AN ONTOLOGY-BASED HYBRID RECOMMENDATION SYSTEM USING SEMANTIC SIMILARITY MEASURE AND FEATURE WEIGHTING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UĞUR CEYLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2011

Approval of the thesis:

AN ONTOLOGY-BASED HYBRID RECOMMENDATION SYSTEM USING SEMANTIC SIMILARITY MEASURE AND FEATURE WEIGHTING

submitted by **UĞUR CEYLAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Dr. Ayşenur Birtürk
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Dept., METU

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU

Assoc. Prof. Dr. Tolga Can
Computer Engineering Dept., METU

Assist. Prof. Dr. Pınar Şenkul
Computer Engineering Dept., METU

Assoc. Prof. Dr. Hasan Oğul
Computer Engineering Dept., Başkent University

Date:

ABSTRACT

AN ONTOLOGY-BASED HYBRID RECOMMENDATION SYSTEM USING SEMANTIC SIMILARITY MEASURE AND FEATURE WEIGHTING

Ceylan, Uğur
M.Sc., Department of Computer Engineering
Supervisor : Dr. Ayşenur Birtürk

September 2011, 81 pages

The task of the recommendation systems is to recommend items that are relevant to the preferences of users. Two main approaches in recommendation systems are collaborative filtering and content-based filtering. Collaborative filtering systems have some major problems such as sparsity, scalability, new item and new user problems. In this thesis, a hybrid recommendation system that is based on content-boosted collaborative filtering approach is proposed in order to overcome sparsity and new item problems of collaborative filtering. The content-based part of the proposed approach exploits semantic similarities between items based on a priori defined ontology-based metadata in movie domain and derived feature-weights from content-based user models. Using the semantic similarities between items and collaborative-based user models, recommendations are generated. The results of the evaluation phase show that the proposed approach improves the quality of recommendations.

Keywords: Recommendation systems, Content-boosted collaborative filtering, Semantic similarity, Ontology, Feature weighting

ÖZ

ANLAMSAL BENZERLİK ÖLÇÜSÜ VE ÖZELLİK AĞIRLIKLANDIRMAYA DAYANAN ONTOLOJİ TABANLI MELEZ BİR TAVSİYE SİSTEMİ

Ceylan, Uğur
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Dr. Ayşenur Birtürk

Eylül 2011, 81 sayfa

Tavsiye sistemlerinin görevi kullanıcıların tercihlerine uygun öğeleri tavsiye etmektir. Tavsiye sistemlerinde iki temel yaklaşım, işbirlikçi filtreleme ve içerik tabanlı filtrelemedir. İşbirlikçi filtrelemenin seyreklik, ölçeklenebilirlik, yeni öğe ve yeni kullanıcı gibi bazı önemli sorunları vardır. İşbirlikçi filtrelemenin seyreklik ve yeni öğe sorunlarının üstesinden gelmek için, içerikle desteklenmiş işbirlikçi filtreleme yaklaşımına dayanan bir melez tavsiye sistemi önerilmektedir. Önerilen yaklaşımın içerik tabanlı bölümü, film alanında önceden tanımlanmış ontoloji tabanlı üstveriye ve içerik tabanlı kullanıcı modellerinden türetilen özellik ağırlıklarına dayanan öğeler arasındaki anlamsal benzerliklerden faydalanmaktadır. Öğeler arasındaki anlamsal benzerlikler ve işbirlikçi tabanlı kullanıcı modelleri kullanılarak tavsiyeler üretilir. Değerlendirme aşamasının sonuçları, önerilen yaklaşımın tavsiyelerin kalitesini arttırdığını göstermektedir.

Anahtar Kelimeler: Tavsiye sistemleri, İçerikle desteklenmiş işbirlikçi filtreleme, Anlamsal benzerlik, Ontoloji, Özellik ağırlıklandırma

To My Parents...

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Dr. Ayşenur Birtürk for her guidance, encouragement and support throughout this thesis work.

I would like to express my special thanks to Prof. Dr. Nihan Kesim Çiçekli, Assoc. Prof. Dr. Tolga Can, Assist. Prof. Dr. Pınar Şenkul and Assoc. Prof. Dr. Hasan Oğul the members of my jury, for reviewing this thesis.

I am greatly indebted to my parents for their love, patience and endless support.

I would like to thank Emre Egüz and İlter Kağan Öcal, the members of the band Bio, for their understanding.

I would like to thank the Scientific and Technological Research Council of Turkey (TUBİTAK) for providing the financial support throughout my M.Sc. study.

Lastly, I would like to thank anonymous reviewers of ACM RecSys'10 Conference, WE-BIST 2011 Conference, and KDD 2011 Workshop on Social Network Mining and Analysis (SNAKDD 2011) for their comments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Accessing information is one of the crucial points of web-based systems. Since the rapid increase in the amount of information over the World Wide Web, information in a specific domain becomes larger and larger. That means, in a particular domain, size of search space for alternatives is huge [1]. Irrelevant information is not necessary to be considered by users because users only want valuable information in that domain. By eliminating information that is irrelevant to users, the search space may become much smaller.

Due to the above reasons, recommendation systems (recommender systems) are widely used in order to overcome information overload problem [2] by generating recommendations. The aim of the recommender systems is to predict the valuable information/items for its users and recommend these items. In literature, ‘recommendation system’ has many definitions. Some of these definitions are the following:

- “Recommender systems are software tools and techniques providing suggestions for items to be of use to a user” [3].
- “Recommender systems are personalized information filtering technology used to either predict whether a particular user will like a particular item or to identify a set of N items that will be of interest to a certain user” [4].

Some popular web-based applications that take advantage of recommendation systems and their domains are as follows: Amazon.com [5] for online shopping, Netflix [6] and MovieLens [7] for movie domain, Last.fm [8], Pandora [9], Grooveshark [10] for music domain.

In this thesis study, we propose a hybrid approach based on content-boosted collaborative filtering presented in [11] to cope with sparsity and new item problems of collaborative filtering. The content-based filtering in our hybrid approach utilizes content-based user models, collaborative-user models and semantic similarity measure on ontology-based metadata in order to generate high-quality recommendations in movie domain. The main contribution of this study is combining semantic similarity measure and content-based user models in recommendation process. The content-based user models are utilized in order to determine the feature-weights that show the preferences of users. Using the feature-weights with the semantic similarity measure, the similarity between items are computed. And then, these similarities and collaborative-based user models are exploited to enhance the rating data.

This thesis consists of 5 chapters. The remaining 4 chapter is organized as follows.

In Chapter 2, the main topics in recommendation systems literature are presented. The recommendation problem is formalized, main recommendation approaches along with the limitations are explained, and most commonly used metrics that evaluate the performance of recommendation systems are presented.

In Chapter 3, the proposed hybrid approach is explained in detail. First, the overview of the proposed approach is presented. After that, the content-based filtering part of the proposed approach, which includes generating ontology-based metadata, the semantic similarity calculation, rating prediction, feature selection and feature weighting, is described. Then, collaborative filtering used in the proposed approach is presented.

In Chapter 4, the evaluation of the proposed approach is explained in detail. The data set and the evaluation metrics used in the evaluation are described. The results of the experiments are presented and discussed.

In Chapter 5, the thesis is concluded and the possible future work to improve the performance of the proposed approach is mentioned.

CHAPTER 2

RELATED WORK

This chapter presents the main topics in the recommendation systems area. First, the recommendation problem is formalized. Then, the main recommendation approaches and their limitations are explained. Finally, some evaluations metrics that are used commonly to evaluate the performance of the recommendation approaches are described.

2.1 Formalization of Recommendation Problem

Formalization of the recommendation problem can be done as follows [2]. U is the set of all users and I is the set of all items in the recommendation system. Movies, songs can be given as an example of such items. Both the space I, which is the set of all items, and U, which is the set of all users, can be very large. In some domains both U and I can be in range of millions. A function ut that measures the quality of being useful of item i to user u is defined as follows.

$$ut : U \times I \rightarrow R \tag{2.1}$$

The function ut is called *utility function*. R is a set which contains non-negative integers or real numbers within a certain range. In order to recommend an item to a user, a recommendation system tries to find such an item that maximizes the utility of the user. Therefore, for each user $u \in U$, the system tries to recommend i'_c which is the following.

$$\forall u \in U,\ i'_c = \underset{i \in I}{argmax}\, ut(u, i) \tag{2.2}$$

In general, rating represents the utility of an item to a user. Rating of an item shows that how much the user, who gave the rating, likes that item. For instance, in a 1 to 5 rating scale, if user1 gives the rating 5 for item1, it means that user1 strongly likes item1.

In recommendation systems, in which the utility function is represented by ratings, it is obvious that some ratings of user-item pairs are not known. The information known about the utility function is the ratings for the items given by the users in the past. If a recommendation system estimates this utility function, in other words predicts ratings for the items that have not been rated by the users, the system can recommend items to its users by considering these predicted ratings. Therefore, it can be said that, given some ratings of the users for the items, the recommendation problem can be reduced to the rating prediction for unknown user-item pairs.

Ratings in a recommendation system can be represented by a matrix which is called *user-item rating matrix*. In a m-by-n user-item rating matrix, m rows represent the users and n columns represent the items. An example of user-item rating matrix is shown in Table 2.1. In Table 2.1, rating scale is 1 to 5 and '**?**' indicates the unknown rating. For instance in Table 2.1, element at the first row and third column of the user-item rating matrix is '**?**' which means that User1 has not rated Item3.

Table 2.1: User-Item Rating Matrix Example

	Item1	**Item2**	**Item3**	**Item4**	**Item5**	**Item6**
User1	2	4	**?**	**?**	1	3
User2	**?**	3	**?**	**?**	1	**?**
User3	**?**	2	2	5	**?**	**?**
User4	3	4	3	**?**	2	**?**
User5	2	**?**	4	4	**?**	2

If '**?**' elements in the user-item rating matrix are predicted, the recommendation system is able to recommend item or items to a user by considering the predicted ratings for the user. For example, unknown ratings in user-item rating matrix, which are shown by '**?**' in Table 2.1, are predicted and the user-item rating matrix that is filled with these predicted ratings is shown in Table 2.2.

After the process of prediction of the unknown ratings, using the Equation 2.2, the system

Table 2.2: User-Item Rating Matrix Filled by Prediction

	Item1	**Item2**	**Item3**	**Item4**	**Item5**	**Item6**
User1	2	4	**2.1**	**4.8**	1	3
User2	**4.2**	3	**3.1**	**4.9**	**2.9**	**4.4**
User3	**3**	2	2	5	**2.3**	**3.2**
User4	3	4	3	**3.8**	2	**1.9**
User5	2	**3.6**	4	4	**3.2**	2

can recommend items to the *active user* (the user whom the prediction is for), based on these predicted ratings. Recommendation system recommends a number of items that have the highest ratings among the predicted ratings of the active user for unrated items. For instance, given the user-item rating matrix in Table 2.1, User2 is the active user. First, the system predicts ratings of the User2 for unrated items (Item1, Item2, Item3, Item4 and Item6) that are given in Table 2.2. Suppose that in our example, the number of the items in recommendation list is 3. Therefore, the recommendation list consists of Item4, Item6, Item1. The other option is that the recommendation system can recommend the items which the active user will like according to the prediction of ratings. Suppose that the threshold rating in our example is 3. In other words, if the prediction is above 3, it means that the user will like corresponding item. In this case, the recommendation list for User2 consists of Item4, Item6, Item1, Item3. In the literature, making a prediction of unknown rating for a user-item pair is called *prediction problem*/*prediction computation* and determining a list of N items which a user will like is called *top-N recommendation problem*/*ranked scoring* [12, 13, 4]. In this thesis, we primarily focus on the prediction problem rather than top-N recommendation problem.

2.2 Recommendation Approaches

The approaches that are used in recommendation systems are mainly divided into three categories which are collaborative recommendation, content-based recommendation and hybrid recommendation approaches [2, 14, 15]. In this section, these recommendation system approaches are explained briefly. Additionally, semantic recommendation approach is also presented.

2.2.1 Collaborative Recommendation

Collaborative filtering (CF) is first mentioned by the developers of the first recommendation system Tapestry [16], which is a mail system that performs filtering by utilizing the users reactions to documents they read, and since then, collaborative filtering has been adopted by other systems [17]. The idea behind collaborative recommendation approach, also known as collaborative filtering, is that the users which have similar interests will like the similar items [18]. Formally, by using the model mentioned in Section 2.1, the utility $ut(u, i)$ of item i for user u is predicted based on utilities $ut(u_j, i)$ where u_j are "similar" to user u [2].

Collaborative filtering algorithms can be grouped into the following two general classes [18]:

1. Memory-based algorithms
2. Model-based algorithms

In order to make predictions for user-item pairs, memory-based algorithms use entire rating values of items given by the users while model-based algorithms use entire rating values to learn a model which is then used for making predictions.

Memory-based algorithms are heuristic algorithms that operate over the whole user-item rating matrix in order to predict unknown ratings. Memory-based algorithms first identifies the group of users who have similar interests with the active user, called neighbors, then use different algorithms to utilize the neighbors' preferences in order to make predictions [19]. This algorithm called user-based CF. Item-based CF uses the similarities between items rather than similarities between users [19]. According to [13], the steps of the neighborhood-based CF algorithm, which is commonly used memory-based algorithm, are the following:

- Similarity Computation
- Prediction Computation

2.2.1.1 Similarity Computation

First step of the neighborhood-based CF is computation of similarity between users/items. These similarities are used for determining the nearest neighbors and making predictions by weighting users/items with respect to these similarity values. In order to calculate similarity between users/items, two most common used approaches are correlation-based and cosine-based approaches [2].

In correlation-based approach, Pearson correlation coefficient [20] or other correlation-based similarities can be used [13]. In user-based CF, Pearson correlation between user u and w is as follows,

$$sim(u,w) = \frac{\sum_{i \in \hat{I}} (r_{u,i} - \overline{r_u})(r_{w,i} - \overline{r_w})}{\sqrt{\sum_{i \in \hat{I}} (r_{u,i} - \overline{r_u})^2 \sum_{i \in \hat{I}} (r_{w,i} - \overline{r_w})^2}} \tag{2.3}$$

where $\hat{I}$ is the set of items that have been rated by both users u and w. Because, in user-based CF, the similarity between two users are computed by considering the ratings of the co-rated items. $r_{u,i}$ is the rating given by the user u for the item i. $\overline{r_u}$ and $\overline{r_w}$ are the average ratings of the co-rated items given by the user u and user w respectively. In item-based CF, Pearson correlation between item i and j is as follows,

$$sim(i,j) = \frac{\sum_{u \in \hat{U}} (r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in \hat{U}} (r_{u,i} - \overline{r_i})^2 \sum_{u \in \hat{U}} (r_{u,j} - \overline{r_j})^2}} \tag{2.4}$$

where $\hat{U}$ is the set of users who rated item i and item j. Similar to the user-based CF, $\overline{r_i}$ and $\overline{r_j}$ are the average ratings given by users in $\hat{U}$ for item i and item j respectively.

Some other correlation-based similarities are constrained Pearson correlation, Spearman rank correlation, and Kendall's correlation [21, 22, 23].

In cosine-based approach [19, 18], the users/items are treated as two vectors and the cosine angle between these vectors is computed [2]. The similarity between vector x and vector y is computed as follows,

$$sim(x,y) = \cos(x,y) = \frac{\sum_{e=1}^{k} x_e y_e}{\sqrt{\sum_{e=1}^{k} x_e^2 \sum_{e=1}^{k} y_e^2}} \tag{2.5}$$

where k is the length of the vectors. In user-based CF, the vectors x and y contain the rating values of the co-rated items. In item-based CF, these vectors contain the rating values of the items given by the users who rated both two items.

2.2.1.2 Prediction Computation

In user-based CF, after computing similarities between active user and other users, a number of nearest neighbors (N) are selected by considering the computed similarities, and then a proper function with the similarities of nearest neighbors and their ratings is used for making a prediction [22, 13].

In [2], three prediction functions are mentioned. These are the following:

1. Average
2. Weighted sum
3. Adjusted weighted sum

In order to predict an unknown rating of user u for item i in user-based CF, the simplest function is the "average" of the ratings of the nearest neighbors given to the item i. Prediction of a rating for an item i given by user u is as follows,

$$pred_{u,i} = \frac{1}{N} \sum_{u' \in U'} r_{u',i} \tag{2.6}$$

where U' is the set of N number of most similar users (nearest neighbors) to user u, and $r_{u',i}$ is the rating given by the user u' for the item i. Similarly in item-based CF an unknown rating can be predicted by taking the average of the ratings of the nearest items given by the user u. Another function that can be used for making prediction is "weighted sum", which is the following.

$$pred_{u,i} = \frac{1}{\sum_{u' \in U'} |sim(u,u')|} \sum_{u' \in U'} sim(u,u') r_{u',i} \tag{2.7}$$

The similarity between an active user and the neighbors are used as weights. Therefore, more similar users have more weights in rating prediction. The above formulation is for user-based

CF, but it can easily be used for item-based CF. Another function, which can be used in user-based CF, is called "adjusted weighted sum". Adjusted weighted sum function is the following,

$$pred_{u,i} = \overline{r}_u + \frac{1}{\sum\limits_{u' \in U'} |sim(u,u')|} \sum_{u' \in U'} sim(u,u')(r_{u',i} - \overline{r}_{u'}) \tag{2.8}$$

where $\overline{r}_u$ is the average of the ratings of user u. The advantage of using adjusted weighted sum is that it takes into account the different rating scales of different users. For instance, for some users, rating value 3 indicates that these users like the rated item, but for some users, rating value 3 indicates that these users do not like the rated item. Therefore, rather than using the rating values in scale of 1 to 5 for all users, rating value minus average rating of the corresponding user is used for making prediction.

As mentioned before, model-based algorithms learn a model by using the rating values, and then this model is used for predicting a rating for an user-item pair [18]. According to [13], some model-based CF techniques are clustering CF models [24, 25], Bayesian belief nets CF models [18, 26, 27], latent semantic CF models [28]. In this thesis, we focus on memory-based CF, therefore model-based algorithms are not explained in detail.

As mentioned in [29, 2], some limitations of collaborative recommendations are the following.

Sparsity: In such domains which have large number of items, users rate only relatively small number of these items. As a result, the user-item rating matrix can be very sparse and this can lead poor predictions/recommendations. For the users who rated a small number of items compared with the large number of items in the system, finding similarities between these users can be difficult or inaccurate. For instance, if there is not any co-rated items for two users in the system, the system can not compute similarity between these users. As a result, quality of recommendations obtained from sparse data can be low. This problem also occurs in the initial phase of a recommender system [11].

New item problem: Another major problem in CF is new item problem. In recommendation systems, it is obvious that new items are added to the system over time. As mentioned before, prediction process in collaborative filtering systems depends on the other users' opinion, in other words their ratings for items. When a new item is added to the system it is clear that the system has no rating data given by the users for this new item. Therefore, CF system can not recommend this item to users until a significant number of users rate this new item. This problem also called first-rater problem [11].

New user problem: Until a new user rates sufficient number of items in a CF system, because of lack of information about the user, the similarities between this user and other users can not be computed accurately. As a result, the predictions/recommendations can be poor. Both new user and new item problems are called “cold start” problems and these problems can be treated as a result of the sparsity problem [13].

