict ratings. The systems combining same kind of recommender approaches implicitly they carry the same problems as with their base approach. Because of this a content-based hybrid approach has the same limitations as content-based normal system, and same agrees with a collaborative filtering based hybrid approach.

There are some systems, which try to merge different approaches in order to overcome the limitations of each of the individual approaches. Content-boosted collaborative filtering is one such approach [29] in which the existing user profiles are enriched with content-based predictions and these are used to generate the collaborative filtering based predictions for the complete set of items. This way, the limitations of collaborative filtering are overcome and also, the limitations of content-based filtering.

A general comparison of the recommender approaches is given in table below, the comparison is very generic and some of the attribute values for comparison can vary depending on the approach used.

Attribute	Content-Based	Collaborative Filtering	Hybrid (non-similar approaches)
Unique Users	✔	✘	✔
Unpopular Items	✔	✘	✔
Non-static item set	✘	✔	✔
Non-static user profiles	✘	✔	✔
Sparse Data Set	✔	✘	✔
Relevant Recommendation	✘	✔	
Complexity	✔	✘	✔

Table 2-1: *Recommender Techniques*

2.2 Diversification in Recommender System

There have been a lot of approaches which have been developed for solving the issues of diversity. Some of them have been item-based diversification, some have been based on user tastes and some have been a mixture of both item based and user preferences. Hence, diversity is not just defined in terms of how different the set of recommended items are from each other, but can be defined in terms of the need of the recommender system. For example, there are systems which provide temporal diversity *i.e.*, diversity over time, some which provide explanation diversity *i.e.*, the diversity which arises from difference in explanation of each recommended

item (basically the approach used to get the item recommendation) and many other kinds of diversities are possible based on the need of the recommender system.

Thus, this section concentrates on the diversification task which is handled in the current thesis work, *i.e.*, item diversity in the set of recommended items. Also, one of the pre-requisite of this task of diversification is maintain a level of relevance, since diversifying the results solely for the purpose of producing diverse results can lead irrelevant recommendations.

Some of the approaches used to produce item diversification are based on item attribute-based diversification or user-preference based diversifications or explanation based diversification.

2.2.1 User preference based diversity

The approach where diversification is done based on user attributes [50] can be beneficial in many ways; here user attributes can refer to attributes which are used to understand the user. The approach used here considers the items into clusters of topics which are relevant to the user based on the user attributes and recommends using the topic probability for all items in the topics. For example, in case of a movie recommender system, the topic might mean genres and item diversification is done based on genre probabilities for the active user. Also, such an approach leads to high level of diversification, while maintaining a reasonable amount of accuracy, the problem with such an approach is that, this might work fine when user preferences don't change, *i.e.*, using static data. But when dealing with changing user preferences this can lead to very bad recommendations. Another issue can be that of understanding user tastes, which itself is an open problem, since choosing the attributes on which the user has to be understood is the very essence of recommender system.

2.2.2 Attribute-based diversity

The approach of attribute-based diversification is to say that the recommended items differ from each other as much as possible. The measure used here can be termed as 'dissimilarity', which is to say the dissimilarity between items. This can be calculated by representing the items using attributes and then concentrating on understanding the dissimilarity of the attributes. For example, Yahoo! Movies [54] recommends movies based on the attributes intrinsic to each movie (such as genre, director, *etc*.). This works reasonably well in comparison with user-based diversification, but the only problem exists when the item set is not clearly

defined. This approach doesn't work properly in the web scenarios where the understanding the items is not easy. The situations in which the items are clearly defined are the ideal situation for such an approach [2].

2.2.3 Explanation based diversity

Another approach of explanation based item diversification [2] is to recommend a set of items greedily to increase the diversity factor. The diversity factor used here is the '*Jaccard diversity distance:* the difference in the explanation sources to two items'. Clearly this is a completely different approach and is not compared to the item diversity produced by other approaches, but here the novel factor is to produce diversity in items using the diversity which is drawn from the recommender algorithm itself. This can lead to varied results in item diversity since; it is possible to have different explanation sources for items even when the items are intrinsically similar.

The approach used in this thesis work, is a combination of some of the above mentioned approaches, to solve the issue of diversity to produce item based diversity. But the user tastes are not considered directly, but are understood from the way of collaborative filtering, since the clusters of similar users are considered to be of the same type as the active user. The approach is to understand the items in-terms of attribute profiles. This is taken from the Content-based approach; I propose to form clusters of items, which are similar to each other. Thus, this actually considers various defining attributes of the items. After clusters of items are generated, trying to assign weights on the clusters based on like-minded user preferences; like-minded users are found from the collaborative filtering approach. The intuition behind using similar users instead of the active user is to provide more diversification since although the users are similar they are not identical so getting their choices helps in providing a variety of choices for the active user. Greedily choose the items, which minimize the relative item similarity from each of the cluster of items based on the weights of the clusters.

3 Modeling text corpora and Clustering

This Chapter is documented to handle the discussions of some of the specific techniques used in this thesis work. In section 3.1.1 some of the information retrieval techniques which have been previously used to model textual corpora are discussed with their pros and cons and also Latent Dirichlet Allocation which is used for topic modeling on item content as a replacement for the previous IR techniques is discussed and in section 1 Clustering Analysis which is used to find similar users and similar items is discussed.

3.1 Modeling text corpora

Information Retrieval is a science of retrieving information from documents, in general sense searching through documents for information pertaining to a natural language query. One of the main goals of a successful information retrieval is the techniques' power of representing a textual corpus. The goal is to find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful for basic tasks such as classification, novelty detection, summarization, and similarity and relevance judgments.

There have been many techniques which have been developed to handle this research problem, the essence of the technique is to significantly reduce the text corpora into vector of real numbers, the vectors might represent keywords, topics, *etc.*,

3.1.1 Information Retrieval Techniques

3.1.1.1 TF-IDF Scheme

A basic vocabulary of 'words' or 'terms' is chosen, and, for each document in the corpus, a count is formed of the number of occurrences of each word. After suitable normalization, this term frequency count is compared to an inverse document frequency count, which measures the number of occurrences of a word in the entire corpus (generally on a log scale, and again suitably normalized). The end result is a term-by-document matrix X whose columns contain the *tf-idf* values for each of the documents in the corpus. Thus the *tf-idf* scheme reduces documents of arbitrary length to fixed-length lists of numbers [37].

While the *tf-idf* reduction has some appealing features—notably in its basic identification of sets of words that are discriminative for documents in the collection—the approach also provides a relatively small amount of reduction in description length and reveals little in the way of inter- or intra-document statistical structure [6].

3.1.1.2 LSI (Latent Semantic Indexing)

LSI uses a singular value decomposition of the *X* matrix to identify a linear subspace in the space of *tf-idf* features that captures most of the variance in the collection. This approach can achieve significant compression in large collections. Furthermore, it can be argued that the derived features of LSI, which are linear combinations of the original *tf-idf* features, can capture some aspects of basic linguistic notions such as synonymy and polysemy [12].

3.1.1.3 pLSI (Probabilistic Latent Semantic Indexing)

Given a generative model of text, however, it is not clear why one should adopt the LSI methodology—one can attempt to proceed more directly, fitting the model to data using maximum likelihood or Bayesian methods.

A significant step forward in this regard was made by *probabilistic LSI (pLSI)* model [19], also known as the *aspect model*, as an alternative to LSI. The pLSI approach, models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as representations of ‘topics’. Thus each word is generated from a single topic, and different words in a document may be generated from different topics. Each document is represented as a list of mixing proportions for these mixture components and thereby reduced to a probability distribution on a fixed set of topics. This distribution is the ‘reduced description’ associated with the document.

In pLSI, each document is represented as a list of numbers (the mixing proportions for topics), and there is no generative probabilistic model for these numbers. This leads to several problems: (1) the number of parameters in the model grows linearly with the size of the corpus, which leads to serious problems with over-fitting, and (2) it is not clear how to assign probability to a document outside of the training set ([6], [15]).

Latent Dirichlet Allocation (LDA) was proposed to alleviate the problems of pLSI and to produce a more generative probabilistic model. A clear depiction of the difference between pLSI, LDA and unigrams is geometrically given in figure below.

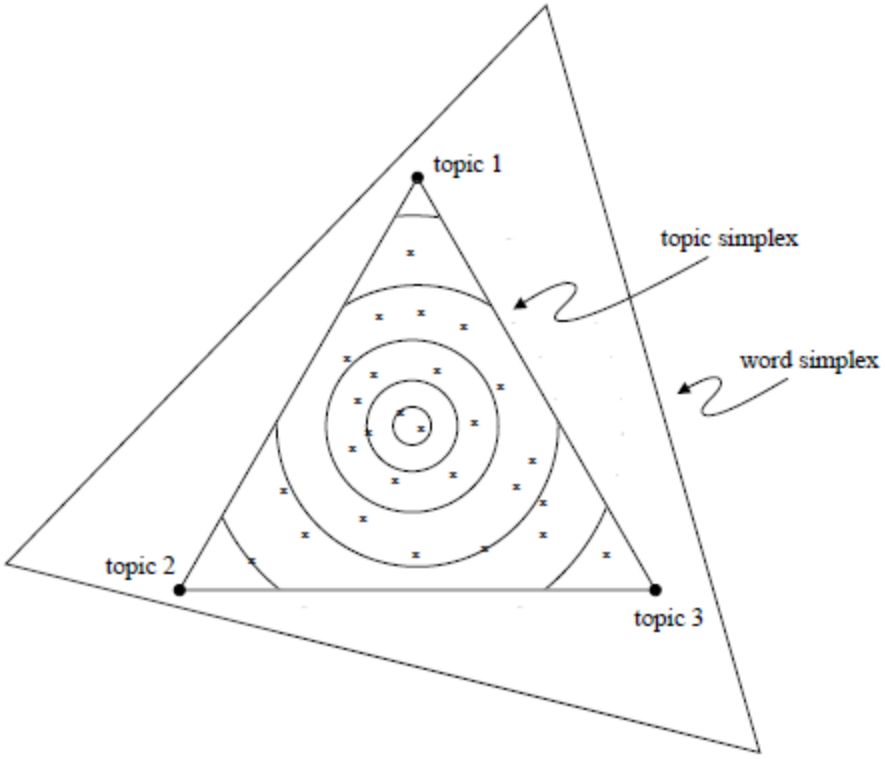

Figure 3-1: *The topic simplex for three topics embedded in the word simplex for three words. The corners of the word simplex correspond to the three distributions where each word (respectively) has probability one. The three points of the topic simplex correspond to three different distributions over words. The mixture of unigrams places each document at one of the corners of the topic simplex. The pLSI model induces an empirical distribution on the topic simplex denoted by x. LDA places a smooth distribution on the topic simplex denoted by the contour lines.[6]*

3.1.2 Latent Dirichlet Allocation

3.1.2.1 Notation and Terminology

The notation used in the methodology of LDA is given below:

- A word is the basic unit of discrete data, defined to be an item from a vocabulary indexed by $\{1 \ldots V\}$. We represent words using unit-basis vectors that have a single component equal to one and all other components equal to zero. Thus,

using superscripts to denote components, the v^{th} word in the vocabulary is represented by a V-vector w such that $w^v = 1$ and $w^u = 0$ for u not equal to v.

- A document is a sequence of N words denoted by $w = (w_1, w_2 \ldots w_n)$, where w_n is the n^{th} word in the sequence.
- A corpus is a collection of M documents denoted by $D = \{w_1, w_2 \ldots w_M\}$.

3.1.2.2 Generative Process of LDA

Here a brief description of the generative process which is used by LDA is given, for further details regarding the generative process and other information refer to [6].

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

1. Choose N ~ Poisson (ξ).
2. Choose θ ~ Dir (α).
3. For each of the N words w_n:
   a) Choose a topic z_n ~ Multinomial (θ).
   b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Several assumptions are made to this basic model.

First, the dimensionality *k* of the Dirichlet distribution (and thus the dimensionality of the topic variable *z*) is assumed known and fixed. Second, the word probabilities are parameterized by a *k* x V matrix β where $\beta_{ij} = p(w^j = 1 \mid z^i = 1)$, which for now we treat as a fixed quantity that is to be estimated. Finally, the Poisson assumption is not critical to anything that follows and more realistic document length distributions can be used as needed.

Next the *k*-dimensional Dirichlet random variable θ can take values in the $(k-1)$-simplex and has a probability density function:

$$p(\theta \mid \alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}, \quad [6]$$

Where the parameter α is a *k*-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function.

Given the parameters α and β, the joint distribution of a topic mixture θ, a set of N topics **z**, and a set of N words **w** is given by:

$$p(\theta, \mathbf{z}, \mathbf{w} | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^{N} p(z_n | \theta) p(w_n | z_n, \beta), \quad [6]$$

Where p (z_n | θ) is simply θ_i for the unique i such that $z^i_n = 1$.

Integrating over θ and summing over z, we obtain the marginal distribution of a document:

$$p(\mathbf{w} | \alpha, \beta) = \int p(\theta | \alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n | \theta) p(w_n | z_n, \beta) \right) d\theta. \quad [6]$$

Finally, taking the product of the marginal probabilities of single documents, we obtain the probability of a corpus:

$$p(D | \alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d | \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} | \theta_d) p(w_{dn} | z_{dn}, \beta) \right) d\theta_d. \quad [6]$$

The LDA model is represented as a probabilistic graphical model in Figure 3-2. As the figure makes clear, there are three levels to the LDA representation. The parameters α and β are corpus level parameters, assumed to be sampled once in the process of generating a corpus. The variables θ_d are document-level variables, sampled once per document. Finally, the variables z_{dn} and w_{dn} are word-level parameters and are sampled once for each word in each document.

The LDA tool which was used in this thesis Mallet [56] uses Gibbs Sampling for estimating the parameters.

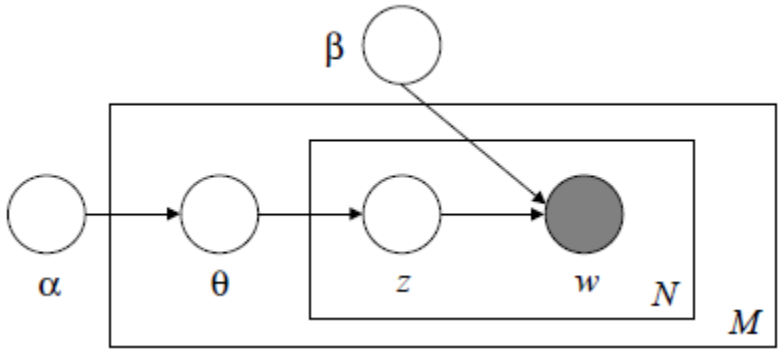

Figure 3-2: *Graphical model representation of LDA. The boxes are 'plates' representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.* **[6]**

3.2. Cluster Analysis

Clustering is one of the major and important techniques used in unsupervised learning in many fields like machine learning, information extraction *etc.*, Unsupervised learning can be termed as a class of problems were given a data one wants to find the structure, features and distortion in the data. In unsupervised learning the data is unlabelled, so no organization is manually done, so given such a data one tries to extract key features, understand the distortion *etc.*, Clustering is a technique used to solve these kinds of problems. Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). It has an appeal and usefulness as one of the steps in exploratory data analysis. However, clustering is a difficult problem combinatorial, and differences in assumptions and contexts in different communities have made the transfer of useful generic concepts and methodologies slow to occur ([34], [21], [41]).

Here I cover some basics of clustering and which of the techniques is used in my work. Going through the stages in clustering [21], the major importance is given to representing the items (data) into an effective feature vector. The starting step is to understand patterns which can help in identifying the features, using this, a set of all possible candidate features can be extracted. In the next step, the extracted candidate features are used and using some selection criteria based on the active problem, the feature selection is done and all the data is represented in the form of feature vectors. Using some clustering technique in the next step these data nodes are grouped into clusters. Normally, the grouping is done based on the distance measures between all the data nodes. Most commonly used distance measures are:

- Euclidean Distance
- Manhattan Distance
- Hamming Distance
- Mahalanobis distance

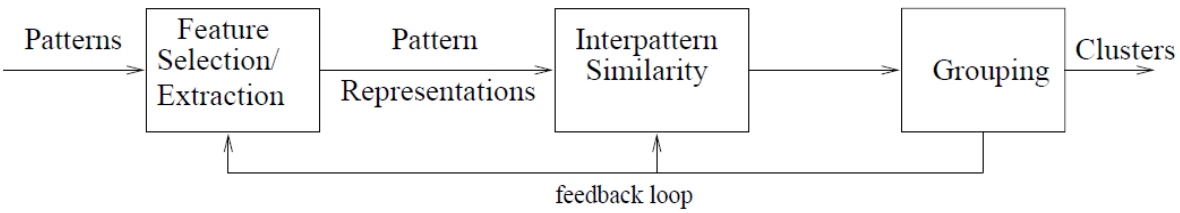

Figure 3-3: *Clustering Stages* **[21]**

Clustering techniques are majorly classified into two categories:

i. *Hierarchical Clustering:* In these types of clusters the clusters at any stage of grouping are formed from the information by the previous stage clusters. These are normally agglomerative (bottom-up) or Divisive (top-down). Clustering techniques are those which are used on the data points which are represented in the form feature vectors. [30]

ii. *Partitioning Clustering:* Here typically the approach tries to determine all clusters at once. They are not dependent on the clusters of previous iterations like the hierarchical clustering techniques. These are mostly used when the data set is large enough that the computational effort used by hierarchical clustering is too expensive. [30]

In this thesis work a partitioning clustering technique is used since the data set is large enough. As mentioned before, clustering is a famous technique used for collaborative filtering approach. Hence for collaborative filtering component of the recommender system clustering is used and also in the approach to solve the problem of diversity.

The clustering technique specifically used is K-Means Clustering using Euclidean Distance. The *k*-means algorithm assigns each point to the cluster whose center (also called centroid) is nearest. The center is the average of all the points in the cluster — that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. The steps used in K-Means algorithm [16] are given below:

1. Initialize algorithm with guessed centers C.
2. For each data point x_i, compute its membership m ($c_j \mid x_i$) in each center c_j and its weight w (x_i).
3. For each center c_j, re-compute its location from all data points x_i according to their memberships and weights: given in the figure below.

$$c_j = \frac{\sum_{i=1}^{n} m(c_j|x_i)w(x_i)x_i}{\sum_{i=1}^{n} m(c_j|x_i)w(x_i)}$$

4. Repeat steps 2 and 3 until convergence.

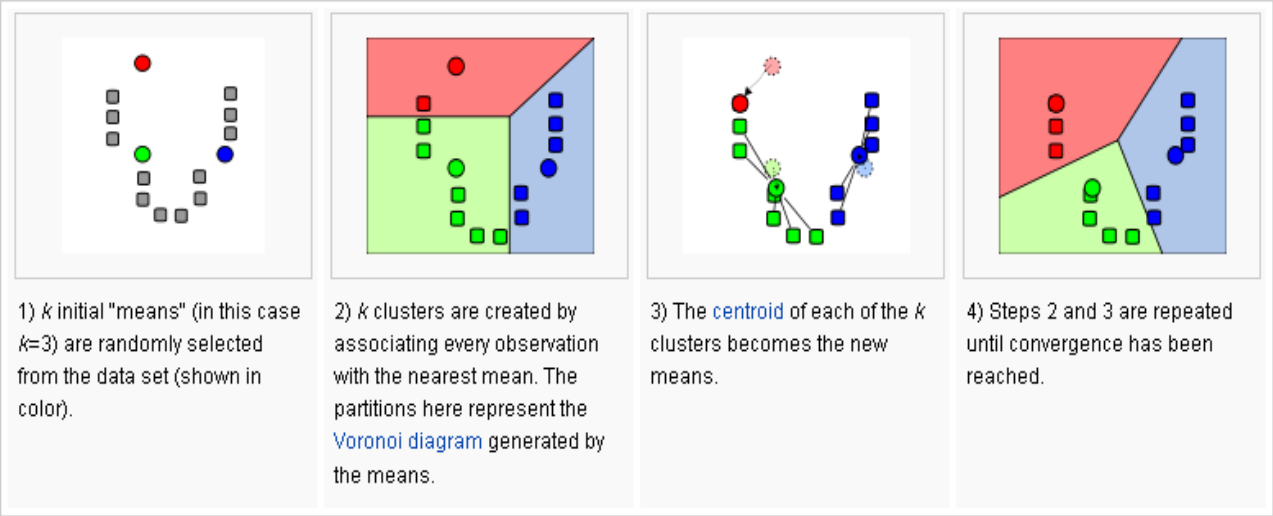

Figure 3-4: *K-Means Example* Error! Reference source not found.

4 Preliminary Experiments & Methodology

This Chapter is documented to address the methodology of the system developed in this thesis work. In section 4.1 preliminary experiments and results are shown, which serve as the rationale behind the decision for the approach chosen to develop the base recommender system algorithm. In section 4.2, the methodology used in this thesis work is described in detail.

