THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

CLUSTERING BASED DIVERSITY IMPROVEMENT IN RECOMMENDER SYSTEMS

Master's Thesis

MAHMUT ÖZGE KARAKAYA

İSTANBUL, 2012

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE
COMPUTER ENGINEERING

CLUSTERING BASED DIVERSITY IMPROVEMENT IN RECOMMENDER SYSTEMS

Master's Thesis

MAHMUT ÖZGE KARAKAYA

Supervisor: ASST. PROF. DR. TEVFiK AYTEKiN

İSTANBUL, 2012

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

Title of the Master's Thesis : Clustering based diversity improvement in recommender systems
Name/Last Name of the Student : Mahmut Özge Karakaya
Date of the Defense of Thesis : 11 June 2012

The thesis has been approved by the Graduate School of Natural and Applied Science.

Asst. Prof. F. Tunç BOZBURA
Acting Director

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Comitee Members	Signature
Asst. Prof. Dr. Tevfik AYTEKİN	----------------------------------
Asst. Prof. Dr. Alper TUNGA	----------------------------------
Asst. Prof. Dr. Egemen ÖZDEN	----------------------------------

ACKNOWLEDGMENTS

I am very thankful to my family, friends and Dr. Tevfik Aytekin who have helped me all along my studies.

Mahmut Özge Karakaya

11 June 2012

ABSTRACT

CLUSTERING BASED DIVERSITY IMPROVEMENT IN RECOMMENDER SYSTEMS

Özge Mahmut Karakaya

Computer Engineering

Supervisor: Asst. Prof. Dr. Tevfik Aytekin

June 2012, 50 Pages

Recommender systems help users to find items (movies, books, music, etc.) of interest based on information about users such as past transactions, explicit ratings, or some implicit feedback. The success of a recommendation system is typically measured by the accuracy of its predictions, i.e., its ability to predict users' ratings for items. Although accurracy is a very important property of recommender systems, it has been recognized that there are other important properties of recommender systems which play valuable roles in user satisfaction.

One such important property is diversity. Diversity measures the variety of a recommendation list. For example, for a movie recommender system, the system might try to suggest movies from different genres in order to increase the diversity of the recommendation lists. In this study, we present a new method to diversify recommendations. The method lets the users to adjust the diversity levels of their recommendation lists by using a tunable parameter. One advantage of our method over previous ones is that it has a low computational time complexity which makes it possible to work online so that users are able to see the results immediately when they change the diversity level of their recommendation lists.

Keywords: Diversity, Collaborative filtering, Recommender systems

ÖZET

TAVSİYE SİSTEMLERİNDE KÜMELEME ALGORİTMASI KULLANARAK ÇEŞİTLİLİĞİN GELİŞTİRİLMESİ

Özge Mahmut Karakaya

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği

Tez Danışmanı: Yrd. Doç.Dr. Tevfik Aytekin

Haziran 2012, 50 Sayfa

Tavsiye sistemleri kullanıcıların geçmiş işlemleri, açık derecelendirmeleri veya örtük geribildirimleri gibi bilgilere dayanarak kullanıcıların ilgisini çekebilecek öğeleri (film, kitap, muzik, vb.) bulmalarına yardımcı olur. Tavsiye sistemlerinin başarısı tipik olarak tavsiye sisteminin doğruluğuna yani kullanıcı derecelendirmelerini ne kadar iyi tahmin edebildiğine göre belirlenir. Doğruluk tavsiye sistemlerinin çok önemli bir özelliği olmakla birlikte kullanıcı tatmini için başka önemli kriterlerin de olduğu kabul edilmiştir.

Bunlardan bir tanesi çeşitliliktir. Çeşitlilik tavsiye listelerinin ne kadar farklı öğelerden oluştuğunu ölçer. Örneğin, bir film tavsiye sistemi, tavsiye listelerinin çeşitliliğini artırmak için farklı türlerden filmler önermeye çalışabilir. Bu çalışmada, tavsiye listelerinin çeşitliliğini artırmak için yeni bir yöntem öneriyoruz. Bu yöntem sayesinde kullanıcılar bir parametreyi kontrol ederek tavsiye listelerinin çeşitliliğini ayarlayabilirler. Metodumuzun öncekilere göre bir avantajı düşük bir hesaplayımsal zaman karmaşıklığına sahip olmasıdır. Bu sayede kullanıcılar çevrimiçi olarak tavsiye listelerin çeşitlilik seviyesini değiştirip sonuçları hemen görebilirler.

Anahtar Kelimeler: Çeşitlilik, İşbirlikci Filtreleme, Tavsiye Sistemleri

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

IR	:	Information Retrieval
MAE	:	Mean Absolute Error
RMSE	:	Root Mean Square Error
RS	:	Recommender Systems
SVD	:	Singular Value Decomposition

1. INTRODUCTION

Recommender systems help users to find items (movies, books, music, etc.) of interest based on information about users such as past transactions, explicit ratings, or some implicit feedback. Many successful techniques have been developed to this end. The success of a recommendation algorithm is typically measured by the accuracy of its predictions, i.e., its ability to suggest relevant items to the user. The accuracy of predictions is no doubt is an important property of recommender systems. And naturally most of the research in recommender systems focused on improving accuracy. However, there are other properties of recommender systems, which play valuable roles in user satisfaction. One such important property, which has gained importance recently is diversity. For example, think of a system which suggests movies to its users. The system might be very accurate, that is, it might predict the real user ratings very well. However, if the recommendation list consists of one type movies (e.g., only science fiction movies) it might not be very satisfactory. A good system should also recommend a diverse set of movies to its users (e.g., movies from different genres). There is a trade-off between accuracy and diversity. That is most of the time diversity can only be increased at the expense of accuracy. But this decrease in accuracy might be preferable if the user satisfaction increases. Moreover, this trade-off can be implemented as a tunable parameter, which the user can adjust according to her needs. In this way the user herself decides how much to sacrifice accuracy for an increase in diversity. This also lets users to experiment (in a systematic way) with the recommendations produced by the system and to discover a diverse set of items.

1.1 PURPOSE OF THESIS

In this thesis, we will describe a novel method which can be used to increase the diversity of recommender lists with little decrease in accuracy. Our idea, basically, is to cluster items into groups and build the recommendation list by selecting items from different groups such that recommendation diversity is maximized without decreasing accuracy too much. The contributions of this thesis include: The proposed method is simple to understand and implement. The algorithm is quite efficient (both in the online and offline phases). The method naturally involves a tunable parameter, which the user can adjust according to her needs. In order for a recommender system to allow users to adjust diversity level the online complexity of the algorithm should be low. As stated above our algorithm has a low online complexity. We cluster items into groups using only the rating information. We use no content information, which can be difficult to compile.

1.2 LITERATURE REVIEW

Several strategies have been proposed to address the problem of diversity. Authors developed a greedy selection algorithm (Smyth and McClave 2001). In this method the items are first sorted according to their similarity to the target query and then the algorithm begins to incrementally build the retrieval set (or the recommendation list if we use the terminology of recommender systems research) such that both similarity and diversity are optimized. This is achieved as follows: in the first iteration the most similar item to the target query is put in the retrieval set, in the next iteration the item, which has the maximum combination of similarity to the target query and diversity with respect to the retrieval set already built is selected. Iterations continue until the desired retrieval set size is achieved.

This greedy selection algorithm is quite inefficient so the authors proposed, in the same paper, a bounded version of the greedy selection algorithm (Smyth and McClave 2001). In this version the algorithm first selects the most similar bk number of items to the target query and then the greedy selection method is applied to this set of items instead of the entire set of items. As bk approaches n (the number of items) the complexity of this bounded version approaches to the complexity of the greedy selection method. Assuming that the items are already sorted according to their similarity to the target query, the bounded greedy algorithm requires k (the retrieval set size) sorts of a list of size O(bk). As we will discuss later the running time of our method is much faster than this method. It only requires O(k) time in the online phase.

Another method is proposed (Hurley and Zhang 2011) (Zhang and Hurley 2008) Here the authors represent the trade-off between similarity and diversity as a quadratic programming problem. Then they offer several solution strategies to solve this optimization problem. They achieve better results in terms of precision and recall in top-

N recommendation compared to other method. However, as they have reported the time-efficiency of their algorithm is slightly worse. Authors use a clustering approach to better diversify recommended items according to the taste of users (Zhang and Hurley.2009). To do this, they clusters items in the user profile and recommend items that match well to these individual clusters rather than the entire user profile. Our method also clusters items into similar groups. However, as will be discussed in detail below, we cluster all the items in the system rather than just the items in the user profile. That is, our aim is not to recommend items, which match users' tastes but rather to recommend a diverse set of items while maintaining accuracy as much as possible. This gives users the opportunity to meet serendipitous items.

The authors define a similarity metric based on classification taxonomies according to which the intra-list similarity is calculated (Ziegler et al. 2005). The authors propose a heuristic algorithm for diversification of recommendation lists based on this similarity metric. Similar to other studies the proposed method increases diversity with some negative effects on accuracy. One important contribution of this work is to show empirically that overall user satisfaction increases with diversified recommendation lists. This result supports the claim that accuracy of recommendation lists is not the only criteria for user satisfaction but other criteria such as being able to suggest novel items are also important.

1.3 THESIS OUTLINE

We organized the thesis as follows. In section 2 we describe recommender system approaches such as collaborative filtering and content based methods. In section 3 we represent evaluation of recommender systems in detail. In section 4 we explicate our methodology and results. Finally, in section 5 we conclude the thesis and point out new directions of research.

2. RECOMMENDER SYSTEM APPROACHES

2.1 COLLABORATIVE FILTERING

The inputs of a collaborative filtering algorithm are just user interactions such as implicit and explicit ratings. Therefore, we can say that users obliquely specify other users' recommendation lists. This feature makes collaborative filtering significant and choices of community in a recommender system direct the recommender system.

Tapestry is assumed as the first system that supports collaborative filtering (Goldberg et al. 1992). This system let its users to tag the documents they read. Nevertheless, it was not a pure collaborative filtering recommender; it was amazing for a recommender engine to produce its recommendations using not only content of the documents themselves, but also on what other users have said about them. Therefore, Tapestry is set as a milestone in history of recommender systems.

Another crunch time for collaborative filtering was Netflix Prize. It was an open competition on 21 September 2009 with the grand prize of US$1,000,000. The aim of the competition was choosing the best collaborative filtering algorithm to predict user ratings for movies, based on previous ratings. BellKor's Pragmatic Chaos team has won the competition with best RMSE 0.8567 and 10.06 percent improvement against the current Netflix Recommender Algorithm (Bell et al 2007). Thanks to Netflix Prize, researchers have learnt much about methodologies of recommendation systems such as binary views of ratings, temporal effects on ratings, choosing models etc.

Netflix Prize has led so many innovations about accuracy on collaborative filtering algorithms beside the short amount of time. However, accuracy of a recommender algorithm is still an active research area in computer science literature. Many other topics such as diversity, novelty, cold start problem, serendipity etc. are also active research areas for collaborative filtering. In research, people did not give much thought to these topics when Netflix Prize competition was active. Nevertheless, nowadays people work through many topics of collaborative filtering and recommender systems in research and practice.

As popularity of social networks and e-commerce has been raised, the importance of the collaborative filtering has become more visible in IR. Authors notified that collaborative filtering is the most successful recommender system technology to date, and is used in many of the most successful recommender systems on the Web (Sarwar et al 2000). For an example, Amazon.com, which has 29 million customers and several million-catalog items, employs collaborative-filtering-based recommendations (Linden et al 2003).

Collaborative Filtering is in demand now and it has a promising future too. Many personalized web and mobile applications are growing together with collaborative filtering algorithms. Collaborative search engines are one of the promising topics of collaborative filtering already now (Dadal 2007).

2.1.1 Memory Based

Memory Based methods use user ratings to generate item-item or user-user similarity matrixes and predict ratings using these matrixes. This is an easy but effective technique and one of most common methodology in recommender systems research. The techniques, also known as nearest neighbor or user based collaborative filtering, are more popular and widely used in practice (Sarwar.et al 2001).

2.1.1.1 Item based

The basic idea in similarity computation between two items *i* and *j* is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine similarity s_{ij} (Sarwar.et al 2001). Item based collaborative filtering is a more common technique than user based technique in practice. In most of systems where a recommender system works number of users is more than number of items. Therefore, building an item item similarity matrix is favored due to time and space complexity. While a recommender system grows, the most important constraints become time and space complexities. In this respect, the recommender systems should be easy to understand, maintain and administrate. In view of this fact most of the famous recommender systems such as Amazon prefers item based approach as we mentioned before. Another important metric is accuracy. Making a choice between item and user based methods in consideration of accuracy usually depends on ratio between the number of users and items. If number of items is less than number of users then more deterministic similarity computation is possible between items. That is the reason why item based methods are more accurate in case of number of users are much more than number of items. We already know that number of items is generally more than number of users in recommender systems. Another reason to choose item based methods instead of user based methods is justifiability. It is possible to explain recommendations using list of neighbor items used in prediction and their similarities when we recommend an item to a user.

A common way to determine similarity between two items is cosine based similarity. Characteristic of this methodology is to represent items or users as vectors. So we determine the similarity between two items or users as similarity between two vectors. The formula of similarity between items i and item j is below.

$$s_{ij} = \cos(\vec{\imath}, \vec{\jmath}) = \frac{\vec{\imath}.\vec{\jmath}}{|\vec{\imath}|^2 * |\vec{\jmath}|^2} \tag{2.1}$$

where "." represents the dot product of two vectors.