2.2.2 Content-Based Recommendation

The idea behind content-based recommendation approach, also known as content-based filtering, is that a user has the same opinion about similar items [22]. Content-based recommendation approaches, also known as content-based filtering (CBF), generate recommendations by comparing representation of content, which defines an item, to representation of content that is liked by the user [30]. By using the model mentioned in Section 2.1, in content-based recommendation, the utility $ut(u,i)$ of item i for user u is predicted based on the utilities $ut(u,i_j)$ where i_j are “similar” to item i [2].

A CBF system generates recommendations based on the similarities between the content of the items (*item profile*) and the preferences of the active user (*user profile*) [31]. A user profile can be constructed using the profiles of the items that have been rated by the corresponding user. After constructing the user profile of the active user, a CBF system can generate recommendations for the active user by considering the similarities between the constructed user profile and the item profiles of the unrated items. This process can be formalized as follows [2]. *ItemProfile(i)* is a set of attributes that represent item i. *ItemProfile(i)* can be constructed by extracting some features from item i. For textual domains in which a CBF system recommends text-based items (or called “documents”), item profile consists of “keywords” and their weights. The terms keyword and document refer different terms in different domains. For instance, in movie domain, document refers a movie item in the CBF system and keyword refers an attribute of the movie item such as actor, genre, runtime etc. Item profile of document d_j is defined as follows,

$$ItemProfile(d_j) = \{w_{1,j}, w_{2,j}, ..., w_{k,j}\} \tag{2.9}$$

where $w_{i,j}$ is the weight of the keyword k_i in document d_j and k is the number of keywords in the system. A weight of a keyword in a document defines the importance of this keyword for corresponding document. The term frequency/inverse document frequency weight [32] is one of the measures for specifying keyword weights [2, 33].

The profile of a user is constructed from profiles of items that have been rated by the corresponding user by using some averaging approaches [2]. User profile is defined as follows,

$$UserProfile(u) = \{w_{1,u}, w_{2,u}, ..., w_{k,u}\} \tag{2.10}$$

where $w_{i,u}$ is the importance of the keyword k_i for user u. After constructing profile of a user, utility of an item to the user is usually computed by using a heuristic, such as cosine similarity [32] between the user profile and item profile vectors which is as follows [2].

$$ut(u,i) = \frac{\sum_{m=1}^{k} w_{m,i} w_{m,u}}{\sqrt{\sum_{m=1}^{k} w_{m,i}^2 \sum_{m=1}^{k} w_{m,u}^2}} \tag{2.11}$$

After finding the utilities of unrated items to the active user, a content-based recommendation system recommend items with the highest utility values. It is obvious that, range of the cosine similarity, which is from 0 to 1, can be mapped into the rating scale, for instance 1 to 5 in order to make a prediction.

Not only heuristics but also other techniques for content-based recommendation approach have also been used in the literature in order to make predictions/generate recommendations [2]. The task of constructing the user profiles can be treated as a classification learning [33]. Items that have been rated by the active user can be used as training data. Categories (classes) of the training data can be rating values, e.g., 5 classes for each rating value in the 1 to 5 rating scale, or can be binary which denotes the corresponding item is liked by the active user or not. After constructing the "model" for the active user, predictions are made or recommendations are generated by classifying unrated items. According to [33, 2, 34, 35], some of the algorithms used for learning the model of a user are naive Bayesian classifiers, decision trees, nearest neighbor methods, linear classifiers, artificial neural networks.

Some problems of content-based recommendation approach are as follows [2, 33].

Limited content analysis: Content-based recommendation approaches are limited by the expressiveness of the features. The features of the items in a content-based recommendation system can be obtained by automatically or assigned manually. Assigning these features to items manually is not practical and analyzing items to extract the features automatically is a difficult task for some non-textual items, such as audio and video [21]. The other problem occurs when the content of the items does not contain enough information to determine whether a user like the corresponding item or not [33]. It is clear that under this circumstance, content-based recommendation generate poor recommendations.

Over-specialization: If a content-based recommendation system can only recommend items that are similar to the profile of the active user, it is impossible that the active user receives recommendation items that are not similar to the previously rated items [14]. For instance, a content-based movie recommendation system may never recommend comedy movies to the active user until the user rates comedy movies.

New user problem: It is the same problem that is mentioned in Section 2.2.1. In order to construct the profile of a user accurately, sufficient number of items has to be rated by the user. If the system can not model the preferences of a user, generated recommendations for the corresponding user can be poor.

2.2.3 Hybrid Recommendation

Hybrid recommendation is referred as a recommendation approach that utilizes multiple recommendation approaches in order to generate recommendations or make predictions [36]. Hybrid recommendation systems are generally implemented by combining collaborative and content-based recommendation approaches to cope with the limitations of these two approaches [37, 2]. Some of the hybrid recommendation approaches are as follows [38].

Weighted: One approach to use both content-based and collaborative recommendation approaches is that the recommendations/predictions of these approaches are combined to produce a single recommendation/prediction. For instance in [39], collaborative and content-based recommenders have initial weights for each user and these weights are used for the prediction of unknown ratings. The weight values of these independent recommenders are initially equal, and adjusted over time by observing the error of the system.

Switching: In this approach, hybrid recommendation system switches between content-based and collaborative recommenders based on a criterion. For instance, a switching hybrid system generates recommendations using different approaches for different users by analyzing the profile of the user using the switching criterion [36].

Mixed: Simply, a mixed hybrid recommendation system presents recommendations by combining recommendations that are generated by different recommenders independently. For instance, a mixed hybrid recommender, which consists of content-based and collaborative recommenders, produces recommendation lists independently, and then presents all recommended item in a single combined list. A major challenge is to determine the way of integrating the rankings of the different recommendations [36].

Feature combination: In this hybrid approach, the features that are obtained by a recommendation approach are injected into the other recommendation approach. For instance, the hybrid approach that is mentioned in [40] uses collaborative features, which are ratings of the users, with the content features to improve the performance of pure collaborative recommendation approach.

Cascade: In cascade hybrid approach, one recommendation approach is utilized in order to refine the other recommendation approaches output that is a recommendation list or predictions. For example, second recommendation approach can be used in order to the determine the ranking of the items that have equal predictions as a result of the first recommendation approach [38].

Feature augmentation: In feature augmentation hybrid approach, the input of the one recommendation approach is the output of the other recommendation approach. "Content-boosted collaborative filtering" [11] can be given as an example of feature augmentation hybrid approach. In this approach, first, a content-based recommender predicts unknown ratings by learning user models in order to complete the missing data in user-item rating matrix. Content-based recommender uses an extended version of a bag-of-words, a vector of bags of words, naive Bayesian text classifier [41]. And then, this enhanced user-item rating matrix is used as an input for collaborative filtering.

Meta-level: In this hybrid approach, the model constructed by a recommender is used as an input for the other recommender. For instance, “collaboration via content” [37] is a meta-level hybrid approach. In collaboration via content, user profiles, which are constructed by content-based recommender, are represented as vectors that contain weights for the terms/keywords. After that, in collaborative filtering part of the hybrid approach, these content-based profiles are used in order to calculate the similarity between users.

The proposed approach in this thesis is based on the content-boosted collaborative filtering which is a feature augmentation hybrid approach. Rather than using a naive Bayesian text classifier [41] to enhance user-item rating matrix, content-based filtering in our approach finds semantic similarities between items by using content-based user models and ontology-based metadata in order to enhance user-item rating matrix by utilizing these semantic similarities.

2.2.4 Semantic Recommendation

In semantic recommendation approach, the recommendation process is generally based on a concept diagram or an ontology describing a knowledge base and uses Semantic Web technologies [42]. An ontology, which consists of concepts in a domain and relations between them, is a form of knowledge representation [43]. Because the semantic recommendation systems are based on a knowledge-base, they can appear in the category of knowledge-based recommender systems [2]. Semantic recommendation systems can be used to limit the sparsity and new item problems of collaborative filtering systems [44].

Some of the semantic recommendation systems in literature are as follows. The study mentioned in[45] presents a multilayered semantic social network model that groups users according to their common preferences in a layered model. Using the correlation of ontology-based user profiles and relations between concepts, concepts are clustered into different concept groups. According to the concept clusters, user profiles are partitioned and therefore similarity between users can be found at different semantic layers. Based on the similarities at different layers, implicit social networks, which can be utilized in collaborative and content-based recommendation systems, are found.

Another study exploits taxonomic knowledge for generating the personalized recommendations [46]. In this approach user profiles are represented as vectors of interest scores assigned to topics taken from taxonomy over product categories. Using these user profiles, similar users to the active user are discovered. In our approach, besides using super-concept and sub-concept relationships between items in taxonomy, we also consider the relations between concepts.

2.3 Evaluation Metrics

In order to evaluate the performance of a recommendation approach, several evaluation metrics are used in the literature. In this section, the evaluation metrics that are commonly used in the literature are examined.

The accuracy of a recommendation approach is simply evaluated by comparing the predicted ratings with the actual ratings. The accuracy metrics are classified into the following two classes [22]:

- Statistical accuracy metrics
- Decision-support accuracy metrics

Statistical accuracy metrics (predictive accuracy metrics [47]) are used to determine how close the predictions which are generated by recommender, to the actual ratings given by users [12]. The most commonly used statistical accuracy metrics are mean absolute error (MAE) and root mean squared error (RMSE). MAE is calculated as follows,

$$MAE = \frac{\sum_{(i,j)\in P} |pred_{i,j} - r_{i,j}|}{|P|} \tag{2.12}$$

where P, which is called test set [48], is the set of predictions made for unknown user-item pairs, $pred_{i,j}$ and $r_{i,j}$ are the predicted rating and actual rating for the user i on item j, respectively. The difference between MAE and RMSE is that RMSE penalizes large errors more than MAE does [48]. RMSE is calculated as follows.

$$RMSE = \sqrt{\frac{\sum_{(i,j)\in P} (pred_{i,j} - r_{i,j})^2}{|P|}} \tag{2.13}$$

Decision-support accuracy metrics are used for evaluating the performance of a recommendation approach from the aspect of distinguishing the high-quality items rather than error on numerical predictions [12]. Suppose that a recommendation system recommends items that have prediction of more than 3 to its active user. For this recommendation system, the difference between predictions of 2 and 3, 4 and 5 is not important. Because, for both prediction of 2 and 3 the item is not recommended, and for both prediction 4 and 5 item is recommended. For evaluation of such recommendation systems, decision-support accuracy metrics is more suitable than the statistical accuracy metrics [22, 49]. In order to evaluate the performance of a recommendation approach using decision-support accuracy metrics, the rating scale has to be transformed into a binary scale which indicates an item is “relevant” or “irrelevant” [47]. For instance, according to [47], in order to transform MovieLens data set [50], which has a rating scale 1 to 5, commonly used process is that the ratings 1, 2, and 3 are converted to irrelevant,and the ratings 4 and 5 are converted to relevant. The confusion matrix, which indicates the classification of items in recommendation process, is shown in Table 2.3.

Table 2.3: Confusion Matrix for Decision-Support Accuracy Metrics

		Predicted	
		Relevant	Irrelevant
Actual	Relevant	tp	fn
	Irrelevant	fp	tn

In Table 2.3, the rows show that whether the items are relevant or irrelevant actually, and the columns indicate whether items are predicted as relevant or irrelevant by the recommendation process. Some of the decision-support accuracy metrics are precision, recall and F-Measure. By using this confusion matrix, precision and recall are defined as follows.

$$Precision = \frac{tp}{tp + fp} \tag{2.14}$$

$$Recall = \frac{tp}{tp + fn} \tag{2.15}$$

As it can be seen from the Equation 2.14, precision is the ratio of relevant items predicted as relevant to the number of items predicted as relevant, and recall is the ratio of relevant items predicted as relevant to the number of actual relevant items. In other words, precision is the probability that a selected item is relevant and recall is the probability that a relevant item will be selected [47]. According to the [49], high precision is important if the task is "recommending some good items", and high recall is important if the task is "recommending all good items". In order to evaluate the recommendation system by using both precision and recall metrics, several approaches combine these two metrics. F-Measure (also known as F1 [12, 47]) is one of these approaches and defined as follows.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.16}$$

As it can be seen from Equation 2.16, F-Measure metric gives equal weights to precision and recall. If the two metrics, precision and recall, are 1, then the F-Measure metric is 1, which is the best value.

CHAPTER 3

SEMCBCF:CONTENT-BOOSTED COLLABORATIVE FILTERING USING SEMANTIC SIMILARITY MEASURE

This chapter presents the proposed hybrid recommendation system approach, *SEMCBCF*, for movie domain. First, a brief overview of the system is given. Then, the content-based filtering of the *SEMCBCF* is presented. After that, some refinements that were applied on the content-based filtering part of the *SEMCBCF* are explained. Finally, the collaborative filtering approach used in *SEMCBCF* is introduced.

3.1 System Overview

In order to reduce the sparsity and new item problems of collaborative filtering, we have developed a hybrid approach, content-boosted collaborative filtering using semantic similarity measure (*SEMCBCF*), that utilizes semantic similarities between items by using ontology-based metadata in the movie domain. The flow diagram of our approach is shown in Figure 3.1.

Our system, first generates ontology-based metadata by using a predefined ontology and the content information of the movies in the system. After generating ontology-based metadata, *SEMCBCF*, which is based on the hybrid methodology in [11], first applies content-based filtering to user-item rating matrix by utilizing the similarities between items in order to predict unknown ratings. Afterwards, *SEMCBCF* applies collaborative filtering to enhanced user-item rating matrix in order to generate recommendations for an active user. Therefore, our hybrid approach consists of four main phases:

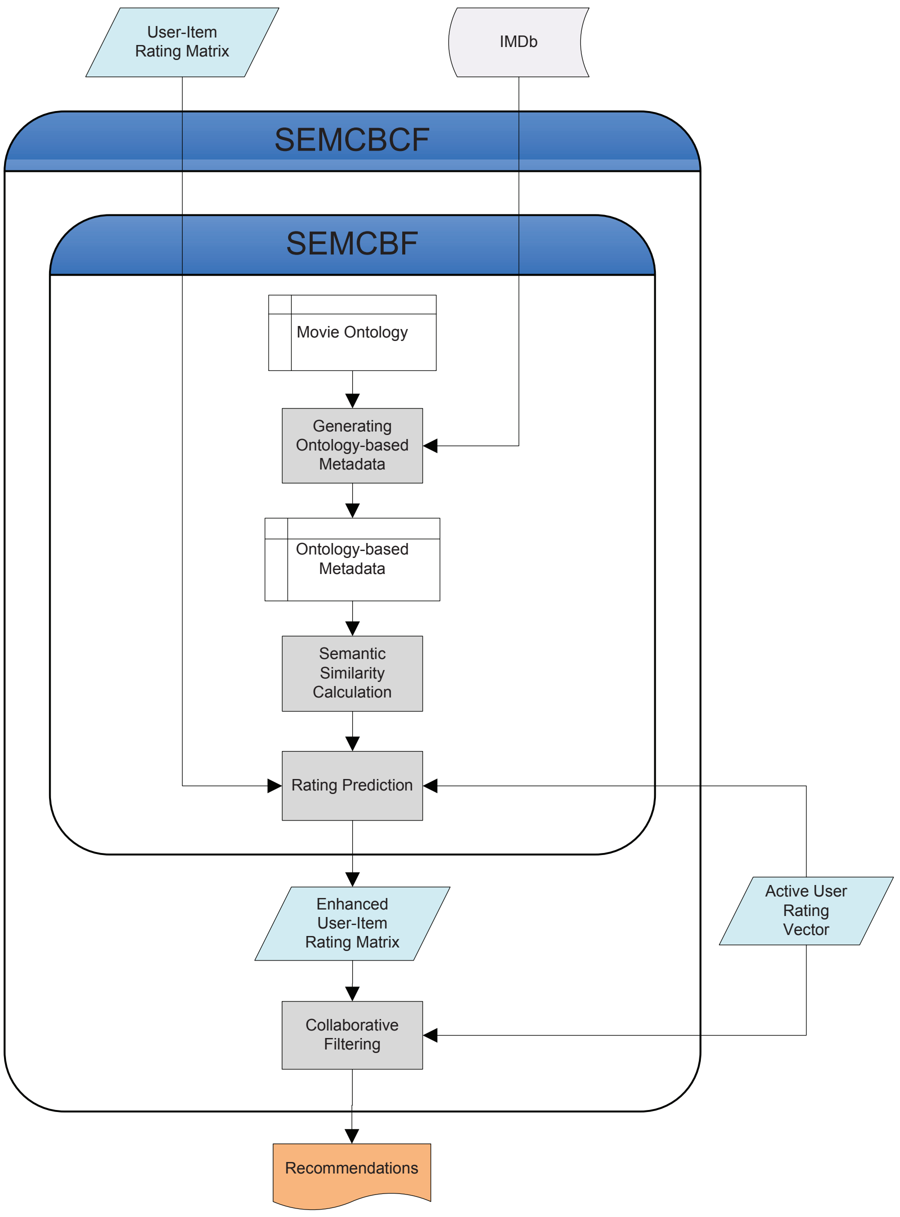


Figure 3.1: Flow Diagram of *SEMCBCF*

1. Generating ontology-based metadata
2. Semantic similarity calculation
3. Rating prediction in order to enhance user-item rating matrix
4. Using collaborative filtering on enhanced user-item rating matrix

As it can be seen form the Figure 3.1, first three phases of *SEMCBCF* constitutes the content-based filtering using semantic similarity measure, named as *SEMCBF*. In Section 3.2, the *SEMCBF* is explained in details.

3.2 SEMCBF:Content-Based Filtering Using Semantic Similarity Measure

Content-based filtering systems recommend items, which are highly similar to the profiles of the users, by considering the content of the items. In *SEMCBF* collaborative-based user model is used for representing user profiles. In this model a user's preferences is represented by a vector that consists of user's ratings for the items in the system. Beside collaborative-based user models, *SEMCBF* utilizes similarities between items in movie domain by using ontology-based metadata in order to predict unknown ratings for user-item pairs. In the following subsections, information about ontology-based metadata, semantic similarity measure to calculate similarities between items and prediction of the ratings is given respectively.

3.2.1 Generating Ontology-Based Metadata

3.2.1.1 Extracting Content of Movies

Each movie item in the system is represented by a unique key which is the url of the movie in IMDb (The Internet Movie Database) [51]. By using these keys, for each movie item, the system gathers content information from IMDb by the implemented web crawler. These content information are kept in the database of the system in order to generate ontology-based metadata.

Content of a movie is organized as follows: the extracted information of a movie consist of a set of *feature-values* and each feature-value belongs to a *feature*. Thus, for each movie, the web crawler parses the IMDb web page of that movie according to the predefined features. In *SEMCBCF*, 10 features of the movie domain are used. These are *cast*, *director*, *writer*, *language*, *genre*, *runtime*, *release date*, *country*, *color* and *IMDb rating*. For some of these features, a movie can have more than one feature-value. For example, if a movie has 2 writers, the number of the feature-values that belongs to *writer* feature is 2. But for some other features, a movie can have at most one feature-value. *Runtime* feature can be given as an example of such features. And also for the features that a movie can have more than one feature-value, there is an upper limit for the number of these feature-values. For example, for the feature *cast*, the upper limit is 3. That means, if a movie has more than 3 actors and/or actresses, the web crawler only gathers the first 3 actors and/or actresses of that movie.

Two features, whose the feature-values are adjusted, are *IMDb rating* and *runtime* features. The adjustment process is that the average IMDb rating of a movie is rounded the nearest integer value for the feature *IMDb rating*. For the feature *runtime*, the digit in the ones place in the feature-value is subtracted from itself. The properties of the features is shown in Table 3.1.