4.1 Preliminary Experiments

The approach used to develop a base recommender algorithm in this thesis work is a weighted hybrid approach on a combination of content-based and collaborative filtering components. To emphasize the need for using a hybrid approach, some initial experiments are run on individual components of content-based and collaborative filtering,

The preliminary experiments are chosen from the most used experimental methodology, *i.e.,* measuring the responsiveness (accuracy) of a recommender system. The metric used to measure this is the Mean Absolute Error (MAE), which is the mean of the absolute values of error in prediction for each user.

The complete dataset used consists of 943 users who have combined to rate a total of 1682 movies and the total number of ratings is 100,000. From the complete data set 5 different 80-20 splits are taken to run the experiments. 80% split is considered as the train dataset and 20% split is considered as the test dataset.

For the content-based component the item representation in the attribute profile is done using the topic probabilities which have been generated using LDA described in the previous Chapter, and predicting the ratings of movies for users in the test set, using the train dataset.

For the collaborative filtering component the users from the train dataset are represented using their ratings as feature instances, and clustered using this format to find a cluster of similar users. Predictions are inferred for movies, using this cluster of similar users.

The results of these experiments for both content-based and collaborative filtering component are given below in Figure 4-1: Collaborative Filtering Component and Figure 4-2: Content-Based Component.

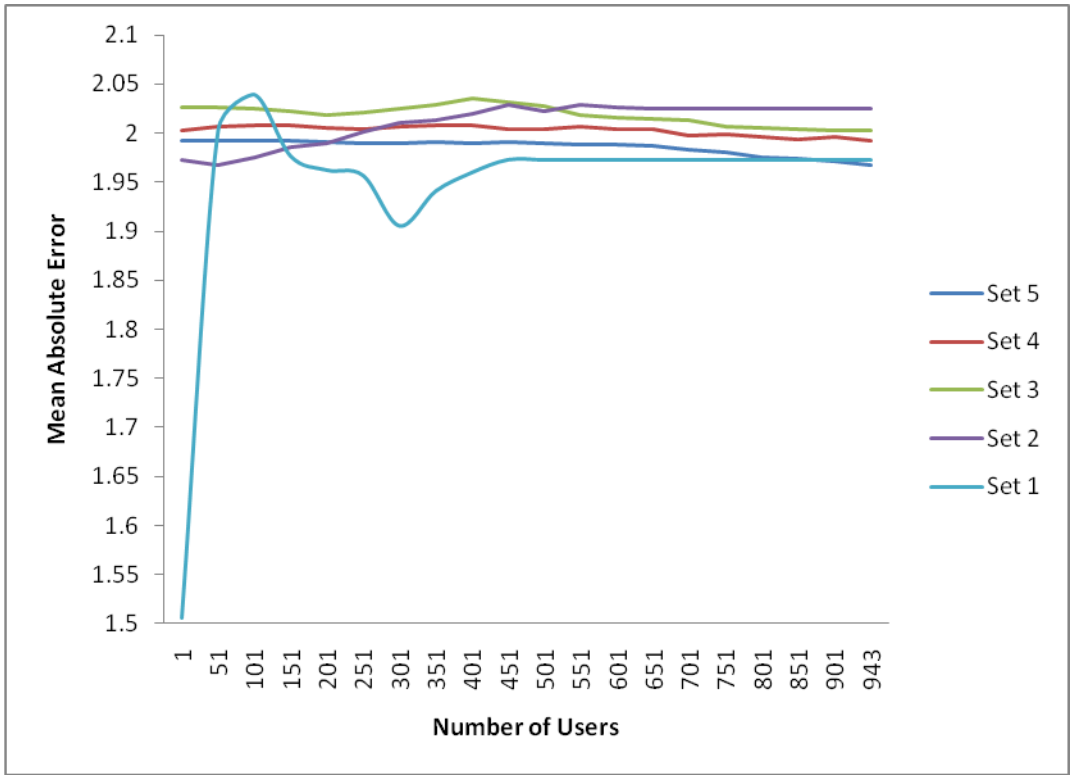

Figure 4-1: *Mean Absolute Error (MAE) to Number of Users, graph for a pure collaborative filtering approach on five different 80-20 splits of the complete data set, each represented by Set1, Set2, Set3, Set4 and Set5.*

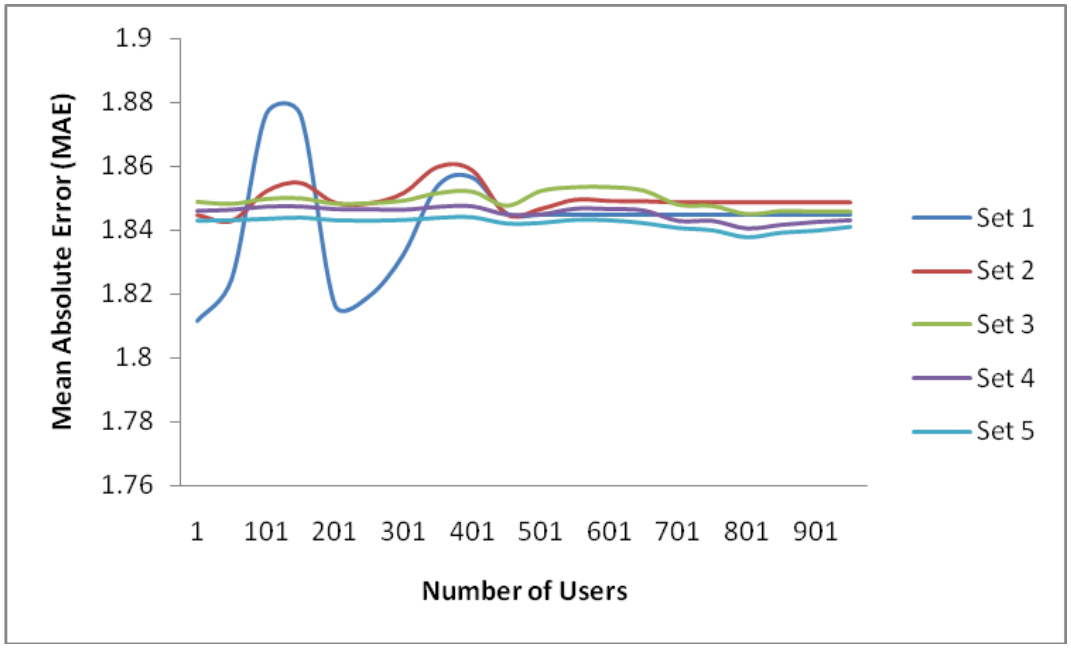

Figure 4-2: *Mean Absolute Error (MAE) to Number of Users, graph for a pure Content-based approach on five different 80-20 splits of the complete data set, each represented by Set1, Set2, Set3, Set4 and Set5.*

From the preliminary experiments it is clear that both the components of content-based and collaborative filtering do not perform well. This can be inferred; when these result values are compared to some hybrid systems which use both the approaches. For example, a hybrid system [32] which is run on the same dataset as the preliminary experiments clearly outperforms the individual content-based and collaborative filtering components. Thus, there is a need to develop a better approach than using individual components.

These preliminary experimental results form the basis for the rationale behind the hybrid approach for the development of a base recommender system algorithm in this thesis work.

4.2 Methodology

The goal of the approach is to develop a recommender system, which can tackle the problem of diversity while maintaining the maximum threshold cut-off MAE (Mean Absolute Error).

The data set used to develop the recommender system is 'Movie Lens data set' by Group Lens Research group (Group Lens is a research lab in the Department of Computer Science and Engineering at the University of Minnesota) [52]. The data set consists of 943 Users who have combined to rate 1682 Movies (Items). The total number of ratings given are 1, 00,000. Apart from this, documents about movies (items) from Wikipedia **Error! Reference source not found.** and IMDB **Error! Reference source not found.** are collected to gather more information about the items. The Wikipedia articles would normally give plot, awards and reception information of the movie. The information from IMDB was the IMDB user rating, number of user votes and the year of release. All this information is used for the prediction.

The approach used to develop the hybrid system is weighted recommendations. 'The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation' [8]. So both content-based recommendations and collaborative recommendations are combined using weighted parameters.

4.2.1. Collaborative Filtering Component

Collaborative filtering takes into account user preferences. As mentioned before, model based collaborative filtering approach is used; this mechanism uses user rating data to compute similarity between users and similar users are found using clustering technique. This is used for making recommendations. This was the earlier mechanism and is used in many commercial systems. It is easy to implement and is effective.

So the first step is the representation of user profiles. Here the information at hand about users is just the ratings they have given for some movies. So, to represent this information, the movies are considered as features for each user and their respective ratings are considered as feature instances. Thus a feature vector for each user is generated. Mathematically this can be represented as: Consider for user u, and the whole set of Movies as $\{m_1, m_2, m_3 \ldots m_{1682}\}$ and the ratings given by the user can be {1, 5} for some of the movies and the rest are unrated by this

user. Unrated values are considered to be 0 in this instance. So ratings be considered as {r_{u1}, r_{u2}… r_{u1682}}

So a user feature vector would be in the form of:

$$U = ru1 * m1 + ru2 * m2 + ru3 * m3 \ldots ru1681 * m1681 + ru1682 * m1682$$

Now, initially the active user for whom the recommendations are to be made is also considered using the cold start initial recommendations (most popular items in terms of number to user ratings for the movie) to have rated some initial set of movies. Thus he is also represented in this format.

The next step is to find the users who are similar to the active user, to do this K-Means' clustering technique is used. The inputs of the clustering technique are the feature vectors of the user profile and the active user. The user-profiles are modified so the feature vectors are constructed for only the movies which have been rated by the active user. This is done to ensure that the similarity is constrained to the point of the information known about the active user. If this is not done, the active user might be placed in a cluster of similar users, yes, but the probability that the user, given initial ratings, might actually differ from the cluster of similar users is very high. To eliminate this scenario, given the current information of the active user similarity is found by only using the given information.

The distance measure used for K-Means Clustering is the 'Euclidean distance'. The Euclidean distance between point's p and q is the length of the line segment $\overline{\mathbf{pq}}$. In Cartesian coordinates, if p = (p_1, p_2 ... p_n) and q = (q_1, q_2 ... q_n) are two points in Euclidean n-space, then the distance from p to q is given by:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Figure 4-3: *Euclidean Distance* Error! Reference source not found.

To eliminate computational complexities, recursive clustering is used to find the best cluster into which the active user has to be placed. Once a cluster of 'n' similar users is found, the next step is to infer predictions for these set of similar users. To do this Cosine Rule is used. The Cosine Rule here states that, the prediction for a movie 'm' is given by the average of ratings of the similar users which are transformed using the cosine similarity between the active user and the similar user. Mathematically, this can be represented as follows: Let movie be

represented by ‘m’ from set of Movies ‘M’ of size ‘k’, which have been rated by active user, and the set of similar user be represented by ‘U’ of size ‘n’, then cosine similarity of user ‘u’ from set ‘U’ to the active user ‘a’ is given by:

$$CosSim(u,a) = \frac{\sqrt{\sum_{i=1}^{k}(u_i - a_i)^2}}{\sqrt{\sum_{i=1}^{k} u_i^2} * \sqrt{\sum_{i=1}^{k} a_i^2}}$$

Now, the complete rating prediction is given by the following formula:

M: Set of movies rated by the active user ‘a’.

k: Size of movie set M.

m: movie which $\in$ set M.

U: Set of similar users to the active user ‘a’.

n: Size of similar user set U.

u: similar user who $\in$ set U.

$$Prediction(m) = \frac{\sum_{i=1}^{n} CosSim(u_i, a) * Rating(m, u_i)}{n}$$

The intuition of the above formula is that the rating of two users are related by the measure of similarity of the users, so for an user y whose prediction for an item is to be made by using the rating of user x who has already rated the item, then prediction of y is similarity of the users x and y times the rating of item. The above formula can be explained as ‘the prediction of unrated item for a user, is said to be the average of all the predicted ratings using all the users’.

4.2.2. Content-Based Component

A content-based recommender learns a profile of the users’ interests based on the features present in items the user has rated. To do this, using the active users’ profile and information about the items rated and information of unrated items, the predictions are to be made. The initial step is to gather the information of items. As the items are movies, to describe an item the information needed in most cases would be plot of the movie, genre of the movie, awards, reception of the movie and additionally cast of the movie can also be considered. All this information is taken out from the Wikipedia articles for each of the 1682 movies manually and is placed into documents for each movie. Now, the next step is to represent these documents in terms of feature vectors. There are many approaches used in these kinds of problems. It is a

problem of text classification in machine learning. Some of the popular approaches would be vector space model and latent semantic analysis. LDA (Latent Dirichlet Allocation), which is discussed in Chapter 3, is used here for this purpose.

Using LDA one can represent the documents of text in terms of topic probabilities. However choosing the number of topics can be an issue, since choosing a very high number of topics can give you fine grained representation of the documents, but it can also lead to lots of noise. Since, some of the topics can be unrelated and unnecessary. Some experiments are conducted to choose the number of topics. These are clearly detailed in Chapter 5. Now, let's say each document is represented in terms of topic probabilities and the number of topics is 'n'.

Thus, using the above approach each item (movie) can be represented in terms of feature vectors in the following format mathematically:

$$m = w_1 * tp_1 + w_2 * tp_2 + \ldots + w_n * tp_t m = \sum_{i=1}^{t} w_i * tp_i$$

In the above formula, m is the movie, 'w_i's' are the weights are for each of the topics, 'tp_i' are the topics and 't' is the total number of topics.

The Next step is to infer the predictions for each movie using the active users' profile. We have the set of movies M, which have been rated by the active user. So constructing an item-item similarity matrix of the set of movies in M and the remaining of the total movies is to be done. This can be done in many ways; one of the simplest ways is to use the Cosine Similarity measure between two feature vectors. Cosine Similarity for item-item co-relation is given by the following formula:

$$CosSim(m1, m2) = \frac{\sqrt{\sum_{i=1}^{t}(m1_i - m2_i)^2}}{\sqrt{\sum_{i=1}^{t} m1_i^2} * \sqrt{\sum_{i=1}^{t} m2_i^2}}$$

Once the item-item similarity matrix has been constructed, the next step is to infer the predictions of the items using the ratings of the items rated by the active user. This is similar to the cosine rule which was used to infer predictions in collaborative filtering component. The mathematical representation of the cosine rule used here is stated below:

$$Prediction(m\prime) = \frac{\sum_{i=1}^{k} CosSim(m\prime, m_i) * Rating(m_i, a)}{k}$$

The intuition of the above formula is that the rating of two items are related by the measure of similarity of the items, so an item y whose prediction is to be made by using the rating of item x which has been rated, then prediction of y is similarity of the items x and y times the rating of item x. The above formula can be explained as 'the prediction of unrated item, is said to be the average of all the predicted ratings from all the rated items'.

4.2.3. Hybrid Step: Weighted Combination

A weighted hybrid recommender is one in which the score of a recommended item is computed from the results of all of the available recommendation techniques present in the system. For example, the simplest combined hybrid would be a linear combination of recommendation scores. Here we have predictions from content-based component and the collaborative filtering component, now these are to be used to predict the final predictions. This is done by predictor component. The predictor considers these two predictions as the baseline predictors and uses additional predictors to make accurate predictions. The following give the predictors used by the component.

Base Line Predictors:

- Content-based prediction.
- Collaborative filtering prediction.

Additional Predictors:

- Time-line predictions.
- IMDB user rating.
- Genre Predications.

The additional predictors are used to reduce the error in prediction.

Time-line predictions are used in a manner, where the active user predominantly prefers watching movies from a specific time period. For example a user might prefer watching movies released in 1980's. To capture these kinds of preferences, this predictor is used. The time-line prediction is done in the same way as previous predictions. For example, a movie 'm' has been rated as 'r' by the active user. Let the year of release be in the period of $|t_1 - t_2|$, then all the movies in the same period are given predictions in a similar way using the cosine rule discussed earlier. Mathematically, this is represented as follows:

$\forall$ *movies* $m\prime \in$ *time period* t_1 *to* t_2 *and* $\notin M$
and
$\forall$ *movies* $m \in$ *time period* t_1 *to* t_2 *and* $\in M$

$$TimePrediction(m\prime) = \frac{\sum_{i=1}^{k\prime} CosSim(m\prime, m_i) * Rating(m_i, a)}{k}$$

IMDB user-rating is used in order to better understand the active users' general opinion of a movie. This majorly comes into play when the active user is very unique and cannot find significant similar users or the items which the active user prefers seem to be very unique and where not rated by many users. To capture these kinds of preferences, this predictor is used. The mathematical representation of the prediction is given below.

We now define the term '*Diff*', as the difference in the IMDB rating of the movie and current rating of the movie. If two movies are different and we have user rating for one and have to predict for another, then although, if we know the similarity measure we need to understand in what way, the similarity affects the predicted rating. For example, a movie x might have been rated as 3 by the user and might have a rating of 5 on IMDB, and for another movie y we try to predict the user rating using IMDB rating. In such case simply using the similarity measure to infer ratings cannot be right solution, since we cannot directly say how the similarity effects the difference in the IMDB and user rating of a movie, sometimes it might be decrease the difference between the IMDB and user rating or sometimes it might just increase. The following solution is used to solve this issue.

$\textsc{Diff}(\textsc{m}) = \textsc{Difference}((rm_{imdb} - rm))$
if $CosSim(m\prime, m) < 0$
  if $Diff < 0$
    Diff Increases.
  else
    Diff Decreases.
  else
    Diff follows the similarity measure.

This mathematically can be represented as:

$$\text{DIFF(M)} = \text{DIFFERENCE}((rm_{imdb} - rm))$$
if $CosSim(m\prime, m) < 0$
  if $Diff < 0$
    $Diff(m\prime) = \frac{|(rm_{i_{imdb}} - rm_i)|}{CosSim(m\prime,m)}$
  else
    $Diff(m\prime) = (rm_{i_{imdb}} - rm_i) * |CosSim(m\prime, m)|$
  else
    $Diff(m\prime) = (rm_{i_{imdb}} - rm_i) * CosSim(m\prime, m)$

The final IMDB prediction is given by:

$$IMDBPrediction(m\prime) = IMDBRating(m\prime) + \frac{\sum_{i=1}^{k} Diff(m\prime)}{k}$$

Genre Predictions are used to understand, if a user predominantly prefers movies from a particular genre. This is a common phenomenon, that many users prefer some genres to other genres. This can help reduce the errors in prediction, if such a case exists. The approach is to initially, find the cluster of similar users and then get an average of the ratings for each genre; these are the genre averages for the active user. Then for each movie an average of all the genres it belongs is taken as the genre prediction of the movie.

The Genre Averages are calculated as following:

$$GenreAverages(g) = \frac{\sum_{i=1}^{g(n)} Rating(m, u)}{g(n)}$$

Here, ‘g’ is the genre; g (n) represents the size of the set of movies in this genre and which have been rated by the cluster of similar users.

Next the genre predictions are calculated using the following formula:

$$GenrePrediction(m) = \frac{\sum_{i=1}^{n(m_g))} GenreAverage(g_i)}{n(m_g)}$$

Now, after calculating the predictions of all movies from the predictors, the Predictor component, tries to combine these to make some meaningful recommendations. The simplest way to combine these would be combining them with uniform weights, but this kind of situation doesn’t actually help in getting accurate predictions. Thus, giving weights to each on them using

the current instance is the best way to do this. Firstly, the final formula for the predictor component looks like the following mathematical representation:

Predicted rating (movie) =

α (Collaborative Prediction) +

β (Content-based Prediction) +

γ (Time Prediction) +

δ (IMDB Prediction) +

λ (Genre Prediction).

The parameters 'α', 'β', 'γ', 'δ', 'λ' are weights given to each of the predictions.

An ideal assigning of values for the parameters would be to depend on the instance of the active user profile and item under considerations to re-arrange the weights. For example, if the user is unique and cannot find many similar users, then the weight on collaborative filtering is to be reduced or if the movie under consideration is very rare and is unique, then collaborative filtering is to be weighed more. Also, considering the other predictors, the user profile is to be considered. However this is not easy to solve and in the current thesis the parameter values are assigned statically using some experiments. Description of how the weighting procedure and fine tuning of these parameters is shown in Chapter 5.

4.2.4. Diversity

The approach where diversification is done based on user tastes [50] can be beneficial in many ways since diversification is done considering the user attributes. Also, such an approach leads to high level of diversification, while maintaining a reasonable amount of accuracy, the problem with such an approach is that, this might work fine when user preferences don't change, *i.e.,* using static data. However when dealing with changing user preferences this can lead to very bad recommendations. Another issue can be that trying to understand user tastes itself is an open problem, since on what attributes the user has to be understood is the very essence of recommender system.

The approach of item-based diversification is to say that the recommended items differ from each other as much as possible. The measure used here can be termed as 'dissimilarity', which is to say the dissimilarity between items. This can be calculated by representing the items using attributes and then concentrating on understanding the dissimilarity of the attributes. For

example, Yahoo! Movies [54] recommends movies based on the attributes intrinsic to each movie (such as genre, director, *etc*.). This works reasonably well in comparison with user-based diversification, but the only problem exists when the item set is not clearly defined. This approach doesn't work properly in the web scenarios where the understanding the items is not easy [47]. The situations in which the items are clearly defined are the ideal situation for such an approach.