Another approach to calculate similarity between two items is correlation based similarity or pearson correlation. A constraint for this similarity measure is to consider just co-rated items. The following is the formula for correlation-based similarity.

$$s_{ij} = \frac{\sum_{u \in U} (\mathrm{r}_{u,i} - \overline{\mathrm{r}}_i)(\mathrm{r}_{u,j} - \overline{\mathrm{r}}_j)}{\sqrt{\sum_{u \in U} (\mathrm{r}_{u,i} - \overline{\mathrm{r}}_i)^2} \sqrt{\sum_{u \in U} (\mathrm{r}_{u,j} - \overline{\mathrm{r}}_j)^2}} \tag{2.2}$$

where $\mathrm{r}_{u,i}$ represents the rating of user u on item i and $\overline{\mathrm{r}}_{i}$ is the average rating of item i.

The last example for item based similarity calculation methodology that we represent is adjusted cosine similarity. Users' rating scales are different from each other. The main advantage of adjusted cosine is to consider different users' rating scales. We represent the formula for adjusted cosine below.

$$s_{ij} = \frac{\sum_{u\in U}(\mathrm{r}_{u,i} - \overline{\mathrm{r}}_{u})(\mathrm{r}_{u,j} - \overline{\mathrm{r}}_{u})}{\sqrt{\sum_{u\in U}(\mathrm{r}_{u,i} - \overline{\mathrm{r}}_{u})^2}\sqrt{\sum_{u\in U}(\mathrm{r}_{u,j} - \overline{\mathrm{r}}_{u})^2}} \qquad \textbf{(2.3)}$$

Here $\overline{\mathrm{r}}_{u}$ is the average rating of user u.

Using these similarities, we can calculate predicted ratings by obtaining weighted average of ratings that is given by the particular user. Following is the formula of this definition.

$$\hat{r}_{ui} = \frac{\sum_{j\in \mathrm{N}_u(\mathrm{i})} \mathrm{s}_{i,j}\, \mathrm{r}_{u,j}}{\sum_{j\in \mathrm{N}_u(\mathrm{i})} |\mathrm{s}_{i,j}|} \qquad \textbf{(2.4)}$$

2.1.1.2 User based

Similar to the item based there are two steps in user-based methodology. These are similarity calculation and prediction steps. Differently from item based method, we calculate similarities between users and construct a user-user similarity matrix. Such like in similarity calculation step prediction step in user based methodology it is little different from item based. As it is expected in prediction step, we obtain weighted average of ratings, which is given by similar users.

In consideration of serendipity user based is a more powerful methodology than item based. Item based method tends to recommend similar items to the items that the user rates. For example, a user A that has watched only comedies may be very similar to a user B only by the ratings made on such movies. However, if B is fond of a movie in a different genre, this movie may be recommended to A through his similarity with B (Desrosiers and Karypis 2011).

We apply cosine similarity to users and user similarity formula is below.

$$s_{uv} = \frac{\sum_{i \in \mathrm{I}_{uv}} \mathrm{r}_{ui}\, \mathrm{r}_{vi}}{\sqrt{\sum_{i \in \mathrm{I}_u} \mathrm{r}_{ui}{}^2} \sqrt{\sum_{j \in \mathrm{I}_v} \mathrm{r}_{vj}{}^2}} \qquad \textbf{(2.5)}$$

where I_{uv} represents the items rated by both user *u* and *v*.

We write *k* most similar neighbors of user *u* who rate item *i* as $\mathrm{N}_i(\mathrm{u})$ and following is the prediction formula for user-based methodology.

$$\hat{r}_{ui} = \frac{1}{|\mathrm{N}_i(\mathrm{u})|} \sum_{v \in \mathrm{N}_i(\mathrm{u})} \mathrm{r}_{vi} \qquad \textbf{(2.6)}$$

The main drawback is that this formula does not consider the user similarities. The similarities between users may be different and we want that most similar user has the strongest effect in prediction formula. Therefore, we change the formula as below where w_{uv} is the similarity between user *u* and user *v*.

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathrm{N}_i(\mathrm{u})} \mathrm{w}_{uv}\, \mathrm{r}_{vi}}{\sum_{v \in \mathrm{N}_i(\mathrm{u})} |\mathrm{w}_{uv}|} \qquad \textbf{(2.7)}$$

2.1.2 Model Based

The design and development of models such as machine learning, data mining algorithms can allow the system to learn to recognize complex patterns based on the training data, and then make intelligent predictions for the collaborative filtering tasks for test data or real-world data, based on the learned models (Su and Khoshgoftaar 2009). We build models using machine learning algorithms (such as Bayesian network, clustering and rule based approaches and so on) to predict user ratings. In consideration of prediction performance, model based approaches are advantageous to Memory Based methods. Moreover, model based methods are better to solve sparsity problem in collaborative filtering. The main disadvantage of model-based methods is model-building complexity. Most of the Model Based methods need so much more space and time than Memory Based methods. We can say that another disadvantage of model based methods is the difficultly in prediction explanation which is commonly used in practice. We have an example for model based methods in next section named Singular Value Decomposition aka SVD.

2.1.2.1 Singular value decomposition

There are some variations of SVD methodology. We choose one of them to explain the general idea, which is relatively easy to understand and implement, as well as we use it for evaluation. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f , such that user-item interactions are modeled as inner products in that space (Koren and Bell 2011). Following is the prediction formula for SVD where μ is the average of all ratings, b_i is standart deviation of the item *i* from μ, b_u is the standart deviation of user *u* from μ and $\mathrm{q}_i{}^{\mathrm{T}}\mathrm{p}_u$ is the dot product which determines overall interest of the user *u* to the item *i*.

$$\hat{r}_{ui} = \mu + \mathrm{b}_i + \mathrm{b}_u + \mathrm{q}_i{}^{\mathrm{T}}\mathrm{p}_u \tag{2.8}$$

For each rating in training set we calculate prediction error as $\mathrm{e}_{ui} = r_{ui} - \hat{r}_{ui}$ and update the values with formulas below.

$$\begin{aligned} b_u &\leftarrow b_u + \Upsilon\,(\,\mathrm{e}_{ui} - \lambda\, b_u\,) \\ b_i &\leftarrow b_i + \Upsilon\,(\,\mathrm{e}_{ui} - \lambda\, b_i\,) \\ q_i &\leftarrow q_i + \Upsilon\,(\,\mathrm{e}_{ui}\, p_u - \lambda\, q_i) \\ p_u &\leftarrow p_u + \Upsilon\,(\,\mathrm{e}_{ui}\, q_i - \lambda\, p_u) \end{aligned} \qquad \textbf{(2.9)}$$

Learning rate ϒ and regularization parameter λ are the constants that we can determine heuristically with cross validation.

We assign all the initial values randomly except the constants μ, *f*, λ and ϒ

2.1.3 Hybrid

Hybrid recommendation within this scope is a methodology that combines Memory Based methods with Model Based methods. Author combines SVD with KNN to achieve better RMSE result than Memory and Model Based methods (Paterek 2007). These types of methodologies are suitable to produce systems that are more accurate and preferable to overcome problems such as sparsity, loss of information etc. However, the main drawbacks are expense of implementation and complexity.

2.2 CONTENT BASED FILTERING

These methods specify the similarity between two items using content of the item. The content is any available data about the item. If the item is a movie then the content may be genre, summary, directors, actors, actress and so on. The basic advantage of content based filtering to collaborative filtering is about new item problem. Collaborative filtering methods are not able to recommend an unrated item to users whereas count of the ratings does not matter for content based filtering methods. On the other hand, main drawback of the content based filtering is limited content analysis. It is not much easy to label all features and add contents for all items correctly.

2.3 HYBRID RECOMMENDER SYSTEMS

Hybrid recommender systems in this context are combining Collaborative Filtering methods with Content Based Filtering methods. Different ways to combine Collaborative and Content Based methods into a hybrid recommender system can be classified as follows: Implementing Collaborative and Content Based methods separately and combining their predictions, incorporating some Content Based characteristics into a Collaborative approach, incorporating some Collaborative characteristics into a Content Based approach, and constructing a general unifying model that incorporates both Content Based and Collaborative characteristics (Adomavicius and Tuzhilin 2005). It is possible and an efficient way to handle drawbacks of the Content Based and Collaborative Filtering methods by combining them.

.

3. EVALUATION OF RECOMMENDER SYSTEMS

In computer science literature, there are some ways to classify recommender systems. Authors classified the recommender systems in three groups.(Schafer et al.1999) and others preferred to classify in eight groups (Montaner et al 2003). Therefore, it is not so clear in computer science literature but in consideration of evaluation, it is sensible to classify recommender systems based on recommendation task that they are designed for (Gunawardana and Shani 2009). We collect recommender systems under three headings predicting ratings, recommending good items and optimizing utility.

3.1 PREDICTING RATINGS

In predicting ratings, task system must provide a set of predicted ratings. Commonly the predicted rating is compared with the actual rating to find out the accuracy of the prediction. Accuracy is the only metric that can be measured for predicting ratings task.

3.1.1 Accuracy

The accuracy of a system is the most investigated metric in recommender systems history. There are some ways to measure accuracy for predicting ratings tasks such as MAE, NMAE, RMSE, NMRSE etc. (Gunawardana and Shani 2011).

Generally the data set is divided into two parts test and training set. The aim is to predict a rating in test set correctly using the training set.

RMSE and MAE are the most common metrics in research and competitions. RMSE is more sensible to MAE metric because given a test set with four hidden items RMSE would prefer a system that makes an error of 2 on two ratings and 0 on others to one that makes an error of 3 on one rating and 0 on all three others, while MAE would prefer the second system. Even in Netflix Prize competition RMSE was the measure metric.

3.1.1.1 Mae

MAE is a main and simple measure to calculate error in predicting ratings task. It is commonly used in statistics. The aim of MAE is to measure how close forecasts or predictions are to the eventual outcomes. If all series are on the same scale, then the

MAE may be preferred because it is simpler to explain (Hyndman and Koehler 2005). The formula for predicting ratings task is below where T is test set of user item pairs.

$$MAE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} |\hat{r}_{ui} - r_{ui} |} \qquad \textbf{(3.1)}$$

3.1.1.2 Rmse

Due to calculation of square of the error, RMSE is more sensible to penalize the large errors. The formula is below.

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (\hat{r}_{ui} - r_{ui})^2} \qquad \textbf{(3.2)}$$

3.2 RECOMMENDING GOOD ITEMS

Although many competitions concentrate on predicting ratings task, in reality as for the primary purpose of a recommender system is to recommend items to its users, recommender engines generally apply recommending good items task. As “good item” applies to the recommender systems literature, it simply means the item that the user likes. The definition of liking is relativistic and it depends on user’s field of interest. We make assumption that there is a large number of good items that may appeal to the user, and the user does not have enough resources to select all items (Gunawardana and Shani 2009). Therefore, it is important not to present any disliked items and put the best of the good items in a right order.

Recommender systems generally list the recommendations shortly. The length of the list may be 10 or 20. Some recommenders such as Movielens prefer long lists so the user herself may filter the recommendation list according to her criteria. At issue the recommendation list, there are so many metrics to evaluate it such as accuracy, diversity, novelty, serendipity etc.

3.2.1 Accuracy

The accuracy in recommending good items task is similar to predicting the ratings task. Such in predicting the ratings task we divide the data set into two parts but differently from predicting ratings task we consider only the high ratings in test set. We generate a recommendation list to the user who has a high rating in test set. In successful scenario, we expect the recommendation list to contain the item with high rating. We apply this scenario until all high rated items in test set are tested.

3.2.1.1 Precision recall

Common accuracy metrics for recommending good items task are precision recall and ROC Curves.

Table 3.1 : Classification of the possible result of a recommendation of an item to a user

	Recommended	Not recommended
Preferred	True-Positive (tp)	False-Negative (fn)
Not preferred	False-Positive (fp)	True-Negative (tn)

As it is seen in formulas below precision penalize the recommended but not preferred items in recommendation list. On the other hand, recall penalizes preferred but not recommended items. It is clear that if the size of the recommendation list equals to count of the items in data set, the recall score is 1 but we expect precision score to be so low. If we reduce the size of the recommendation list, the precision score increases but the score of recall decreases and vice versa. So it is a tradeoff between precision and recall. ROC Curves are useful to present results fairly and clearly but discussing ROC Curves is beyond the scope of this thesis.

$$\text{Precision} = \frac{\#tp}{\#tp + \#fp} \quad \textbf{(3.3)}$$

$$\text{Recall} = \frac{\#tp}{\#tp + \#fn} \quad \textbf{(3.4)}$$

3.2.2 Diversity

Verbal definition of diversity is the variety in a recommendation list. Diversity enhancement in a recommender system aims to increase the number of different items in a single recommendation list. The difference between two items can be measured using contents of the items (if exist), distance between two items etc.

There are different metrics to measure different dimensions of diversity in a recommendation system. However, a general definition for collaborative filtering is one minus similarity.

Diversity implies loss in accuracy so in research of diversity showing results for accuracy is essential.

3.2.3 Novelty

In the scope of novelty, recommender systems try to recommend unknown but relevant items to the user. In a recommender system, the recommendation lists usually contains popular items in virtue of the nature of the recommendation algorithms. However, recommending Titanic movie to a user is not so surprising at all. The user loyalty depends on recommending unknown items to the user. User should also like the unknown items in list.

Authors defined novelty as the formula below (Zhou et al. 2010).

$$I_a = log_2(\frac{u}{k_a}) \tag{3.5}$$

where a is an item and u is the number of users. k_a the number of users who rated item a. So $\frac{k_a}{u}$ is the chance a randomly selected user has collected it. So in consideration of this formula novel item is an item, which is collected or rated by least amount of users.

Another definition is below.

$$n_L(i) = \frac{1}{\mathrm{p} - 1} \sum_{j \in L} d(\mathrm{i}, \mathrm{j}) \tag{3.6}$$

In the formula above p is the size of the recommendation list, $d(\mathrm{i}, \mathrm{j})$ is the distance between two items, L is the user likes. With this definition, set diversity is the average novelty of the items in the set (Hurley and Zhang 2011).