Table 3.1: Properties of the Features

Feature	Description	Single Value	Upper Limit for Number of Feature-Values	Adjustment Rule
Cast	Actors and/or actresses starred in the movie	No	3	-
Director	Directors of the movie	No	2	-
Writer	Writers of the movie	No	2	-
Language	Languages spoken in the movie	No	-	-
Genre	Genres of the movie	No	-	-
Runtime	Runtime of the movie in minutes	Yes	-	$value - (value \bmod 10)$
Release Date	Year of the release date of the movie	Yes	-	-
Country	Countries in which the movie produced	No	-	-
Color	Color technology used in the movie	No	-	-
IMDb Rating	Average rating of the movie given by the IMDb users	Yes	-	$\lfloor value + 0.5 \rfloor$

3.2.1.2 Ontology and Ontology-Based Metadata Models

Based on the studies in [52] and [53], our approach finds semantic similarities between items, which are movies in movie recommendation domain, by using the contents of items and their relations in the system. Therefore, in order to find semantic similarities between items, ontology model and ontology-based metadata model have to be defined.

Ontology model represents the concepts, attributes and relationships between them in a domain. Ontology model is defined as follows [52]:

$$O := \{\tilde{C}, \tilde{P}, \tilde{A}, H^c, prop, att\} \tag{3.1}$$

For ontology model, $\tilde{C}$, $\tilde{P}$ and $\tilde{A}$ are sets which consist of concepts,relation and attributes identifiers respectively. H^c is called concept hierarchy which defines the hierarchical relations between concepts. $prop$ is the function that defines non-hierarchical relations between concepts. Additionally, *att* is the function that defines non-hierarchical relations between literal values and concepts.

Ontology-based metadata model represents the instances, literals and relationships between them in a domain. Ontology-based metadata model is defined as follows [52]:

$$MD := \{O, \tilde{I}, \tilde{L}, inst, instl, instr\} \tag{3.2}$$

For metadata model, $\tilde{I}$ and $\tilde{L}$ are sets which consist of instances and literal values respectively. O is the ontology that the metadata relies on. *inst*, *instr*, *instl* are the functions that define concept instantiation, relation instantiation and attribute instantiation. Predicates and the meanings of them that is used in the rest of the thesis are shown in Table 3.2.

As an example, an ontology and ontology-based metadata for the movie domain are shown in Figure 3.2. *Men in Black* is the only movie that exists in the metadata in Figure 3.2. The feature-values that exist in the metadata belong to features *genre*, *release date*, *cast* and *director*. In the Table 3.3 some predicates of the ontology and ontology-based metadata in Figure 3.2 are shown.

Table 3.2: Predicates and Meanings

Predicate	**Meaning**
$H^c(C_1, C_2)$	Concept C_1 is a sub-concept of concept C_2
$P(C_1, C_2)$	P is a relation with domain C_1 and range C_2
$A(C_1)$	A is an attribute of C_1
$C(I)$	I is an instance of concept C
$P(I_1, I_2)$	Instance I_1 has a P relation to instance I_2
$A(I_1, L)$	Instance I_1 has an A attribute with literal value of L

Table 3.3: Ontology and Ontology-Based Metadata Predicates of the Example

Ontology Predicates	$H^c(Comedy, Movie)$ $H^c(Action, Movie)$ $H^c(Romance, Movie)$ $H^c(Cast, Person)$ $H^c(Director, Person)$ $H^c(Person, Feature)$ $hasCast(Movie, Cast)$ $hasDirector(Movie, Director)$ $ReleaseDate(Movie)$

Ontology-Based Metadata Predicates	*Action(Men In Black)*
	Comedy(Men In Black)
	Cast(Will Smith)
	Director(Barry Sonnenfeld)
	hasCast(Men In Black,Will Smith)
	hasDirector(Men In Black,Barry Sonnenfeld)
	ReleaseDate(Men In Black,”1997”)

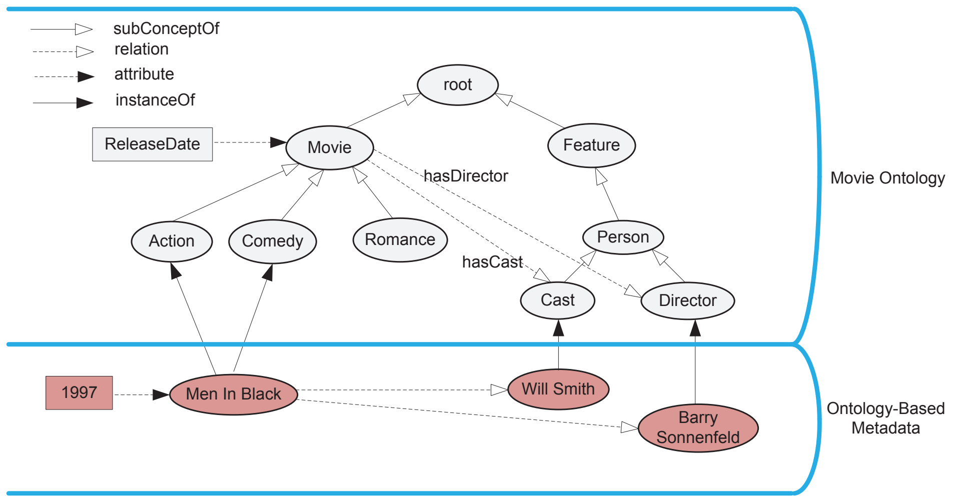


Figure 3.2: An Example of Ontology and Ontology-Based Metadata

3.2.1.3 Characteristics of Ontologies

As mentioned in Section 3.2.1.1, content of a movie is represented as a number of feature-values for 10 features. In order to represent movies, features and feature-values in the system, different ontologies can be used. We generate 5 different movie ontologies manually by using a free, open source ontology editor and knowledge-base framework called Protégé [54]. After generating movie ontologies manually, the metadata is generated based on the defined movie ontology and the content information of movies extracted from IMDb [51].

In all of the 5 movie ontologies, the following characteristics are the same:

1. *Movie* and *Feature* concepts are sub-concepts of *Root* concept.
2. The feature-values of *genre* feature are represented by concepts.
3. The concepts that represent the feature-values of *genre* feature are linked to *Movie* concept by $subConceptOf$ links.
4. *Person* concept is sub-concept of *Feature* concept.
5. *Cast*, *Director* and *Writer* concepts are sub-concepts of *Person* concept.
6. The relations are:
   - *hasCast(Movie,Cast)*

- *hasDirector(Movie,Director)*
- *hasWriter(Movie,Writer)*

The features *cast*, *director* and *writer* are represented by relations in all of the movie ontologies. Other features, which are *language*, *runtime*, *release date*, *country*, *color*, *IMDb rating*, can be represented by relations or attributes in the ontology. The feature-values, whose features are represented by relations, must be represented by instances in ontology-based metadata. And also, for the features that are represented by relations, a proper concept or concepts has to be defined in the ontology. For example, *cast* feature is represented by *hasCast* relation. For that relation, *Cast* concept is defined. Additionally, the feature-values, whose features are represented by attributes, must be represented by literal values in ontology-based metadata. Briefly, the 5 movie ontologies differ from each other by the following characteristics:

- Concept hierarchy
- Relations that represent features
- Attributes that represent features

The last two characteristics of these movie ontologies are shown in Table 3.4. In the rest of the thesis, when we say ontology, unless explicitly expressed, we mean ontology and ontology-based metadata models.

Movie Ontology 1 The basic ontology used in our approach is Movie Ontology 1. A part of the Movie Ontology 1 is shown in Figure 3.3. It can be seen from the Figure 3.3 that all feature-values of *genre* feature are sub-concept of *Movie* concept. For example *comedy*, which is feature-value of *genre* feature, is a sub-concept of concept *Movie*. In order to represent a movie, an instance is inserted into the ontology. The crucial point is that this movie instance is instance of the concepts which are the movies feature-values that belong to the *genre* feature. The movie *Men In Black* in the Movie Ontology 1 is represented by an instance which is instance of the concepts *Action* and *Comedy*. For each remaining feature, a concept and a relation exist in the ontology. For example, in order to represent *language* feature, *Language* concept and *hasLanguage* relation exist in the ontology. The feature-values of the features

Table 3.4: Characteristics of Ontologies from the point of Feature-Value Representation

Ontology	Features whose Feature-Values are Instances	Features whose Feature-Values are Literal Values	Features whose Feature-Values are Concepts
Movie Ontology 1 Movie Ontology 2	cast,director, writer, language, runtime, release, date, country, color, average rating		genre
Movie Ontology 3 Movie Ontology 4 Movie Ontology 5	cast,director, writer, language, country, color	runtime, release date, IMDb rating	genre

other than *genre*, are instances of their corresponding concepts that represents the features. *English* feature-value of *language* feature can be given as an example. At last, the relations between the movie instance and corresponding feature-value instances are set.

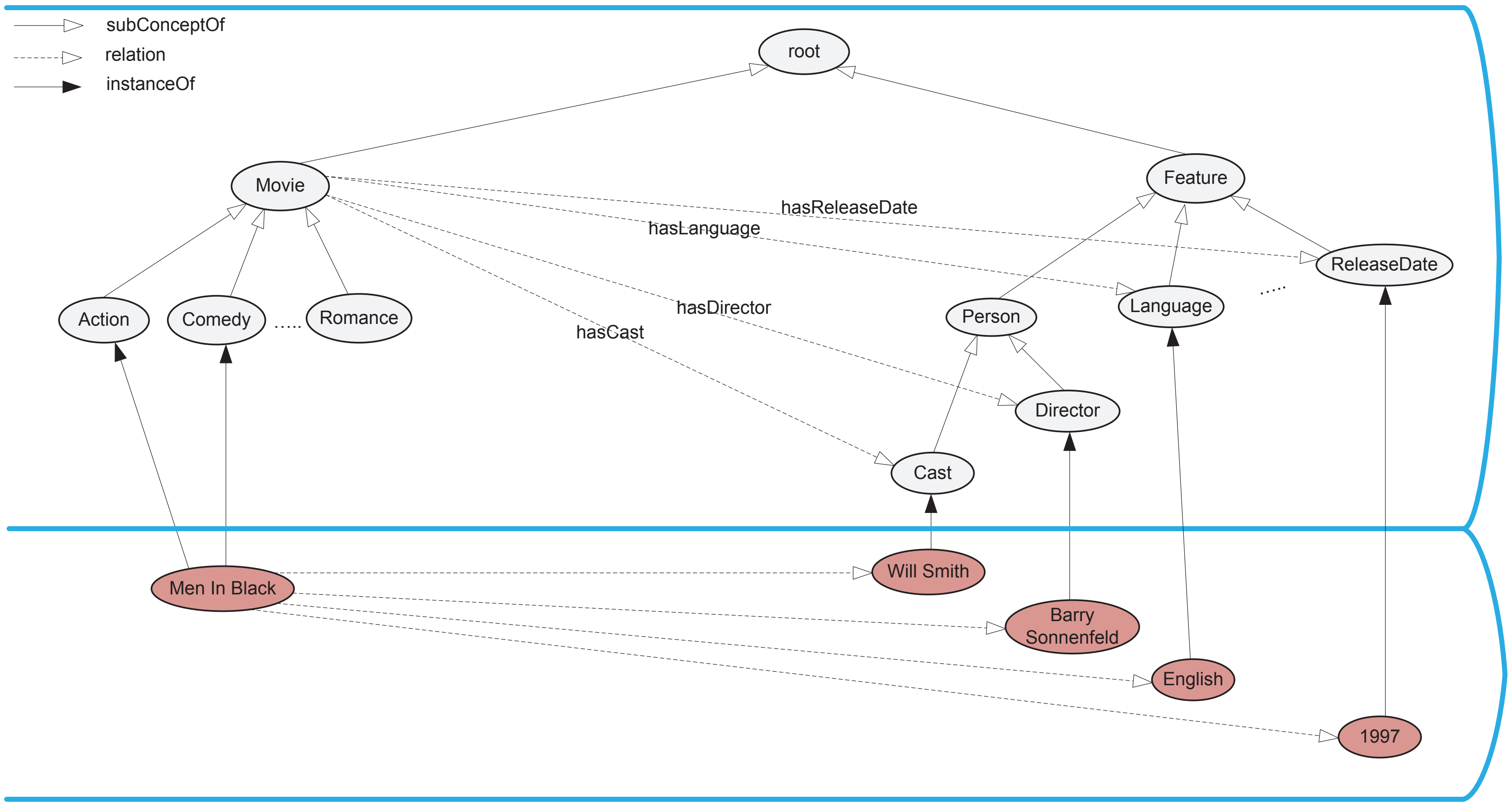
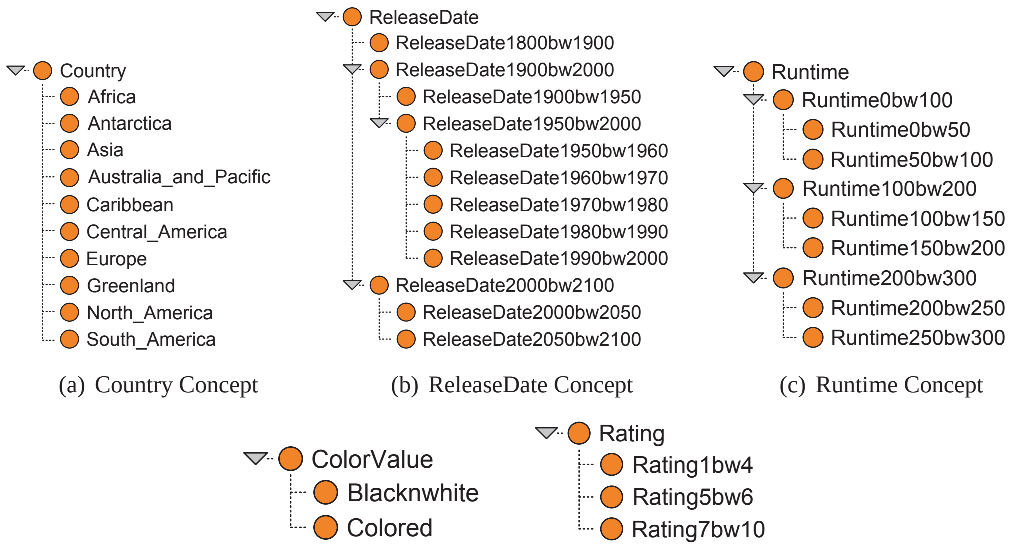


Figure 3.3: A Part of Movie Ontology 1



(a) Country Concept (b) ReleaseDate Concept (c) Runtime Concept

(d) ColorValue Concept (e) Rating Concept

Figure 3.4: The Hierarchy of Concepts defined for Country, Color, IMDb Rating, Release Date and Runtime Features in Movie Ontology 2

Movie Ontology 2 Another ontology used in our approach is Movie Ontology 2. Movie Ontology 2 is similar to Movie Ontology 1. The difference between these two ontologies is the hierarchy of the concepts that are defined for *country*, *color*, *IMDb rating*, *release date* and *runtime* features. The hierarchy of these concepts is shown in Figure 3.4. These concepts, which are linked to *feature* concept by *subConceptOf* links, has more sub-concepts in Movie Ontology 2 than it has in Movie Ontology 1. For example concept that represents *runtime* feature has a number of sub-concepts that defines the runtime intervals. Similarly it can be seen from the Figure 3.4 that *Rating* concept has three sub-concepts that defines the intervals of the average rating of the IMDb users.

Movie Ontology 3 Another ontology used in our approach is Movie Ontology 3. Only difference between Movie Ontology 2 and Movie Ontology 3 is that *IMDb rating*, *release date* and *runtime* features are represented by the attributes in Movie Ontology 3. As it can be seen from Figure 3.5, unlike in Movie Ontology 2, *IMDb rating* feature is represented by attribute *Rating* rather than by a relation. Thus, in Movie Ontology 3, a feature-value of a feature, other than *genre* feature, can be an instance or a literal depending on the representation of the feature. For example, *men in black* has a feature-value *1997* that belongs to the *release* *date* feature. As a result of representing *release date* feature by an attribute, after inserting the instance that represents the *Men In Black* into the Movie Ontology 3, the attribute *ReleaseDate* is set to the literal value *1997* for the movie *Men In Black*.