Another approach might be diversifying based on the item explanations. *i.e.,* to diversify based on how each item is recommended and calculating the similarity on explanations. This approach produces a different way of handling diversity, which is based on the explanations of the recommended items. However the drawback of this approach can be that it does not ensure that the items themselves are diverse. It might be possible that two items which are recommended by different explanations might be similar [2].

Now coming to the research part of the system *i.e.,* the issues of diversity. To deal with this, the approach is to understand the items, and trying to come up with features which represent these items, and next try to sort them out into sets. As, here the items are movies we can try to sort them out into sets in two ways.

One implementation is to take the pre-defined genres and sort the movies by the weights of the genres. Using these genre weights and converting them into genre ratios can be useful in solving this problem of diversity, but this is true in case of a well defined dataset, but in most cases, finding the clusters of items is difficult. Also, using this approach it is not guaranteed that the items recommended are actually diverse, since movies in different genre can be similar too.

The implementation used in this thesis work is to understand the items from a different perspective rather than pre-defined clusters (genres). Thus, I propose my novel approach to solving the issue of diversification. For this approach, both the item similarity measure and user tastes although in an indirect way are considered. The approach is to understand the items in-terms of feature vectors. This is taken from the Content-based approach, thus using the feature vector of items; clusters of items which are similar to each other are formed. Thus, this actually considers various factors of the items rather than just the genres. After clusters of items are generated weights are given to the clusters based on user predictions; here users are just the similar users who have been clustered from the collaborative filtering approach. The intuition behind using similar users instead of the active user / or complete set if users is to provide more

diversification since although the users are similar they are not identical so getting their choices helps in providing a variety of choices for the active user.

Thus, once cluster weights are assigned to each of the cluster of items, these weights are used to get ratios of weights of each cluster. Then, a greedy algorithm is used to recommend movies. This algorithm, greedily chooses the items from the cluster which maximize the diversity *i.e.*, minimum amount similarity. Also, there is a restriction on the item relevance to the active user. Since, if the restriction on relevance is removed, although the algorithm might recommend diverse movies, it might consider the items which are very irrelevant to the active user.

DIVERSIFIER()

CandidatePredictions ← FinalPredictions + PredictabilityFactor
GETITEMCLUSTERS(items[ ])
for $i \leftarrow 0$ **to** n(item clusters) **by** 1
  for $i \leftarrow 0$ **to** n(similar users) **by** 1
    Sum ← Sum + Rating of all movies in the cluster i by the user j;

  ClusterAverage(i) ← $\frac{Sum}{n(ratings\ by\ similar\ users\ for\ all\ the\ movies\ in\ cluster\ i)}$

ClusterRatios ← GETCLUSTERRATIOS(ClusterAverages[ ])

for $i \leftarrow 0$ **to** n(item clusters) **by** 1
  for $j \leftarrow 0$ **to** n(items in cluster) **by** 1
    if CandidatePrediction[j] ≥ relevance cut-off
      CandidateItem[] ← Item[j]

  SORTONSIMILARITY(Candidate Items[ ])
  Recommend Items[] ← GETITEMSFROMCLUSTER(Candidate Items[ ])
return Recommended Items [ ]

The figure above shows the working of algorithm used for diversification in the current recommender system.

The issue of freshness is dealt in somewhat a random way here; predictability comes when already well known items are predicted again. Because of this, I treat each item equally and in the later stages by the use of pattern seen in user profile to those IMDB data, I put weights to parameters which affect the predicted rating of the movie. Here the number of votes given for a movie in IMDB and the number of users who have rated the movie in the current dataset are considered. The concept is the movie with majority of votes normally will be known to the user

is other ways rather than the recommender system. So using the weights as indirectly proportional to the number of votes can give a higher recommendation to movies which might have been seen the by user. Also, considering another perspective, the number of users who rate the movie in the current dataset gives a measure of the movie been watched and seen by the active user. However it would be even more interesting to see the number of similar users who have rated the movie, thus considering this intuition the following algorithm is used to deal with the issue of freshness.

Although the freshness component is built at a later part, this component is imbibed into the system as part of the predictor component, since diversifying and predictability are done in different ways, diversifying is done to produce a recommended set of items, but predictability factor is produced for each movie individually.

PREDICTABILITY()

GETIMDBVOTES(unrated items[ ], Predictions[ ]) SORTITEMSONIMDB-VOTES()
for $j \leftarrow 0$ **to** n(Sorted Items) **by** 1

if j $\leq$ 20
part1 $\leftarrow$ 1
else
if j $\leq$ 40
part1 $\leftarrow$ 2
else
if j $\leq$ 60
part1 $\leftarrow$ 3
else
if j $\leq$ 80
part1 $\leftarrow$ 4
else
part1 $\leftarrow$ 5
SORTITEMSONUSERRATINGS()
for $j \leftarrow 0$ **to** n(Sorted Items) **by** 1

if j $\leq$ 20
part2 $\leftarrow$ 1
else
if j $\leq$ 40
part2 $\leftarrow$ 2
else
if j $\leq$ 60
part2 $\leftarrow$ 3
else
if j $\leq$ 80
part2 $\leftarrow$ 4
else
part2 $\leftarrow$ 5

for $i \leftarrow 0$ **to** n(unrated items) **by** 1

PredictabilityFactor(i) $\leftarrow \frac{part1+part2}{2}$
return Predictability Factor

The complete algorithm of the recommender system is given below:

Recommender System

RECOMMENDATION SYSTEM()

GETTOPICS(*items*[ ])
CALCOSINESIM(*items*[ ])
InitialSet ← popular movies
RemSet ← TotalMovieSet − InitialSet
ActiveUser ← GETUSERRATING(*InitialSet*)
UserProfiles ← READUSERPROFILES()
while n(RemSet) ≠ 0
  SimilarUsers ← K-MEANCLUSTERER(UserProfiles, ActiveUser)
  cbPredictions ← CONTENT BASED()
  cfPredictions ← COLLABORATIVE FILTERING()
  timePredictions ← TIME-LINE()
  imdbPredictions ← IMDB()
  genrePredictions ← GENRE()
  FinalPredictions() ← α(cbPredictions) + β(cfPredictions) + γ(timePredictions) + δ(imdbPredictions) + λ(genrePredictions)
  Recommendations ← DIVERSIFIER(FinalPredictions[ ] + Predictability-Factor[ ])

  ActiveUser ← GETUSERRATINGS(Recommendations[ ])

COLLABORATIVE FILTERING(Similar Users, Active User)
for $i \leftarrow 0$ **to** n(unrated items) **by** 1
  for $j \leftarrow 0$ **to** n(Similar Users) **by** 1

    Prediction(i) ← Prediction(i) + $CosSim(u_j, u_a) * Rating(i, j)$

  Prediction(i) ← $\frac{Prediction(i)}{n(Similar\ user\ ratings\ for\ i)}$
return Predictions

CONTENT BASED()
for $i \leftarrow 0$ **to** n(unrated items) **by** 1
  for $j \leftarrow 0$ **to** n(Rated Movies) **by** 1

    Prediction(i) ← Prediction(i) + $CosSim(m_i, m_j) * Rating(j)$

  Prediction(i) ← $\frac{Prediction(i)}{n(Rated\ Movies)}$
return Predictions

TIME-LINE()

for $i \leftarrow 0$ **to** n(unrated items) **by** 1
  Movies $\leftarrow$ getMovies(rated movies with in timeline)
  for $j \leftarrow 0$ **to** n(Movies) **by** 1

    Prediction(i) $\leftarrow$ Prediction(i) + $CosSim(m_i, m_j) * Rating(j)$

  Prediction(i) $\leftarrow \frac{Prediction(i)}{n(Movies)}$
return Predictions

GENRE(similar users)

for $i \leftarrow 0$ **to** n(genres) **by** 1
  for $i \leftarrow 0$ **to** n(similar users) **by** 1
    Sum $\leftarrow$ Sum + Rating of all movies in the genre i by the user j;

  GenreAverage(i) $\leftarrow \frac{Sum}{n(ratings\ by\ similar\ users\ for\ all\ the\ movies\ in\ genre\ i)}$

for $i \leftarrow 0$ **to** n(unrated items) **by** 1

  GenrePredictions(i) $= \frac{\sum_j^{n(genres\ of\ i)} GenreAverages(j)}{n(genres\ of\ i)}$

return GenrePredictions

IMDB()

for $i \leftarrow 0$ **to** n(unrated items) **by** 1
  SumDiff $\leftarrow 0$
  for $j \leftarrow 0$ **to** n(rated items) **by** 1
    if $CosSim(m_i, m_j) < 0$
      Diff $\leftarrow r_{j_{imdb}} - r_j$
      if Diff < 0
        NewDiff $\leftarrow \frac{|(r_{j_{imdb}} - r_j)|}{CosSim(m_i, m_j)}$
      else
        NewDiff $\leftarrow (r_{j_{imdb}} - r_j) * |CosSim(m_i, m_j)|$
    else
      NewDiff $\leftarrow (r_{j_{imdb}} - r_j) * CosSim(m_i, m_j)$

    SumDiff $\leftarrow$ SumDiff + NewDiff

  IMDBPrediction(i) $\leftarrow$ IMDBRating(i) + $\frac{SumDiff}{n(rated\ items)}$

return IMDB Predictions

PREDICTABILITY()

GETIMDBVOTES(unrated items[ ], Predictions[ ]) SORTITEMSONIMDB-VOTES()
for $j \leftarrow 0$ **to** n(Sorted Items) **by** 1

```
   if j ≤ 20
      part1 ← 1
   else
      if j ≤ 40
         part1 ← 2
      else
         if j ≤ 60
            part1 ← 3
         else
            if j ≤ 80
               part1 ← 4
            else
               part1 ← 5
```

SORTITEMSONUSERRATINGS()
for $j \leftarrow 0$ **to** n(Sorted Items) **by** 1

```
   if j ≤ 20
      part2 ← 1
   else
      if j ≤ 40
         part2 ← 2
      else
         if j ≤ 60
            part2 ← 3
         else
            if j ≤ 80
               part2 ← 4
            else
               part2 ← 5
```

for $i \leftarrow 0$ **to** n(unrated items) **by** 1

PredictabilityFactor(i) $\leftarrow \frac{part1+part2}{2}$
return Predictability Factor

DIVERSIFIER()

CandidatePredictions ← FinalPredictions + PredictabilityFactor
GETITEMCLUSTERS(items[ ])
for $i \leftarrow 0$ **to** n(item clusters) **by** 1
  for $i \leftarrow 0$ **to** n(similar users) **by** 1
    Sum ← Sum + Rating of all movies in the cluster i by the user j;

  ClusterAverage(i) ← $\frac{Sum}{n(ratings\ by\ similar\ users\ for\ all\ the\ movies\ in\ cluster\ i)}$

ClusterRatios ← GETCLUSTERRATIOS(ClusterAverages[ ])

for $i \leftarrow 0$ **to** n(item clusters) **by** 1
  for $j \leftarrow 0$ **to** n(items in cluster) **by** 1
    if CandidatePrediction[j] $\geq$ relevance cut-off
      CandidateItem[] ← Item[j]

  SORTONSIMILARITY(Candidate Items[ ])
  Recommend Items[] ← GETITEMSFROMCLUSTER(Candidate Items[ ])
return Recommended Items [ ]

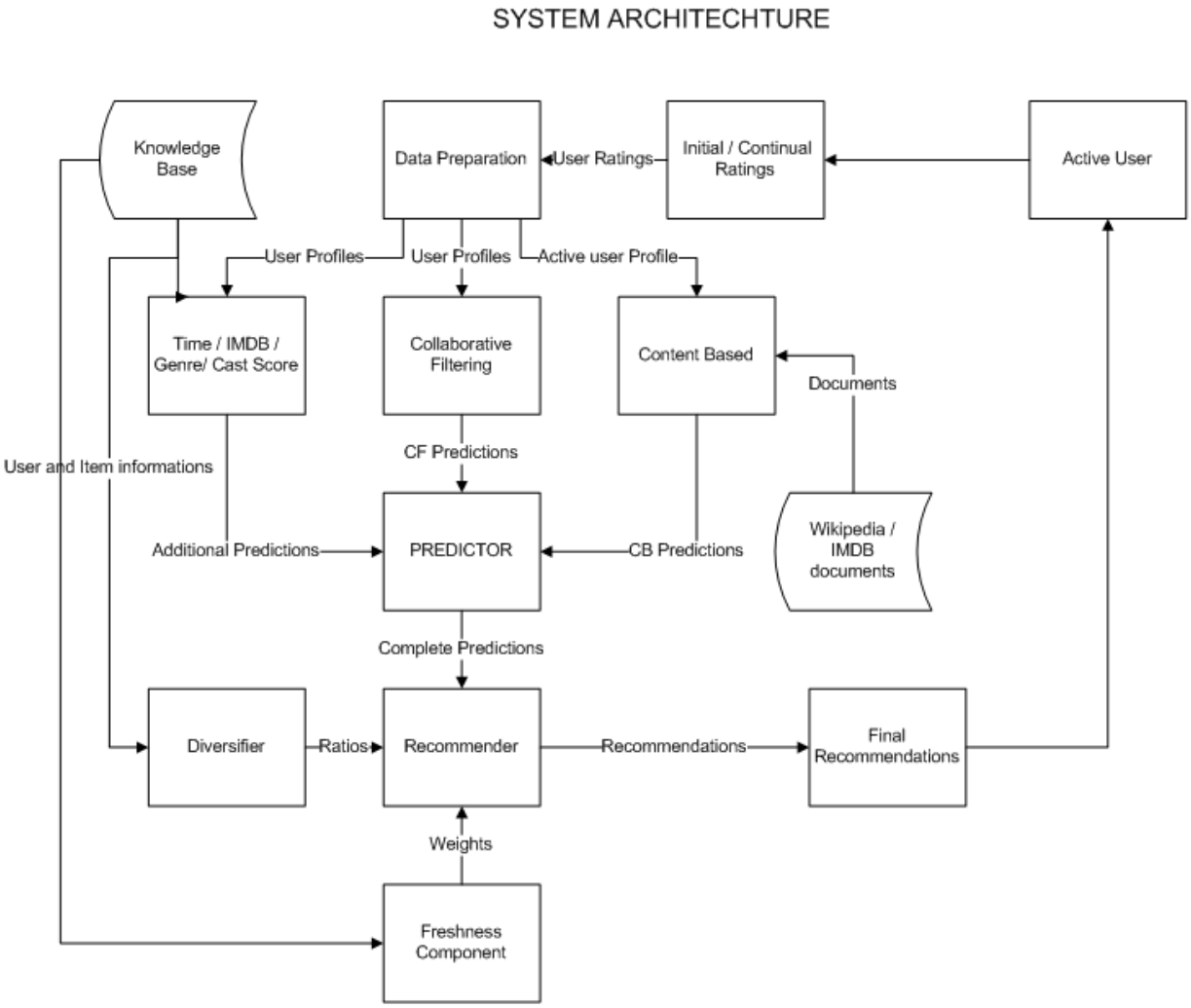

Figure 4-4: *System Architecture*

5 Experimental Setup

In this Chapter, we describe the dataset used in this work and the experiments designed to calculate the best-fit for choosing the number of topics using LDA and the reasons for choosing the cluster sizes for finding similar users. Some experiments are done to evaluate the approach and design of the recommender system and describe evaluation metrics used for evaluation. This Chapter is organized as follows: In Section 5.1, the experiments conducted to choose the number of topics for LDA is discussed. In Section 5.2, the reasons behind the cluster sizes are discussed. In Section 5.3, various evaluation metrics of the recommender system are recommender system. In section 5.4, experiments conducted for evaluating the recommender system are discussed.

5.1.LDA, No. of Topics.

Using LDA one can represent the documents of text in terms of topic probabilities. However choosing the number of topics can be an issue, since choosing a very high number of topics can give you fine grained representation of the documents, but it can also lead to lots of noise. Since, some of the topics can be unrelated and unnecessary.

For choosing the number of topics, some experiments were conducted. In the experiments, for each number of topics each item is represented by a vector of the topic probability and each item are given a class based on the Genre it belongs to. To construct a balanced data set for testing, the genre ‘drama’, which actually classifies the items into an almost 50-50 balanced dataset, is used.

The intuition behind choosing such an experiment was that, the most ideal number of topics would generally give an ideal description of each item. Thus the more ideal the descriptions better the accuracy in classifying the items in to classes.

This dataset is given to five different classifiers, to check the accuracy with changing number of topics; the results for these experiments are shown in Figure 5-1:

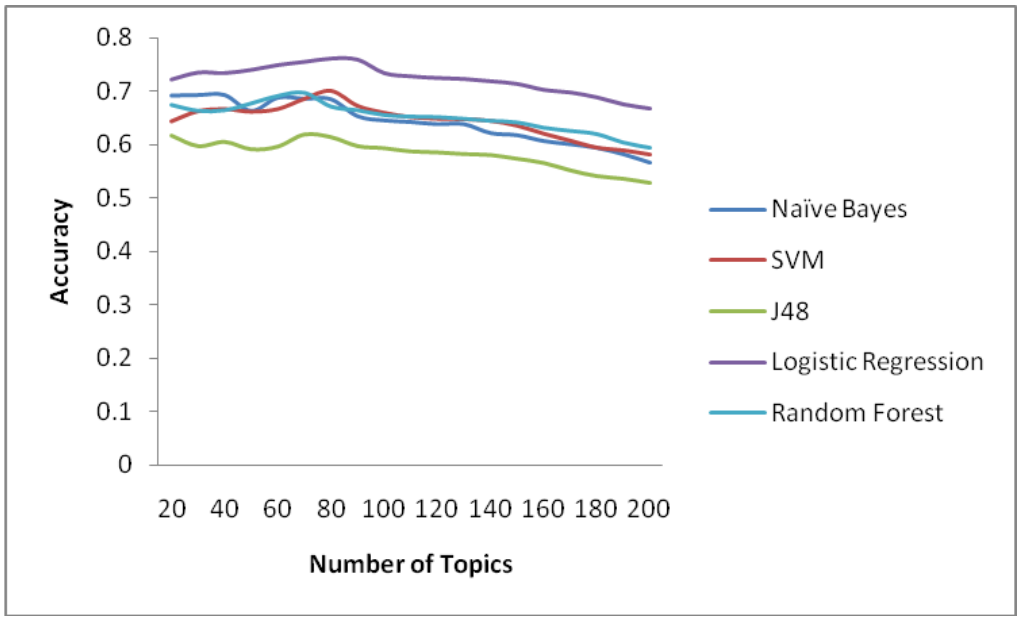

Figure 5-1: *LDA - no. of topics to Accuracy, with different classifiers like Naïve Bayes, Support Vector Machine (SVM), J48 (), Logistic Regression, Random Forest.*

The dataset was run on an 80-20 % split for training and testing. The results show the variance in accuracy with changing number of topics. From the experiments, it can be evaluated that most of the classifiers show a peak in the topic number ranging from 60-80, but clearly 4/5 classifiers have a peak at the value of ~80, because of these results, the number of topics ideally suited to represent the items (movies) is chosen to be ~80.

Also, the classifiers used here are the implementations from the Waikato Environment for Knowledge Analysis (WEKA). It came about through the perceived need for a unified workbench that would allow researchers easy access to state-of-the-art techniques in machine learning [55].

5.2. Cluster Sizes

5.2.1. Collaborative Filtering

Choosing the cluster sizes is very important, since if there is no restriction on the number of users who are similar to the active user, then the cluster might contain users who actually might be dissimilar (*i.e.*, very less similarity). To solve this, the average numbers of users who

have a rated a movie is taken. On an average, given the dataset, the average number of users who have rated a random movie is around ~59.4. So the restriction is to cluster recursively until the total number of similar users found is utmost 60.

The intuition behind this is '*too much of a good thing can turn bad*'. If the cluster size is too large, then the possibility of finding users who hold very little similarity to the active user is considerably high. This can result in decreasing the predicted rating of movies which the active user might like and can increase predicted ratings of other movies.

5.2.2. Diversifier

In the system, the diversifier component also clusters items to get cluster weights. Here there is no restriction on the cluster sizes but the number of clusters. K-Means clustering algorithm requires the number of clusters to be defined clearly, so to choose the number of clusters, some experiments have been done with varying number of clusters. The experiment is use the topic vectors with ~80 as the topic size as the input and for varying number of clusters, the average similarity between the cluster centers is calculated.

The result of the experiment is shown below.