3.2.4 Serendipity

Serendipity of a recommender system depends on recommending unexpected and useful items to its users. Authors analyzed the etymology of this word. They found that serendipity is mostly related to the quality of recommendations in RS-related research, that it largely depends on subjective characteristics, and that it is a difficult concept to study (Iaquinta et al 2008).

Recommendations of a recommender system must be for the benefit of the user. The benefit in clearly saying can be defined as useful, relevant, unexpected, accurate etc. Serendipity is a metric that tries to meet most of these terms. So definition of the metrics such as serendipity, diversity and novelty may mix the readers up. In order to differentiate between novelty, diversity and serendipity, authors provide the following example. In the list of recommended action movies, the user might also find an unknown movie, which is interesting to her. This action movie is then called novel recommendation instead of serendipitous recommendation because she might discover this movie by herself. If we found that the user likes action movies might also like comedy movies, we can diversify the recommendation list by adding a comedy movie. It is then called a diverse recommendation instead of a serendipitous recommendation, as she might be not surprised about the recommendation (Ge at al. 2010).

3.3 OPTIMIZING UTILITY

This topic is about optimizing the user interactions in recommender systems. Presentation of the recommendations is as important as what you recommend. Presentation and user interaction are trendy fields of study nowadays.

As an example of presentation, horizontally listed top five recommendations eases user to observe the list at first glance. Another option may be the long lists but most of the users do not bother to scan down the entire list. Therefore, the recommendations in top five or ten are the outstanding recommendations and must be stunning for the user.

Users like to see changes in recommendation lists. It might be annoying to the user if the item is always in the list until she rates it. Therefore, in consideration of user interactions, the system may eject the recommended item from the list unless the user rates or collects it in a stated period.

Authors suggested a method called half-life utility score where the system produces an unbounded list (Breese et al 1998). This approach drops down items exponentially, which are top of the recommendation list. Users generally look the items starting from the top. Assuming this fact the probability of being viewed formula for an item is below where k is the position of an item and α is a half-life parameter. Half-life period may depend on time or viewing number of the recommendation list. The probability score P specifies the new location of the item.

$$P = \frac{1}{2^{\frac{(k-1)}{(\alpha-1)}}} \tag{3.7}$$

4. OUR METHODOLOGY

4.1 SIMILARITY MEASURE

We use cosine similarity to determine similarities between items or users. We create two vectors and compute the similarity as cosine of the angle. We consider not only co-rated items but also at least one of the items has been rated. We assign 0 to the unrated items.

4.2 CLUSTERING

Clustering aims to assign items to group which are similar to each other. Clustering is a widely used technique that is used in computer science literature. Two main types of clustering are hierarchical and partitional. Partitional clustering algorithms divide items into non-overlapping clusters. On the other hand, hierarchical algoritms assign items to the found clusters.

4.2.1 K-Means

K-Means is the most common partitional clustering algorithm. This algorithm firstly selects *k* initial cluster centroids in data set where *k* is the number of clusters. Then all non centroid items are assigned to the nearest cluster. After we assign all items, we recalculate the cluster centroids. We loop until none of the centroids change.

4.3 EVALUATION MEASURES

We test our algorithm using three different measures. These are recall, diversity and novelty. As for the scope of the thesis, diversity is the main measure. However, generally in research, balancing diversity and accuracy is an optimization problem so trade-off between these measures is inevitable. Archiving improvement on diversity with minimum loss of accuracy is essential. So accuracy is another significant measure in our tests. An interesting research direction would be to develop a new measure that captures both of these aspects in a single metric (Adomavicius and Kwon 2011).

Another research direction may be increasing diversity without loss of accuracy where satisfying improvement has not been achieved.

The subtitle of diversity measure, diversity histogram, is not a measure itself but it is important to observe the diversity improvement of the users whose diversity score for their recommendation list is low. Therefore, we tabularize diversity histograms.

Finally, novelty is the last measure we use for evaluation. Novelty is a notable measure because diversity applies to a set of items and is related between items but novelty is about each item in a set.

4.3.1 Recall

Authors develop a methodology to test recall for top-n recommendation tasks (Cremonesi et al 2010). We apply this methodology without any major changes. We split data set into two subsets training set M and test set T where T is 1.4% of all ratings. We use only the high ratings in T. The high rating for Movielens means 5-star ratings. For Jester and Bookcrossing data sets we assume 9 and above scores as high ratings. For all the high rated items in T we randomly select 400 additional unrated items. We may assume that the user is not interested in most of them. We predict the ratings for the test item i and for the additional 400 items. Because of the number of items in Jester data set we select all unrated items. We rank 401 items order by their predicted ratings and generate a top n recommendation list. In thesis we assume n=20 for Bookcrossing and Movielens datasets. As for Jester we assumed n=10 by the reason of all item count in data set. If the high rated item is in the recommendation list we have a hit. So the recall formula is;

$$\text{recall} = \frac{\#hits}{|T|} \qquad \textbf{(4.1)}$$

where |T| is the number of high rated items in T.

4.3.2 Diversity

In this thesis we use the metric as Barry at al. (2011) and Neil at al. (2011) described for measuring the diversity of a recommendation list of a particular user. This metric measures the diversity as the average dissimilarity of all pairs of items in the recommendation list. Formally, it canbe defined as follows:

$$\mathrm{D(R)} = \frac{1}{N(N+1)} \sum_{i \in R} \sum_{y \in R, j \neq i} d(\mathrm{i}, \mathrm{j}) \qquad \textbf{(4.2)}$$

where R is the recommendation list of a user and N = |R|. d(i, j) is the dissimilarity of items i and j which is defined as one minus the similarity of items i and j. We use cosine similarity for measuring the similarity between two items. In measuring cosine similarity we only considered co-ratings and if there are no co-ratings between two items we take their similarity to be 0.

4.3.2.1 Diversity histogram

Although the overall performance of an algorithm is good, the algorithm may influence some users negatively. If we observe only the overall performance, we may not see the whole picture. If some users' diversity score stay the same or decrease while the overall performance of the system increases, we have to admit that the algorithm does not address all users' needs. As we mentioned in previous sections diversity histogram is not a measure itself. But we want to observe the diversity improvement of the users who has low diversity score for their recommendation list. Two dimensional graphics show number of users and diversity score of a list. When the diversity increases, we expect diversity histogram bars gather around the high diversity scores.

4.3.3 Novelty

Authors formulate novelty as the average distance of the recommended items to the items that user likes (Hurley and Zhang 2011). We convert this to a more general definition. We define novelty as the average distance of the recommended items to all items in data set. So novelty of an item is stable and does not change depending upon a user. The formula for novelty of an item i is below.

$$n(i) = \frac{1}{|\mathrm{I}| - 1} \sum_{j \in I, j \neq i} d(\mathrm{i}, \mathrm{j}) \qquad \textbf{(4.3)}$$

where $d(\mathrm{i}, \mathrm{j})$ is the distance between two items, I is the all items in data set.

4.4 DATA SETS

We used three data sets to evaluate our algorithm. These are Movielens, Jester and Bookcrossing data sets. All these data sets are available online. Movielens dataset is a commonly used dataset in recommender systems research. Jester is another dataset that contains just 101 items. Challenge for this dataset is number of items. There are not many variations of recommendation lists so it does not seem easy to find out novel and diverse recommendation lists. Bookcrossing is the sparsest dataset that we used for evaluation. Considering the similarity measure this data set is relatively the most diverse one. Diversity enhancement in a diverse dataset is much harder than in a not diverse data set. The main success of the algorithms for diversity becomes more visible if we consider the mean and standard deviation of the data sets for diversity that we will describe in detail next sections.

We represent the item and user counts of data sets that we use in a table below.

Table 4.1 : Item and user counts of data sets that we use.

	Item	**User**
Movielens	3953	6041
Jester	101	5000
Bookcrossing	2177	6358

4.4.1 Movielens

We used Movielens data set that consists of 1 million ratings from 6000 users on 4000 movies. Movielens data set were explicitly rated integers ranging from 1 to 5. In tests, we kept whole data without any changes.

4.4.2 Jester

Different types of Jester dataset are available online. We choose data from 24,938 users who have rated between 15 and 35 jokes, a matrix with dimensions 24,938 X 101. Due to lack of memory and time, we reduced the dataset picking randomly 5000 users. Ratings are decimal values ranging from -10 to +10.

4.4.3 Bookcrossing

Bookcrossing dataset contains 278,858 users providing 1,149,780 explicit and implicit ratings about 271,379 books. We reduced the dataset picking users who rate 20 or more items and items that are rated 10 or more times.

4.4.4 Analyzing Data Sets Via Diversity

We construct item similarity matrixes for each data set by using cosine similarity measure, which we explain in detail previous sections. We calculated the mean and standard deviation scores of each data set.

Mean score is the average of (1 – similarity) for each item pair. We expect mean score to be equal to the diversity score if we randomly generate a recommendation list using entire set of items. By the reason we cannot recommend item, which is already rated by a user, mean score may be a little different from diversity score although we generate recommendation lists randomly. This constraint is trivial for Movielens and Bookcrossing data sets but in Jester data set it is not. Because Jester data set contains just 101 items and users rate approximately 30 percent of the items.

Standard deviation shows how much variation exists from the mean. A low standard deviation value specifies that the similarity of the item pairs tends to be very close to the mean whereas high standard deviation value indicates that the similarities of the item pairs are spread out over a large range of values. If standard deviation score is high, relatively recommending diverse lists is easier and vice versa. The mean and standard deviation scores are below.

Table 4.2 : Mean and standard deviation scores of data sets.

	Movielens	**Jester**	**Bookcrossing**
Mean	0.924	0.847	0.982
Standard Deviation	0.075	0.170	0.031

In consideration of mean and standard deviation scores Jester seems like the easiest data set to achieve diversity improvement. Because, mean score of Jester is lowest and the standard deviation score is the highest. Bookcrossing appears like the challenging data set. Because mean score shows that Bookcrossing is already a diverse data set depending upon the similarity measure. Standard deviation score of Bookcrossing shows that it is not easy to find out pairs that are more diverse. Movielens data set is middle of two others.

4.5 PREDICTION ALGORITHMS

We use two Memory Based and one Model Based algorithms to predict ratings. These are item-based, user-based and SVD algorithms. These algorithms, which we explain in previous sections, are most common algorithms in research and practice. We believe that these three algorithms are enough to represent all collaborative filtering algorithms due to time and space constraints. In addition, we apply normalization steps for item based and user based algorithms, which we describe extensively in next section.

4.6 RATING NORMALIZATION

Normalization step is important for recommender systems. Rating scales of the users may be different from each other. And also rating scale of an individual user may change in time, depends on her mood, time of day, seasons etc. A recommendation algorithm should consider these kinds of changes Two of the most popular rating normalization schemes that have been proposed to convert individual ratings to a more universal scale are mean-centering and Z-score (Gunawardana and Shani 2011).

Mean centering approach tries to determine the rating is negative or positive. Let us say Bob gives a rating of 4 to a movie and Alice rates 3. If the average rating of Bob is 4.5, we assume that Bob's rating as a negative rating for his scale although it looks like a positive rating. In a similar way, Alice's rating of 3 is a positive rating if her average rating is 2.5.

Mean approach determines the mean centered rating h(r_{ui}) simply by subtracting to r_{ui} the average $\overline{\mathrm{r}}_{u}$ of the ratings given by the user u.

$$\mathrm{h}(r_{ui}) = r_{ui} - \overline{\mathrm{r}}_{u} \tag{4.4}$$

In consideration of formula above, user based prediction of a rating r_{ui} is obtained as follows.

$$\hat{r}_{ui} = \overline{\mathrm{r}}_{u} + \frac{\sum_{v \in N_i(u)} w_{uv}(r_{vi} - \overline{\mathrm{r}}_{v})}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{4.5}$$

In a similar way, we get the formula for item based prediction of a rating r_{ui} .

$$\hat{r}_{ui} = \overline{\mathrm{r}}_{\mathrm{i}} + \frac{\sum_{\mathrm{j} \in \mathrm{N_u(i)}} q w_{uv}(r_{vi} - \overline{\mathrm{r}}_{v})}{\sum_{\mathrm{j} \in \mathrm{N_u(i)}} |\mathrm{w_{ij}}|} \tag{4.6}$$

We applied mean centering approach for item based and user based collaborative filtering on tests.

Another rating normalization approach is z-score. Let's say that both Alice and Bob's average rating is 3 but Alice's rating scale is between 1-5, Bob's rating scale is between

2 and 4 so a rating of 5 given to an item by Alice is more exceptional than Bob. It is not possible to consider this case using mean centering but z score takes into consideration of this case. We name б_u as standard deviation of the ratings given by user u and h(r_{ui}) is;

$$\mathrm{h}(\, r_{ui}\,) = \frac{r_{ui} - \overline{\mathrm{r}}_u}{\text{б}_u} \tag{4.7}$$

So the user based predicted rating r_{ui} is;

$$\hat{r}_{ui} = \overline{\mathrm{r}}_u + \text{б}_u \frac{\sum_{v \in N_i(u)} w_{uv}(r_{vi} - \overline{\mathrm{r}}_v)/\text{б}_v}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{4.8}$$

And finally, item based predicted rating is obtained as follows.

$$\hat{r}_{ui} = \overline{\mathrm{r}}_i + \text{б}_i \frac{\sum_{j \in N_u(i)} w_{ij}(r_{uj} - \overline{\mathrm{r}}_j)/\text{б}_j}{\sum_{j \in N_u(i)} |w_{ij}|} \tag{4.9}$$

.