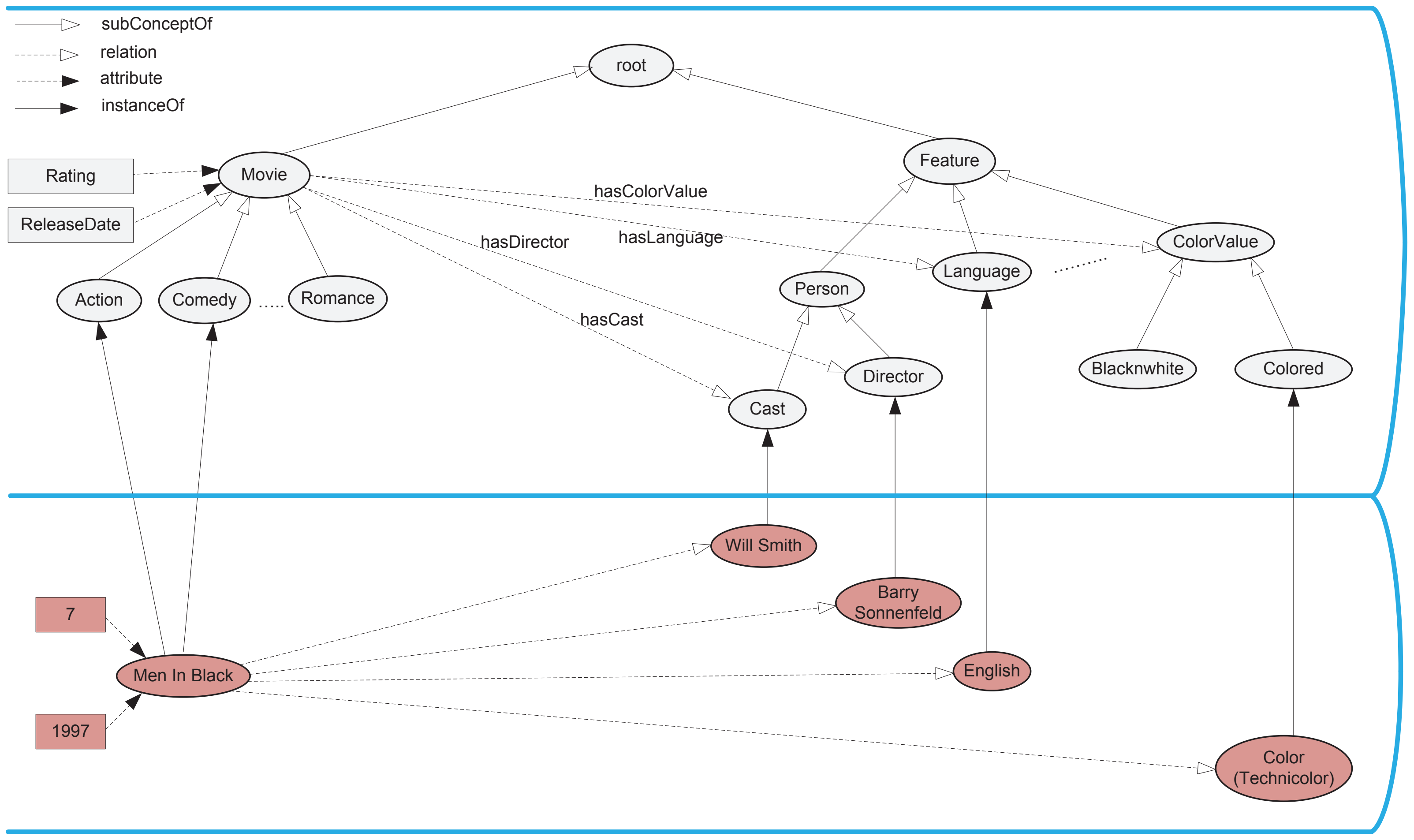


Figure 3.5: A Part of Movie Ontology 3

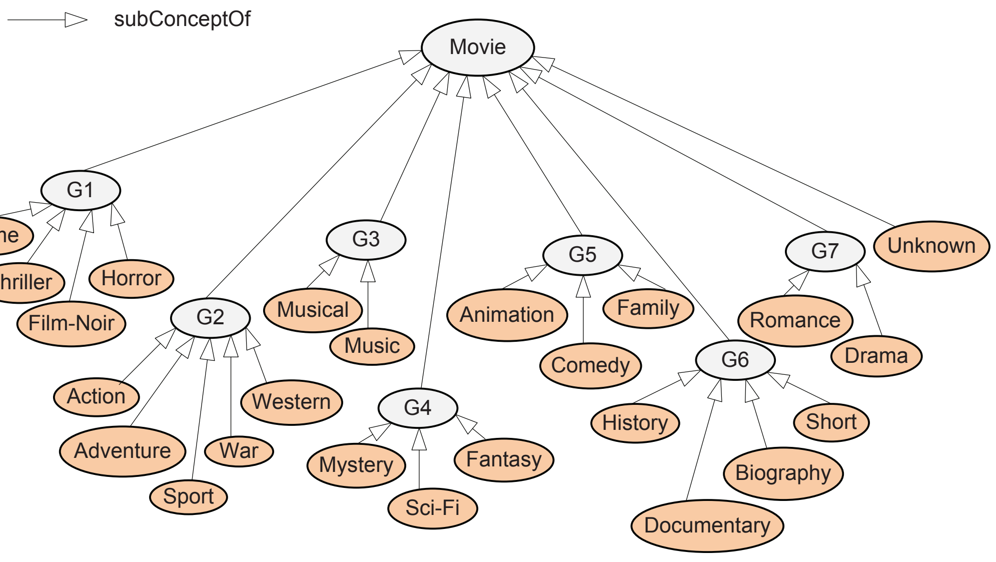


Figure 3.6: The Hierarchy of Descendants of *Movie* Concept in Movie Ontology 4

Movie Ontology 4 Movie Ontology 4 is another ontology used in our approach. It is very similar to Movie Ontology 3. The difference them is the hierarchy of the concepts that represents the feature-values of *genre* feature. In order to identify the effect of the hierarchy of the concepts that represent the feature-values of *genre* feature, the feature-values of *genre* feature are grouped into six sets by common sense. In Movie Ontology 4, each of these sets is represented by a concept that is sub-concept of *Movie* concept. Additionally, each feature-value of *genre* feature is sub-concept of its corresponding concept. The hierarchy of descendants of *Movie* concept is shown in the Figure 3.6.

Movie Ontology 5 Another ontology used in our approach is Movie Ontology 5. The only difference between Movie Ontology 3 and Movie Ontology 5 is the hierarchy of the descendants of *Movie* concept. Considering the feature-values of *genre* feature, the hierarchy of the descendants of *Movie* concept is created by the help of a group of students studying Radio-Television and Cinema. The other characteristics of Movie Ontology 5 are the same as Movie Ontology 3's. The hierarchy of descendants of *Movie* concept in Movie Ontology 5 is shown in the Figure 3.7.

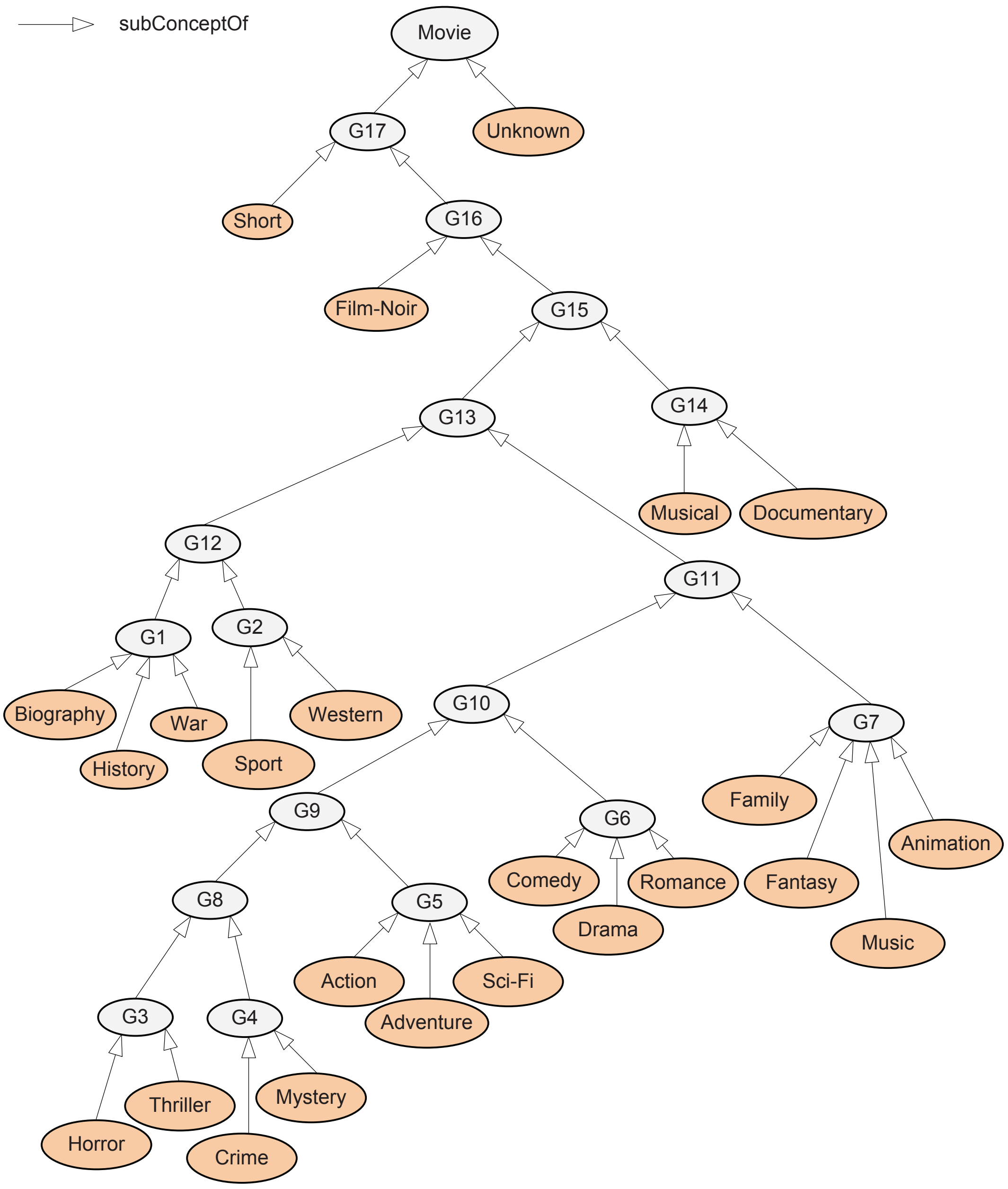


Figure 3.7: The Hierarchy of Descendants of *Movie* Concept in Movie Ontology 5

3.2.2 Semantic Similarity Calculation

In $SEMCBF$, after generating ontology-based metadata by using a predefined ontology and content of the movie items, three types of similarity measures [52] are used in order to calculate similarities between items which are instances in ontology-based metadata:

- Taxonomy similarity (TS)
- Relation similarity (RS)
- Attribute similarity (AS)

For each pair of item in the system *SEMCBF* computes a semantic similarity using the above similarity measures and weight values of these measures. $SS(I_i, I_j)$, semantic similarity between instance I_i and instance I_j, is calculated by the weighted arithmetic mean of TS, RS and AS.

$$SS(I_1, I_2) = \frac{a \times TS(I_1, I_2) + b \times RS(I_1, I_2) + c \times AS(I_1, I_2)}{a + b + c} \tag{3.3}$$

3.2.2.1 Taxonomy Similarity

Taxonomy similarity between two instances (TS) is based on their corresponding concepts' positions in concept hierarchy (H^c) which is defined in ontology model. Basically, the idea behind taxonomy similarity is that closer concepts in taxonomy are more similar. For instance, MovieA is a biography, MovieB is a war and MovieC is a comedy movie. In Movie Ontology 5, it is expected that taxonomy similarity between MovieA and MovieB is higher than the taxonomy similarity between MovieA and MovieC.

An instance can be instance-of more than one concept in ontology. For example in Figure 3.2, *Men In Black* is instance-of both *Action* and *Comedy* concepts. In order to find taxonomy similarity between two instances, the corresponding concepts of these instances have to be considered. Therefore, at the beginning, taxonomy similarity between two concepts (TSC) have to be defined. After finding similarities between concepts in ontology, similarity between two instances can be found by considering the similarities between corresponding concepts of these instances.

In order to calculate the taxonomy similarity between two concepts, the following 4 different measures can be used:

- TSC_{CM}
- $TSC_{Wu\&Palmer}$
- TSC_{Lin}
- TSC_{Mclean}

The first method to calculate TSC is taxonomy similarity between concepts using concept match (TSC_{CM}) which is mentioned in [52]. TSC_{CM} is based on the distance between two concepts in ontology. Concept match between two concepts (CM), which is used for TSC_{CM}, is defined as follows,

$$CM(C_i, C_j) = \frac{|UC(C_i, H^c) \cap UC(C_j, H^c)|}{|UC(C_i, H^c) \cup UC(C_j, H^c)|} \tag{3.4}$$

where UC (upwards cotopy) is the following.

$$UC(C_i, H^c) = \{C_j \in \tilde{C} | H^c(C_i, C_j) \vee C_i = C_j\} \tag{3.5}$$

UC defines the set of concepts that form the path from a given concept to the root of a given concept hierarchy. Then, TSC_{CM} is defined as follows.

$$TSC_{CM}(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j \\ \dfrac{CM(C_i, C_j)}{2}, & \text{otherwise} \end{cases} \tag{3.6}$$

Suppose that, we try to find the similarity between *Horror* and *Action* concepts using TSC_{CM} in Movie Ontology 5. $TSC_{CM}(Horror, Action)$ can be computed using the upwards cotopy of these concepts.

$$\begin{aligned} UC(Horror, H^c) &= \{Horror, G3, G8, G9, G10, G11, G13, G15, G16, G17, Movie, root\} \\ UC(Action, H^c) &= \{Action, G5, G9, G10, G11, G13, G15, G16, G17, Movie, root\} \end{aligned}$$

After finding the upwards cotopies of *Horror* and *Action* concepts, the concept match between these concepts is computed by dividing the cardinality of intersection of the corresponding upwards cotopies by the cardinality of union of the corresponding upwards cotopies. Using the concept match between these concepts, $TCS_{CM}(Horror, Action)$ can be computed.

$$\begin{aligned} TSC_{CM}(Horror, Action) &= \frac{CM(Horror, Action)}{2} \\ &= \frac{9}{14} \cdot \frac{1}{2} \\ &= 0.321 \end{aligned}$$

The second measure used for calculating TSC is a measure proposed by Wu and Palmer in [55]. Taxonomy similarity between concepts using Wu and Palmer's measure ($TSC_{Wu\&Palmer}$) is the following.

$$TSC_{Wu\&Palmer}(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j \\ \dfrac{2 \cdot N_3}{N_1 + N_2 + 2 \cdot N_3}, & \text{otherwise} \end{cases} \quad (3.7)$$

N_1 and N_2 are the number of $subConceptOf$ links from C_i and C_j to their most specific concept C_k that subsumes both of them. Additionally, N_3 is the number of $subConceptOf$ links from C_k to the root of the ontology(*root* concept). Similar to TSC_{CM}, $TSC_{Wu\&Palmer}$ based on the distance between concepts in ontology.

If the previous example, where the concepts are *Horror* and *Action*, is considered, the most specific concept that subsumes both *Horror* and *Action* is *G9* concept in Movie Ontology 5. N_1 is the *subConceptOf* links from *Horror* concept to *G9* concept which is 3 and N_2 is the *subConceptOf* links from *Action* concept to *G9* concept which is 2. And N_3 is the *subConceptOf* links from *G9* concept to *root* concept which is 8. Based on the values of N_1, N_2 and N_3, $TSC_{Wu\&Palmer}(Horror, Action)$ can be computed.

$$\begin{aligned} TSC_{Wu\&Palmer}(Horror, Action) &= \frac{2 \cdot 8}{3 + 2 + 2 \cdot 8} \\ &= \frac{16}{21} \\ &= 0.761 \end{aligned}$$

Lin's taxonomy similarity [56] is selected as the third measure for calculating TSC in *SEMCBF*. Lin's taxonomy similarity is an information theoretic approach based on probabilistic model. Taxonomy similarity between concepts using Lin's taxonomy similarity (TSC_{Lin}) is the following.

$$TSC_{Lin}(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j \\ \dfrac{2 \cdot \log Pr(C_k)}{\log Pr(C_i) + \log Pr(C_j)}, & \text{otherwise} \end{cases} \tag{3.8}$$

$Pr(C_n)$ is the probability that a randomly selected instance belongs to concept C_n, and C_k is the most specific concept that subsumes both C_i and C_j. Since the ontologies that are utilized in *SEMCBF* has two main concepts, *Movie* concept and *Feature* concept, the instances that represent movies and the instances that represent feature-values does not effect each others probabilities. For example, when the probability of a concept that is descendant of *Movie* concept, only the movie instances are taken into account. Therefore $Pr(C_n)$ is the following.

$$Pr(C_n) = \begin{cases} \dfrac{|ISET(C_n)|}{|ISET(Movie)|}, & \text{if } Movie \in UC(C_n, H^c) \\ \\ \dfrac{|ISET(C_n)|}{|ISET(Feature)|}, & \text{if } Feature \in UC(C_n, H^c) \end{cases} \tag{3.9}$$

$ISET(C_n)$ represents the set of instances that are instances of the concepts which are connected to the C_n concept by $subConceptOf$ links. $ISET(C_n)$ can be formulated is as follows,

$$ISET(C) = \{I \in \tilde{I} | C \in UC(CSET(I), H^c)\} \tag{3.10}$$

where

$$CSET(I) = \{C \in \tilde{C} | C(I)\} \tag{3.11}$$

$CSET(I)$ represents the set of concepts which instance I is connected to by *instanceOf* links. When the similarity between *Horror* and *Action* concepts in Movie Ontology 5 is considered, at first $|ISET(Movie)|$ have to be computed in order to find the probabilities needed. The number of the movie instances, in other words the number of the instances which are connected to the *genre* concepts by *instanceOf* links is 1659 in the system. Similarly, $|ISET(Horror)|$ is 98 and $|ISET(Action)|$ is 254. The most specific concept that subsumes both *Horror* and *Action* is *G9*. Additionally $|ISET(G9)|$ is 802. Using these computed values $TSC_{Lin}(Horror, Action)$ can be computed.

$$\begin{aligned} TSC_{Lin}(Horror, Action) &= \frac{2 \cdot \log(802/1659)}{\log(98/1659) + \log(254/1659)} \\ &= \frac{2 \cdot -0.3156}{-1.2286 + -0.815} \\ &= \frac{-0.6312}{-2.0436} \\ &= 0.308 \end{aligned}$$

The last measure used in *SEMCBF* is mentioned in [57]. In [57] different similarity calculation strategies are evaluated and the following similarity measure, which we call taxonomy similarity between concepts using Mclean's taxonomy similarity (TSC_{Mclean}), yields the best performance.

$$TSC_{Mclean}(C_i, C_j) = \begin{cases} 1, & \text{if } C_i = C_j \\ e^{-\alpha l} . \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, & \text{otherwise} \end{cases} \tag{3.12}$$

l is the shortest path length between C_i and C_j, h is the depth of most specific concept in ontology. According to [57], optimal values of parameters α and β is 0.2 and 0.6 respectively. Similar to TSC_{CM} and $TSC_{Wu\&Palmer}$, TSC_{Mclean} is based on the distance between concepts rather than an information theoretic approach.

If the same example is examined for TSC_{Mclean}, l, which is the shortest path between *Horror* and *Action* concepts, is 5. h, which is the depth of $G9$ concept in Movie Ontology 5, is 8. Based on this values, α and β constants $TSC_{Mclean}(Horror, Action)$ can be computed.

$$\begin{aligned} TSC_{Mclean}(Horror, Action) &= e^{-0.2 \cdot 5} . \frac{e^{0.6 \cdot 8} - e^{-0.6 \cdot 8}}{e^{0.6 \cdot 8} + e^{-0.6 \cdot 8}} \\ &= 0.378 \end{aligned}$$

After defining the taxonomy similarity between concepts, calculating taxonomy similarity between instances is reduced to calculating the similarity between two sets containing concepts.

TS between instance I_i and instance I_j is defined as follows.

$$TS(I_i, I_j) = \begin{cases} 1, & \text{if } I_i = I_j \\ SSIM(CSET(I_i), CSET(I_j)), & \text{otherwise} \end{cases} \tag{3.13}$$

$CSET$ was previously defined in the Equation 3.11. $SSIM(S_1, S_2)$ is the similarity between set S_1 and set S_2. Similarity between two sets can be found using the similarities between their elements, in this case TSC of concepts, and a method that defines a way of utilizing these similarities. These methods are mentioned later in Section 3.2.2.4.

3.2.2.2 Relation Similarity

The second type of similarity measure using ontology-based metadata is relation similarity. Relation similarity between two instances (RS) is based on their relations to other instances in ontology-based metadata. Suppose that DirectorX is the director of MovieA and MovieB, DirectorY is the director of MovieC. In this example, relation similarity between MovieA and MovieB is higher than the one between MovieA and MovieC because director of MovieA and MovieB is the same. For relation similarity measure, we use a modified version of Maedche and Zacharias's relation similarity measure [52]. RS between instance I_i and instance I_j can be calculated as follows:

$$RS(I_i, I_j) = \begin{cases} 1, & \text{if } I_i = I_j \\ \dfrac{\sum\limits_{p \in P_{co-I}} OR(I_i, I_j, p, IN) + \sum\limits_{p \in P_{co-O}} OR(I_i, I_j, p, OUT)}{|P_{co-I}| + |P_{co-O}|}, & \text{otherwise} \end{cases} \tag{3.14}$$

P_{co-I} stands for *incoming relations* and is the set of relations that allows $UC(C(I_i), H^c)$ and $UC(C(I_j), H^c)$ as range. Similarly, P_{co-O} stands for *outgoing relations* and is the set of relations that allows $UC(C(I_i), H^c)$ and $UC(C(I_j), H^c)$ as domain. $OR(I_i, I_j, p, DIR)$ stands for the similarity for relation p and direction DIR between instances I_i and I_j where $DIR \in \{IN, OUT\}$. Thus, relation similarity between two instances is computed by calculating similarities for each *incoming* and *outgoing relations* of these instances and taking average of them.