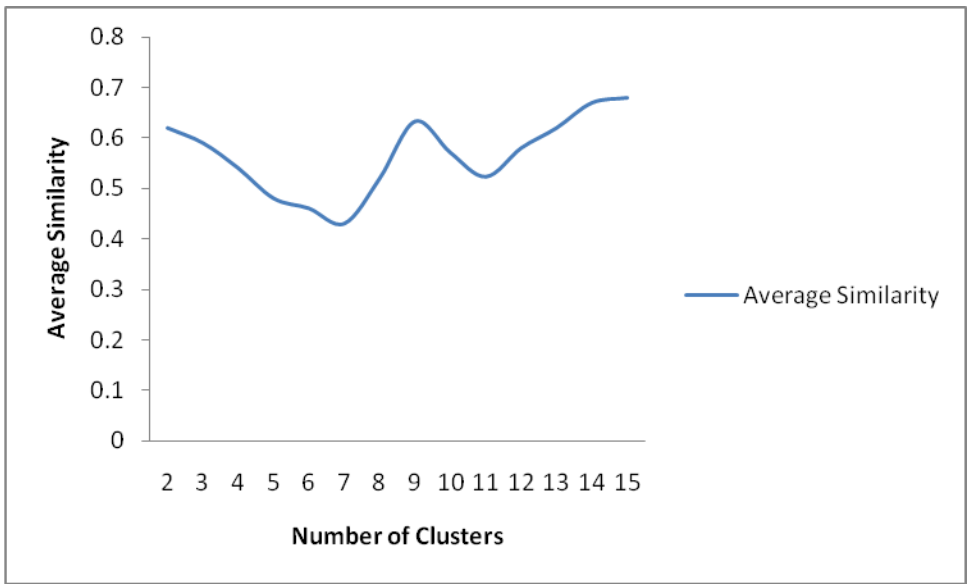

Figure 5-2: *Average Similarity of Cluster centers to Number of Clusters*

From the graph, it is clear with varying number of clusters, the items which are clustered based on similarity vary. From the graph, it is clear that around 7 clusters are needed to ideally, get the best distribution of items based on similarity.

5.3.Evaluation Metrics

There have been many metrics in the past which have been used for evaluating recommender systems.

Mean Absolute Error: The main purpose of recommender system is to successfully predict the items with same amount of similarity to the active user. This main goal of the recommender system is measured through the Mean Absolute Error (MAE). This metric allows the base recommender algorithm to be successfully validated with other recommender algorithms. There have been many evaluation metrics which have been done off-line using test and train datasets. Here in the current recommender system, the calculation of MAE is done off-line using test and train datasets.

The next evaluation metric is the goal of the recommender system, once the base recommender system is measured above the accuracy threshold; it has to be evaluated based on the goal, on which the recommender algorithm is developed. So, the next metric is to measure the diversity of the system. This is done by introducing a new measure 'Average Similarity' (AS). This is measured for the recommended items as the average of the similarity of items recommended. The calculation of this measure is done off-line using test and train datasets, but it can be done online or offline.

Evaluating the Freshness component of current recommender system is an open-problem. Since, to evaluate whether a item recommended is fresh for the user or not cannot be done off-line and even online it has to been done with keeping track of user profiles, and asking the user explicitly of the freshness factor. This is a limitation of this current recommender system, since this recommender system is developed using an assumption of static data set.

5.4. Evaluation - Experiments.

Here the recommender system has 5 parameters, which change the values of the predictions and the recommendations considerably. So for evaluating the best fit recommender system model, one has to find the best (ideal) parameters which make the recommender system to be most effective. These five parameters are the weights for each component of the predictions: Content-based prediction, Collaborative Filtering prediction, Time-Line prediction, IMDB prediction and Genre Predictions.

- *Measuring MAE (Mean Absolute Error):*

The experiments used to measure the MAE, consists of varying input values to find the best parameters and the MAE for each of the parameter set. There are a total of five test and train data sets. Each of the datasets used for testing is a 90-10, 80-20, 70-30, 60-40, and 50-50 split of the original complete dataset. These set of experiments is called as Experiment Set 1.

Another point to be noted is that, this evaluation is done, without considering the diversity component of the system. Since, this metrics is to evaluate the actual base recommender algorithm; the evaluation is done without involving both diversifier component and the freshness component.

- *Measuring AS (Average Similarity):*

The experiments used to measure the AS are pretty much similar to the measuring of MAE. AS, is measured as the average of similarities of the items that are recommended. However once the MAE for the recommender system is found to be within the threshold, we can use the same recommender system measure the diversity factor of the system. These set of experiments are referred as Experiment Set 2.

These experiments are done on a recommender system which holds the threshold cut-off MAE and the experiments are not done on varying splits of datasets but an 80-20 split of dataset. The intuition is that once the recommender system can make descent predictions, then evaluating it for a particular dataset is sufficient to understand its diversity factor rather to understand at each and every split.

Another point to be noted is that, the freshness component which was not included in the experiments conducted to measure MAE is used here, since diversifier component is the last component which makes the recommendations. So, the system has to be evaluated on the basis of all the working components.

6 Evaluations and Results

In this Chapter, using the experimental set-up mentioned in the previous Chapter, the methodology is evaluated using various experiments. In section 6.1, the experiments conducted to evaluate the recommender system based on Mean Absolute Error are discussed and results of those experiments are given. In section 6.2, the experiments conducted to evaluate the recommender system, based on how it tackles the diversity issues are discussed; the results of those experiments are also shown. In section 6.3, the research questions which this recommender system answers are documented.

6.1.Evaluating based on MAE

This section concentrates on evaluating the base recommender algorithm developed in this thesis work using a cross validation approach (section 6.1.1) and to further support the claim of meeting the thesis goal experiments done are noted (section 6.1.2) and comparison (section 6.1.3) of the base recommender algorithm experimental results with other algorithms using the same data set.

6.1.1 Experiment Set 1

The first set of experiments is used to fine tune the parameters which can effectively produce a recommender system, which can produce the threshold cutoff which was put as a goal of the recommender system in Chapter 1. For this purpose the approach of cross validation is used. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the *training set*), and validating the analysis on the other subset (called the *validation set* or *testing set*).

Here 10-fold cross validation is used to evaluate the parameters for the recommender system. In 10-fold cross validation the complete data set is randomly split into 10% subsets and in each round of evaluation 9 splits are used as training set and the remaining one split is used as testing split.

The parameter sets used in these experiments are given below:

α **Content-based**	**β** **Collaborative Filtering**	**γ** **Time Prediction**	**δ** **IMDB Prediction**	**λ** **Genre Prediction**
0.7	0.1	0.05	0.1	0.05
0.6	0.2	0.05	0.1	0.05
0.5	0.3	0.05	0.1	0.05
0.4	0.4	0.05	0.05	0.1
0.4	0.4	0.05	0.1	0.05
0.4	0.4	0.1	0.05	0.05
0.3	0.5	0.05	0.1	0.05
0.2	0.6	0.05	0.1	0.05
0.1	0.7	0.05	0.1	0.05
0.1	0.1	0.1	0.6	0.1

Table 6-1: *Experimental Parameter Set*

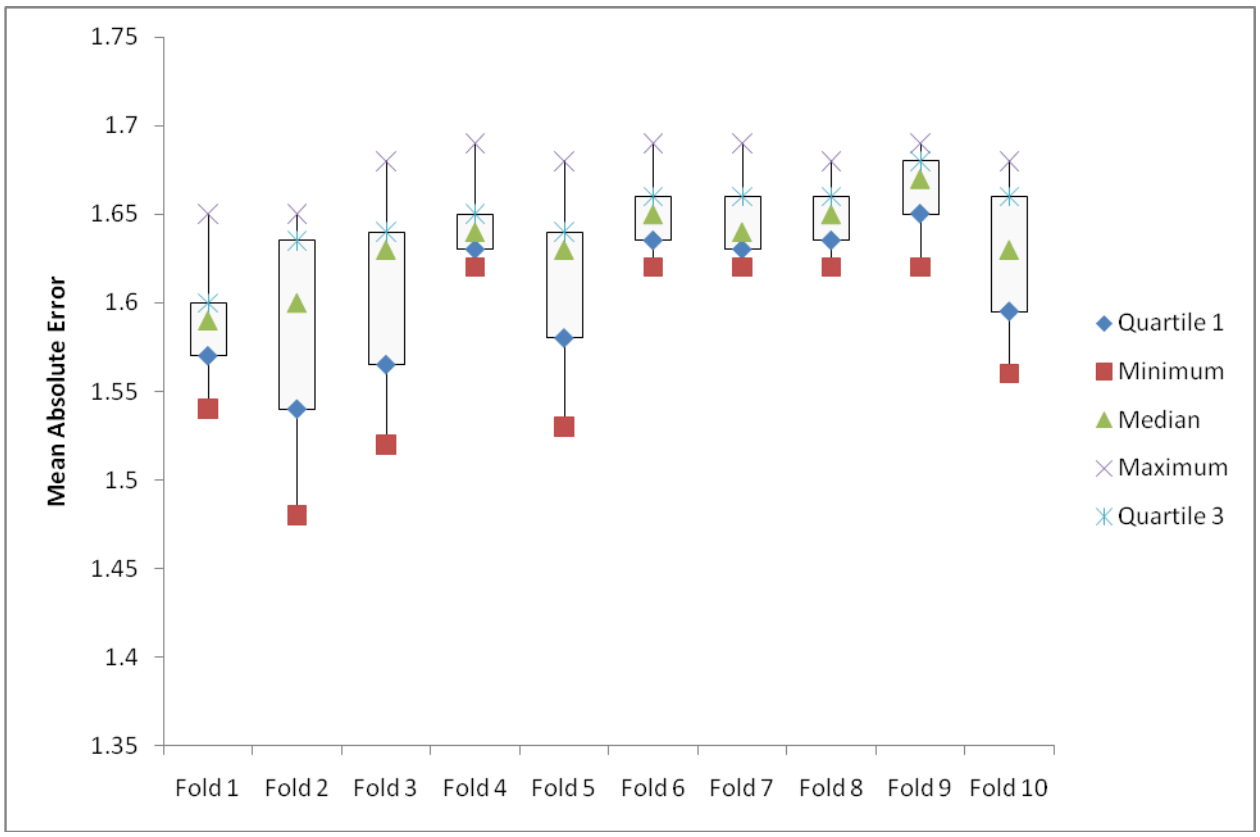

Figure 6-1: *Cross Validation Plot for the parameter set ($\alpha = 0.7$, $\beta = 0.1$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.57 to 1.67 and clearly this is more than the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set is needed.

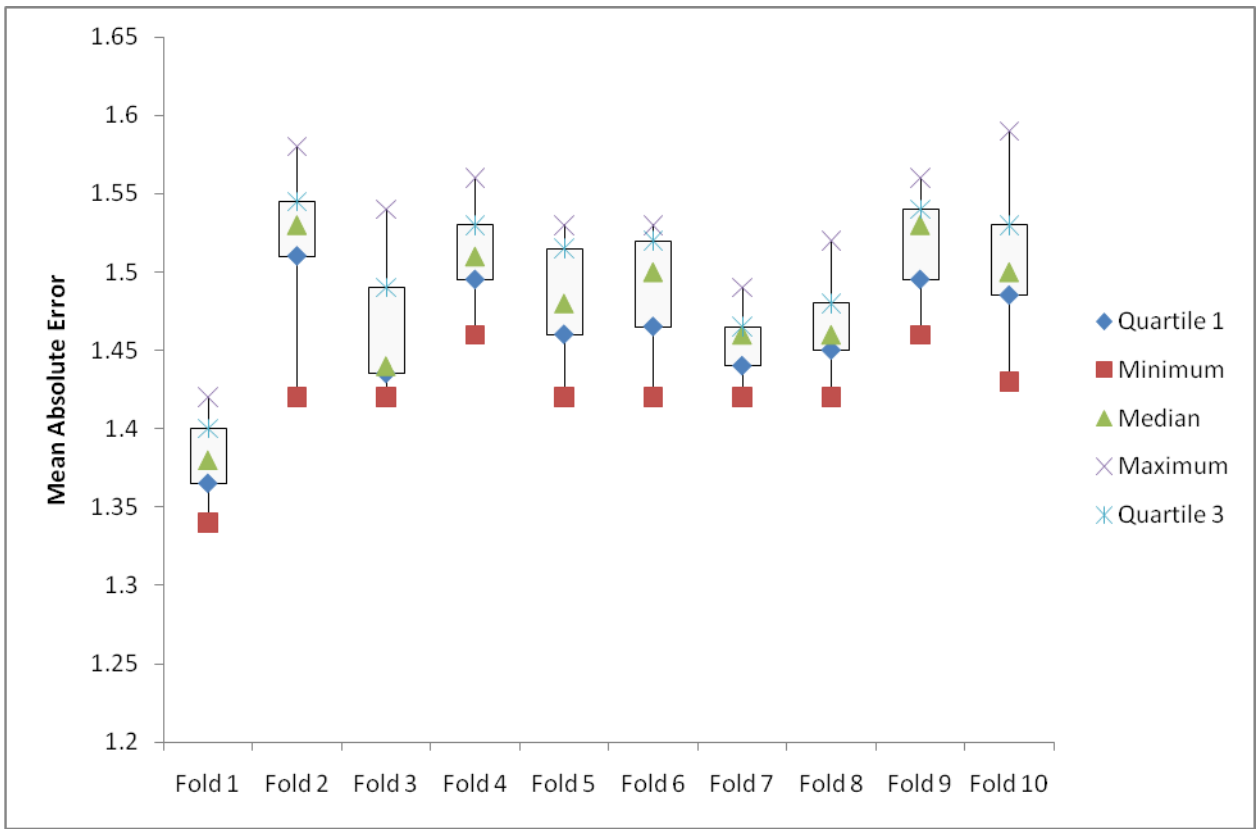

Figure 6-2: *Cross Validation Plot for the parameter set ($\alpha = 0.6$, $\beta = 0.2$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.40 to 1.55 and clearly this is more than the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set is needed.

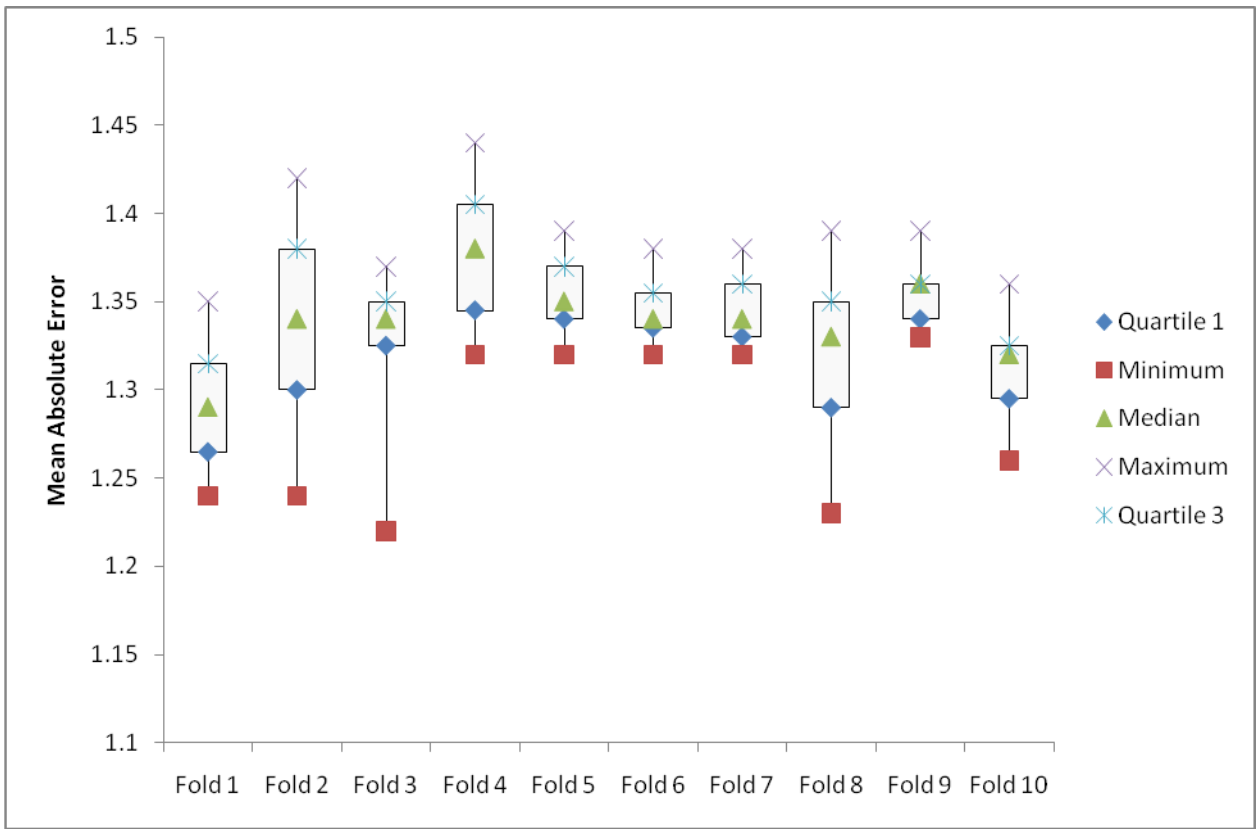

Figure 6-3: *Cross Validation Plot for the parameter set ($\alpha = 0.5$, $\beta = 0.3$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.28 to 1.38 and this is within the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set could prove to produce better results.

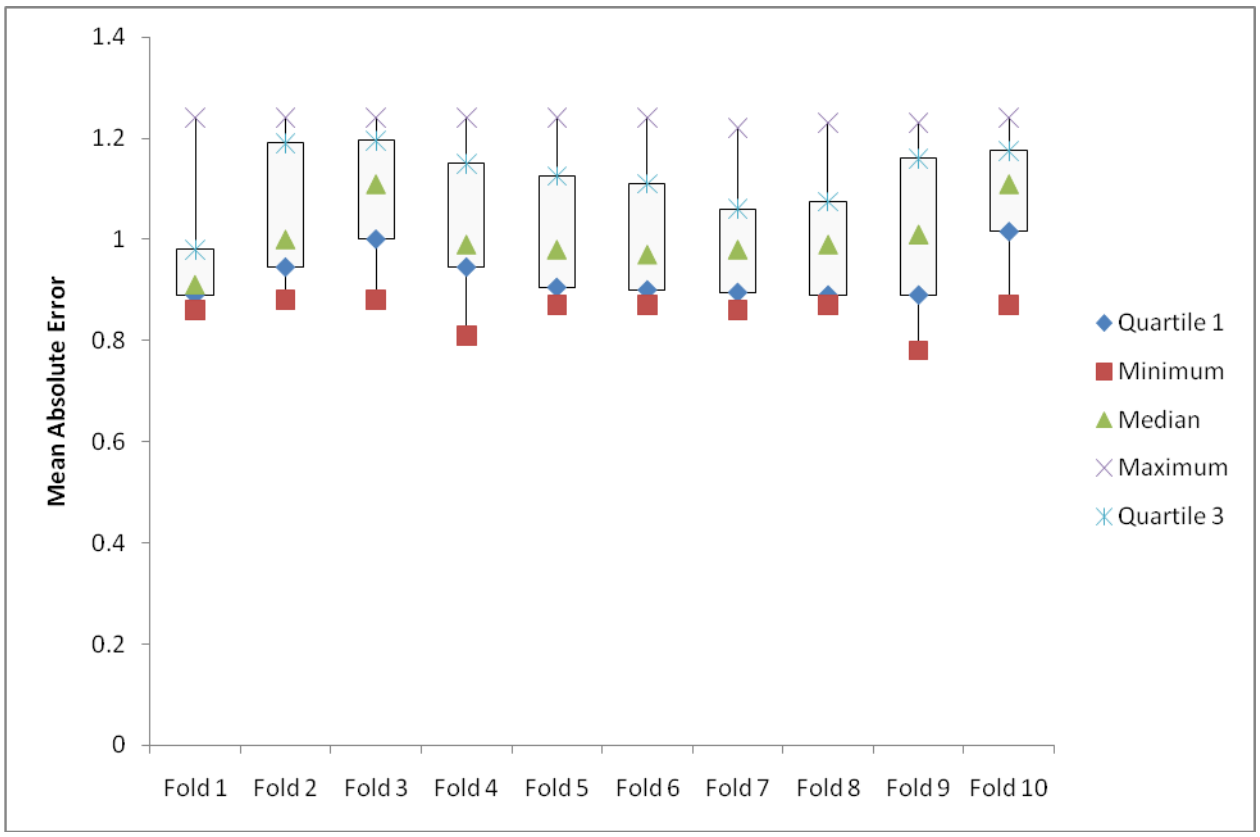

Figure 6-4: *Cross Validation Plot for the parameter set ($\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.05$, $\delta = 0.05$ and $\lambda = 0.1$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 0.90 to 1.1 and this is within the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set could prove to produce better results.