4.7 DESCRIBING RANDOM METHODOLOGY

We apply random methodology as a baseline method to compare our results. The simplest strategy for increasing the diversity of a set of k items is the Bounded Random Selection method (Bradley and Smyth 2001). We arrange the items on descending order by their predicted ratings for a particular user. For Movielens and Bookcrossing data sets firstly, we pick top 50 items and generate a recommendation list by selecting randomly 20 of them. Then we enlarge the bounds of picking top n items to 100, 150, 200 and so on. For Jester data set we pick top 20 items in first step and similarly extend the bounds of picking top n items.

4.8 DESCRIBING OUR METHODOLOGY

In this section, we will describe our algorithm in detail. Similar to many recommendation algorithms our algorithm has offline and online phases. In the offline phase, apart from model building, we build N (where N is the size of the recommendation list) item clusters and compute the similarities between them. We build item clusters using the K-Means clustering algorithm with some changes. We cluster items based on their ratings given by the users of the system. No content information about items is used. This is one of the strengths of our approach because content information about items is usually difficult to obtain.

On item clustering step we apply the following way: Before we begin, it is useful to specify that we calculate the similarities between items as we mention in former sections and each item has *n* dimension where *n* is the number of users. Firstly, we assign random and unreal centroid items to all clusters. Then for each item, we determine the nearest cluster by comparing distances of the item to all centroids. When we assign all items to clusters, we reappoint the centroid items and determine the nearest clusters for items again. We loop until none of the centroid items change.

After item clusters are constructed, we calculate the similarities between clusters. We calculate the similarities between clusters as the average similarity of pairs of items from different clusters. Formally, the similarity between cluster $C_i$and C_j is calculated as

$$\mathrm{sim}(C_i, C_j) = \frac{1}{|C_i|.|C_j|} \sum_{x \in C_i} \quad \sum_{y \in C_j} s(\mathrm{x}, \mathrm{y}) \qquad \mathbf{(4.10)}$$

where s(x, y) is the cosine similarity between items x and y.

At the heart of our algorithm lies the construction of cluster weights (CW). CW is a vector whose ith element (CW_i) holds the number of items which cluster C_i will contribute to the recommendation list of a particular user. Every user has own CW vector. So, for example, if CW_i = 5 for user u then it means that cluster C_i will contribute 5 items (those which have the highest predicted ratings in C_i) to the recommendation list of user u. It follows that the sum of cluster weights for any user should be equal to N (the recommendation list size).

The last step of offline phase is to arrange items on descending order by their predicted ratings for a particular user.

The online phase stands for determining how many items will be selected for the recommendation list from each cluster and finding out these items. The following is the algorithm that we construct to determine the count of selected items from each cluster, which we call cluster weights.

Algorithm 1 Cluster Weights

```
Input: topN: top-N list for a particular user, C: Item clusters
Output: CW: Cluster weights
1: $CW_i = |C_i \cap topN|$
2: for all $C_i \in C$ do
3:     while $|C_i| > th$ do
4:         $c = argmax_j\ d(C_i, C_j)\ s.t\ |C_j| > th$
5:         $CW_c = CW_c + 1$
6:         $CW_i = CW_i - 1$
7:     end while
8: end for
```

$$d(C_i, C_j) = \frac{1 - sim(C_i, C_j)}{CW_j + 1} \qquad \textbf{(4.11)}$$

ClusterWeights algorithm computes the cluster weights for a particular user. It takes two parameters: the top-N recommendation list for the target user u which is generated by the prediction algorithm, and the set of clusters $C \quad = C_1, C_2, , C_k$.

In line 1 we initialize cluster weights. Each cluster weight CW_j is assigned the number of items in C_i which are also in topN. In other words, initially the set of items contributed by all the clusters is exactly same as the top-N recommendation list generated by the prediction algorithm. Then, in line 2, the algorithm begins to distribute the cluster weights of those clusters whose weights are larger than a given threshold (*th*). This *th* value allows us to control the trade-off between accuracy and diversity. As *th* gets larger the user gets more accurate but less diverse results and as *th* gets smaller the user gets more diverse but less accurate results. In the extreme case if *th* is equal to N then the recommendation list generated by the prediction algorithm does not altered at all. The algorithm distributes the weights of clusters whose weights are larger than the threshold value to other clusters whose weights are smaller than the threshold value. The algorithm tries to distribute weights so as to achieve high item diversity without decreasing accuracy too much. In order to increase diversity the algorithm selects those clusters, which are far away from the source cluster, and also whose weights are small. We achieve this using the distance function defined above. Distributing weights to far away clusters increases diversity because items in distant cluster are also less similar to each other. The algorithm also tends to distribute to clusters whose weights are small because concentration of weights in a small number of highly weighted clusters will tend to decrease diversity. Therefore, the algorithm tries to distribute the weights to different clusters as much as possible.

The strength of our algorithm comes from the ability to change the diversity of recommendation list online. As for the Big-*O* notation for our algorithm is $O(n)$, the users can diversify their recommendation list using threshold parameter on runtime.

4.9 EXPERIMENTAL RESULTS

Th = 20 (or *th* = 10 for Jester data set) is default values for prediction algorithms. Figure A.2, Figure A.5 and Figure A.8 show diversity scores for different threshold on different data sets. For all data sets, we can clearly see that when the *th* value decreases, the diversity of recommendation lists increases. We assume that the loss in accuracy is inevitable and Figure A.1, Figure A.4 and Figure A.7 show recall values in different thresholds for three data sets. In Figure A.3, Figure A.6 and Figure A.9 we can also see that this algorithm is able to increase the novelty of recommendation lists. Diversity histograms are helpful to observe whole diversity changes in recommendation lists. Figure A.10 shows that some users' recommendation lists have so low diversity scores. As the threshold value decreases in Figure A.11, Figure A.12, Figure A.13, Figure A.14 and Figure A.15, we can see that all recommendation lists have good diversity scores and algorithm affects all users positively. Therefore, we can say that this algorithm addresses all users' needs. Figures from A.15 to Figure A.24 have the same results for Jester and Bookcrossing data sets. Figure A.25, Figure A.26 and Figure A.27 are Movielens results for random methodology that we described in previous sections. N is top N items of predicted ratings list that we use to generate a recommendation list by picking randomly some of them (10 for Jester data set and 20 for others). Figure A.28, Figure A.29, Figure A.30, Figure A.31, Figure A.32 and Figure A.33 represent results of the same experiment for Jester and Bookcrossing data sets. We expect as the N parameter increases, diversity score will increase. However, until N=300 it is not as we expected for user based and item based methods. We enlarge the N parameter to item count in data set and see that diversity scores are just like the diversity mean scores as we mention in alter sections. Therefore, random method is a way to increase diversity for recommender systems but enlarging N parameter does not always increase diversity.

5. CONCULUSION

We proposed a methodology to diversify the recommendation lists, which is suitable for recommender systems. We can easily integrate this methodology to a living recommender system. Using this methodology, users can have more diverse or accurate lists immediately using a tunable parameter. We tested our algorithm using different data sets and different prediction algorithms. We compared our method with random method and we showed that our method has better results in all measures. In addition, we showed that this algorithm is able to address all users' needs.

For future work, this method can be reconsidered in content based recommendation perspective. It would be beneficial to see the performance of this method on content based recommenders. Therefore, we can have an enhanced algorithm, which can address both content based and collaborative filtering methods' needs. Another important measure is aggregate diversity, which measures the diversity of whole system. We can apply some new features to deal with aggregate diversity. Finally, we can also define a new diversity measure, which considers mean and standard deviation of data sets to measure our algorithm.

REFERENCES

Books

Desrosiers, C. & Karypis, G., 2011. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*

Gunawardana, A. & Shani, G., 2011. *Evaluating Recommendation Systems*

Koren, Y. & Bell, R., 2011, *Advances in Collaborative Filtering*

Periodicals

Adomavicius, G. & Kwon, Y., 2011, *Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques.*

Adomavicius, G. & Tuzhilin, A., 2005, *Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.*

Bell, R., Koren, Y & Volinsky, C., 2007, *The BellKor Solution to the Netflix Prize.*

Breese, J., Heckerman, D. & Kadie C., 1998, *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.*

Cremonesi, P., Koren, Y. & Turrin, R., 2010, *Performance of Recommender Algorithms on Top-N Recommendation Tasks.*

Goldberg, D., Oki, B., Nichols, D. & Douglas, B., 1992, *Using Collaborative Filtering to Weave an Information Tapestry.*

Gunawardana, A. & Shani, G., 2009, *A Survey of Accuracy Evaluation Metrics of Recommendation Tasks.*

Hurley, N. & Zhang, M., 2011, *Novelty and Diversity in Top-N Recommendation – Analysis and Evaluation.*

Hyndman, R. & Koehler, A., 2005, *Another Look at Measures of Forecast Accuracy.*

Iaquinta, L., Gemmis, M., Lops, P., Semeraro, G., 1999, *Recommender systems in e-commerce.*

Linden, G., Smith, B. & York, J., 2003, *Amazon.com Recommendations Item-to-Item Collaborative Filtering.*

Montaner, M., L´opez, B. & De La Rosa, J. L., 2003, *A taxonomy of recommender agents on the internet.*

Mouzhi, G., Carla, D. & Dietmar, J., 2010, *Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity.*

Mukesh, D., 2007, *Personalized Social & Real-Time Collaborative Search, ACM 978-1-59593-654-7/07*

Paterek, A., 2007, *Improving regularized singular value decomposition for collaborative filtering.*

Sarwar, B., Karypis, G., Konstan, J. & Riedl J., 2000, *Analysis of Recommendation Algorithms for E-Commerce ACM Conf. Electronic Commerce, ACM Press.*

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J., 2001, *Item-Based Collaborative Filtering Recommendation Algorithms.*

Schafer, J., Konstan J. & Riedi J., 1999, *Recommender systems in e-commerce.*

Smyth, B. & Bradley, K., 2001, *Improving Recommendation Diversity*.

Su, X. & Khoshgoftaar, T., 2009, *A Survey of Collaborative Filtering Techniques.*

Zhou, T., Kuscsika, Z., Liua J., Medoa, M., Wakelinga, J.& Zhanga, Y., 2010, *Solving the apparent diversity-accuracy dilemma of recommender systems.*

Other References

Bookcrossing Data Set. [Online] Available at: http://www.grouplens.org/node/74
Accessed: 01 June 2012

Jester Data Set. [Online] Available at: http://www.grouplens.org/node/75
Accessed: 01 June 2012

Movielens Data Set. [Online] Available at: http://www.grouplens.org/node/73
Accessed: 01 June 2012