$OR(I_i, I_j, p, DIR)$ can be calculated by considering associated instances of I_i and I_j with respect to the relation p and direction DIR. For example, if we consider the similarity for the relation *hasDirector* and direction OUT between two movie instances in Movie Ontology 5, we consider the directors of two movies. Similarly, if we consider the similarity for the relation *hasDirector* and direction IN between two directors, we consider the movies directed by these directors. Associated instances (A_s) of instance I_n with respect to the relation P and direction DIR is the following.

$$A_s(P, I_n, DIR) = \begin{cases} \{I_k : I_k \in \tilde{I} \wedge (P(I_k, I_n)\}, & \text{if DIR = IN} \\ \{I_k : I_k \in \tilde{I} \wedge (P(I_n, I_k)\}, & \text{if DIR = OUT} \end{cases} \tag{3.15}$$

After defining $A_s(P, I_n, DIR)$, calculating $OR(I_i, I_j, p, DIR)$ is reduced to calculating the similarity between two sets that contain associated instances. $OR(I_i, I_j, p, DIR)$ is defined as follows.

$$OR(I_i, I_j, p, DIR) = \begin{cases} 0, & \text{if } (A_S(P, I_i, DIR) = \emptyset \\ & \vee A_S(P, I_j, DIR)) = \emptyset) \\ SSIM(A_S(p, I_i, DIR), A_S(p, I_j, DIR)), & \text{otherwise} \end{cases} \tag{3.16}$$

As mentioned in Section 3.2.2.1, to find the similarity between two sets ($SSIM$), similarities between their elements are considered using a method. In this case, associated instances consist these sets and semantic similarities between the elements of these sets are used. The problem is that to calculate SSs between instances, RS is used and to calculate RSs between instances, SSs between associated instances are used. Therefore, in order to prevent calculation from infinite cycles, a *maximum recursion depth* has to be defined.

The advantage of using relation similarity is that the similarities between associated instances are taken into account. If the relation similarity between movie instances are considered, associated instances are feature-values of these movies. Similarly, if the relation similarity between instances, which represent feature-values, are considered, associated instances are the instances that represent movies. Suppose that, in a system, movies have only one feature which is an actor starred in the movie and we try to find the similarity between two movies, MovieX and MovieY. MovieX has a feature-value ActorA and MovieY has a feature-value

ActorB. If a user only rates movies in which only ActorA starred, it is unable to predict the rating of MovieY by using naive Bayesian classifier [41] which is used in [11]. But in *SEM-CBF*, relation similarity between MovieX and MovieY depends on the semantic similarity between ActorA and ActorB. In a recursive manner, relation similarity between ActorA and ActorB depends on the semantic similarity between other instances which have relations to ActorA and ActorB. Therefore, a similarity value between these two movies can be computed and then, a rating prediction can be made.

3.2.2.3 Attribute Similarity

Attribute similarity (AS) is the third similarity measure that is used for calculating semantic similarities in ontology-based metadata. Similar to the relation similarity, attribute similarity between two instances depends on their attribute values. AS between instance I_i and instance I_j is the following.

$$AS(I_i, I_j) = \begin{cases} 1, & \text{if } I_i = I_j \\ \dfrac{\sum\limits_{a \in P_A} OA(I_i, I_j, a)}{|P_A|}, & \text{otherwise} \end{cases} \tag{3.17}$$

P_A represents the set of attributes that are attributes of both $UC(C(I_i), H^c)$ and $UC(C(I_j), H^c)$. $OA(I_i, I_j, a)$ is the similarity between instances I_i and I_j for attribute a. Therefore, attribute similarity between two instances is computed by calculating similarities for each attribute in the set P_A and taking average of these similarities.

Similar to the calculation of $OR(I_i, I_j, a)$, $OA(I_i, I_j, a)$ is calculated by considering associated literals of I_i and I_j with respect to the attribute a. For example if the attribute is *ReleaseDate*, associated literals of two movies are years of the release dates of these movies in Movie Ontology 5. Associated literal (A_l) of I_n with respect to the attribute A is the following.

$$A_l(A, I_n) = \begin{cases} L_x, & \text{if } L_x \in \tilde{L} \wedge A(I_n, L_x) \\ \emptyset, & \text{otherwise} \end{cases} \tag{3.18}$$

The difference between A_s and A_l is that A_l can contain at most one literal unlike A_s. The reason of this difference is that the characteristics of the ontologies that are utilized in *SEM-*

CBF. In ontologies that are used in *SEMCBF*, an instance can have at most one attribute value (literal) with respect to an attribute. For instance, a movie can have more than one director, on the other hand, a movie has at most one release date. Therefore, rather than calculating similarity between two sets, similarity between attribute values is focused in order to calculate *OA*.

$$OA(I_i, I_j, a) = \begin{cases} 0, & \text{if } (A_l(A, I_i) = \emptyset \\ & \vee A_l(A, I_j) = \emptyset) \\ LSIM(L_i, L_j, a), & \text{otherwise} \end{cases} \tag{3.19}$$

$L_i = A_l(a, I_i)$ and $L_j = A_l(a, I_j)$. In ontologoies that are used in *SEMCBF*, all attributes (*ReleaseDate*, *Runtime*, *Rating*) represent numeric features of the items. Therefore, we have to translate difference of numeric values to a similarity value that is between 0 and 1. For this translation, maximum difference of a numeric attribute A ($MDIF(A)$) has to be defined.

$$MDIF(A) = max\{(L_i - L_j) : A(I_1, L_i) \wedge A(I_2, L_j) \wedge I_1, I_2 \in \tilde{I}\} \tag{3.20}$$

Suppose that we want to find the maximum difference of the attribute *Runtime*. Maximum difference, 260, is achieved by the shortest movie, which has an attribute value 20 for *Runtime* attribute, and the longest movie, which has an attribute value 280 for *Runtime* attribute. After defining maximum difference of a numeric attribute, we can define the similarity ($LSIM$) between attribute value L_1 and attribute value L_2 of an attribute a.

$$LSIM(L_1, L_2, a) = 1 - \frac{|L_1 - L_2|}{MDIF(a)} \tag{3.21}$$

3.2.2.4 Similarity Between Sets

In order to calculate taxonomy similarity and relation similarity, we have to define the similarity between sets of elements. Similarity between two sets depends on the similarities between their elements and the similarity method used. Elements of these sets are *concepts* for taxonomy similarity and *instances* for relation similarity calculation. Similarities between elements are TSCs for taxonomy similarity and SSs for relation similarity. In *SEMCBF*, six different methods can be used in order to find the similarity between two sets.

The first method ($SSIM_1$) used for calculating the similarity between two sets is mentioned in [52]. Similarity between set $S1$ and set $S2$ ($SSIM_1(S1, S2)$) is the following.

$$SSIM_1(S1, S2) = \begin{cases} \dfrac{\sum\limits_{a \in S1} max\{SIM(a, b) | b \in S2\}}{|S1|}, & \text{if } |S1| \geq |S2| \\ \\ \dfrac{\sum\limits_{a \in S2} max\{SIM(a, b) | b \in S1\}}{|S2|}, & \text{otherwise} \end{cases} \tag{3.22}$$

In $SSIM_1$, after selecting the set with the greater size, for each element in the selected set, an element which has the maximum similarity with the corresponding element is selected from the other set. Then, the similarity between these two sets is the average of these calculated similarities between selected element pairs.

The method mentioned in [58] is the second method ($SSIM_2$) used for calculating the similarity between two sets of elements.

$$SSIM_2(S1, S2) = \frac{\sum\limits_{a \in S1} max\{SIM(a, b) | b \in S2\} + \sum\limits_{b \in S2} max\{SIM(b, a) | a \in S1\}}{|S1| + |S2|} \tag{3.23}$$

In $SSIM_2$, for each element of the two sets, an element which has the maximum similarity with the corresponding element is selected from the other set. Then, the similarity between these two sets is the average of these calculated similarities between selected element pairs.

The third method, which is used for calculating similarity between two sets, forms pairs of elements by using these two sets [59]. Pairing of elements is done by selecting two elements a and b, whose similarity ($SIM(a, b)$) is maximum, from sets $S1$ and $S2$ respectively. Then a and b are removed from their belonging sets. This pairing process is done until one of these sets has no more elements. $SSIM_3(S1, S2)$ using element pairing is the following,

$$SSIM_3(S1, S2) = \frac{\sum\limits_{(a,b) \in Pairs(S1,S2)} SIM(a, b)}{max(|S1|, |S2|)} \tag{3.24}$$

where $Pairs(S1, S2)$ contains the pairs of elements that are formed by the above process.

The other methods used for calculating the similarity between two sets are based on the methods used for calculating the distance of pair of clusters in hierarchical clustering algorithms. These methods are single-link, complete-link and average-link [60].

In the single-link hierarchical clustering, distance between two clusters is the *minimum* distance between the pairs of elements belong to these clusters. Since distance and similarity are inversely proportional, similarity between two sets is the *maximum* similarity between elements from these sets. Similarity between two sets using single-link method ($SSIM_S$) is the following.

$$SSIM_S(S1, S2) = max\{SIM(a, b) | a \in S1 \wedge b \in S2\} \tag{3.25}$$

Complete-link hierarchical clustering is the opposite of single-link hierarchical clustering. In the complete-link hierarchical clustering, distance between two clusters is the *maximum* distance between the pairs of elements of these clusters. Therefore, we take the *minimum* similarity between elements of two sets to determine the similarity between these sets using complete-link method ($SSIM_C$).

$$SSIM_C(S1, S2) = min\{SIM(a, b) | a \in S1 \wedge b \in S2\} \tag{3.26}$$

The last method used for calculating the similarity between two sets is average-link. In the average-link hierarchical clustering, distance between two clusters is the *average* of all distances between the pairs of elements of these clusters. By using this definition, similarity between two sets using average-link method ($SSIM_A$) can be calculated as follows.

$$SSIM_A(S1, S2) = \frac{\sum\limits_{a \in S1} \sum\limits_{b \in S2} SIM(a, b)}{|S1||S2|} \tag{3.27}$$

3.2.3 Rating Prediction

The last step of *SEMCBF* is prediction of the unknown ratings in order to enhance user-item rating matrix. In order to predict the unknown ratings, *SEMCBF* uses a prediction function (PF), semantic similarities between items (SS) and collaborative-based user models which consist of ratings given by users, on a neighborhood-based method [22] [19]. In order to compute a prediction for a user-item pair, two prediction functions can be used after selecting a number of (k) most similar items (*k nearest neighbors*) to the item in the pair.

By using the first prediction function (*pred1*), the predicted rating of user u for item i can be calculated by taking the average of the ratings given by the user u for $\hat{I}$ which is the set of k most similar items to i (according to the SSs) that have been rated by user u.

$$pred1_{u,i} = \frac{1}{k} \sum_{\acute{i} \in \hat{I}} r_{u,\acute{i}} \tag{3.28}$$

By using the second prediction function (*pred2*), the predicted rating of user u for item i can be calculated by taking the weighted average of the ratings given by the user u for $\hat{I}$ which is the set of k most similar items to i that have been rated by user u. Weights of the ratings are set according to the sematic similarities between items.

$$pred2_{u,i} = \frac{1}{\sum_{\acute{i} \in \hat{I}} SS(i, \acute{i})} \sum_{\acute{i} \in \hat{I}} SS(i, \acute{i}) r_{u,\acute{i}} \tag{3.29}$$

SEMCBF creates enhanced user-item rating matrix by predicting all unknown user-item pairs. In other words, the sparsity of user-item rating matrix is reduced. Additionally, new item problem is avoided. Even if an item has no explicit rating given by any user in the system, by using *SEMCBF*, our approach predicts a rating given by every user for that item. Thus, a new item which has no ratings given by any user in the system has a chance of being recommended.

3.3 Refinements of *SEMCBF*

There are two refinements that are applied in order to improve the quality of *SEMCBF*'s recommendations. One of these refinements is excluding unnecessary attributes and relations from recommendation process. The other refinement is determining feature-weights and utilizing these weight values in *SEMCBF*.

3.3.1 Feature Selection

Feature selection is a way of dimensionality reduction which is a process of finding low dimensional representations for high-dimensional data [61]. The aim of a feature selection method is to find the *best subset* of the input feature set [62]. Best subset refers to the set of features that optimizes a given objective/criterion function. The size of the subset can be predetermined or can be optimized by the feature selection method. In fact there are optimal methods, e.g. exhaustive search, due to the computational constraints, we apply Sequential Forward Selection (*SFS*) [63] which is a sub-optimal feature selection algorithm. SFS starts from an empty set and at each iteration, adds a feature, which maximizes the objective function, to the set. This process ends when the size of the subset reaches the predetermined value or there are no more features remaining.

SFS is used for determining a sub-optimal subset of relations and attributes used in *SEMCBF* in order to improve the performance of *SEMCBF*. Therefore, relations and attributes constitutes the features and any performance metric that is used for evaluating the performance of recommendation systems can be used as the objective function. Additionally, the size of the subset is not predetermined. Algorithm 1 computes the *SFS* used for *SEMCBF*.

Algorithm 1 Sequential Forward Selection for SEMCBF

```
Input: $X$ — the set of relations and attributes of utilized ontology in $SEMCBF$
  $OF$ — evaluation function of $SEMCBF$ using relations and attributes set as input
Output: $SF$ — the set of selected relations and attributes $\vee\ SF \subseteq X$
  $Y \Leftarrow \emptyset$
  $maxOF \Leftarrow 0$
  while $X \neq \emptyset$ do
    $x \Leftarrow \text{argmax}_{x \in X}[OF(Y + x)]$
    $Y \Leftarrow Y + x$
    $X \Leftarrow X - x$
    if $OF(Y) > maxOF$ then
      $SF \Leftarrow Y$
    end if
  end while
```

In Algorithm 1, it is assumed that minimum value of the evaluation function is 0 and higher values of the function indicates better performance of *SEMCF*. For each iteration, if the performance of *SEMCBF* using new subset exceeds the *maxOF* that keeps the value of the best evaluation value of the previous subsets, selected subset is set to the new subset.

3.3.2 Feature Weighting

As mentioned in Section 3.2.2, two of the similarity measures used for calculating semantic similarity between two instances are relation similarity and taxonomy similarity. Relation similarity is calculated by considering similarities for each relation of these two instances' relations and taking average of these similarities. In other words, every relation has the same weight, same effect. It is also the same for calculation of the attribute similarity between two instances. On the other hand, rather than using equal weights, determining these weights can improve the quality of recommendations.

In *SEMCBF*, content-based user models are used for calculating the similarities between items in the system. The reason for using content-based user models is to discover effects of each feature to the preference of the users. For instance, some users consider *writer* feature of the movie is important more than *director* feature of the movie. For such users, when calculating the similarities between items, *SEMCBF* gives more weight to the similarity of the relation that represents *writer* feature than the similarity of the relation that represents *director* feature. In order to determine the feature-weights, multiple criteria approach in [64] is used.

Every movie in the system is represented as a vector called *movie feature vector* (*MFV*). *MFV* of the movie *m* is as follows.

$$MFV_m = ((f_{1,1}, ..., f_{1,k_1}), ..., (f_{N,1}, ..., f_{N,k_N})) \tag{3.30}$$

Each feature-value has an index number for its corresponding feature. This index number has a range between 1 and the number of values belonging to the corresponding feature. And also each feature has a number in the system. In *MFV*, $f_{i,j}$ represents the value with the index j of the feature i, k_i represents the number of values for the feature i and N represents the total number of features. For instance, *language* feature, which is the 4rd feature, has 76 values such as *Turkish*, *Danish*, *English* etc., and *Turkish* is the 5th value for the *language* feature. Then, if the language of a movie is not Turkish, then $f_{3,5}$ is set to 0. But if a movie has a feature-value that is represented by $f_{i,j}$, $f_{i,j}$ is set a value that depends on the selected method. We use three types of methods to set $f_{i,j}$ in *MFV* of a movie:

- $Default$
- Idf
- Ifw

Simplest method used to construct a *MFV* is *Default*. In this method if a movie has a feature-value that is represented by $f_{i,j}$, then $f_{i,j}$ is set to 1. In the second method, *inverse document frequency* (*Idf*) of the feature-values, which is mentioned in [2], is used. *Idf* is defined as follows,

$$Idf_{f_{i,j}} = \log \frac{M}{m_{f_{i,j}}} \tag{3.31}$$

where M is the number of all movies and $m_{f_{i,j}}$ is the number of movies that has a feature-value that is represented by $f_{i,j}$. If a movie has a feature-value that is represented by $f_{i,j}$, then $f_{i,j}$ is set to $Idf_{f_{i,j}}$. Similarly, in the third method, $f_{i,j}$ is set to *item feature weight* (*Ifw*) [65] of the feature-value. *Ifw* is defined as follows,

$$Ifw_{f_{i,j}} = \log \frac{M}{m_{f_{i,j}}} \times \log k_i \tag{3.32}$$

where k_i is the number of values for feature i. Therefore, the importance of a feature-value is proportional to the number of feature-values of its corresponding feature. After constructing the *MFV*s of all movies in the system, in order to determine the users' preferences, for each user, a *user preference vector* (*UPV*) is constructed. *UPV* of the user u is as follows.

$$UPV_u = ((pr_{1,1}, ..., pr_{1,k_1}), ..., (pr_{N,1}, ..., pr_{N,k_N})) \tag{3.33}$$

$pr_{i,j}$ represents the opinion of the corresponding user about the feature-value that is represented by $f_{i,j}$. In order to construct *UPV* of a user, *MFV*s of the movies that has been rated by the user and the rating values are used.

$$UPV_u = \frac{\sum\limits_{m \in \hat{M}} 0.2 \times r_{u,m} \times MFV_m}{|\hat{M}|} \tag{3.34}$$

$\hat{M}$ is the set of movies that has been rated by the user, and $r_{u,m}$ is the rating given by the user u for the movie m. In the equation 3.34, 0.2 is used for transforming the rating value in range 1-5 to normalized rating value in range 0-1.

User weight vector (*UWV*) represents the opinion of a user about features by using weight values. For a user, weight values of the features are obtained from the user's *UPV*. *UWV* of the user u is as follows,

$$UVW_u = (w_1, w_2 ..., w_N) \tag{3.35}$$

where $w_i = pr_i / \sum_{i=1}^{N} pr_i$. w_i represents the opinion of the corresponding user about the feature i.

In order to find the value of w_i in *UVW*, we use two different methods to set pr_i:

- $Default$
- $Stddev$

Using *Default* method, pr_i is the biggest value of the values that represent the preference of the user about feature-values of the feature i in the corresponding *UPV*. In the second method, *Stddev*, pr_i is standard deviation of the values that represent the preference of the user about feature-values of the feature i in the corresponding *UPV*.

After constructing the *UWV*s of all users in the system, in order to avoid scalability problem, we apply clustering on *UWV*s for grouping users that have similar *UWV*s. In order to cluster users, we apply *K-Means* and *Expectation Maximisation* algorithms separately by using *Weka* API [66]. One crucial point is that when UWVs are clustered, the weights of the *genre* feature are ignored. Because the *genre* feature is not represented by a relation or attribute in any of the ontologies used.

Cluster weight vector (*CWV*) represents the opinion of the group of users in a cluster about features. Feature-weights of the clusters are calculated by the UWVs in the system. CWV of the cluster Cl_k is as follows,

$$CWV_{Cl_k} = \frac{\sum\limits_{u \in U(Cl_k)} UWV_u}{|U(Cl_k)|} \tag{3.36}$$

where $U(Cl_k)$ is the users in cluster Cl_k. Briefly, in order to determine the feature-weights of a cluster, average of feature-weights of the users that belong to the cluster is computed.