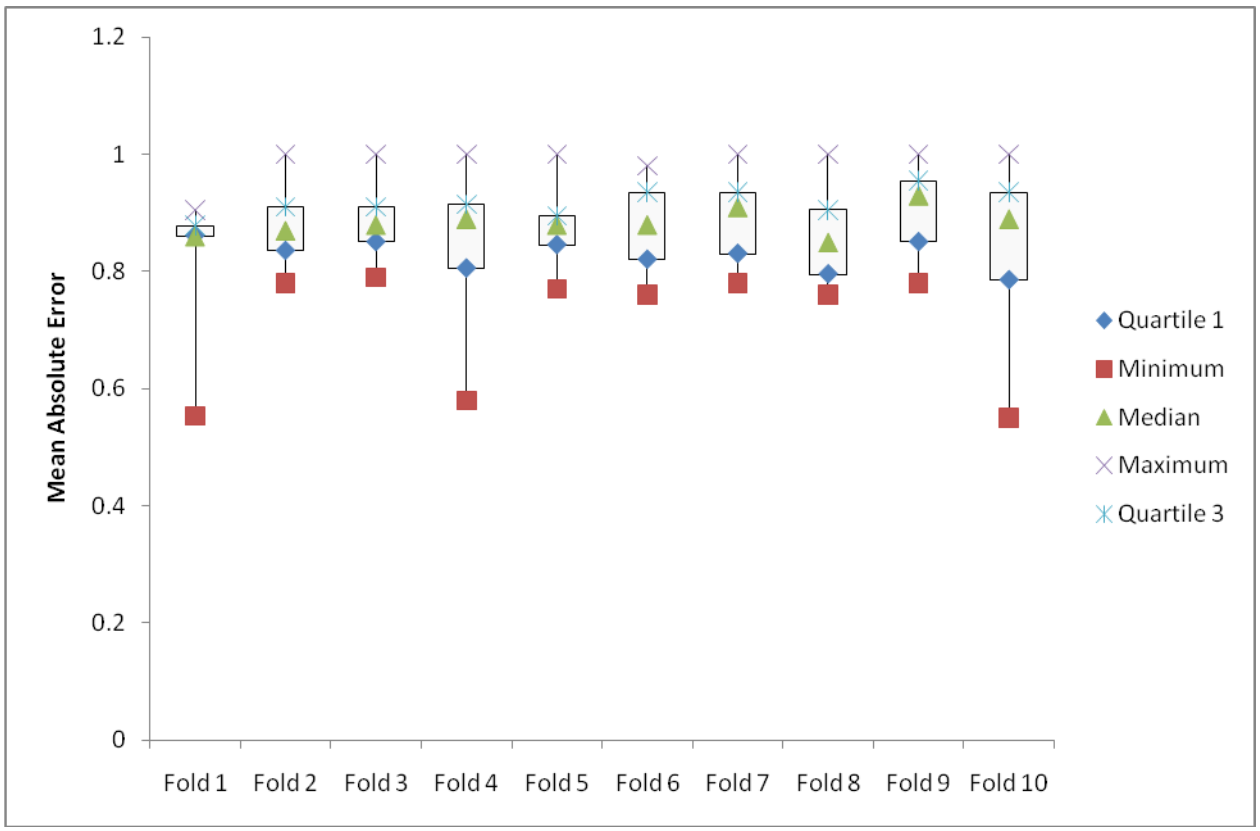

Figure 6-5: *Cross Validation Plot for the parameter set ($\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 0.9 to 0.95 and this is within the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set could prove to produce better results.

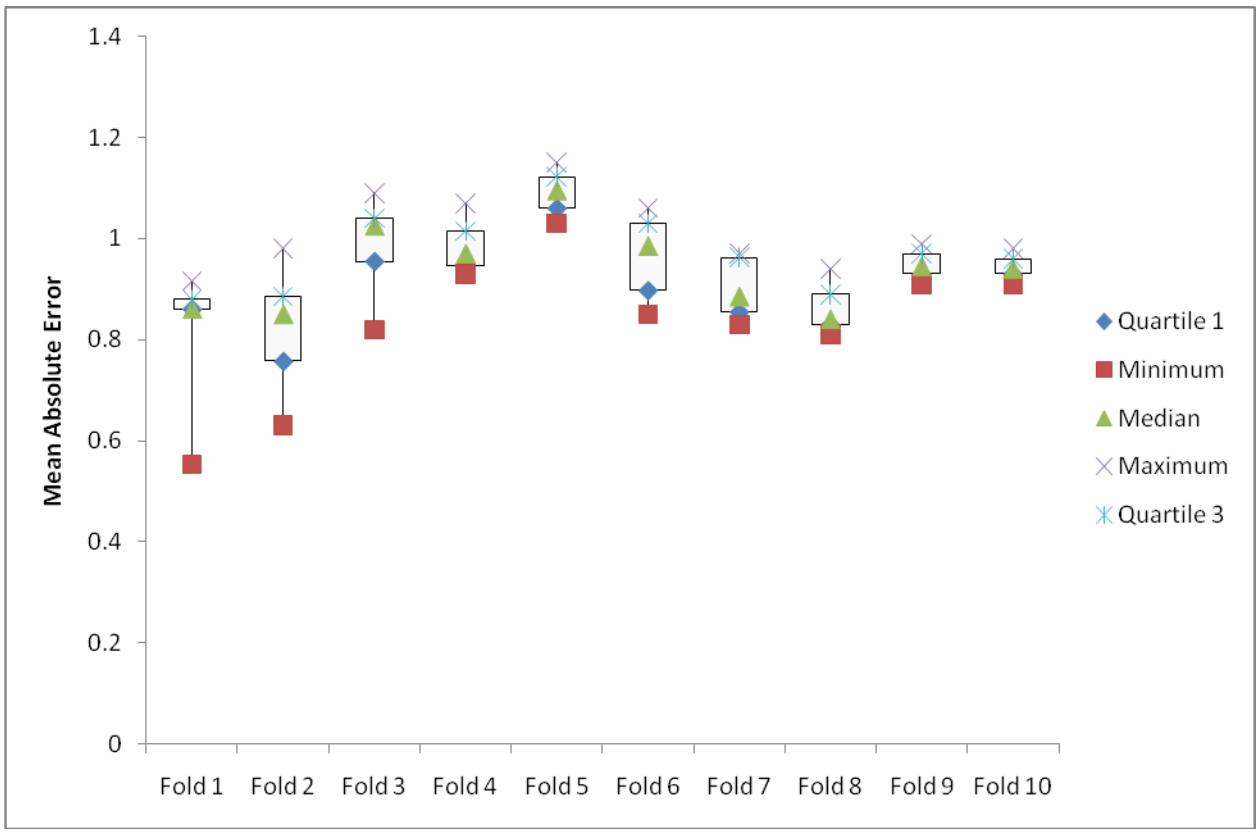

Figure 6-6: *Cross Validation Plot for the parameter set ($\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.1$, $\delta = 0.05$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 0.9 to 1.1 and this is within the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set could prove to produce better results.

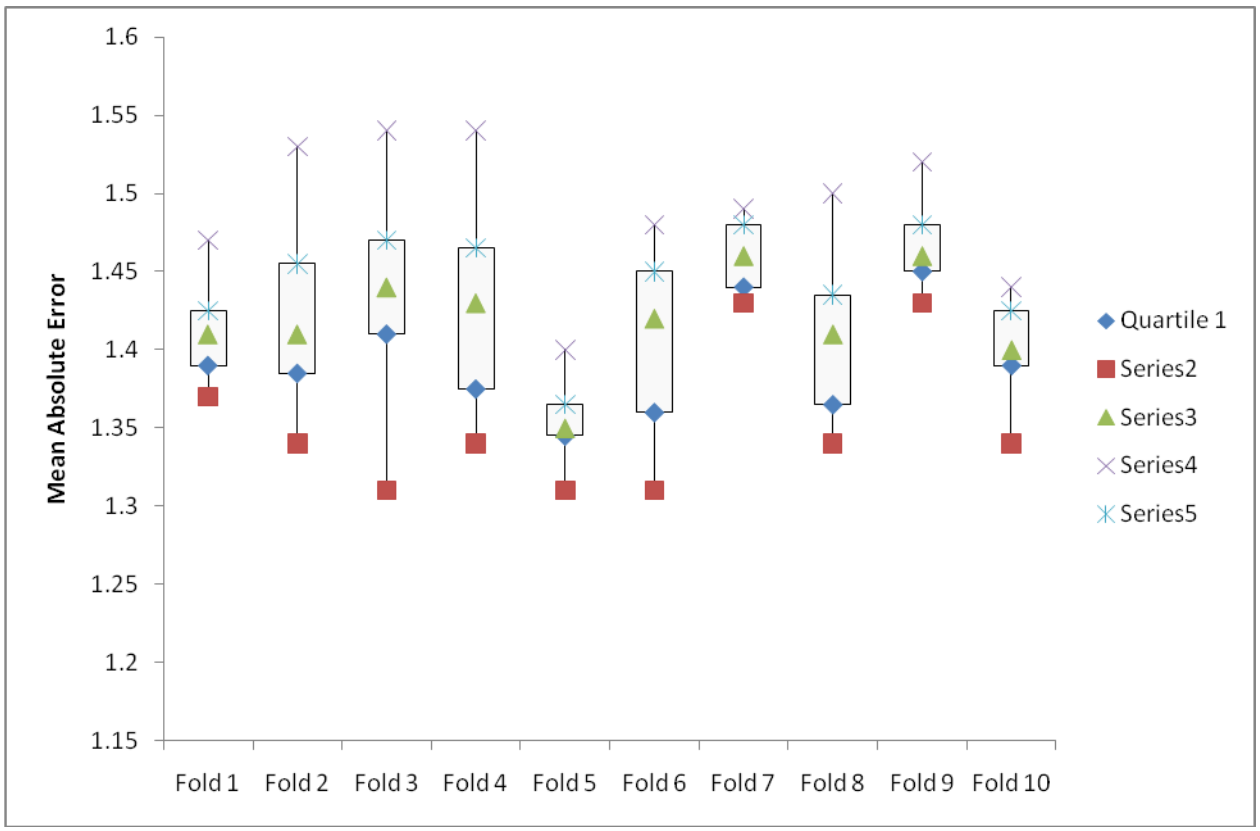

Figure 6-7: *Cross Validation Plot for the parameter set ($\alpha = 0.3$, $\beta = 0.5$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.35 to 1.46 and this is within the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Further tampering with the parameter set could prove to produce better results.

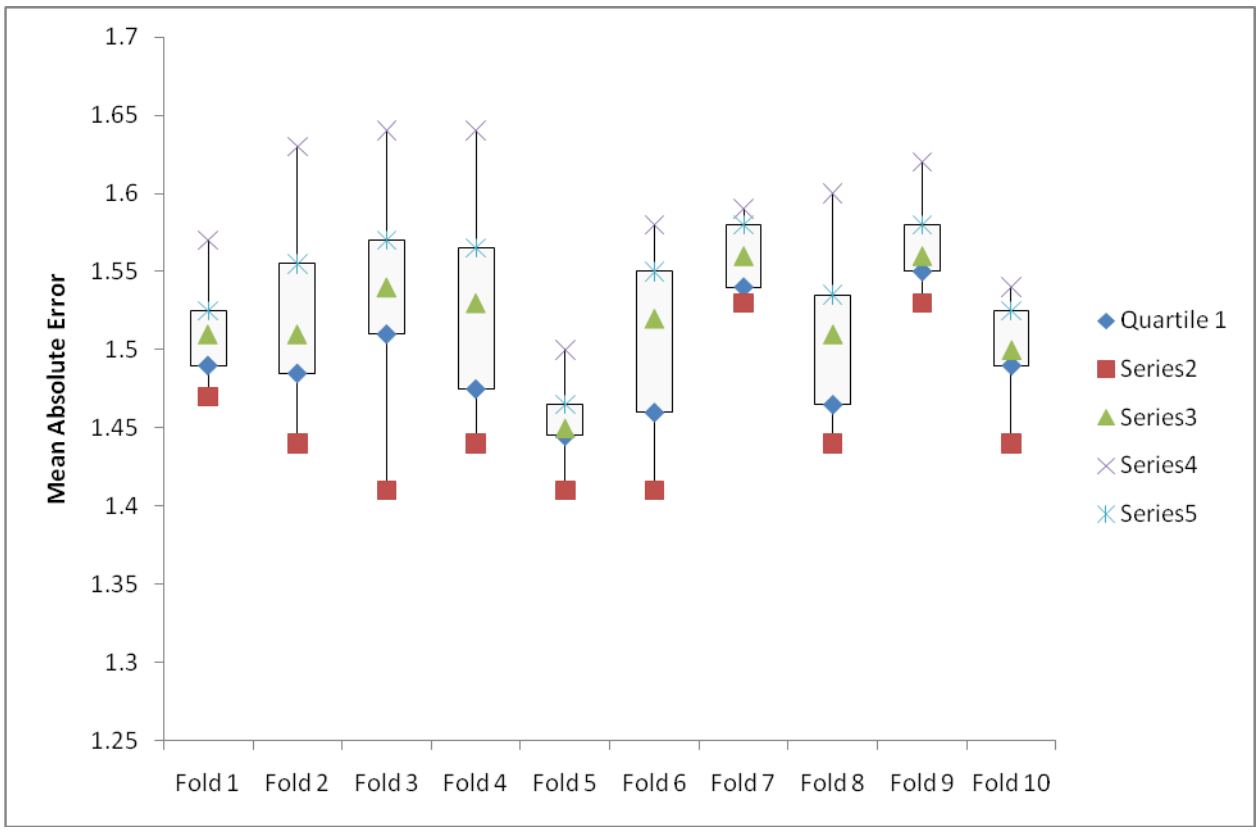

Figure 6-8: *Cross Validation Plot for the parameter set ($\alpha = 0.2$, $\beta = 0.6$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.45 to 1.55 and this is within the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy.

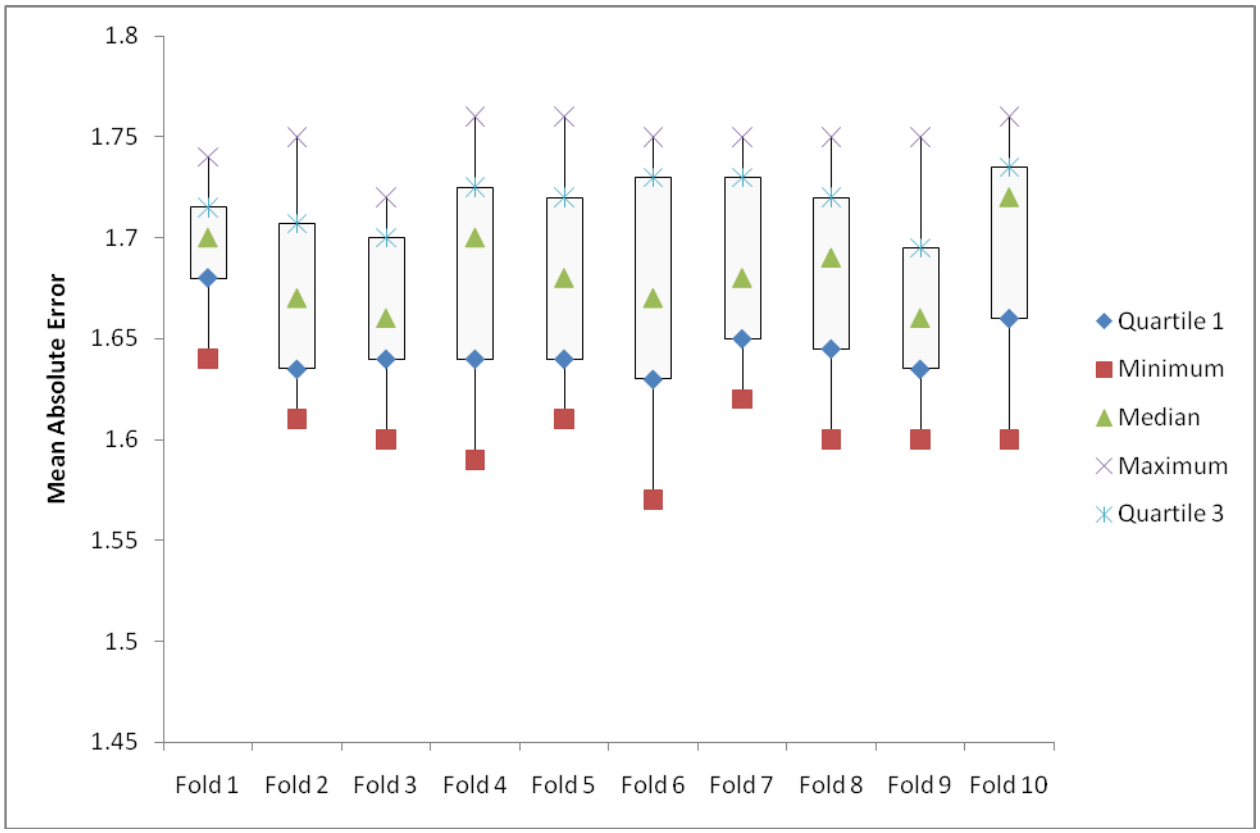

Figure 6-9: *Cross Validation Plot for the parameter set ($\alpha = 0.1$, $\beta = 0.7$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.66 to 1.72 and this is clearly more than the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy.

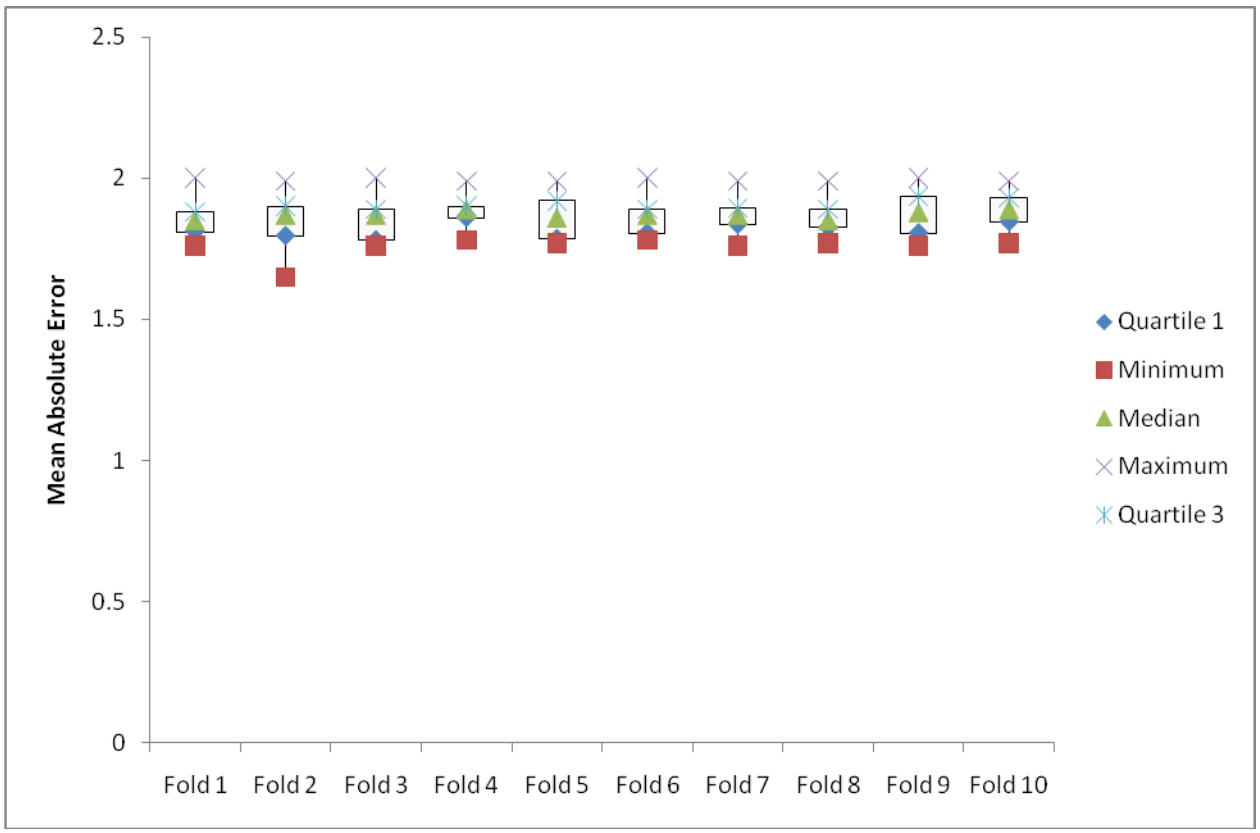

Figure 6-10: *Cross Validation Plot for the parameter set ($\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0.1$, $\delta = 0.6$ and $\lambda = 0.1$) with Folds to Mean Absolute Error.*

From the box plot above, the cross validation folds are drawn against mean absolute error for each fold. From the graph it is clear that for most of the instances in all the plots range from a 1.80 to 1.95 and this is clearly more than the threshold cutoff which was put up as a goal for the recommender system in terms of accuracy. Also, this parameter set was used to show the importance of base predictors over the additional predictors and also to prove that base predictors need to be given more weight when compared to the other additional predictors to get better accuracy.

From the experimental results given above, indeed some of the parameter sets do produce the base recommender algorithm, which can gold the threshold cutoff which was given in this thesis goal.

6.1.2 Experiment Set 1.2

To further evaluate the recommender system with parameter sets, which are evaluated in the previous experiments, a new set of experiments are used to evaluate the recommender system based on MAE (Mean Absolute Error) responsiveness of the system, the system is asked to predict on various datasets. These datasets are generated randomly, to represent x-y % splits of the complete dataset and one part is considered for training (knowledge base) and another part for testing. Care is taken to generate the datasets in such a way that they consist of all the users in the training dataset and also in the test dataset.