APPENDICES

APPENDIX A: Experimental Results Details

Figure A.1: Movielens Recall

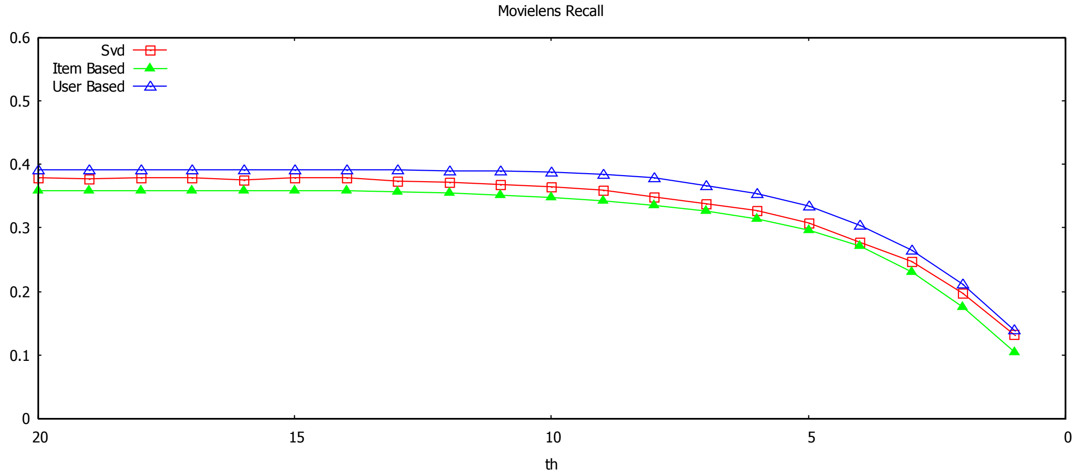


Figure A.2: Movielens Diversity

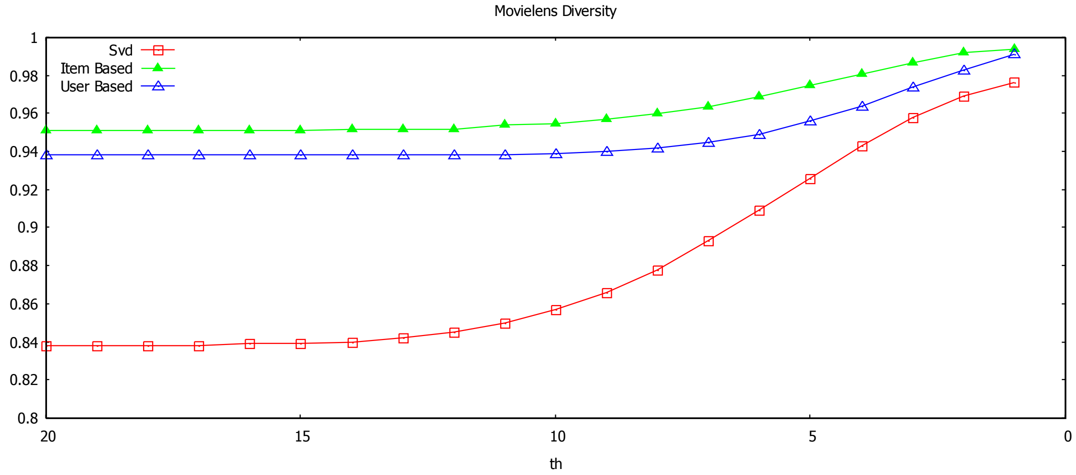


Figure A.3: Movielens Novelty

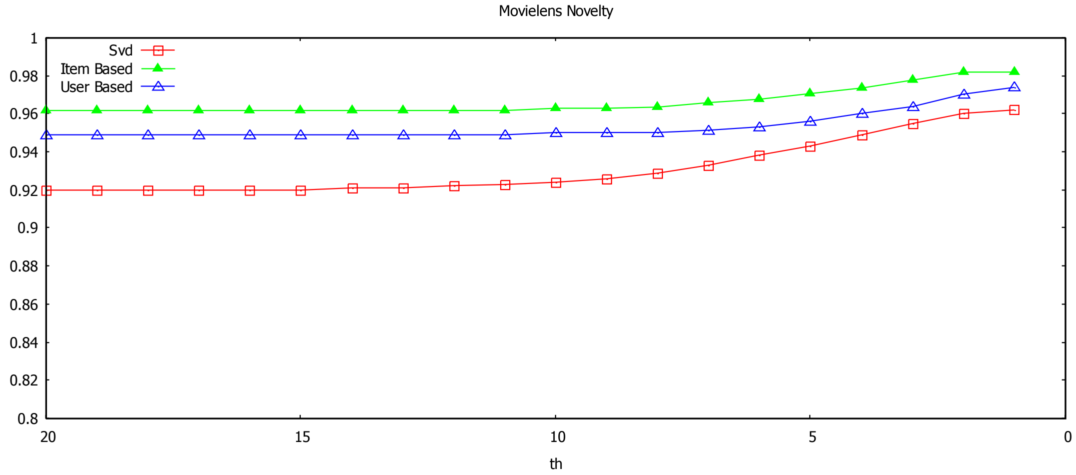


Figure A.4: Jester Recall

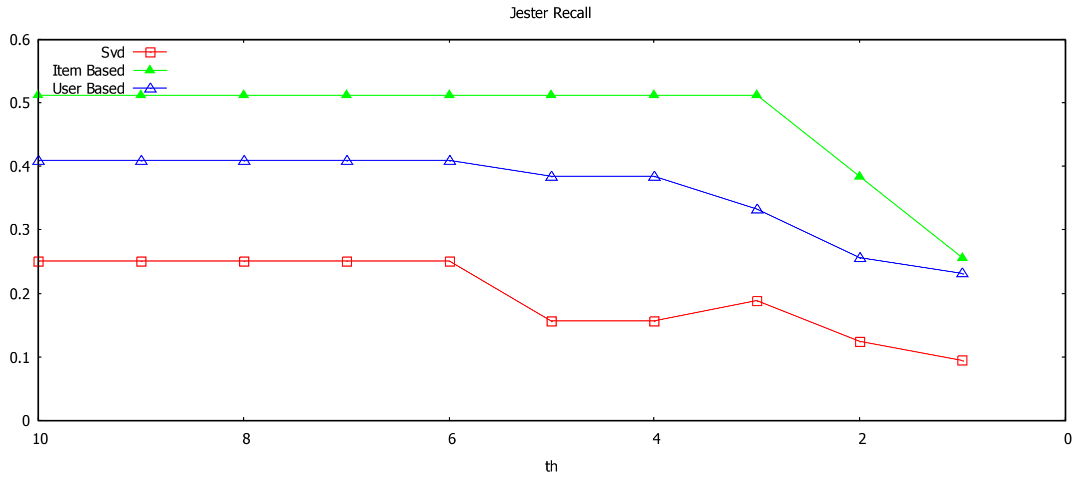


Figure A.5: Jester Diversity

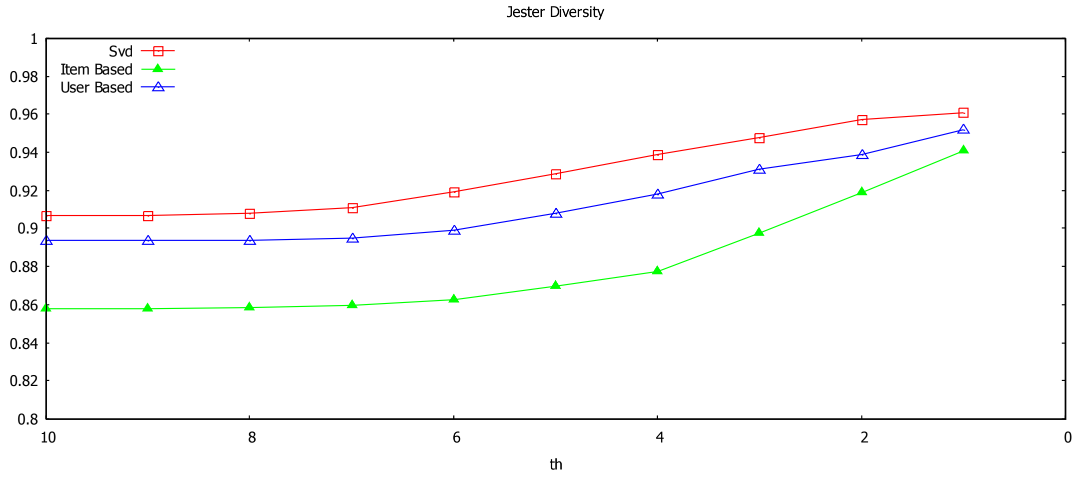


Figure A.6: Jester Novelty

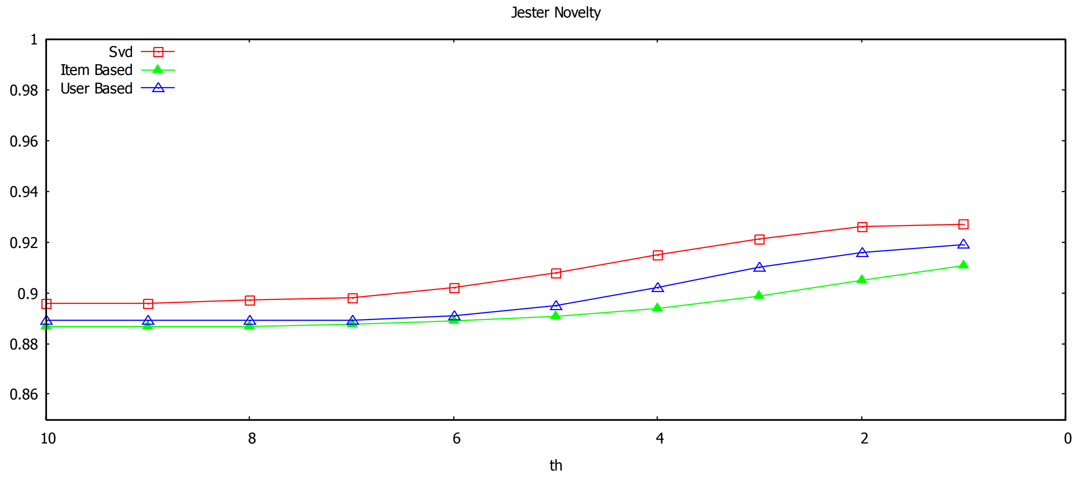


Figure A.7: Bookcrossing Recall

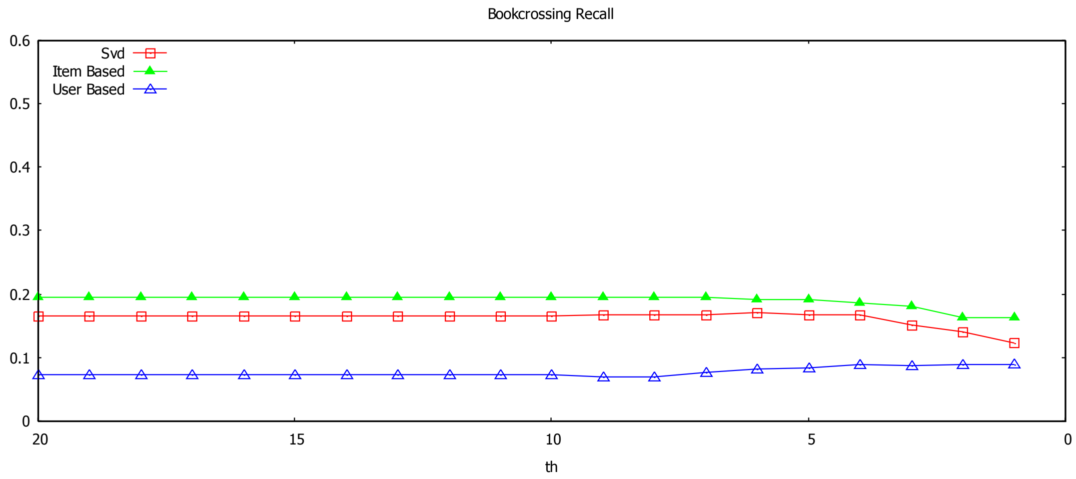


Figure A.8: Bookcrossing Diversty

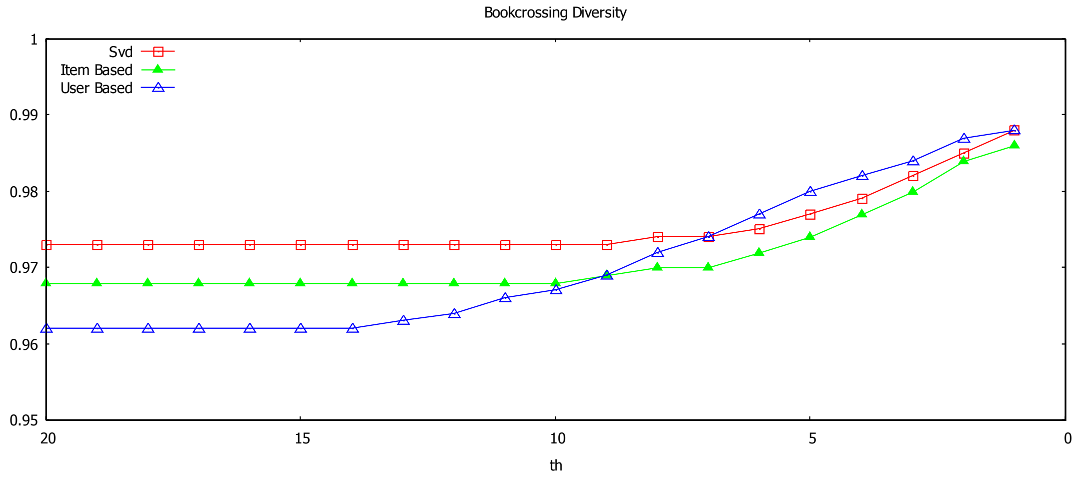


Figure A.9: Bookcrossing Novelty

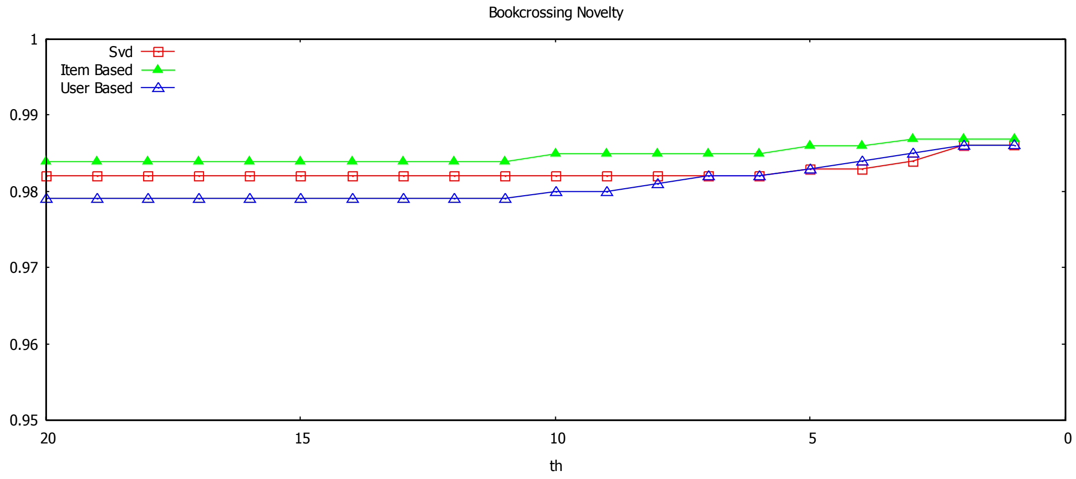


Figure A.10: Movielens Diversity Histogram TH:20