So far, feature-weights, or in other words weights of relations and attributes of the clusters, are determined. These feature-weights are used in relation and attribute similarity measures of semantic similarity calculation. For each cluster, semantic similarities between instances are calculated separately. In order to consider the different weights in different clusters, the Equation 3.3 is changed into a new one. SS between two instances for cluster Cl_k is defined as follows.

$$SS_{Cl_k}(I_i, I_j) = \frac{aTS(I_i, I_j) + bRS_{Cl_k}(I_i, I_j) + cAS_{Cl_k}(I_i, I_j)}{a + b + c} \tag{3.37}$$

$RS_{Cl_k}(I_i, I_j)$, which represents relation similarity between instance I_i and instance I_j for cluster Cl_k, is the following,

$$RS_{Cl_k}(I_i, I_j) = \begin{cases} 1, & \text{if } I_i = I_j \\ \dfrac{\sum\limits_{p \in P_{co-I}} w(Cl_k, p) OR(I_i, I_j, p, IN)}{\sum\limits_{p \in P_{co-I}} w(Cl_k, p) + \sum\limits_{p \in P_{co-O}} w(Cl_k, p)} + & \\ \dfrac{\sum\limits_{p \in P_{co-O}} w(Cl_k, p) OR(I_i, I_j, p, OUT)}{\sum\limits_{p \in P_{co-I}} w(Cl_k, p) + \sum\limits_{p \in P_{co-O}} w(Cl_k, p)}, & \text{otherwise} \end{cases} \tag{3.38}$$

where $w(Cl_k, p)$ is the weight value of the feature, which is represented by the relation p, in CWV_{Cl_k}. Similarly, $AS_{Cl_k}(I_i, I_j)$, which represents attribute similarity between instance I_i and instance I_j for cluster Cl_k, is the following:

$$AS(I_i, I_j)_{Cl_k} = \begin{cases} 1, & \text{if } I_i = I_j \\ \dfrac{\sum\limits_{a \in P_A} w(Cl_k, a) OA(I_i, I_j, a)}{\sum\limits_{a \in P_A} w(Cl_i, a)}, & \text{otherwise} \end{cases} \tag{3.39}$$

where $w(Cl_k, a)$ is the weight value of the feature, which is represented by the attribute a, in CWV_{Cl_k}. After finding semantic similarities between instances for each cluster, in order to predict unknown ratings of user-item rating matrix, the last step of *SEMCBF* is rating prediction that is mentioned in Section 3.2.3. For the first prediction function (*pred1*), the only difference in Equation 3.28 is the definition of $\hat{I}$. Since the semantic similarities between items are calculated for each cluster, items $\hat{I}$ is the set of k most similar items to i for the cluster that user u belongs to. Similarly, for the second prediction function (*pred2*), the Equation 3.29 is changed into the following,

$$pred2_{u,i} = \frac{1}{\sum_{\acute{i} \in \hat{I}} SS_{Cl_k}(i, \acute{i})} \sum_{\acute{i} \in \hat{I}} SS_{Cl_k}(i, \acute{i}) r_{u,\acute{i}} \tag{3.40}$$

where Cl_k is the cluster that user u belongs to.

3.4 Collaborative Filtering

In fourth phase of *SEMCBCF*, a neighborhood-based [22] collaborative filtering algorithm is performed on enhanced user-item rating matrix and active user ratings vector which consists of actual ratings given by the active user and the ratings predicted by *SEMCBF*.

The collaborative filtering algorithm first computes the similarity between the active user and other users by using the enhanced user-item rating matrix. And then, a number of most similar users called neighbors are selected. In order to calculate similarities between the active user and other users, Pearson correlation coefficient [20] is used.

After computing similarities between the active user and other users by using enhanced user-item rating matrix, n most similar users, which are called n-nearest neighbors, are selected. An unknown rating for a user-item pair in active user ratings vector is predicted by calculating the adjusted weighted sum of the user's n-nearest neighbors' ratings for the item. The advantage of using adjusted weighted sum is that it also considers difference of rating scales of the users.

The weight of nearest neighbors depends on the similarities between users.

At the end of the recommendation process, the system recommends a number of unrated items which have the highest predicted rating to the active user.

CHAPTER 4

EVALUATION AND RESULTS

This chapter presents the evaluation of the system. First, the data set used in experiments is described. Then, evaluation metrics that are used to evaluate the system is explained. After that, the results of the experiments are shown and discussed.

4.1 Data Set

The data set used in experiments is the MovieLens 100k data set [50] which is publicly available. This data set contains 100000 ratings that are collected from 943 users for 1682 movies. The ratings in MovieLens 100k data set are on a scale of 1 to 5. 1 indicates strongly dislike and 5 indicates strongly like. Each user in this data set has rated at least 20 movies. This data set contains some demographic information about the users (age, gender, occupation, zip) and some information about the items/movies. The information about an item consists of *title*, *release date*, *IMDb URL* and *genre* of that item. *IMDb URL* is used for extracting the content of a movie that is explained in Section 3.2.1.1. Additionally, rather than using the genre information provided by this data set, genre information of each movie is extracted from IMDb.

In order to apply 5-fold cross-validation, the disjoint test sets (20% of rating data) and their corresponding training sets (80% of rating data) are provided in MovieLens 100k dataset. Therefore, 100000 ratings are divided into 5 disjoint sets and each of these sets contains 20000 ratings. For each of these disjoint sets, experiments are performed using the remaining sets as training data set and the selected set as test data set. And finally, the results are averaged.

MovieLens 100k data set has some inconsistencies. For some movies, *IMDb URL* of that movie is unavailable. For some different movieIDs in data set, the title and *IMDb URL* are the same. In order to fix these inconsistencies, all movies in this data set are processed. The movies, whose *IMDb URL*s are unavailable, are removed with corresponding rating data. For duplicate movies, just one movie entity and its corresponding rating data are kept in the data set. After processing the data set, disjoint sets that are used for 5-fold cross-validation contain the following numbers of ratings:

- Set1 contains 19964 ratings.
- Set2 contains 19954 ratings.
- Set3 contains 19930 ratings.
- Set4 contains 19925 ratings.
- Set5 contains 19905 ratings.

4.2 Evaluation Metrics

In experiments, in order to evaluate the performance of the system, we use mean absolute error (MAE) and F-Measure performance metrics that are mentioned in Section 2.3. In order to use F-Measure performance metric, actual and predicted ratings below or equal 3 indicates that the item is irrelevant and those above 3 indicates that the item is relevant.

4.3 Evaluation of *SEMCBF* and *SEMCBCF*

The evaluation process consists of three phases. In the first phase, in order to find the most appropriate values of *SEMCBF* parameters, *SEMCBF* is evaluated without considering the refinements (feature selection and feature weighting) mentioned in Section 3.3. In other words, all users are grouped in only one cluster with equal feature-weight values. Therefore, CWV of this cluster has the value of 1 for weight of each feature. In the second phase, the refinements of *SEMCBF*, which are feature selection and feature weighting, are evaluated. In the third phase, the performance of *SEMCBF* and *SEMCBF* is compared with some other approaches [7, 67, 1, 68, 11] in the literature.

4.3.1 First Phase of Evaluation

The content-based filtering part of *SEMCBCF*, named as *SEMCBF*, consists of some parameters as mentioned in Chapter 3. These parameters and their possible values are shown in Table 4.1. The performance of *SEMCBF*, thereby *SEMCBCF*, depends on the values of these parameters.

Table 4.1: Parameters and Possible Values of *SEMCBF*

Parameter	**Possible Values**
Ontology (O)	Movie Ontology 1 Movie Ontology 2 Movie Ontology 3 Movie Ontology 4 Movie Ontology 5
Max. Recursive Depth (rd)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Weight of TS (a)	0, 0.1, 0.2, 0.3, ..., 1.0
Weight of RS (b)	0, 0.1, 0.2, 0.3, ..., 1.0
Weight of AS (c)	0, 0.1, 0.2, 0.3, ..., 1.0
Measure for Taxonomy Similarity Between Concepts (TSC)	TSC_{CM} $TSC_{Wu\&Palmer}$ TSC_{Lin} TSC_{Mclean}
$SSIM$ Method for TS ($SSIM_{TS}$)	$SSIM_1, SSIM_2,$ $SSIM_3, SSIM_S,$ $SSIM_C, SSIM_A$
$SSIM$ Method for RS ($SSIM_{RS}$)	$SSIM_1, SSIM_2,$ $SSIM_3, SSIM_S$ $SSIM_C, SSIM_A$
Number of Nearest Neighbors (k)	5, 10, 15, 20, 30, 50, 100, 200
Prediction Function (PF)	$pred1$, $pred2$

Because semantic similarity between two instances is computed by the weighted arithmetic mean of TS, RS and AS, the parameters a, b, c, satisfies the constraint $(a + b + c) = 1$ during the evaluation process. For each parameter we try to find the most appropriate value. In order to find the most appropriate value of a parameter, *SEMCBF* is evaluated for each of the values of the corresponding parameter, k and O, while other parameters are set to selected constant values from the possible values. For each analysis of parameters, the determined values of the previously analyzed parameters are kept same. Therefore at the end of the first phase of evaluation, for each of the ontologies, appropriate values of the parameters mentioned in

Table 4.1 are determined. The experiments are conducted in order to analyze the parameters in the following order:

1. PF
2. TSC and $SSIM_{TS}$
3. $SSIM_{RS}$
4. a, b and c
5. rd

4.3.1.1 Experiment for PF

After SSs between items are computed in the system, PF parameter of *SEMCBF* is used in the rating prediction phase. In order to find the most appropriate value of PF, parameters rd, a, b, c, TSC, $SSIM_{TS}$, $SSIM_{RS}$ are set to their initial values and *SEMCBF* is evaluated using for each value of k, O and PF. We call combination of the values of parameters, *SEMCBF configuration*. Therefore, using *SEMCBF* configurations shown in Table 4.2, the performance of *SEMCBF* in MAE and F-Measure metrics is observed. The results of the experiment are shown in Table 4.3.

Table 4.2: *SEMCBF* Configurations of the Experiment for PF

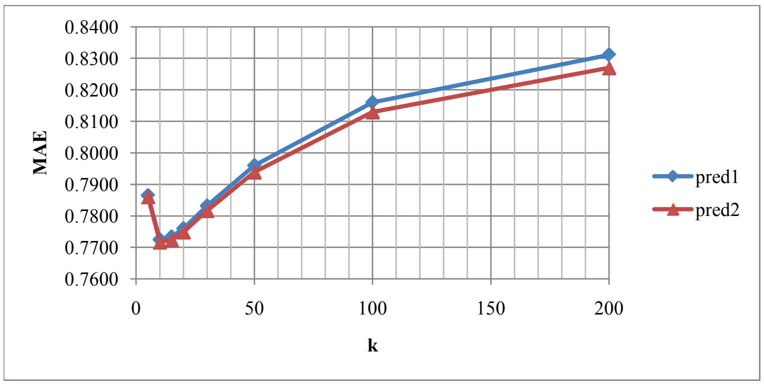
O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	All Values	All Values	All Values	All Values	All Values
TSC	TSC_{CM}	TSC_{CM}	TSC_{CM}	TSC_{CM}	TSC_{CM}
$SSIM_{TS}$	$SSIM_1$	$SSIM_1$	$SSIM_1$	$SSIM_1$	$SSIM_1$
$SSIM_{RS}$	$SSIM_1$	$SSIM_1$	$SSIM_1$	$SSIM_1$	$SSIM_1$
a	0.4	0.4	0.4	0.4	0.4
b	0.3	0.3	0.3	0.3	0.3
c	0.3	0.3	0.3	0.3	0.3
rd	1	1	1	1	1

Table 4.3: Results of the Experiment for *PF*

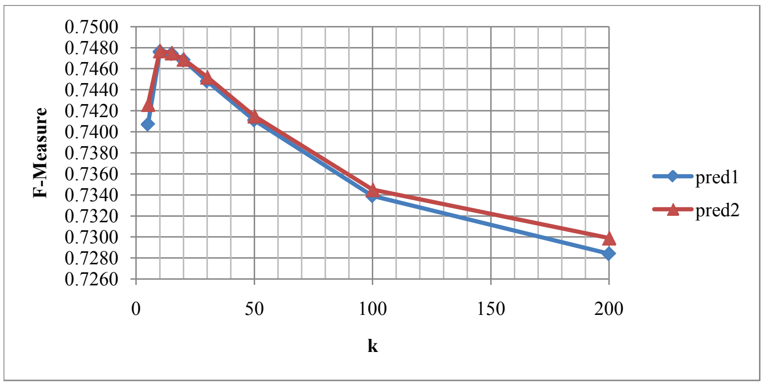
		Movie Ontology 1		**Movie Ontology 2**		**Movie Ontology 3**		**Movie Ontology 4**		**Movie Ontology 5**	
		MAE		MAE		MAE		MAE		MAE	
		PF		*PF*		*PF*		*PF*		*PF*	
		*pred*1	*pred*2	*pred*1	*pred*2	*pred*1	*pred*2	*pred*1	*pred*2	*pred*1	*pred*2
k	5	0.8076	0.8070	0.8003	0.7997	0.7871	0.7867	0.7923	0.7919	0.7865	0.7861
	10	0.7907	0.7895	0.7845	0.7833	0.7725	0.7717	0.7770	0.7762	0.7725	0.7717
	15	0.7896	0.7880	0.7834	0.7817	0.7725	0.7714	0.7767	0.7755	0.7734	0.7724
	20	0.7916	0.7897	0.7860	0.7840	0.7750	0.7737	0.7793	0.7779	0.7760	0.7749
	30	0.7978	0.7954	0.7925	0.7899	0.7820	0.7804	0.7858	0.7840	0.7832	0.7817
	50	0.8071	0.8042	0.8027	0.7994	0.7941	0.7919	0.7971	0.7947	0.7960	0.7939
	100	0.8219	0.8180	0.8192	0.8146	0.8147	0.8114	0.8161	0.8125	0.8160	0.8130
	200	0.8326	0.8277	0.8317	0.8259	0.8305	0.8259	0.8307	0.8258	0.8311	0.8270
		F-Measure		F-Measure		F-Measure		F-Measure		F-Measure	
		PF		*PF*		*PF*		*PF*		*PF*	
		*pred*1	*pred*2	*pred*1	*pred*2	*pred*1	*pred*2	*pred*1	*pred*2	*pred*1	*pred*2
k	5	0.7330	0.7360	0.7370	0.7397	0.7406	0.7422	0.7382	0.7404	0.7407	0.7426
	10	0.7416	0.7419	0.7443	0.7443	0.7474	0.7475	0.7458	0.7459	0.7476	0.7477
	15	0.7419	0.7419	0.7446	0.7448	0.7476	0.7477	0.7463	0.7463	0.7474	0.7475
	20	0.7410	0.7414	0.7435	0.7435	0.7465	0.7467	0.7460	0.7461	0.7468	0.7469
	30	0.7390	0.7394	0.7410	0.7417	0.7444	0.7449	0.7436	0.7438	0.7448	0.7452
	50	0.7360	0.7371	0.7377	0.7383	0.7413	0.7416	0.7402	0.7406	0.7411	0.7415
	100	0.7310	0.7322	0.7318	0.7329	0.7340	0.7348	0.7333	0.7342	0.7339	0.7345
	200	0.7270	0.7285	0.7273	0.7292	0.7285	0.7301	0.7282	0.7297	0.7284	0.7299

The results show that better performance in MAE and F-Measure metrics is achieved by using *pred2* for *PF* rather than using *pred1* for *PF*. The reason of *pred*2 gives better results is that the ratings of more similar items have more effect on the rating prediction by using a weighted sum. However, using *pred*1, these similarities are not considered. Considering these similarities in rating prediction step improves the performance of *SEMCBF*.

Among five different ontologies used in this experiment, Movie Ontology 5 gives the best result. In order to analyze the effect of *k* to the performance of *SEMCBF*, MAE and F-Measure obtained from *SEMCBF* using Movie Ontology 5 and different values of *k* are illustrated in Figure 4.2. The vertical axes show MAE and F-Measure and the horizontal axes show the values of *k*. It can be seen that, performance of *SEMCBF* in MAE and F-Measure initially improves with the increase of the value of *k* and then declines after $k = 10$. Therefore, after the point of $k = 10$, additional neighbors of the corresponding item have negative effect on rating prediction.



(a) MAE vs *k*



(b) F-Measure vs *k*

Figure 4.1: Results of the Experiment for *PF* using Movie Ontology 5

4.3.1.2 Experiment for TSC and $SSIM_{TS}$

The value *pred2* is determined as the most appropriate value of *PF* in the previous experiment. This value is used for the remaining evaluation phases. Similar to the previous experiment, *SEMCBF* is evaluated using each of the values of *k*, *O*, TSC and $SSIM_{TS}$ while the values of other parameters remain constant. *SEMCBF* configurations that are used for this experiment are shown in Table 4.4. The reason behind analyzing combination of TSC and $SSIM_{TS}$ rather than analyzing them separately is that these two parameters are highly related to each other while calculating semantic similarities between two instances in ontology. The results of the experiment are shown in Table 4.5. These are the best results of *SEMCBF* among all values of *k*. For each ontology, the best MAE and F-Measure values are in bold in Table 4.5.