The various sets of parameter values used on the recommender system are given below:

Train Dataset (Number of Ratings)	**Test Dataset (Number of Ratings)**
90,000	10,000
80,000	20,000
70,000	30,000
60,000	40,000
50,000	50,000

Table 6-2: *Experimental Splits (Data Sets).*

To test the current recommender system, just running experiments using the datasets is not feasible since, the recommender system is completely dependent on the parameters which determine the weights of the various components like content-based, collaborative filtering *etc.,* Because of this approach, the recommender systems makes various prediction depending on the values of the parameters. So a various list of experiments are run using various parameter values.

The various sets of parameter values are given below:

α **Content-based**	**β** **Collaborative Filtering**	**γ** **Time Prediction**	**δ** **IMDB Prediction**	**λ** **Genre Prediction**
0.7	0.1	0.05	0.1	0.05
0.6	0.2	0.05	0.1	0.05
0.5	0.3	0.05	0.1	0.05
0.4	0.4	0.05	0.05	0.1
0.4	0.4	0.05	0.1	0.05
0.4	0.4	0.1	0.05	0.05
0.3	0.5	0.05	0.1	0.05
0.2	0.6	0.05	0.1	0.05
0.1	0.7	0.05	0.1	0.05
0.1	0.1	0.1	0.6	0.1

Table 6-3: *Experimental Parameter Set*

The intuition behind choosing parameter values in a manner as above, is that since all the components try to normalize the ratings in such a way that they are in the range of 1-5, which is the base rating range of the dataset, to keep the predictions in range and not to over fit the dataset, the parameter set is chosen such that the sum of all parameters is 1.0. Clearly, we know that the main two base predictors are Content-based and Collaborative Filtering; hence these two must have a combined weight which is greater than the sum of the rest of the parameters. The last experimental parameter set, is an example to show, what happens if the baseline predictors are violated to consider the additional parameters. Other than that, most of the parameter sets are constructed in a manner to get a pattern on how the MAE varies with changing parameters and also, this is a trial and error process. Because of this the recommender system may or may not

give the best or worst MAE of possible MAE values, but this definitely gives a range on how the MAE varies with changing parameter set.

Next there are 50 experiments which are conducted, using the parameter sets given above and 5 different data splits given above. For, each of the parameter set a 'MAE to number of users' graph is constructed for each of the data set. Also, these experiments are done without considering the freshness and diversity component, since this is the evaluation of the base recommender algorithm.

The results of the all the experiments and discussions of each of the result are given below:

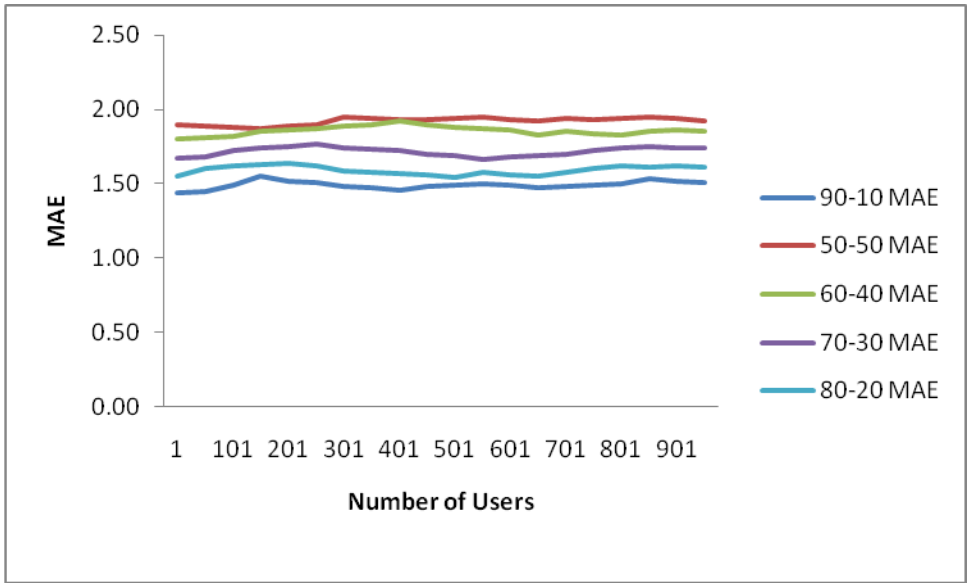

Figure 6-11: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.7$, $\beta = 0.1$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

From the graph, it is clear that the MAE clearly seems to increase as the dataset used to train decreases. This can be attributed by two possible reasons, one is that since the collaborative filtering component needs to find cluster of similar users, with decreasing datasets, the user-profiles seem to shorten and thus giving clusters of users who might actually be differing in opinion to an extent and another reason for such a behavior is that, with decreasing dataset the active user profile also decreases, thus reducing the possibility of a accurate prediction by the content-based component, which infers predictions using the cosine rule from the previous

ratings of the active user. These two seem to be the main reasons behind such a behavior. Also, one important aspect which needs to be discussed is that the average drop in MAE for each decrease train dataset is about ~0.07, which signifies the weight of collaborative filtering component; this behavior is further discussed with varying parameter set. Also, the range of MAE for all the datasets combined is ~1.45 to ~1.85. These results are not the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity.

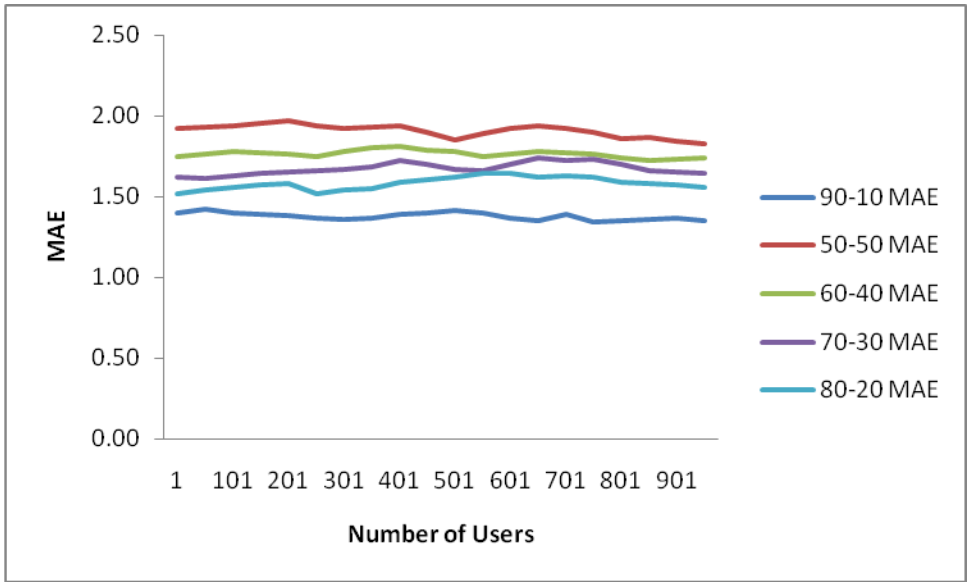

Figure 6-12: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.6$, $\beta = 0.2$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.09, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~1.35 to ~1.80. These results are not the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness.

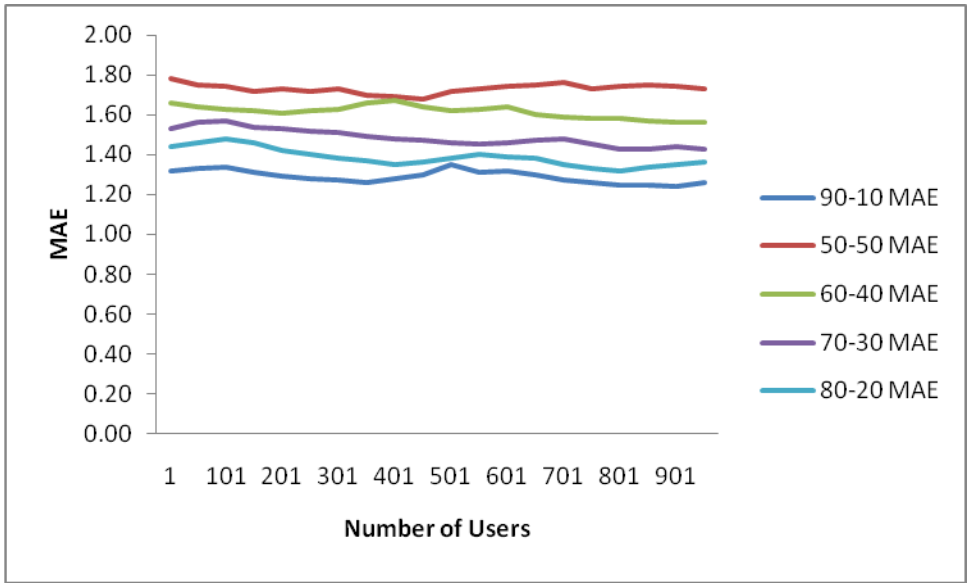

Figure 6-13: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.5$, $\beta = 0.3$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.10, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~1.30 to ~1.80. These results are not the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness.

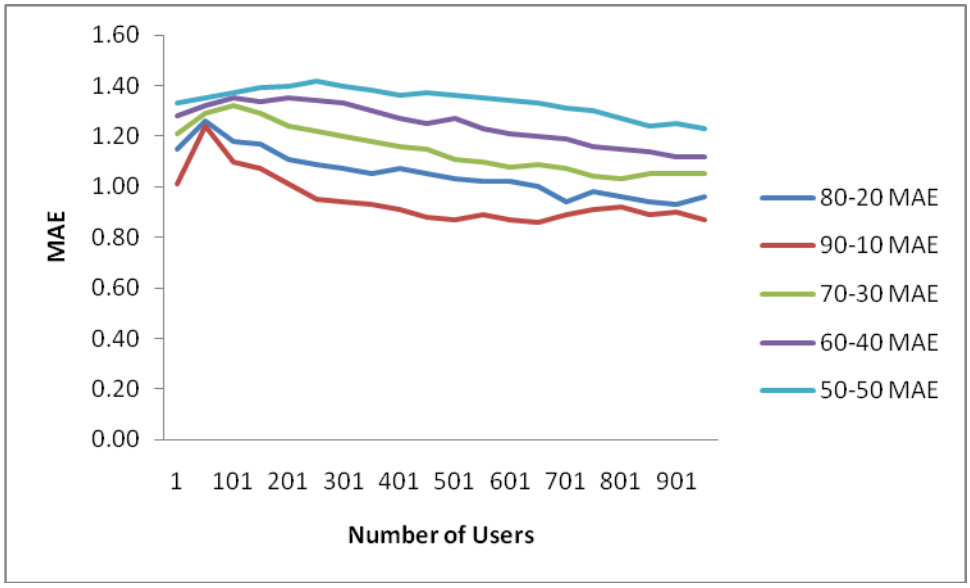

Figure 6-14: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.1$, $\delta = 0.05$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.11, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~0.85 to ~1.40. These results are in the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness. So further tampering with the parameter set with varying additional parameters could prove to show better results.

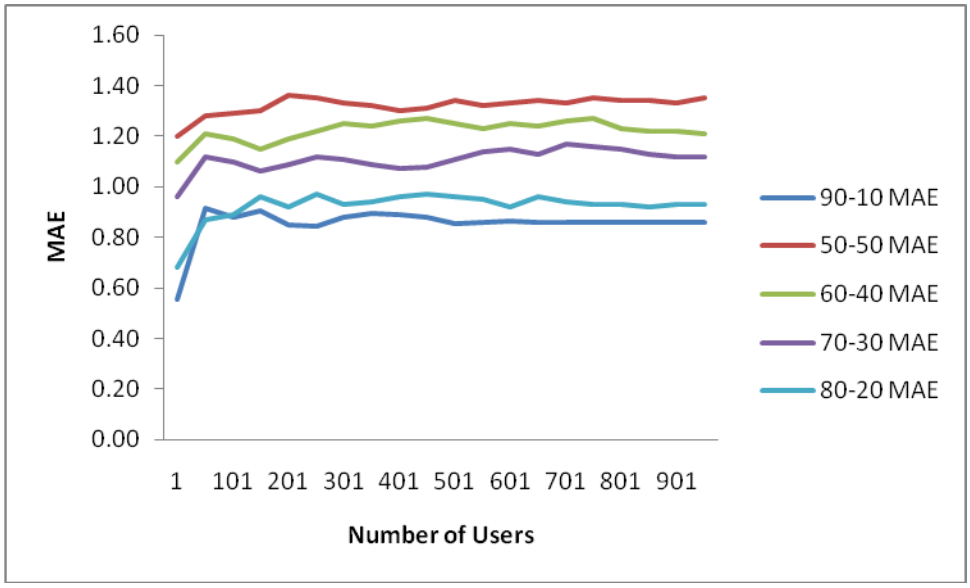

Figure 6-15: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.1, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~0.60 to ~1.35. These results are in the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness. So further tampering with the parameter set with varying additional parameters could prove to show better results.

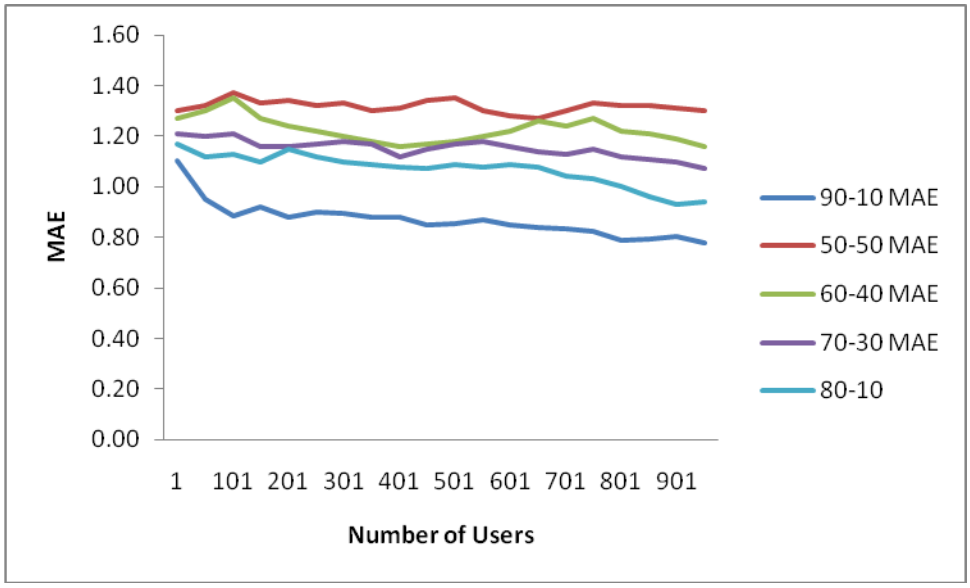

Figure 6-16: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.05$, $\delta = 0.05$ and $\lambda = 0.1$)*

The average drop in MAE is about ~0.9, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~0.80 to ~1.40. These results are in the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness. Since, tampering with the additional parameters doesn't show any significant decrease in MAE for any of the datasets, it can be concluded that the baseline predictors are the main weights which alter the MAE majority of Users, which implicitly signifies the fact that the additional parameters are used to handle unique users in most cases and also to capture a part of user preferences.

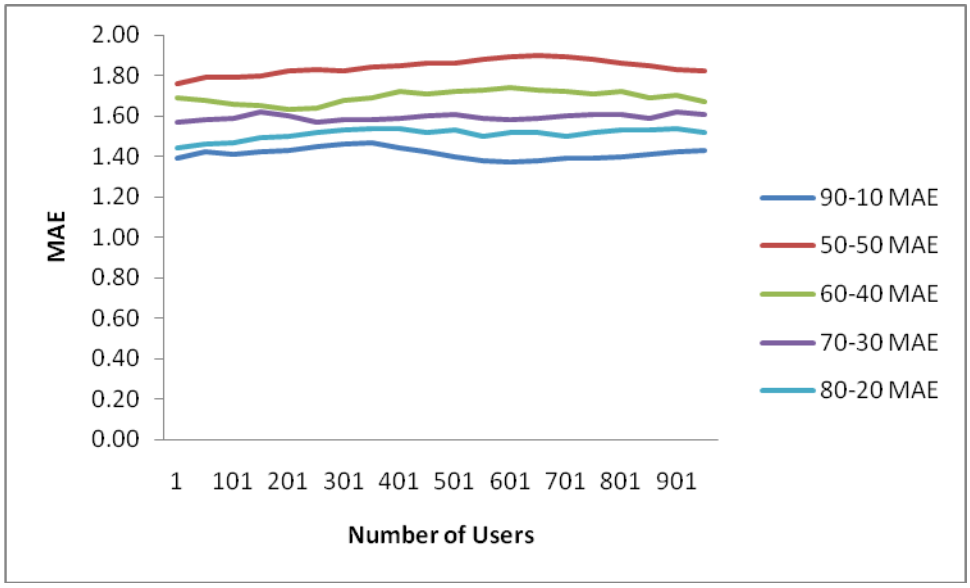

Figure 6-17: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.3$, $\beta = 0.5$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.9, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~1.35 to ~1.85. These results are not in the threshold cut-off.

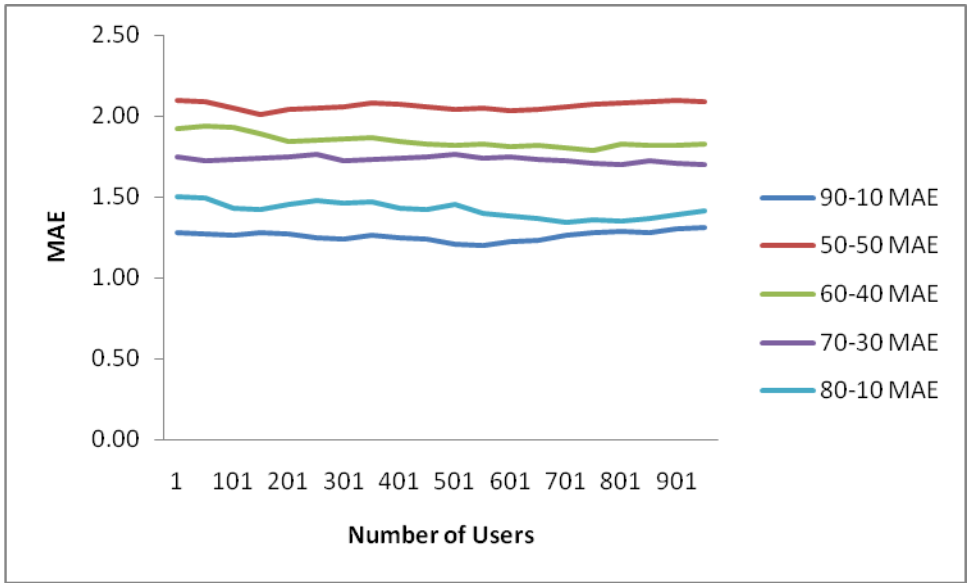

Figure 6-18: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.2$, $\beta = 0.6$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.11, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~1.35 to ~2.10. These results are not in the threshold cut-off.

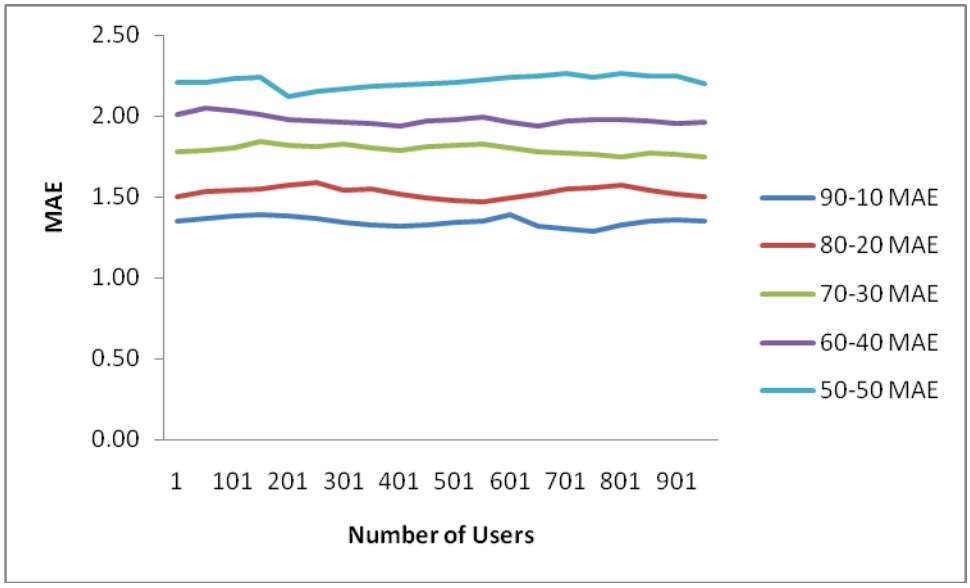

Figure 6-19: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.1$, $\beta = 0.7$, $\gamma = 0.05$, $\delta = 0.1$ and $\lambda = 0.05$)*

The average drop in MAE is about ~0.13, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~1.40 to ~2.25. These results are not in the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness.