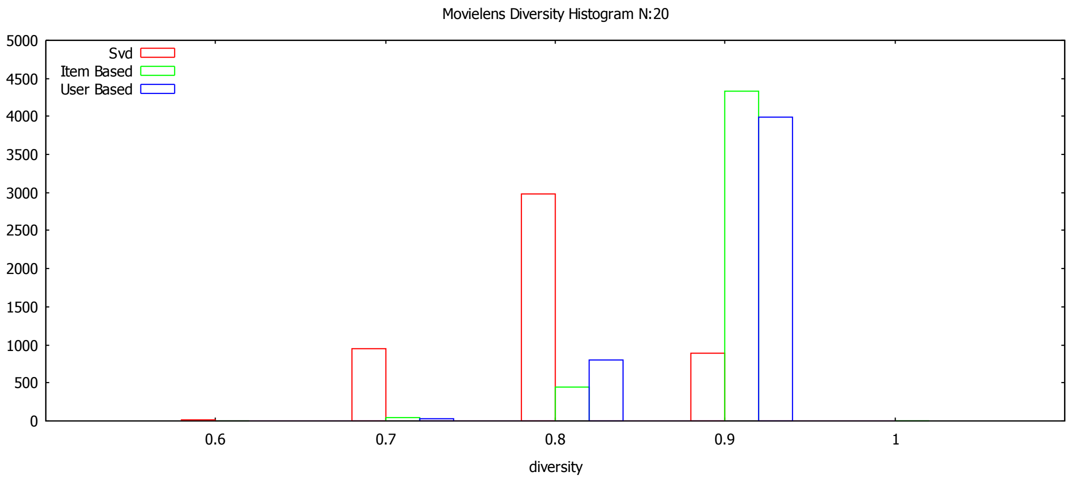


Figure A.11: Movielens Diversity Histogram TH:15

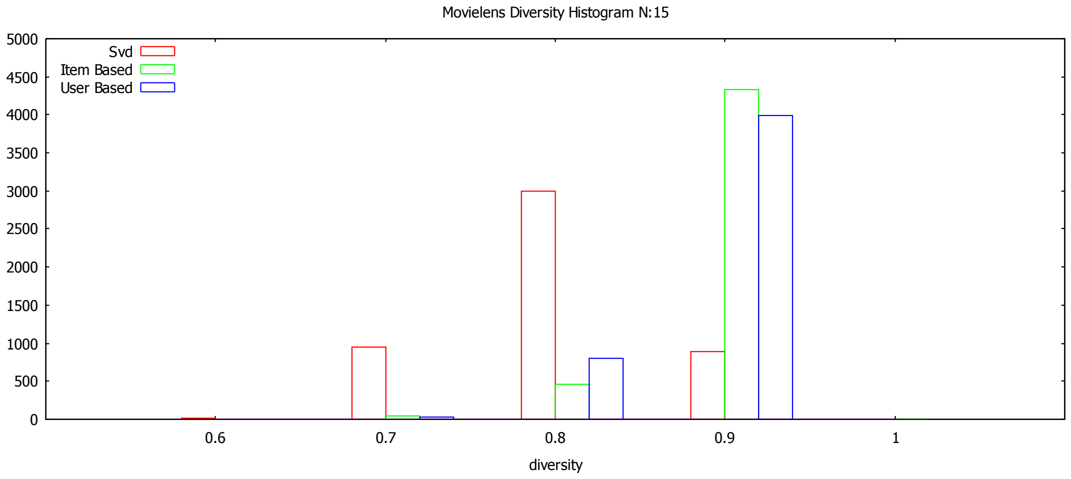


Figure A.12: Movielens Diversity Histogram TH:10

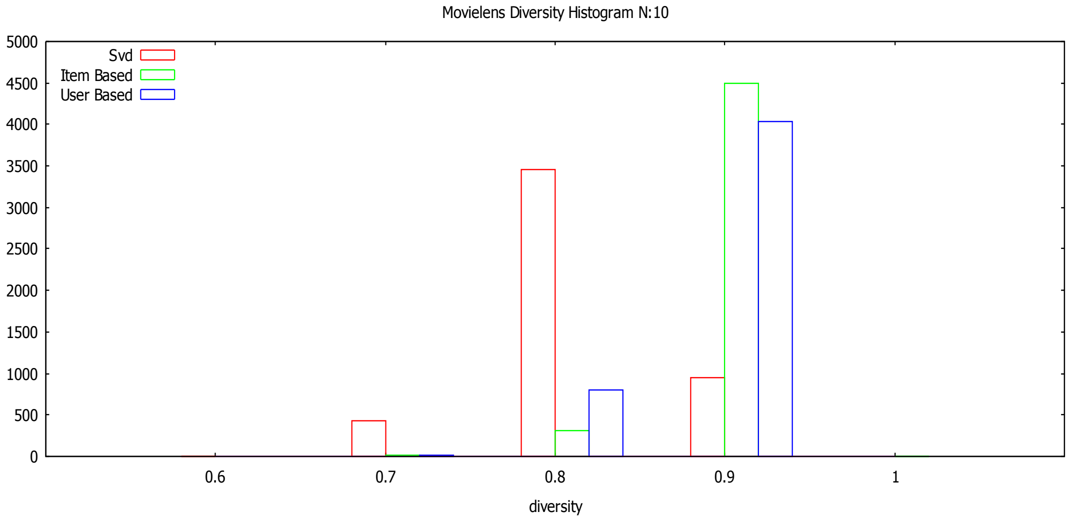


Figure A.13: Movielens Diversity Histogram TH:5

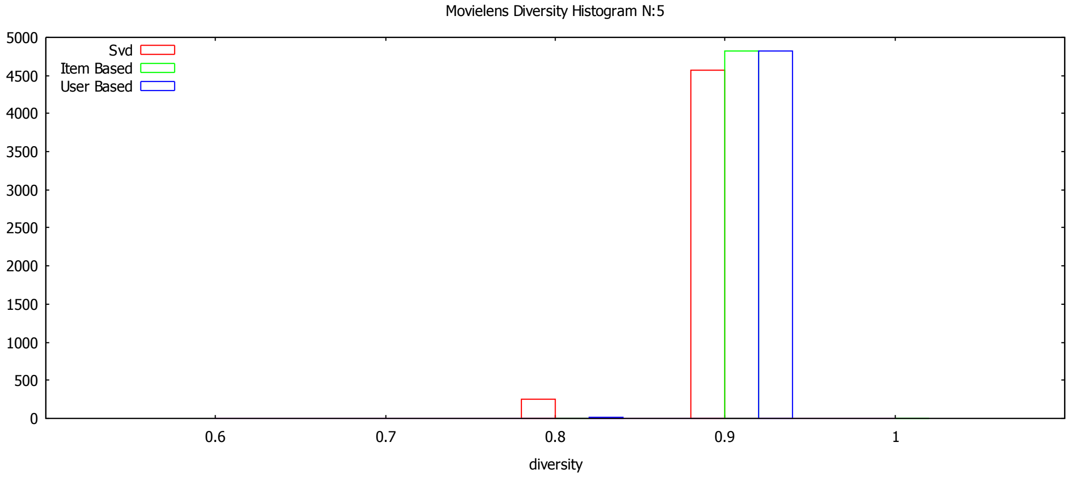


Figure A.14: Movielens Diversity Histogram TH:1

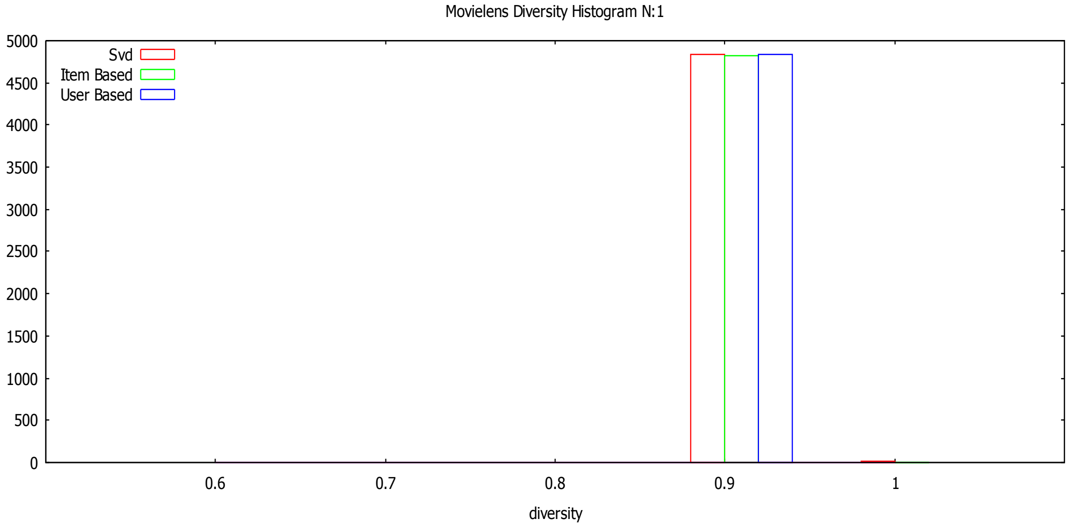


Figure A.15: Jester Diversity Histogram TH:10

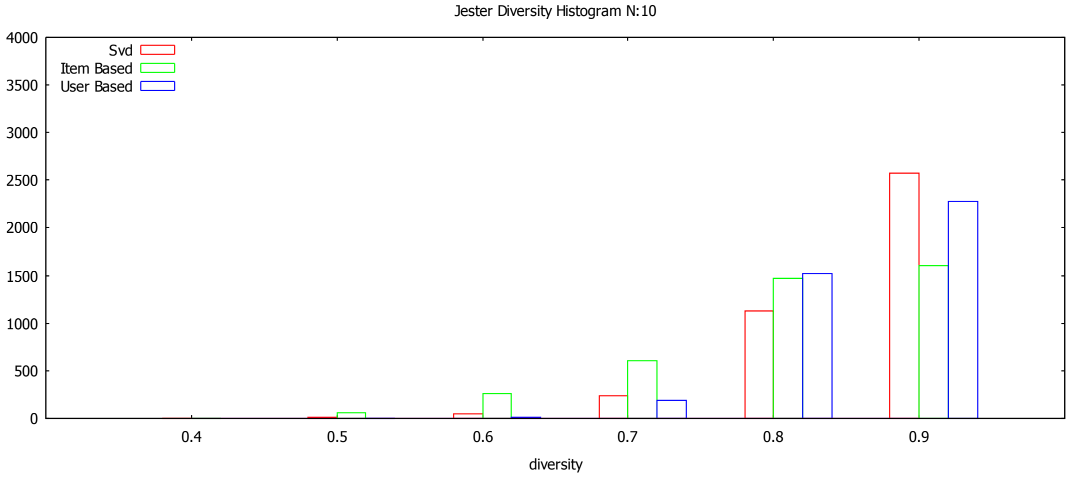


Figure A.16: Jester Diversity Histogram TH:7

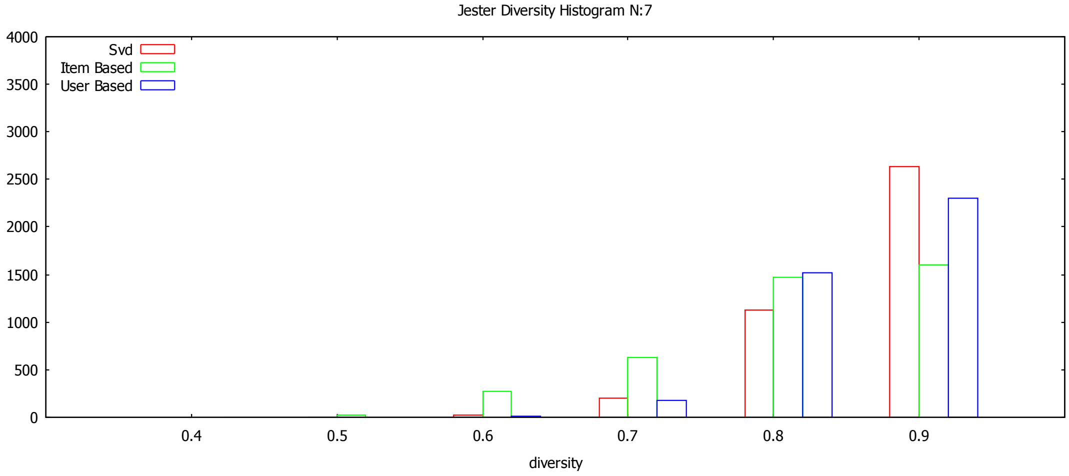


Figure A.17: Jester Diversity Histogram TH:5

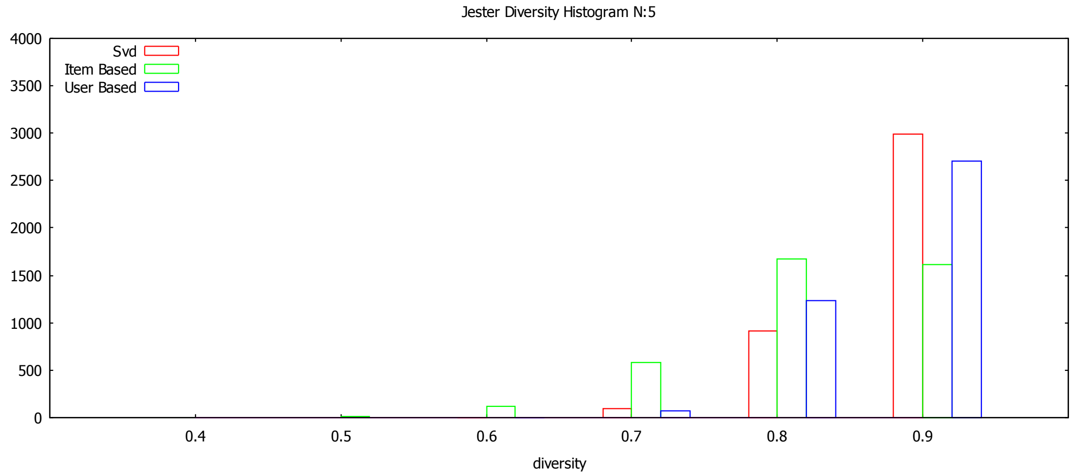


Figure A.18: Jester Diversity Histogram TH:3

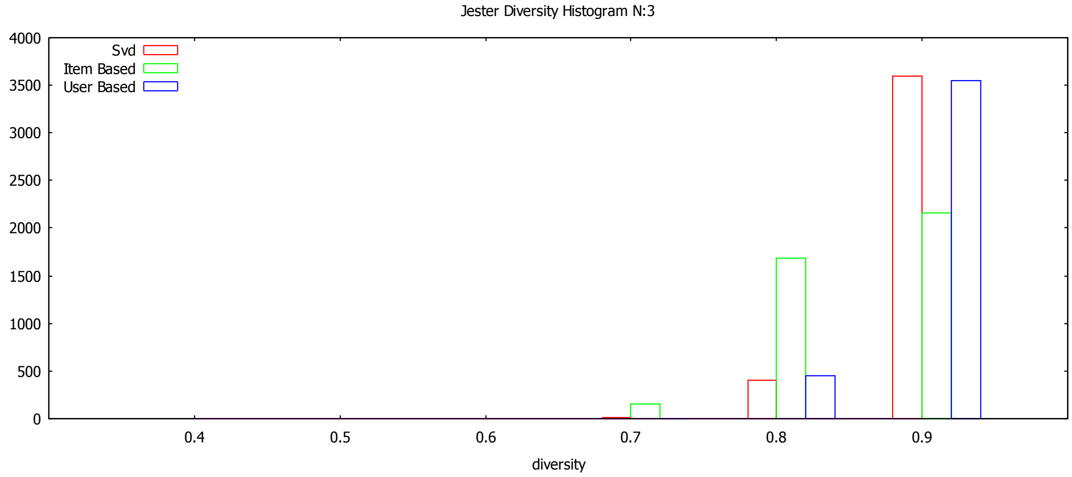


Figure A.19: Jester Diversity Histogram TH:1