Table 4.4: *SEMCBF* Configurations of the Experiment for TSC and $SSIM_{TS}$

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	All Values	All Values	All Values	All Values	All Values
$\boldsymbol{SSIM_{TS}}$	All Values	All Values	All Values	All Values	All Values
$\boldsymbol{SSIM_{RS}}$	$SSIM_1$	$SSIM_1$	$SSIM_1$	$SSIM_1$	$SSIM_1$
a	0.4	0.4	0.4	0.4	0.4
b	0.3	0.3	0.3	0.3	0.3
c	0.3	0.3	0.3	0.3	0.3
rd	1	1	1	1	1

Table 4.5: Results of the Experiment for TSC and $SSIM_{TS}$

MAE						
Movie Ontology 1						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7880	0.7880	0.7997	0.7898	0.7920	**0.7850**
$TSC_{Wu\&Palmer}$	0.8084	0.8107	0.8088	0.7921	0.7963	0.8070
TSC_{Mclean}	0.8085	0.8108	0.8088	0.7921	0.7973	0.8073
TSC_{Lin}	0.8087	0.8109	0.8088	0.7921	0.8016	0.8074
Movie Ontology 2						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7817	0.7823	0.7939	0.7835	0.7833	**0.7787**
$TSC_{Wu\&Palmer}$	0.8024	0.8048	0.8026	0.7809	0.7813	0.7972
TSC_{Mclean}	0.8009	0.8033	0.8011	0.7805	0.7805	0.7950
TSC_{Lin}	0.8061	0.8082	0.8062	0.7851	0.7926	0.8029
Movie Ontology 3						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7714	0.7711	0.7814	0.7729	0.7740	0.7689
$TSC_{Wu\&Palmer}$	0.7923	0.7948	0.7928	0.7666	0.7711	0.7820
TSC_{Mclean}	0.7925	0.7948	0.7929	0.7671	0.7726	0.7823
TSC_{Lin}	0.7927	0.7945	0.7927	**0.7659**	0.7747	0.7827
Movie Ontology 4						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7755	0.7745	0.7807	0.7731	0.7738	0.7703
$TSC_{Wu\&Palmer}$	0.7962	0.7939	0.7940	0.7668	0.7721	0.7856
TSC_{Mclean}	0.7951	0.7940	0.7933	0.7673	0.7735	0.7844
TSC_{Lin}	0.7972	0.7937	0.7943	**0.7662**	0.7758	0.7863
Movie Ontology 5						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7717	0.7696	0.7845	0.7719	0.7766	0.7711
$TSC_{Wu\&Palmer}$	0.7744	0.7701	0.8008	**0.7658**	0.7784	0.7704
TSC_{Mclean}	0.7883	0.7845	0.7957	0.7670	0.7815	0.7795
TSC_{Lin}	0.7943	0.7911	0.7941	0.7659	0.7816	0.7852

F-Measure						
Movie Ontology 1						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7419	0.7424	0.7377	0.7420	0.7430	**0.7440**
$TSC_{Wu\&Palmer}$	0.7344	0.7334	0.7345	0.7419	0.7419	0.7347
TSC_{Mclean}	0.7344	0.7335	0.7344	0.7419	0.7420	0.7346
TSC_{Lin}	0.7344	0.7334	0.7344	0.7419	0.7400	0.7347
Movie Ontology 2						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7448	0.7444	0.7399	0.7450	0.7451	0.7460
$TSC_{Wu\&Palmer}$	0.7367	0.7354	0.7365	0.7462	0.7463	0.7381
TSC_{Mclean}	0.7370	0.7358	0.7370	**0.7465**	0.7464	0.7390
TSC_{Lin}	0.7355	0.7345	0.7355	0.7445	0.7425	0.7365
Movie Ontology 3						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7477	0.7476	0.7438	0.7473	0.7475	0.7483
$TSC_{Wu\&Palmer}$	0.7398	0.7394	0.7400	0.7492	0.7485	0.7432
TSC_{Mclean}	0.7399	0.7395	0.7402	0.7491	0.7480	0.7432
TSC_{Lin}	0.7401	0.7396	0.7401	**0.7493**	0.7480	0.7436
Movie Ontology 4						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7463	0.7471	0.7442	0.7472	0.7474	0.7479
$TSC_{Wu\&Palmer}$	0.7390	0.7394	0.7397	0.7491	0.7478	0.7424
TSC_{Mclean}	0.7401	0.7398	0.7399	0.7489	0.7474	0.7427
TSC_{Lin}	0.7390	0.7399	0.7395	**0.7491**	0.7472	0.7424
Movie Ontology 5						
	$SSIM_1$	$SSIM_2$	$SSIM_3$	$SSIM_S$	$SSIM_C$	$SSIM_A$
TSC_{CM}	0.7477	0.7485	0.7428	0.7475	0.7469	0.7482
$TSC_{Wu\&Palmer}$	0.7470	0.7487	0.7380	**0.7493**	0.7464	0.7480
TSC_{Mclean}	0.7421	0.7436	0.7389	0.7491	0.7449	0.7448
TSC_{Lin}	0.7402	0.7412	0.7397	0.7492	0.7448	0.7423

The best result in MAE is obtained by using TSC_{CM} and $SSIM_A$ and the best result in F-Measure is obtained by using TSC_{Mclean} and $SSIM_S$ in Movie Ontology 2. These two performance metrics are commonly used for evaluating recommendation systems. Therefore, in the rest of the first phase of evaluation, the most appropriate values of *SEMCBF* parameters are found by considering MAE and F-Measure separately. The ontologies other than Movie Ontology 2, using the same TSC and $SSIM_{TS}$ pair yields best results both in MAE and F-Measure.

As it can be seen from the results, different TSC and $SSIM_{TS}$ values yield best results for different ontologies. The reason of that is the characteristics of the ontologies that are representation of the features and hierarchy of the concepts. For example, the depth of the hierarchy of descendants of *Movie* concept in Movie Ontology 5 is bigger than it is in Movie Ontology 3 and Movie Ontology 4. Additionally, this hierarchy in Movie Ontology 5 is created using the domain knowledge. When the values of TSC parameter are considered, TSC_{Lin} gives the best result for Movie Ontology 3 and Movie Ontology 4, $TSC_{Wu\&Palmer}$ gives the best result for Movie Ontology 5. As mentioned in 3.2.1.3, the difference between Movie Ontology 2 and Movie Ontology 3 is the representation of *rating*, *release date* and *runtime* features. When these features are represented by attributes as in Movie Ontology 3 rather than relations and corresponding concepts as in Movie Ontology 2, TSC_{Lin} and $SSIM_S$ give better results. Furthermore, it can be generalized from the results that, in order to calculate taxonomy similarity between two instances, taking the maximum similarity between the corresponding concepts of these instances, which is the $SSIM_S$ method, provides better results. For instance, MovieA is an action and thriller movie, MovieB is an horror and thriller movie. Taxonomy similarity between MovieA and MovieB is calculated regardless of considering *Action* and *Horror* concepts because both MovieA and MovieB are instances-of *Thriller* concept.

4.3.1.3 Experiment for $SSIM_{RS}$

In order to find the most appropriate value of $SSSIM_{RS}$ for each ontology, each of the values of k, O and $SSSIM_{RS}$ are used for evaluating *SEMCBF* while the values of other parameters remain constant. The determined values of PF, TSC and $SSIM_{TS}$ in previous experiments are used in this experiment. In order to obtain best results in MAE and F-Measure using Movie Ontology 2, different TSC and $SSIM_{TS}$ values are determined as appropriate in the previous experiment. Therefore, beginning from this experiment, in order to obtain best MAE and F-Measure, the experiments are separated into two branches. *SEMCBF* configurations that are used for MAE branch of this experiment are shown in Table 4.6. Similarly, *SEMCBF* configurations that are used for F-Measure branch of this experiment are shown in Table 4.7. It can be seen from the Tables 4.6 and 4.7, *SEMCBF* configurations of MAE and F-Measure branches differs from each other for Movie Ontology 2. Best results among all values of *k* are shown in Table 4.8. For each ontology, the best MAE and F-Measure values are in bold.

Table 4.6: *SEMCBF* Configurations of MAE Branch of the Experiment for $SSIM_{RS}$

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	TSC_{CM}	TSC_{CM}	TSC_{Lin}	TSC_{Lin}	$TSC_{Wu\&Palmer}$
$\boldsymbol{SSIM_{TS}}$	$SSIM_A$	$SSIM_A$	$SSIM_S$	$SSIM_S$	$SSIM_S$
$\boldsymbol{SSIM_{RS}}$	All Values	All Values	All Values	All Values	All Values
a	0.4	0.4	0.4	0.4	0.4
b	0.3	0.3	0.3	0.3	0.3
c	0.3	0.3	0.3	0.3	0.3
rd	1	1	1	1	1

Table 4.7: *SEMCBF* Configurations of F-Measure Branch of the Experiment for $SSIM_{RS}$

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	TSC_{CM}	TSC_{Mclean}	TSC_{Lin}	TSC_{Lin}	$TSC_{Wu\&Palmer}$
$\boldsymbol{SSIM_{TS}}$	$SSIM_A$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$
$\boldsymbol{SSIM_{RS}}$	All Values	All Values	All Values	All Values	All Values
a	0.4	0.4	0.4	0.4	0.4
b	0.3	0.3	0.3	0.3	0.3
c	0.3	0.3	0.3	0.3	0.3
rd	1	1	1	1	1

The results in MAE show that, best results are obtained by using $SSIM_S$ for $SSIM_{RS}$ in all of the ontologies. In addition, for Movie Ontology 1 and Movie Ontology 5, *SEMCBF* gives the best result in F-Measure using $SSIM_S$ for $SSIM_{RS}$. When the ontologies other than Movie Ontology 1 and Movie Ontology 5 are considered, *SEMCBF* gives the best result in F-Measure using $SSIM_2$ for $SSIM_{RS}$. From these results it can be generalized that, in order to

Table 4.8: Results of the Experiment for $SSIM_{RS}$

MAE						
	$\boldsymbol{SSIM_1}$	$\boldsymbol{SSIM_2}$	$\boldsymbol{SSIM_3}$	$\boldsymbol{SSIM_S}$	$\boldsymbol{SSIM_C}$	$\boldsymbol{SSIM_A}$
Movie Ontology 1	0.7850	0.7844	0.7856	**0.7827**	0.7897	0.7869
Movie Ontology 2	0.7787	0.7781	0.7797	**0.7764**	0.7827	0.7805
Movie Ontology 3	0.7659	0.7648	0.7726	**0.7643**	0.7708	0.7675
Movie Ontology 4	0.7662	0.7651	0.7730	**0.7646**	0.7712	0.7678
Movie Ontology 5	0.7658	0.7648	0.7718	**0.7641**	0.7711	0.7672
F-Measure						
	$\boldsymbol{SSIM_1}$	$\boldsymbol{SSIM_2}$	$\boldsymbol{SSIM_3}$	$\boldsymbol{SSIM_S}$	$\boldsymbol{SSIM_C}$	$\boldsymbol{SSIM_A}$
Movie Ontology 1	0.7440	0.7440	0.7432	**0.7445**	0.7420	0.7430
Movie Ontology 2	0.7465	**0.7466**	0.7456	0.7461	0.7447	0.7461
Movie Ontology 3	0.7493	**0.7496**	0.7477	0.7495	0.7481	0.7488
Movie Ontology 4	0.7491	**0.7495**	0.7476	0.7494	0.7480	0.7487
Movie Ontology 5	0.7493	0.7495	0.7477	**0.7498**	0.7483	0.7493

calculate relation similarity between two instances, taking the maximum similarity ($SSIM_S$) between the instances that are the feature-values of the corresponding movies provides better results. For example, ActorA, ActorB and ActorC are associated instances of instance MovieX with respect to the relation *hasCast* and direction *OUT*, ActorA, ActorD and ActorE are associated instances of instance MovieY with respect to the relation *hasCast* and direction *OUT*. The similarity for relation *hasCast* and direction *OUT* between MovieX and MovieY using $SSIM_S$ for $SSIM_{RS}$ is 1 because ActorA appears in both sets of associated instances.

4.3.1.4 Experiment for *a*, *b* and *c*

This experiment is conducted in order to determine the appropriate weight values of TS, RS and AS measures (a, b, c) that are mentioned in Section 3.2.2. In this experiment, *SEMCBF* is evaluated by setting the a, b and c parameters to interval [0,1] with one decimal place under the constraint of $a + b + c = 1$. The determined values of parameters in previous experiments are used and remaining parameter, rd, has a value of 1. As mentioned in Section 3.2.1.3, in Movie Ontology 1 and Movie Ontology 2, attributes are not used in order to represent features of the movies. In these ontologies AS between instances is always 0. However, in Equation 3.3, the weight of attribute similarity is used in order to calculate SS of items. Therefore, in this experiment, weight of AS is also considered for Movie Ontology 1 and Movie Ontology 2. Similar to the previous experiment, this experiment is separated into two branches in order to obtain best results in MAE and F-Measure separately. *SEMCBF* configurations that are used for MAE branch and F-Measure branch of this experiment are shown in Table 4.9 and Table 4.10 respectively. Best results among all values of *k*, *a*, *b* and *c* are shown in Table 4.11.

Table 4.9: *SEMCBF* Configurations of MAE Branch of the Experiment for *a*, *b* and *c*

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	TSC_{CM}	TSC_{CM}	TSC_{Lin}	TSC_{Lin}	$TSC_{Wu\&Palmer}$
$\boldsymbol{SSIM_{TS}}$	$SSIM_A$	$SSIM_A$	$SSIM_S$	$SSIM_S$	$SSIM_S$
$\boldsymbol{SSIM_{RS}}$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$
a	All Values	All Values	All Values	All Values	All Values
b	All Values	All Values	All Values	All Values	All Values
c	All Values	All Values	All Values	All Values	All Values
rd	1	1	1	1	1

Table 4.10: *SEMCBF* Configurations of F-Measure Branch of the Experiment for *a*, *b* and *c*

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	TSC_{CM}	TSC_{Mclean}	TSC_{Lin}	TSC_{Lin}	$TSC_{Wu\&Palmer}$
$\boldsymbol{SSIM_{TS}}$	$SSIM_A$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$
$\boldsymbol{SSIM_{RS}}$	$SSIM_S$	$SSIM_2$	$SSIM_2$	$SSIM_2$	$SSIM_S$
a	All Values	All Values	All Values	All Values	All Values
b	All Values	All Values	All Values	All Values	All Values
c	All Values	All Values	All Values	All Values	All Values
rd	1	1	1	1	1

Best results in MAE and F-Measure using Movie Ontology 3, Movie Ontology 4 and Movie Ontology 5 are obtained by using a configuration in which the values of *a* and *b* are the same and *c* is higher than these values. Therefore, TS and RS have equal effects and AS has the most effect on finding the similarities between items accurately. For Movie Ontology 1 and Movie Ontology 2, it can be generalized from the results that, RS has more effect on finding the similarities between items accurately than TS has.

Table 4.11: Results of the Experiment for *a*, *b* and *c*

MAE				
	a	***b***	***c***	**MAE**
Movie Ontology 1	0.1	0.4	0.5	0.7795
Movie Ontology 2	0.4	0.6	0	0.7726
Movie Ontology 3	0.1	0.1	0.8	0.7582
Movie Ontology 4	0.1	0.1	0.8	0.7574
Movie Ontology 5	0.1	0.1	0.8	0.7564
F-Measure				
	a	***b***	***c***	**F-Measure**
Movie Ontology 1	0.3	0.3	0.4	0.7455
Movie Ontology 2	0.4	0.5	0.1	0.7479
Movie Ontology 3	0.1	0.1	0.8	0.7522
Movie Ontology 4	0.1	0.1	0.8	0.7525
Movie Ontology 5	0.2	0.2	0.6	0.7525

4.3.1.5 Experiment for *rd*

As mentioned in Section 3.2.2, *RS* is a measure that is used for calculating SSs between items. In order to calculate *RS* between two instance, SS between associated instances are used. Consequently, *rd* parameter is used for preventing infinite cycles when calculating SSs between instances in ontology. In order to find the most appropriate value of *rd* for each ontology, each of the values of *k*, *O* and *rd* are used for evaluating *SEMCBF* while the values of other parameters remain constant. The determined values of the parameters in previous experiments are kept same in this experiment. Similar to the previous experiments, this experiment is separated into two branches in order to obtain best results in MAE and F-Measure separately. *SEMCBF* configurations that are used for MAE branch and F-Measure branch of this experiment are shown in Table 4.12 and Table 4.13 respectively. Best results in MAE are obtained by using $k = 15$ and best results in F-Measure are obtained by using $k = 10$ among all possible values of *k*. These results are shown in Table 4.14. For each ontology, the best MAE and F-Measure values are in bold.

Among five different ontologies used in this experiment, Movie Ontology 5 gives the best result in MAE, Movie Ontology 4 and Movie Ontology 5 give the best result in F-Measure. In order to analyze the effect of *rd* to the performance of *SEMCBF*, MAE and F-Measure obtained from *SEMCBF* using Movie Ontology 5 and different values of *rd* are illustrated in

Table 4.12: *SEMCBF* Configurations of MAE Branch of the Experiment for *rd*

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	TSC_{CM}	TSC_{CM}	TSC_{Lin}	TSC_{Lin}	$TSC_{Wu\&Palmer}$
$\boldsymbol{SSIM_{TS}}$	$SSIM_A$	$SSIM_A$	$SSIM_S$	$SSIM_S$	$SSIM_S$
$\boldsymbol{SSIM_{RS}}$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$
a	0.1	0.4	0.1	0.1	0.1
b	0.4	0.6	0.1	0.1	0.1
c	0.5	0	0.8	0.8	0.8
rd	All Values	All Values	All Values	All Values	All Values

Table 4.13: *SEMCBF* Configurations of F-Measure Branch of the Experiment for *rd*

O	**Movie Ontology 1**	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
k	All Values	All Values	All Values	All Values	All Values
PF	*pred*2	*pred*2	*pred*2	*pred*2	*pred*2
TSC	TSC_{CM}	TSC_{Mclean}	TSC_{Lin}	TSC_{Lin}	$TSC_{Wu\&Palmer}$
$\boldsymbol{SSIM_{TS}}$	$SSIM_A$	$SSIM_S$	$SSIM_S$	$SSIM_S$	$SSIM_S$
$\boldsymbol{SSIM_{RS}}$	$SSIM_S$	$SSIM_2$	$SSIM_2$	$SSIM_2$	$SSIM_S$
a	0.3	0.4	0.1	0.1	0.2
b	0.3	0.5	0.1	0.1	0.2
c	0.4	0.1	0.8	0.8	0.6
rd	All Values	All Values	All Values	All Values	All Values

Figure 4.2. In Figure 4.2(a), MAE of *SEMCBF* is obtained by using $k = 15$, in Figure 4.2(b), F-Measure of *SEMCBF* is obtained by using $k = 10$. As it can be seen from these figures, the performance of *SEMCBF* improves with the increase of the value of *rd* and remains almost constant after a value of *rd*. The reason is that higher values of *rd* hardly effect the performance of *SEMCBF* because of the nature of recursion.

4.3.2 Second Phase of Evaluation

After determining the appropriate values of *SEMCBF* parameters, the refinements of *SEMCBF*, which are mentioned in Section 3.3, are evaluated in the second phase of evaluation. At the end of the first phase of evaluation, Movie Ontology 5 is selected as the ontology that performed best among the other ontologies. Additionally, for the evaluation metric, F-Measure

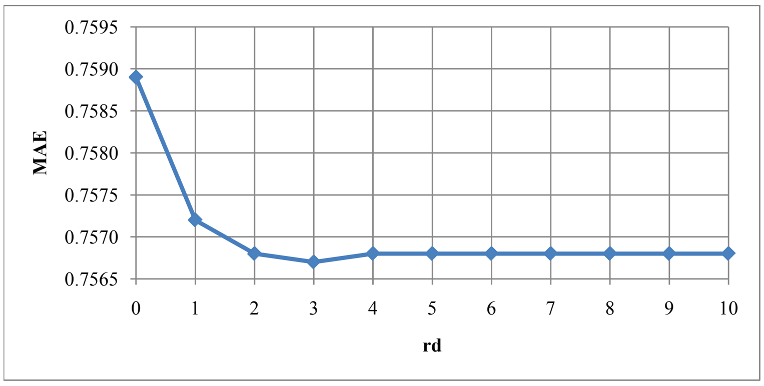
Table 4.14: Results of the Experiment for *rd*

		MAE (*k* = 15)				
		Movie Ontology 1	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
rd	0	0.8274	0.8274	0.7601	0.7594	0.7589
	1	0.7795	**0.7726**	0.7582	0.7574	0.7572
	2	0.7779	0.7751	**0.7578**	**0.757**	0.7568
	3	**0.7772**	0.773	0.7579	0.7571	**0.7567**
	4	0.7774	0.7737	0.7579	0.7571	0.7568
	5	0.7774	0.7737	0.7579	0.7571	0.7568
	6	0.7773	0.7736	0.7579	0.7571	0.7568
	7	0.7773	0.7737	0.7579	0.7571	0.7568
	8	0.7773	0.7737	0.7579	0.7571	0.7568
	9	0.7773	0.7737	0.7579	0.7571	0.7568
	10	0.7773	0.7738	0.7579	0.7571	0.7568
		F-Measure (*k* = 10)				
		Movie Ontology 1	**Movie Ontology 2**	**Movie Ontology 3**	**Movie Ontology 4**	**Movie Ontology 5**
rd	0	0.7234	0.7152	0.7504	0.7506	0.7510
	1	0.7450	0.7479	0.7522	**0.7525**	**0.7525**
	2	0.7455	0.7482	**0.7523**	0.7525	0.7523
	3	0.7456	**0.7486**	0.7523	0.7525	0.7524
	4	0.7456	0.7484	0.7523	0.7525	0.7524
	5	0.7456	0.7485	0.7523	0.7525	0.7523
	6	0.7456	0.7485	0.7523	0.7525	0.7523
	7	0.7456	0.7485	0.7523	0.7525	0.7523
	8	0.7456	0.7484	0.7523	0.7525	0.7523
	9	**0.7457**	0.7484	0.7523	0.7525	0.7523
	10	0.7457	0.7484	0.7523	0.7525	0.7523

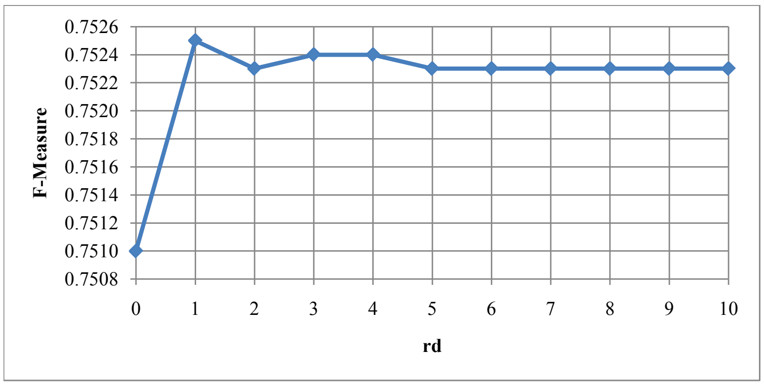
is used in this evaluation phase. Therefore, in this phase of evaluation, in order to evaluate the performance of *SEMCBF*, Movie Ontology 5 and the determined values of the parameters for F-Measure branch is used. The experiments are conducted in order to analyze the refinements in the following order:

1. Feature Selection

2. Feature Weighting



(a) MAE vs *rd* (k = 15)



(b) F-Measure vs *rd* (k = 10)

Figure 4.2: Results of the Experiment for *rd* using Movie Ontology 5

4.3.2.1 Experiment for Feature Selection

We apply a feature selection algorithm, *sequential forward selection (SFS)* [63], in order to find a subset of features that improves the F-Measure of *SEMCBF*. As mentioned in Section 3.3.1, by applying SFS, we can determine unnecessary attributes and relations in ontology. Therefore, these unnecessary attributes/relations can be removed from the recommendation process to improve the performance of *SEMCBF*. In Figure 4.3, horizontal axis shows the selected relation/attribute that represent a feature and vertical axis shows the performance of *SEMCBF* in F-Measure for each iteration. Table 4.15 shows the exact F-Measure values obtained from *SEMCBF* at each iteration.