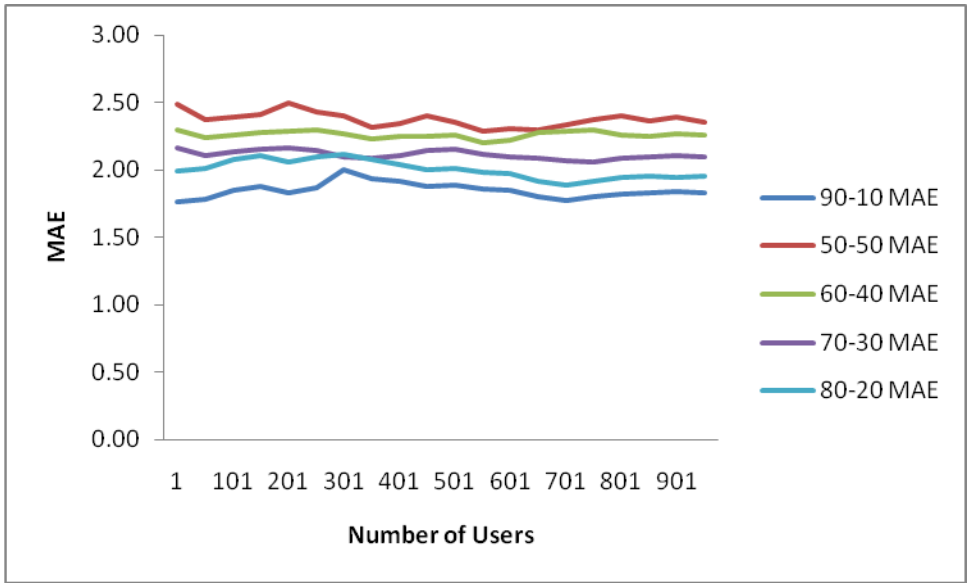

Figure 6-20: *Number of Users to MAE (Mean Absolute Error), for the parameter set ($\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0.1$, $\delta = 0.6$ and $\lambda = 0.1$)*

The average drop in MAE is about ~0.1, which signifies the weight of collaborative filtering component; this behavior is attributed to the weight since, for collaborative filtering component bases its need on finding the cluster of similar users. Also, the range of MAE for all the datasets combined is ~1.75 to ~2.50. These results are not in the threshold cut-off which was defined as the prerequisite for the recommender system, to tackle the problem of diversity and freshness. Now to signify the fact the additional parameters can handle only a few of the total set of users, the graph shows a significant increase in the range of MAE when compared to the previous parameter set.

From all the experiments and their results as shown above, the base recommender algorithm which was developed to get a MAE, which is less than the threshold-cutoff that was defined before the system was developed, is achieved.

6.1.3 Comparison with other base recommender algorithms

When compared to three different base recommender algorithms, which have been run on the same dataset as 'Movie Lens' [52], the current recommender system seems to run reasonably well with the other base algorithms. Three different base recommender algorithms are considered

here for comparison they are: Delanny & Verleysen, 2007 [13], Zaier, Godin, & Faucher, 2008 [32] and Park & Tuzhilin, 2008 [48].

The system in Delanny & Verleysen, 2007 [13] uses a Collaborative Filtering approach with the use of generalized linear models to represent user profiles. In such a representation, the user profiles are represented with much more richer information than a normal representation would do. From the paper, the results of the implementation and experiments on an 80-20 % data split of complete dataset yields good results. The Mean Absolute Error of this system is between 0.875 and 0.975. When the base recommender algorithm developed in this thesis work is compared to Delanny & Verleysen, 2007 [13], it shows a reasonable amount of accuracy, although Delanny & Verleysen, 2007 [13] gives a very uniform range of error in terms of MAE, the current algorithm has a bigger range of error from 0.65 to 1.22.

The system in Park & Tuzhilin, 2008 [48] uses a collaborative filtering approach with the use of Pearson Correlation Coefficient similarity measure. This is a collaborative filtering approach in which the instead of using Euclidean distance to find the similar user cluster, the Pearson correlation coefficient distance is to measure the similarity between users. The Mean Absolute Error of this system is between 0.88 – 1.12 for an 80-20 % split of Movie Lens dataset [52]. When the base recommender algorithm developed in this thesis work is compared to Park & Tuzhilin, 2008 [48], it performs better with a lower MAE than Park & Tuzhilin, 2008 [48] for some users and worse than Park & Tuzhilin, 2008 [48] with higher MAE.

The system in Zaier, Godin, & Faucher, 2008 [32] uses a hybrid approach with model based collaborative filtering. In this approach the concept is a content-boosted collaborative filtering, that is a model is generated to predict unknown ratings for the user profiles and the complete user profiles are then used to find similar users and thus a better cluster of similar users, thus providing more accurate predictions. The Mean Absolute Error of this system is between 0.70 – 1.13 for an 80-20 % split of Movie Lens dataset [52]. When the base recommender algorithm developed in this thesis work is compared to Zaier, Godin, & Faucher, 2008 [32], it performs better with a lower MAE than Zaier, Godin, & Faucher, 2008 [32] for some users and worse than Zaier, Godin, & Faucher, 2008 [32] with higher MAE.

The reasons for such a behavior of the current base recommender algorithm can be attributed to that fact that this algorithm concentrates more than the rest of the algorithms on user

preferences. This might help in some cases with more accurate prediction when the user actually has those predictions, but when the user actually does not have those kinds' preferences then this might produce less accurate predictions. Thus having the parameters static can produce good results in some cases but in some can produce bad results. This is one of the limitations of the system that the parameters are chosen statically.

The results of the comparisons are given below:

Base Recommender Algorithm	Range of MAE (Mean Absolute Error) on an 80-20 split dataset.
Delanny & Verleysen, 2007 [13] (Collaborative Filtering with interlaced Generalized Linear Models)	0.875 – 0.975
Park & Tuzhilin, 2008 [48] (Collaborative filtering with Pearson Correlation Coefficient similarity measure)	0.88 – 1.12
Zaier, Godin, & Faucher, 2008 [32] (Hybrid Approach, with model based collaborative filtering)	0.70 – 1.12
Current Algorithm (Weighted Hybrid Approach, Content-based and collaborative filtering)	0.65 – 1.22

Table 6-4: *Comparison with various base recommender algorithms.*

6.2.Evaluation of Diversity Metrics of the recommender system

From the previous section 6.1.2, it is clear that some of the parameter sets do hold the prerequisite of the threshold cutoff, which was defined to examine a working recommender system, which can diversify over the recommendations. To evaluate the approach described in Chapter 4 for diversification, experiment set 2 described in Chapter 5 are run on a working recommender system from 6.1.2.

The perquisite of a recommender system which holds the cutoff MAE is found from 6.1.2, so one of the three sets of parameters which hold the cut-off in the previous experiments is taken to run the experiments to evaluate the diversity component of the recommender system. The metric used to evaluate is ‘Average Similarity’ Metric which signifies the average similarity of the top-n recommendations for each user. The data set used here is different from the previous datasets used for experiment set 1. There are a total of five datasets, which are constructed in the same manner as the previous datasets, but only difference is that all the datasets are 80-20 % splits of the complete dataset. The intuition behind choosing such a dataset is to make sure that every user can get n recommendations which are actually relevant, considering 50-50 datasets might make the system, provide less than n recommendations for some users. Also, care is taken that all the five datasets are dissimilar. Also, the predictions are modified to include the freshness factor into them. As mentioned in Chapter 4, the diversifier component is the one which makes the final recommendations, keeping this mind the freshness component is imbibed as mentioned in the Chapter 4, and the complete system is evaluated based on average similarity metrics.

The 80,000 ratings part of the datasets is used for training (knowledge base) and the remaining dataset is used to test the recommender system. The following figures show the results for the experiments in an ‘Average Similarity to Number of Users’ graph.

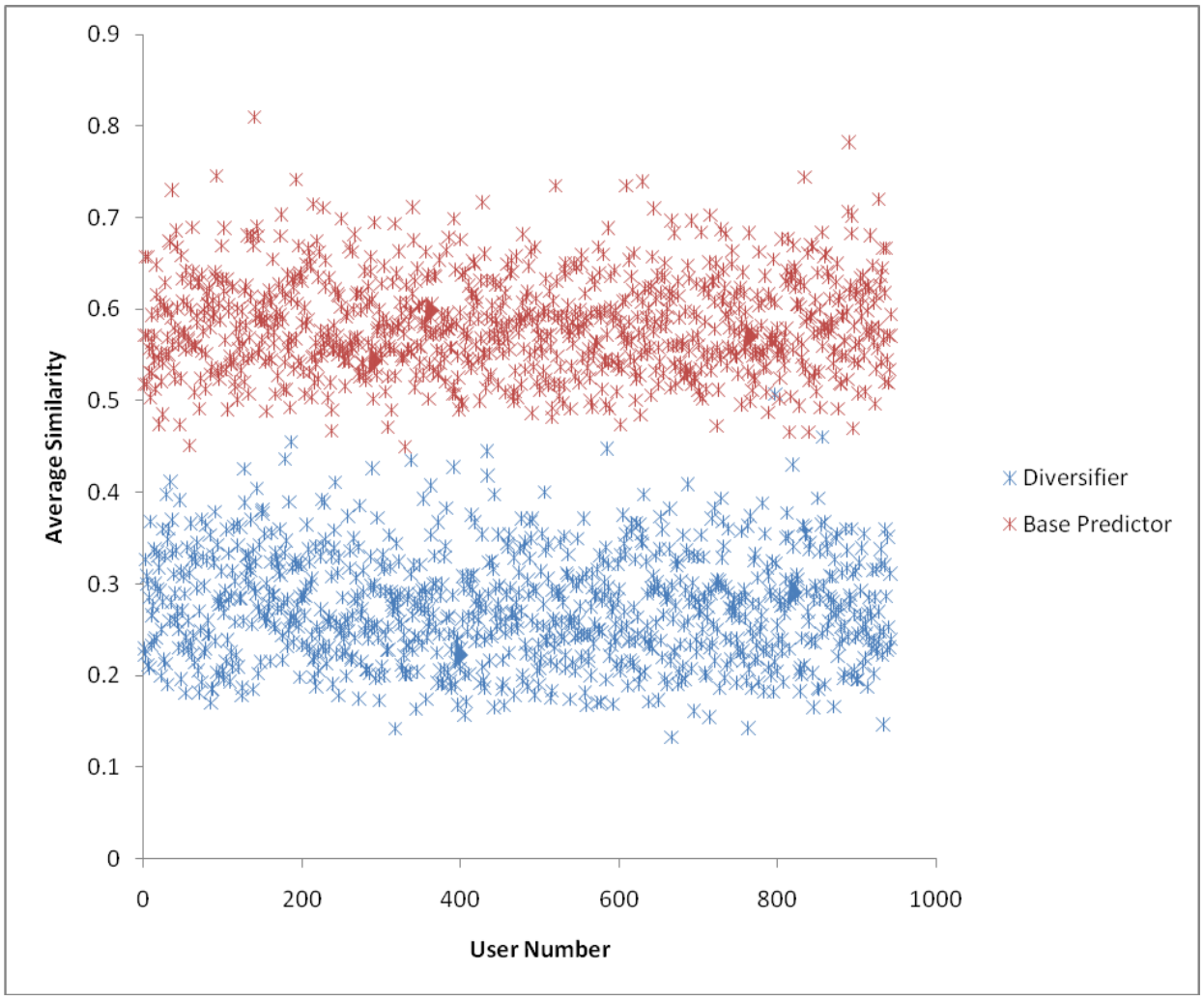

Figure 6-21: '*Average Similarity to Number of Users*' *for 1st 80-20 split dataset.*

From, the graph it can be inferred that the average similarity for the dataset ranges from ~0.15 to ~0.51 when the diversifier is used and the average similarity without any diversity included is from ~0.45 to ~0.82. There also seems to be varying range of average similarities, this can be attributed to the fact that the set of item recommendations change with users and also the user set do consists of some unique users, although the system recommends items to those users, the system seems to recommend more familiar items, this behavior can be supported by the claim that recommender systems' main goal is to recommend relevant items, even though they are similar. So maintaining a level of relevance, the recommender system tries to maximize the diversity of the items recommended. The diversifier does provide a drop in average similarities proving that the recommended items are more diverse than previous expected.

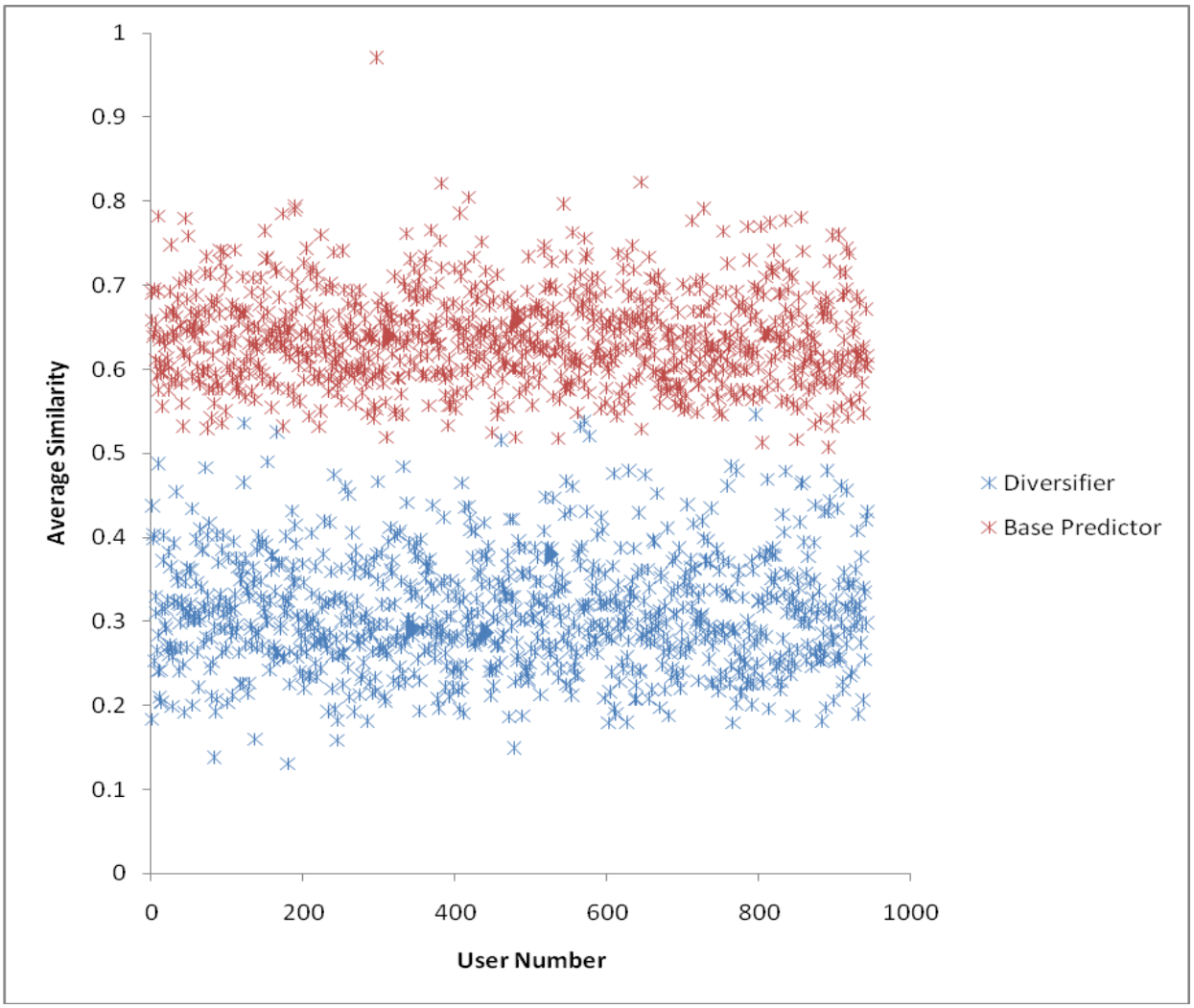

Figure 6-22: '*Average Similarity to Number of Users*' *for 2nd 80-20 split dataset.*

From, the graph it can be inferred that the average similarity for the dataset ranges from ~0.12 to ~0.55 when the diversifier is used and the average similarity without any diversity included is from ~0.44 to ~0.87. The diversifier does provide a drop in average similarities proving that the recommended items are more diverse than previous expected, but for some users the diversifier provides a lesser diverse set of items than the base algorithm, this can understood from the fact that diversifier uses cluster weights of item clusters to get the number of items from each cluster and some unique user / unpopular items might make a drop in cluster weights providing lesser diverse items.

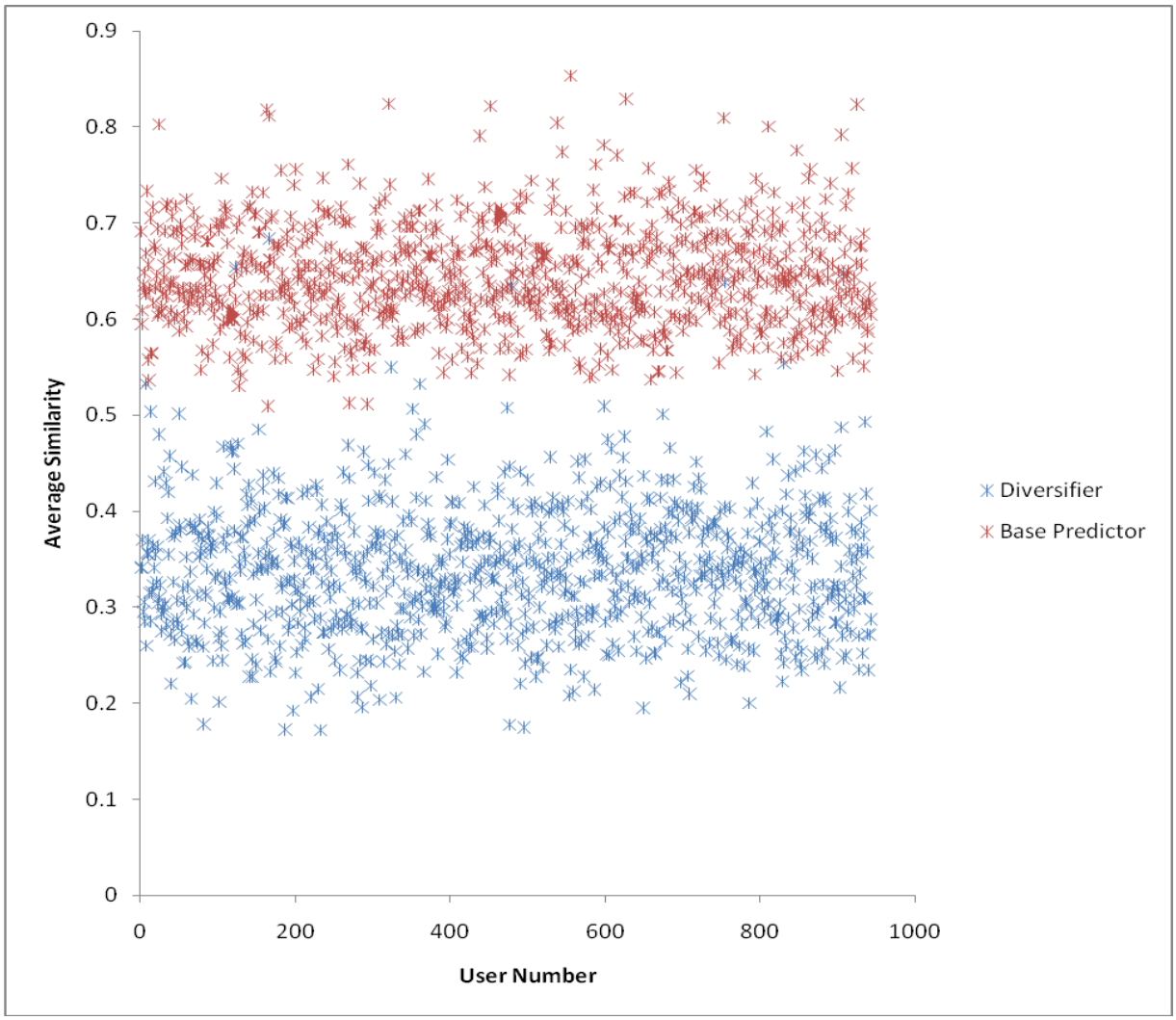

Figure 6-23: '*Average Similarity to Number of Users*' *for 3rd 80-20 split dataset.*

From, the graph it can be inferred that the average similarity for the dataset ranges from ~0.18 to ~0.69 when the diversifier is used and the average similarity without any diversity included is from ~0.52 to ~0.84. The diversifier does provide a drop in average similarities proving that the recommended items are more diverse than previous expected, but for some users the diversifier provides a lesser diverse set of items than the base algorithm, this can understood from the fact that diversifier uses cluster weights of item clusters to get the number of items from each cluster and some unique user / unpopular items might make a drop in cluster weights providing lesser diverse items.