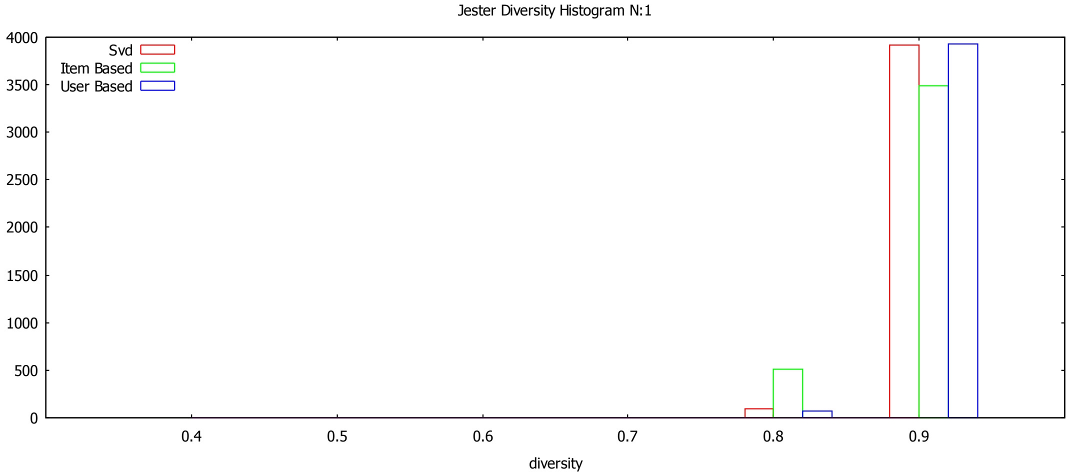


Figure A.20: Bookcrossing Diversity Histogram TH:20

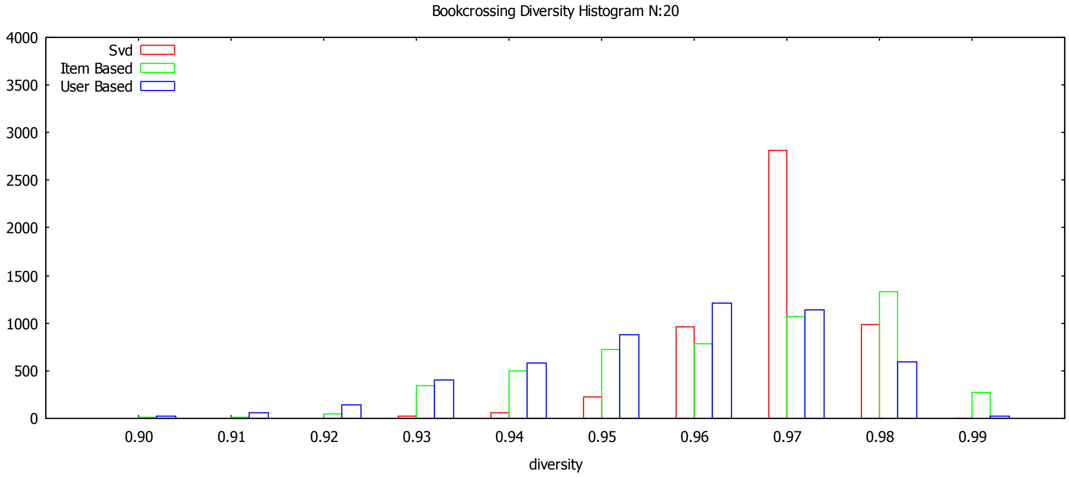


Figure A.21: Bookcrossing Diversity Histogram TH:15

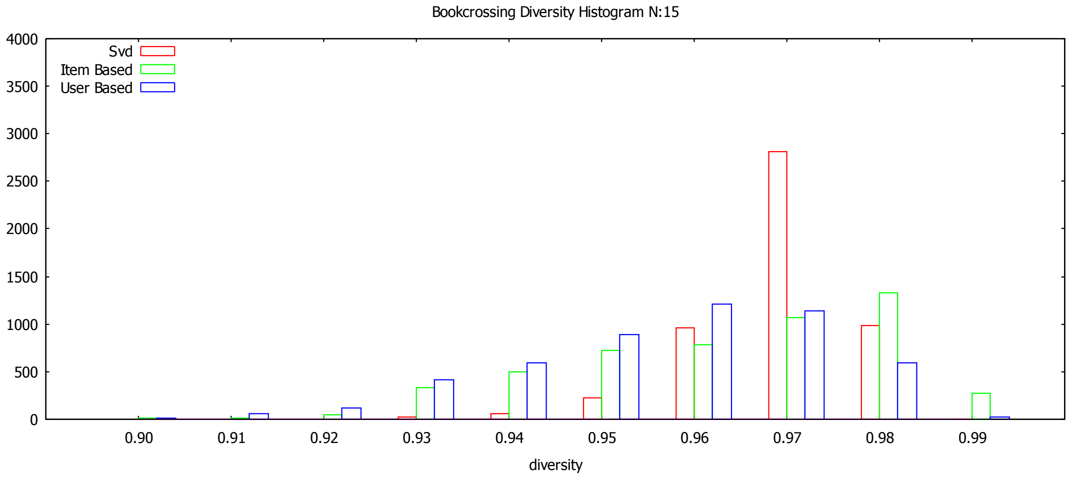


Figure A.22: Bookcrossing Diversity Histogram TH:10

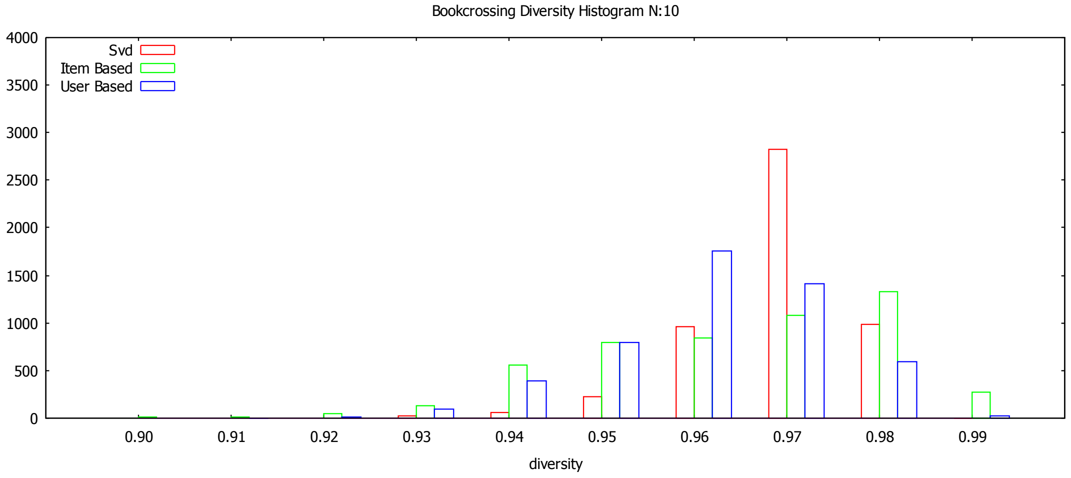


Figure A.23: Bookcrossing Diversity Histogram TH:5

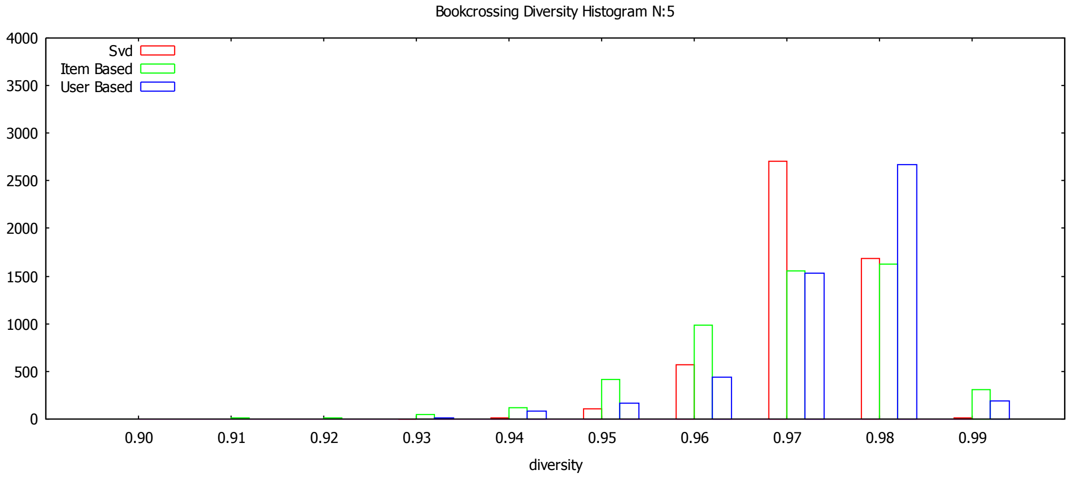


Figure A.24: Bookcrossing Diversity Histogram TH:1

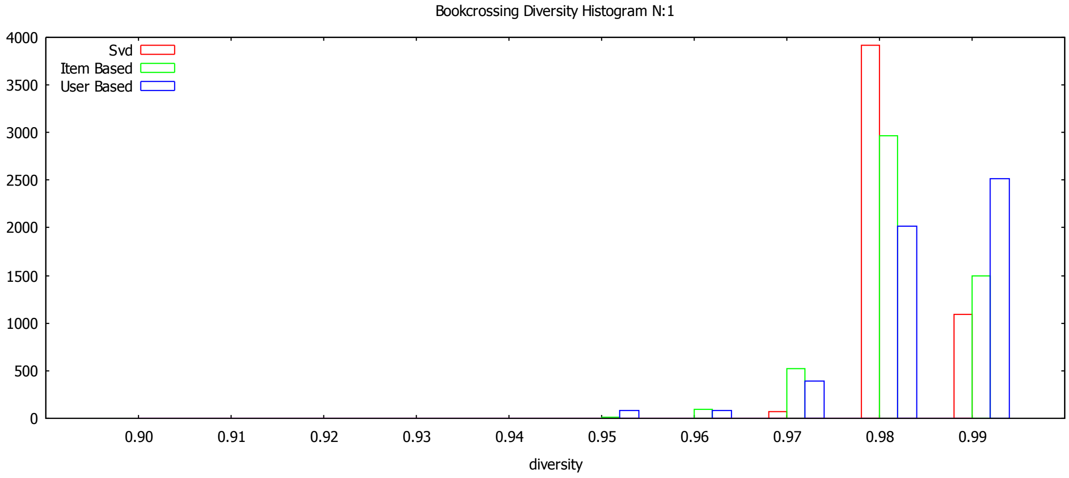


Figure A.25: Movielens Random Recall

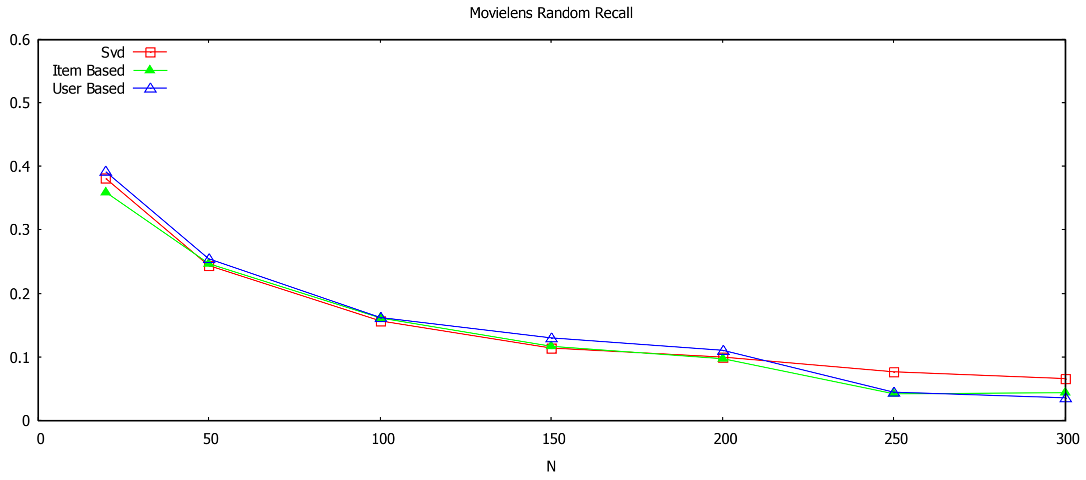


Figure A.26: Movielens Random Diversity

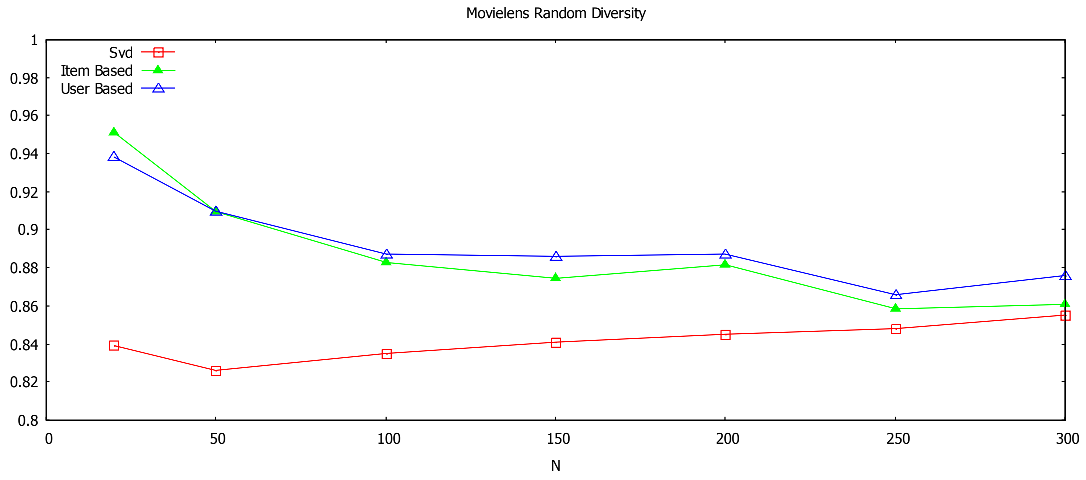


Figure A.27: Movielens Random Novelty

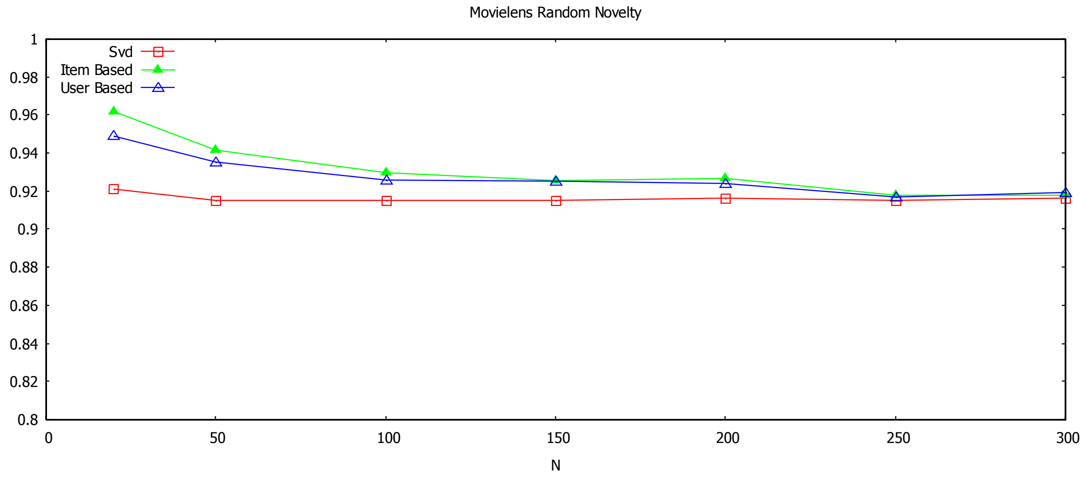