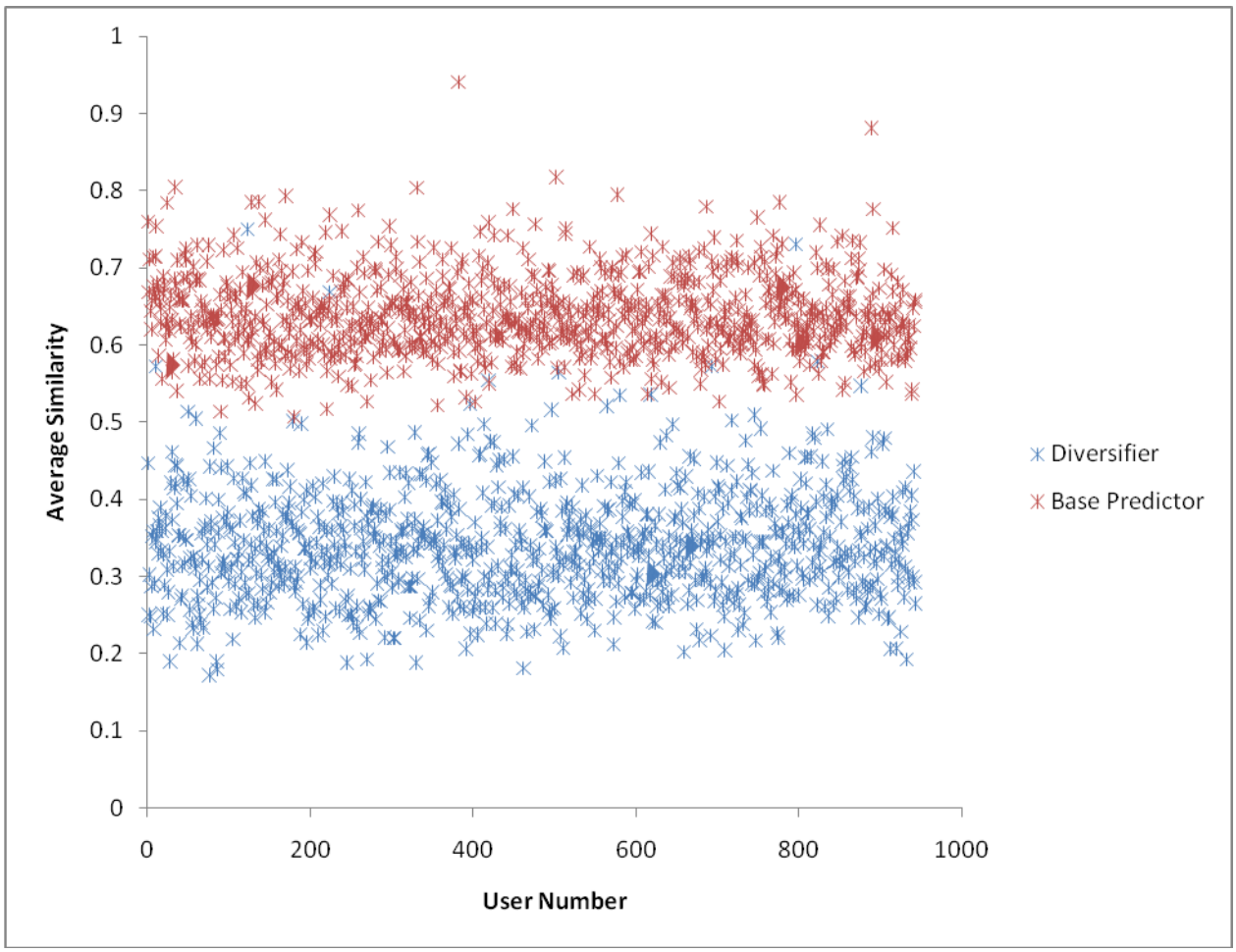

Figure 6-24: '*Average Similarity to Number of Users*' *for 4th 80-20 split dataset.*

From, the graph it can be inferred that the average similarity for the dataset ranges from ~0.16 to ~0.73 when the diversifier is used and the average similarity without any diversity included is from ~0.51 to ~0.94. The diversifier does provide a drop in average similarities proving that the recommended items are more diverse than previous expected, but for some users the diversifier provides a lesser diverse set of items than the base algorithm, this can understood from the fact that diversifier uses cluster weights of item clusters to get the number of items from each cluster and some unique user / unpopular items might make a drop in cluster weights providing lesser diverse items.

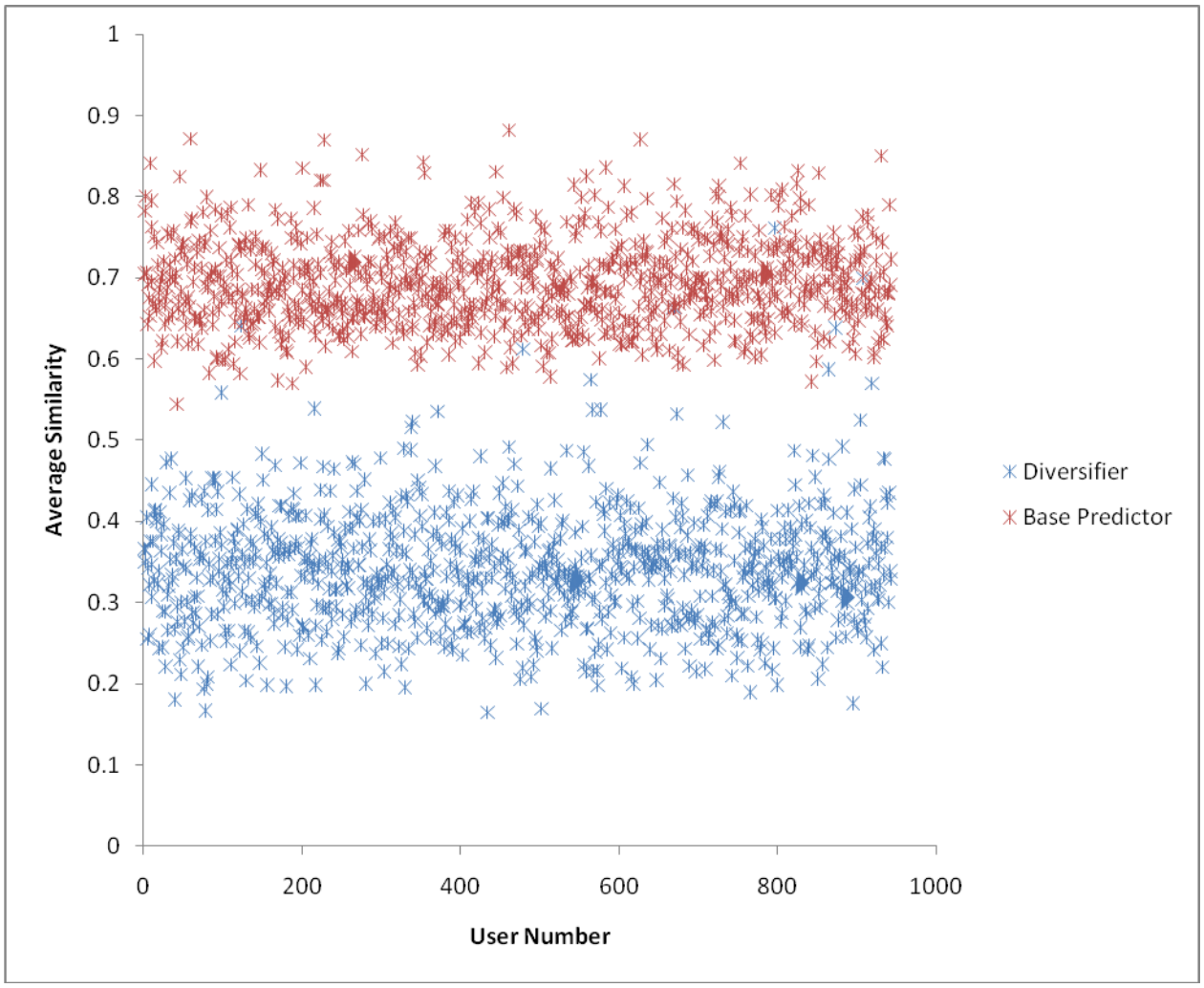

Figure 6-25: '*Average Similarity to Number of Users*' *for 5th 80-20 split dataset.*

From, the graph it can be inferred that the average similarity for the dataset ranges from ~0.17 to ~0.78 when the diversifier is used and the average similarity without any diversity included is from ~0.54 to ~0.87. The diversifier does provide a drop in average similarities proving that the recommended items are more diverse than previous expected, but for some users the diversifier provides a lesser diverse set of items than the base algorithm, this can understood from the fact that diversifier uses cluster weights of item clusters to get the number of items from each cluster and some unique user / unpopular items might make a drop in cluster weights providing lesser diverse items.

6.3. Conclusion and Research Questions

- The first of the goals of this thesis work ‘Develop a recommender algorithm which can produce accurate predictions. Accuracy is measured in terms of Mean Absolute Error (MAE), on a range of 1-5 accuracy greater than 70% (*i.e.*, MAE less than 30% (1.5)) is considered to be reasonable.’ is achieved from the experiment set 1 result shown in section 6.1.2. The base recommender algorithm which was developed to get a MAE, which is less than the threshold-cutoff that was defined, is achieved.
- The second of the goals of this thesis work ‘Prove that the generated recommender algorithm when compared to other hybrid recommender algorithms performs reasonably well.’ is achieved from section 6.1.3. When compared to three different base recommender algorithms, which have been run on the same dataset ‘Movie Lens’ [[52]], the current recommender system seems to run reasonably well in comparison the other base algorithms. From the results, it is clear that the variation in MAE is more than all of the approaches, but it does achieve an acceptable MAE for 80-20 dataset split.
- The last of the foals of this thesis work ‘Diversify the set of item recommendations, without major loss of relevance. Prove that once a threshold of the accuracy is reached diversity can be increased further without any major loss of relevance.’ is achieved from the results shown in section 6.2. From the results of experiment set 2 in section 6.2, the system does give a descent amount of diversity and performs better than a system without any implementation for diversity, since for most of users on an average the similarity is less than 0.4. This suggests that system does provide reasonably diverse at the same time fresh and relevant recommendations.

7 Future Work and Limitations

In this Chapter the limitations of the current thesis work are listed out in section 7.1 and a proposed future plan of work is discussed briefly in section 7.2.

7.1.Limitations

The current recommender system has several limitations, they are discussed below:

- The current recommender system considers the data to be static, and uses this static data to make predictions for a new user. This recommender system cannot handle any changes to the data. Although it uses data from IMDB which is non-static, the data from IMDB is downloaded during the process of developing the system and it doesn't handle any changes to IMDB data at run-time, any changes have to be made before the systems' inception.
- Also, a limitation of the system is that it uses LDA in content-based component, and even in the diversifier, to represent items. However the representation itself can be incomplete since, LDA doesn't capture any correlation between topics which is one of major principles in textual representations of items.
- Also, choosing the parameters is done statically and the user preferences are not captured to the utmost capacity. Although the system works reasonably well, with the static parameters, usage of dynamic parameters can increase the efficiency of the recommender system. If the parameters are modified dynamically, the fluctuations in the average similarities and MAE of the predictions for users can be reduced.
- Also, the concept of relevance is not standard, since the recommender system considers the item whose rating is better than 2.5 out of 5 to be relevant. However the concept of relevance is dependent on the user. So, this makes the system to be static to be limited in the sense of user preference.
- Another limitation is that the system doesn't provide any offline evaluation metrics for the problem of freshness (predictability). Since, from the approach it is defined that evaluating predictability depends only on the user, so it has to be done at run-time and also on the long run.

- Also, one major limitation in the implementation of the system is the time-complexities and storage requirements the system requires to have a descent response time. With a reasonable size of dataset, the response time is not huge but with increasing dataset the approach requires a lot of computations, thus requiring large memory spaces and processor speed to have a smaller response time.

7.2.Future Work

The future work consists of tackling the limitations and handling other open problems in recommender systems. Some of the possible future plans of actions are given below:

- First step is to understand the best way to tackle the problem of non-static data, while maintaining a descent level of computational overheads required.
- Another plan is to consider dynamic assignment of parameter set, for each different user. One plan of action is to implement a classifier like artificial neural network to compute the weights based on the active user profile. Also, the problem to threshold for relevance is to be set dynamically for a user, rather a standard procedure.
- Also, to solve the problem of cold-start in a better way rather than considering just the most popular items initially. One solution might be to use tree structure (Decision Trees) to get a directed way of understanding new user.

Bibliography

[1] Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 734-749.

[2] Amer-Yahia, S., Lakshmanan, L. V., Vassilvitskii, S., & Yu, C. (2009). Battling Predictability and Overconcentration in Recommender Systems. *IEEE Data Engineering Bulletin*.

[3] Balabanovic, M., & Shoham, Y. (1997). Content-based, collaborative recommendation. *Communication of the ACM* , 66-72.

[4] Belkin, N. J., & Croft, W. B. (1992). Information filtering and information retrieval: two sides of the same coin? . *Communications of the ACM* , 29-38.

[5] Billsus, D., & Pazzani, M. J. (2000). User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction* , 147-180.

[6] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Machine Learning Research* , 993-1022.

[7] Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *In Proceedings of the 14th Conference on Uncertainity in Artificial Intelligence*, (pp. 43-52).

[8] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* , 331-370.

[9] Burke, R. (2007). Hybrid Web Recommender Systems. *The Adaptive Web: Methods and Strategies of Web Personalization*, Lecture Notes, Vol. 4321, 377-408.

[10] Cazella, S. C., Reategui, E., & Alvares, L. O. (2006). E-commerce recommenders' authority: Applying the user's opinion relevance in recommender systems. *In Proceedings of the 12th Brazilian Symposium on Multimedia and the Web*, (pp. 71-78).

[11] Debnath, S., Ganguly, N., & Mitra, P. (2008). Feature weighting in content-based recommendation system using social network analysis. *In Proceedings of the 17th International Conference on World Wide Web*, (pp. 1041-1042).

[12] Deerwester, S. C., Susan T, D., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by Latent Semantic Analysis. *American Society of Information Sciences* , 391-407.

[13] Delanny, N., & Verleysen, M. (2007). Collaborative filtering with interlaced generalized linear models. *In Proceedings of the European Symposium on Atrificial Neural Networks*, (pp. 247-252). Bruges, Belgium.

[14] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM* , (pp. 61-70).

[15] Griffiths, T. L., & Steyvers, M. (2007). Topics in Semantic Representation. *Pyschological Review* , 211-244.

[16] Hamerly, G., & Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. *In Proceedings of the 11th International Conference on Information and Knowledge Management.* (pp. 600-607)

[17] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *In Proceedings of the ACM Transactions on Information Systems*. (pp. 5-53)

[18] Hill, W., Stead, L., Rosentein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. *In Proceedings of the SIGCHI Conference on Human factors in computing systems.* (pp. 194-201)

[19] Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. *In Proceedings of the 2nd Annual International SIGIR Conference.*

[20] Hosmann, T., & Puzieha, J. (1999). Latent Class Models for Collaborative Filtering. *In Proceedings of the International Joint Conference on Artificial Intelligence*, (pp. 688-693).

[21] Jain, A., Murthy, M., & Flyn, P. (1999). Data Clustering: A Review. *ACM Computing Surveys,*31(3): 264-323.

[22] Jiang, J., M., W. B., Donato, J., Ostrouchov, G., & Grady, N. (1999). Mining Consumer product data via latent semantic indexing. *Intelligent Data Analysis* , 377-398.

[23] Kim, B. M., & Li, Q. (2004). Probabilistic Model Estimation for Collaborative Filtering Based on Items Attributes. *In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, (pp. 185-191).

[24] Kohrs, A., & Merialdo, B. (1999). Clustering for Collaborative Filtering Applications. *Computational Intelligence for Modelling, Control & Automation.* (pp. 27-36)

[25] Koren, Y. (2009). Collaborative filtering with temporal dynamics. *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery*, (pp. 447-456).

[26] Lang, K. (1995). Newsweeder: Learning to filter Netnews. *In Proceedings of the 12th International Machince Learning Conference.* (pp. 331-339)

[27] Manning, C. D., Raghavan, P., & Schutze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

[28] McNee, S. M., Riedl, J., & Konstan, J. A. (2006). Being accurate is not enough: how accuracy metrics have hurt recommender systems. *In Proceedings of the CHI '06 Extended abstracts on Human factors in computing systems*, (pp. 1097-1101).

[29] Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. *In Proceedings of the 18th National Conference on Artificial Intelligence*, (pp. 187-192).

[30] Mitchell, T. (1997). *Machine Learning.* McGraw Hill.

[31] Park, S.-T., Pennock, D., Madani, O., Good, N., & DeCoste, D. (2006). Naïve filterbots for robust cold-start recommendations. *In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and DataMining*, (pp. 699-705).

[32] Park, Y.-J., & Tuzhilin, A. (2008). The long tail of recommender systems and how to leverage it. *In Proceedings of the ACM conference on Recommender systems.*

[33] Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., & Welling, M. (2008). Fast collapsed gibbs sampling for latent dirichlet allocation. *In Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 569-577).

[34] Raatikainen, K. E. (1993, May). Cluster analysis and workload classification. *In Proceedings of the ACM SIGMETRICS Performance Evaluation Review* , pp. 24-30.

[35] Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM* , 56-58.

[36] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering on netnews. *In Proceedings of the 8th ACM Conference on Computer Supported Cooperative Work*, (pp. 175-186). New York, NY, USA.

[37] Roelleke, T., & Wang, J. (2008). TF-IDF uncovered: as study of theories and probabilities. *In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 435-442).

[38] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *In Proceedings of the 10th International Conference on World Wide Web*, (pp. 285-295). New York, NY, USA.

[39] Schafer, J. B., Konstan, J., & Riedl, J. (1999). Recommender Systems in E-Commerce. *In Proceedings of the 1st ACM Conference on Electronic Commerce*, (pp. 158-166). Denver, Colorado, USA.

[40] Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 253-260). Tampere, Finland.

[41] Schkolnick, M. (1997). A clustering algorithm for hierarchical structures. *ACM Transactions on Database Systems* , 27-44.

[42] Semeraro, G., Lops, P., & Basile, P. (2009). Knowledge infusion into content-based recommender systems. *In Proceedings of the 3rd ACM Conference on Recommender Systems*, (pp. 301-304).

[43] Taghipour, N., & Kardan, A. (2008). A hybrid web recommender system based on Q-learning. *In Proceedings of the SCM Symposium on Applied Computing*, (pp. 1164-1168).

[44] Ungar, L., Foster, D., Andre, E., Wars, F. S., Wars, D. S., & Whispers, J. H. (1998). Clustering Methods for Collaborative Filtering. *In Proceedings of the 15th National Conference on Artificial Intelligence.* AAAI Press.

[45] Xue, G.-R., Lin, C., Yang, Q., Xi, W. S., Zeng, H.-J., & Yu, Y. (2005). Scalable collaborative filtering using cluster-based smoothing. *In Proceedings of the ACM SIGIR Conference*, (pp. 114-121). Salvador, Brazil.

[46] Xue, G.-R., Lin, C., Yang, Q., Xi, W. S., Zeng, H.-J., & Yu, Y. (2005). Scalable collaborative filtering using cluster-based smoothing. *In Proceedings of the ACM SIGIR Conference*, (pp. 114-121). Salvador, Brazil.

[47] Yu, C., Lakshmanan, L., & Amer-Yahia, S. (2009). It takes variety to make a world: Diversification in Recommender Systems. *In Proceedings of the 12th International Conference on Extending Database Technology*, (pp. 368-378). Saint Petersburg, Russia.

[48] Zaier, Z., Godin, R., & Faucher, L. (2008). Evaluating Recommender Systems. *In Proceedings of the International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution*, (pp. 211-217).

[49] Zanker, M. (2008). A collaborative constraint-based meta-level recommender. *In Proceedings of the ACM Conference on Recommender Systems*, (pp. 139-146).

[50] Ziegler, C.-N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving Recommendation Lists Through Topic Diversification. *In Proceedings of the 14th International Conference on World Wide Web*, (pp. 22-32). Chiba, Japan.

[51] IMDB (2010). *Internet Movie Database*. Retrieved June 2010, from http://www.imdb.com

[52] GroupLens. (1997, September 19). *MovieLens Dataset*. Retrieved June 2010, from GroupLens: http://www.movielens.org

[53] Wikipedia. (2004, July 22). *Wikipedia: The free encyclopedia*. Retrieved July 2010, from http://www.wikipedia.org

[54] Yahoo! (1998, May 12). Retrieved June 2010, from Yahoo! Movies: http://movies.yahoo.com

[55] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutmann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations* .

[56] McCallum, A. K. (2002). *MALLET: A Machine Learning for Language Toolkit.* http://mallet.cs.umass.edu. 2002.