Figure A.28: Jester Random Recall

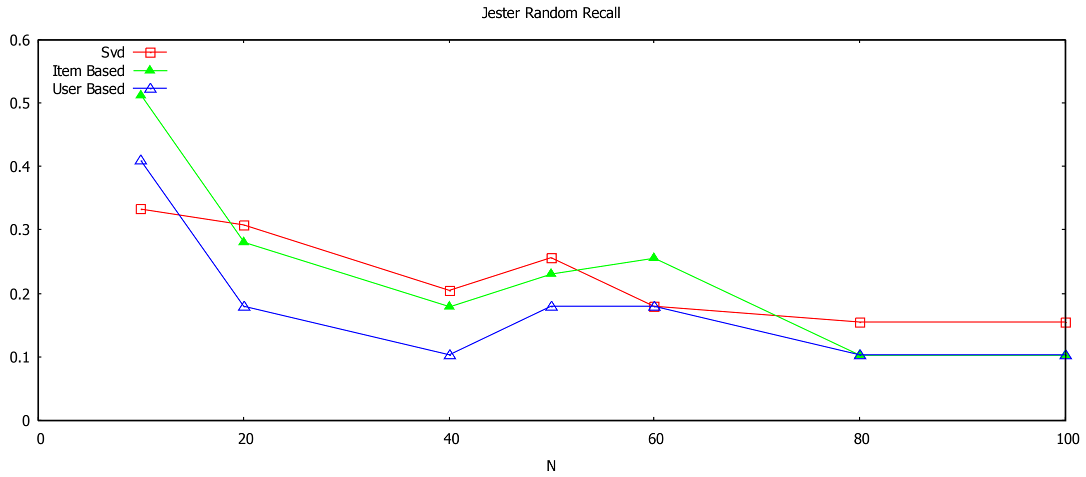


Figure A.29: Jester Random Diversity

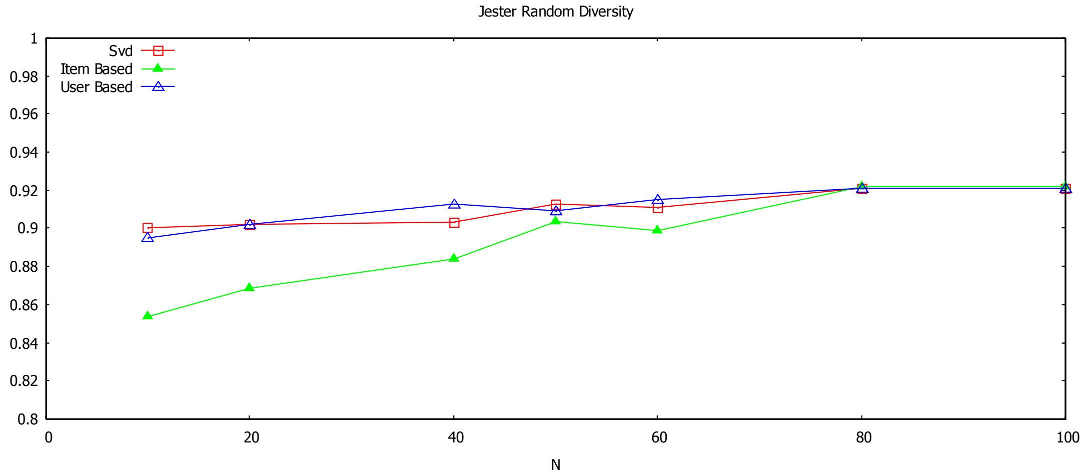


Figure A.30: Jester Random Novelty

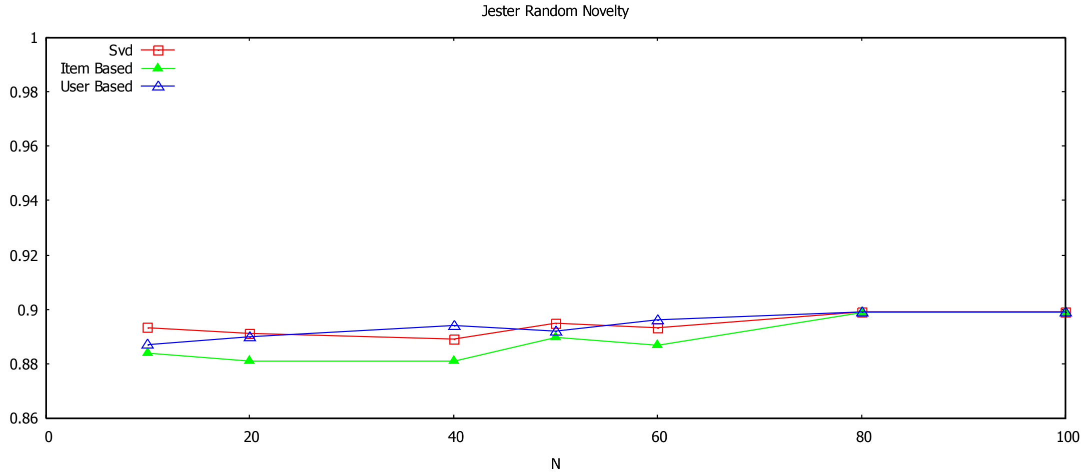


Figure A.31: Bookcrossing Random Recall

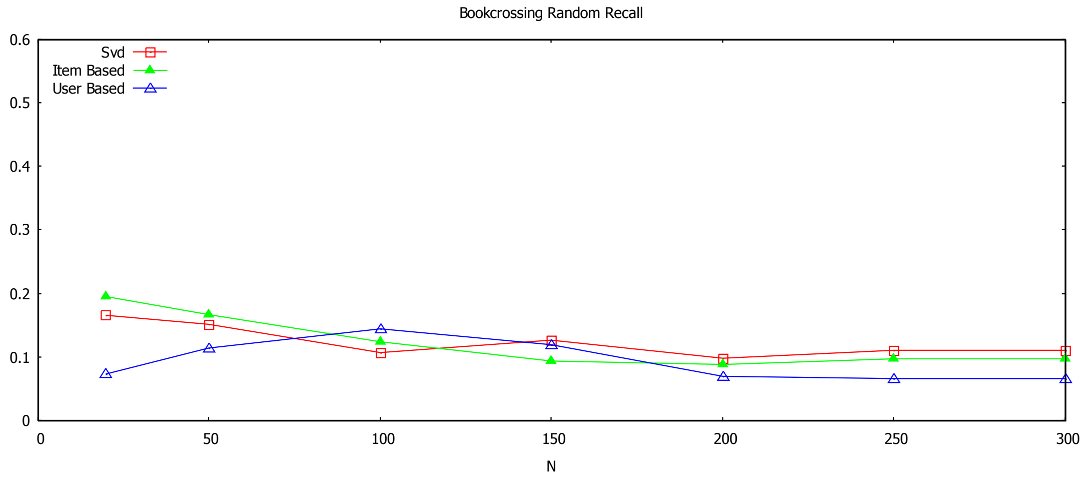


Figure A.32: Bookcrossing Random Diversity

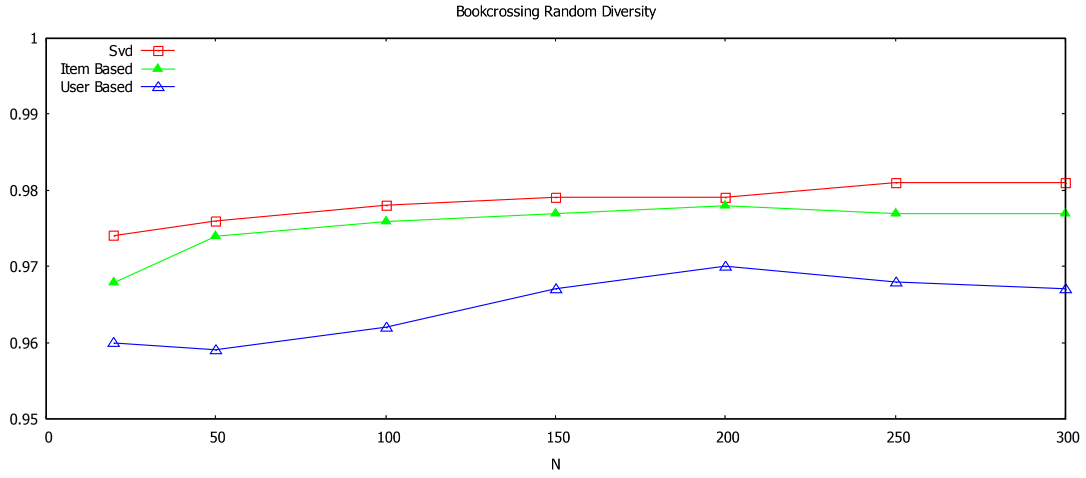
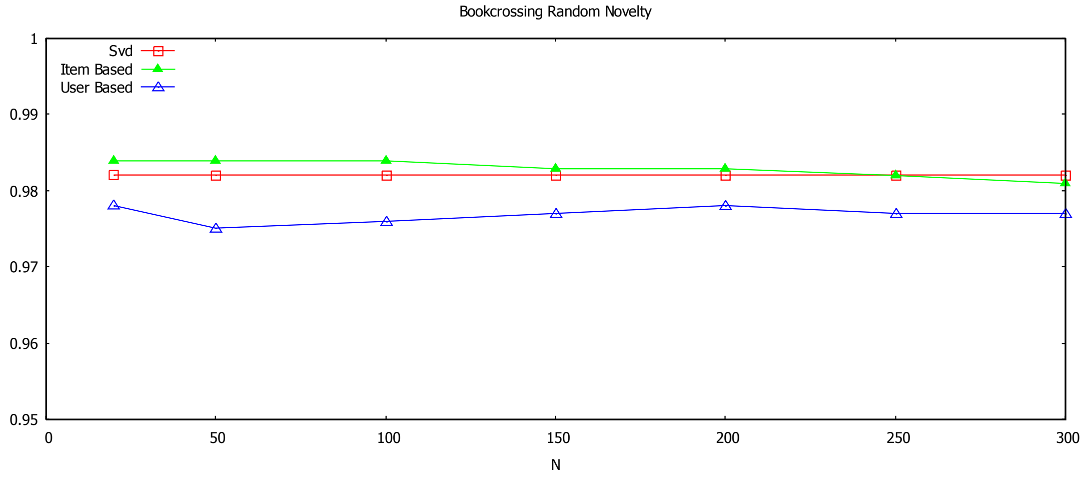


Figure A.33: Bookcrossing Random Novelty



CURRICULUM VITAE

Name & Surname : Mahmut Özge Karakaya

Place and Year of Birth: Afyon 1985

Foreign Language: English

Undergraguate: Ege University Computer Engineering 2008

Working Life : Usta Yazılım – Software Engineer 07.2007- 03.2008
Anadolu Sigorta– Software Engineer 03.2008- Present