As it can be seen from Table 4.15, the best F-Measure value is obtained at the end of 6th iteration. This result shows that excluding *Runtime* attribute, *hasColorValue* relation and *has-*

Table 4.15: Result of the Experiment for SFS

Iteration	Selected Relation/Attribute	F-Measure
1	Rating	0.7512
2	hasDirector	0.7518
3	hasLanguage	0.7520
4	ReleaseDate	0.7521
5	hasWriter	0.7519
6	hasCountry	**0.7529**
7	hasCast	0.7527
8	Runtime	0.7526
9	hasColorValue	0.7525

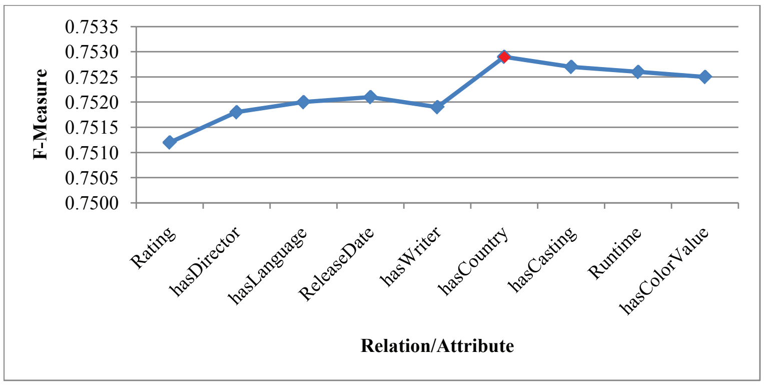


Figure 4.3: Selected Relation/Attribute vs. F-Measure

Cast relation from the Movie Ontology 5, in other words removing from semantic similarity calculation, improves the F-Measure of *SEMCBF*. The relations and attributes selected from the beginning to the 6th iteration are used in the later evaluations.

4.3.2.2 Experiment for Feature Weighting

This experiment is conducted in order to determine the appropriate values of the parameters that are mentioned in Section 3.3.2. These parameters are MFV, UWV and the algorithm used for clustering UWVs of users. At first, methods for MFV and UWV are determined. For this process, for each pair of methods for MFV and UWV, *SEMCBF* is evaluated using the corresponding pair and each of the values of k. Additionally, all users are grouped into only one cluster while analyzing methods for MFV and UWV. The values of other parameters, which are determined in the previous experiments, remain constant. The results of the experiment for *MFV* and *UWV* are shown in Table 4.16. For each combination of *MFV* and *UWV*, the best F-Measure value is shown in bold.

Table 4.16: Results of the Experiment for *MFV* and *UWV*

k	***MFV***	***UWV***	**F-Measure**		***k***	***MFV***	***UWV***	**F-Measure**
5	Default	Default	0.7471		5	Default	Stddev	0.7476
10	Default	Default	0.7521		10	Default	Stddev	**0.7525**
15	Default	Default	**0.7522**		15	Default	Stddev	0.7521
20	Default	Default	0.7507		20	Default	Stddev	0.7510
30	Default	Default	0.7491		30	Default	Stddev	0.7495
50	Default	Default	0.7450		50	Default	Stddev	0.7458
100	Default	Default	0.7374		100	Default	Stddev	0.7380
200	Default	Default	0.7312		200	Default	Stddev	0.7314
5	Idf	Default	0.7484		5	Idf	Stddev	0.7482
10	Idf	Default	**0.7530**		10	Idf	Stddev	**0.7530**
15	Idf	Default	0.7526		15	Idf	Stddev	0.7526
20	Idf	Default	0.7509		20	Idf	Stddev	0.7515
30	Idf	Default	0.7492		30	Idf	Stddev	0.7497
50	Idf	Default	0.7453		50	Idf	Stddev	0.7456
100	Idf	Default	0.7375		100	Idf	Stddev	0.7379
200	Idf	Default	0.7311		200	Idf	Stddev	0.7314
5	Ifw	Default	0.7481		5	Ifw	Stddev	0.7482
10	Ifw	Default	**0.7527**		10	Ifw	Stddev	**0.7528**
15	Ifw	Default	0.7515		15	Ifw	Stddev	0.7524
20	Ifw	Default	0.7503		20	Ifw	Stddev	0.7510
30	Ifw	Default	0.7481		30	Ifw	Stddev	0.7492
50	Ifw	Default	0.7444		50	Ifw	Stddev	0.7453
100	Ifw	Default	0.7365		100	Ifw	Stddev	0.7374
200	Ifw	Default	0.7302		200	Ifw	Stddev	0.7311

The results show that, for *MFV*, using *Idf* method rather than using *Default* or *Ifw* methods gives better results. Therefore, in order to generate *MFV* for each movie in the system, using *Idf* values of the feature-values are determined as appropriate. Similarly, for *UWV*, using *Stddev* method rather than using *Default* gives better or same results. It can be said that it gives better result using standard deviation of the values, which represent the preference of the user about feature-values of a feature, in *UPV* than using the biggest value of the values, which represent the preference of the user about feature-values of a feature, in *UPV* as the weight of the corresponding feature. As a result, *Idf* and *Stddev* methods are determined as appropriate for *MFV* and *UWV*, respectively.

After determining the appropriate methods for *MFV* and *UWV*, the next experiment is conducted in order to analyze the clustering algorithm that is applied to *UWV*s in the system. For this step, *SEMCBF* is evaluated using clustering algorithms, K-Means and Expectation Maximisation (EM), separately with each of the values of *k*. When using K-Means clustering algorithm, *number of clusters* parameter is set to 15. The determined values of other parameters remain constant. The results of the experiment for *UWV clustering* are shown in Table 4.17. For each clustering algorithm, the best F-Measure value is in bold.

Table 4.17: Results of the Experiment for *UWV* Clustering

k	**Algorithm**	**F-Measure**
5	EM	0.7487
10	EM	**0.7532**
15	EM	0.7525
20	EM	0.7515
30	EM	0.7498
50	EM	0.7462
100	EM	0.7382
200	EM	0.7316
5	K-Means	0.7484
10	K-Means	**0.7531**
15	K-Means	0.7526
20	K-Means	0.7515
30	K-Means	0.7499
50	K-Means	0.7462
100	K-Means	0.7382
200	K-Means	0.7316

It can be seen from the Table 4.17 that, the results of both EM and K-Means algorithms are nearly same. One of the possible reasons for these results is that these two clustering algorithms produce nearly same clusters. The other possible reason is that the *UWV*s of all users are very similar to each other. Thus, even the produced clusters for EM and K-Means algorithms are different, the *CWV*s of these clusters are very similar. Although the results of these two algorithms are nearly same, EM algorithm is determined for *UWV* clustering.

At the end of the second phase of the evaluation, the determined values of parameters of *SEMCBF* that gives the best result in F-Measure metric are shown in Table 4.18.

Table 4.18: Determined Values of Parameters

Parameter	**Determined Value(s)**
O	Movie Ontology 5
rd	1
a	0.2
b	0.2
c	0.6
TSC method	$TSC_{Wu\&Palmer}$
$SSIM_{TS}$ method	$SSIM_S$
$SSIM_{RS}$ method	$SSIM_S$
k	10
PF	$pred2$
Relations used	hasDirector, hasLanguage, hasWriter, hasCountry
Attributes used	Rating, Runtime
MFV method	Idf
UWV method	$Stddev$
Clustering Algorithm	EM

4.3.3 Third Phase of Evaluation

SEMCBF is used for enhancing user-item rating matrix in the proposed approach called *SEMCBCF*. Therefore, the performance of *SEMCBCF* is dependent to the performance of *SEMCBF*. Because of that, in the third phase of the evaluation, both the performance of *SEMCBF* and *SEMCBCF* are compared with some other approaches. Table 4.19 gives precision, recall and F-Measure metrics of *SEMCBF*, *SEMCBCF* and some approaches [7, 67, 1, 68, 11].

Table 4.19: Comparison of *SEMCBF* and *SEMCBCF* with Other Approaches

Approach	**Precision (%)**	**Recall (%)**	**F-Measure (%)**
MovieLens	66	74	69.8
MovieMagician Feature-Based	61	75	67.3
MovieMagician Clique-Based	74	73	73.5
MovieMagician Hybrid	73	56	63.4
OPENMORE	62	91.7	74.1
ReMovender	72	78	74.9
CBCF	60	95.2	73.6
SEMCBF	***63.7***	***92.2***	***75.3***
SEMCBCF	***63.8***	***93.1***	***75.7***

Movielens [7] uses collaborative filtering approach. MovieMagician [67] is a hybrid recommender that uses kinds, actors and directors as features of a movie. OPENMORE [1] is a content-based recommender based on fine-tuning the constructed user models. ReMovender [68] is a content-based recommender that is empowered by collaborative missing data prediction. Additionally, *CBCF* [11], which is a hybrid recommender that uses Naive Bayesian classifier as a content-based predictor to enhance user-item rating matrix, is implemented using the same dataset to make a fair comparison. Number of nearest neighbors are set to 30 as mentioned in [11] for collaborative filtering parts of both *SEMCBCF* and *CBCF*. It can be seen from the Table 4.19 that *SEMCBF* and *SEMCBCF* outperform these approaches according to F-Measure metric.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis presents an ontology-based hybrid approach, which uses both content-based and collaborative filtering, in order to overcome the sparsity and new item problems of collaborative filtering. The proposed approach (*SEMCBCF*) is based on content-boosted collaborative filtering presented in [11]. *SEMCBCF* first uses content-based filtering to enhance the user-item rating matrix, then performs collaborative filtering using this enhanced user-item rating matrix.

The contribution of our approach is that it uses content-based user models and semantic similarity measure on ontology-based metadata to calculate the similarities between items in content-based filtering. The content-based part of *SEMCBCF*, *SEMCBF*, utilizes content-based user models to obtain the feature-weights that represent the opinion of the users on the features. According to these user models, users are clustered and for each cluster, *SEMCBF* finds semantic similarities between items by using feature-weights of the corresponding cluster and ontology-based metadata. After finding the similarities between items for each cluster, user-item rating matrix is enhanced according to the similarities between items that are calculated for the active user's cluster and active user's explicit ratings (active user's collaborative-based model). At the last step, collaborative filtering is performed on the enhanced user-item rating matrix to compute predictions. Our hypothesis was that using semantic similarity measures rather than naive Bayesian classifier [41], which is used in [11], will improve the quality of recommendations.

In the evaluation phase, first, *SEMCBF* was fine-tuned by determining the values of its parameters, methods for creating user models and clustering algorithm. Additionally, a feature selection algorithm was applied for determining the relations and attributes that are redundant.

Then, using the determined values and features, *SEMCBF* and *SEMCBCF* were evaluated. The results showed that *SEMCBF* and *SEMCBCF* outperform content-boosted collaborative filtering presented in [11].

Among five different ontologies, Movie Ontology 5 gives the best result in the evaluation phase. The first reason Movie Ontology 5 gives the best result is that the numeric features are represented as attributes in Movie Ontology 5. Therefore, by considering the attribute similarity between movie instances, more accurate similarity values are found. The second reason is that the hierarchy of concepts, which represents the feature-values of the *genre* feature, is generated by the help of a group of Radio-Television and Cinema students. Using the domain knowledge to generate the hierarchy of concepts, taxonomy similarity between movie instances becomes more accurate.

The results obtained in evaluation phase show that the characteristics of the ontology, such as the hierarchy of concepts, representation of the features significantly effect the performance of *SEMCBF*. For further research, generating a more detailed ontology can be focused to improve the performance of *SEMCBF*.

SEMCBF consists of many parameters and these parameters have many possible values. In the evaluation phase, most of the *SEMCBF* parameters are evaluated individually. As a future work, the cross-dependencies between these parameters can be analyzed and evaluated. Additionally, the values and algorithms that are not evaluated in this thesis can be evaluated. For instance, in order to determine the feature-weights, some other user modeling algorithms can be used rather than the multiple criteria approach which is used in [64]. Also, some other alternatives for clustering and feature selection algorithm can be evaluated.

In addition, *SEMCBCF* can be applied to another domain such as music recommendation for further research. Using the features of music domain and the feature-values of songs, a music ontology can be generated. *SEMCBCF* can utilize the generated music ontology, the feature-values of items and the user-item rating matrix in order to generate recommendations.

REFERENCES

[1] O. Kirmemis, "Openmore: A content-based movie recommendation system." M.Sc. Thesis in Computer Engineering Department of Middle East Technical University, 2008.

[2] G. Adomavicius and A. Tuzhilin, "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*, vol. 17, pp. 734–749, june 2005.

[3] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*, pp. 1–35, Springer, 2011.

[4] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, pp. 143–177, January 2004.

[5] "Amazon.com." http://www.amazon.com/, last visited on August 2011.

[6] "Netflix." http://www.netflix.com/, last visited on August 2011.

[7] "Movielens." http://movielens.umn.edu/, last visited on August 2011.

[8] "Last.fm." http://www.last.fm/, last visited on August 2011.

[9] "Pandora radio." http://www.pandora.com/, last visited on August 2011.

[10] "Grooveshark." http://grooveshark.com/, last visited on August 2011.

[11] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, (Edmonton, Alberta), pp. 187–192, 2002.

[12] E. Vozalis and K. G. Margaritis, "Analysis of recommender systems' algorithms," in *Proceedings of the 6th Hellenic European Conference on Computer Mathematics & its Applications (HERCMA 2003)*, 2003.

[13] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–20, January 2009.

[14] M. Balabanović and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, pp. 66–72, March 1997.

[15] L. Candillier, K. Jack, F. Fessant, and F. Meyer, *State-of-the-Art Recommender Systems*, pp. 1–22. IGI Global, 2009.

[16] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, pp. 61–70, December 1992.

[17] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, pp. 56–58, March 1997.

[18] J. S. Breese, D. Heckerman, and C. M. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, (San Francisco), pp. 43–52, Morgan Kaufmann, 1998.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th International Conference on World Wide Web*, WWW ’01, (New York, NY, USA), pp. 285–295, ACM, 2001.

[20] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “Grouplens: an open architecture for collaborative filtering of netnews,” in *Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work*, CSCW ’94, (New York, NY, USA), pp. 175–186, ACM, 1994.

[21] U. Shardanand and P. Maes, “Social information filtering: algorithms for automating “word of mouth”,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI ’95, pp. 210–217, ACM Press/Addison-Wesley Publishing Co., 1995.

[22] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, SIGIR ’99, (New York, NY, USA), pp. 230–237, ACM, 1999.

[23] D. Almazro, G. Shahatah, L. Albdulkarim, M. Kherees, R. Martinez, and W. Nzoukou, “A survey paper on recommender systems,” *CoRR*, vol. abs/1006.5278, 2010.

[24] L. Ungar and D. Foster, “Clustering methods for collaborative filtering,” in *Proceedings of the Workshop on Recommendation Systems*, AAAI Press, Menlo Park California, 1998.

[25] S. H. S. Chee, J. Han, and K. Wang, “Rectree: An efficient collaborative filtering method,” in *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, DaWaK ’01, (London, UK), pp. 141–151, Springer-Verlag, 2001.

[26] K. Miyahara and M. J. Pazzani, “Collaborative filtering with the simple bayesian classifier,” in *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, PRICAI’00, (Berlin, Heidelberg), pp. 679–689, Springer-Verlag, 2000.

[27] X. Su and T. M. Khoshgoftaar, “Collaborative filtering for multi-class data using belief nets algorithms,” in *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI ’06, (Washington, DC, USA), pp. 497–504, IEEE Computer Society, 2006.

[28] T. Hofmann, “Latent semantic models for collaborative filtering,” *ACM Transactions on Information Systems (TOIS)*, vol. 22, pp. 89–115, January 2004.

[29] “Do you have any recommendations? an introduction to recommender systems.” http://www.cdlib.org/services/uxdesign/docs/2005/recSystemIntro_2005.pdf, last visited on August 2011.

[30] P. Melville and V. Sindhwani, “Recommender systems,” in *Encyclopedia of Machine Learning* (C. Sammut and G. I. Webb, eds.), pp. 829–838, Springer, 2010.

[31] R. V. Meteren and M. V. Someren, “Using content-based filtering for recommendation,” in *Proceedings of ECML/MLNET Workshop on Machine Learning and the New Information Age*, vol. 4203/2006, pp. 47–56, Citeseer, 2000.

[32] G. Salton, *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[33] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*, pp. 325–341, Springer-Verlag, 2007.

[34] B. Xu, M. Zhang, Z. Pan, and H. Yang, “Content-based recommendation in e-commerce,” in *Computational Science and Its Applications (ICCSA 2005)* (O. Gervasi, M. Gavrilova, V. Kumar, A. Laganà, H. Lee, Y. Mun, D. Taniar, and C. Tan, eds.), vol. 3481 of *Lecture Notes in Computer Science*, pp. 72–73, Springer Berlin / Heidelberg, 2005.

[35] R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *Proceedings of the 5th ACM Conference on Digital Libraries*, DL ’00, (New York, NY, USA), pp. 195–204, ACM, 2000.

[36] R. Burke, “Hybrid web recommender systems,” in *The adaptive web*, pp. 377–408, Berlin, Heidelberg: Springer-Verlag, 2007.

[37] M. J. Pazzani, “A framework for collaborative, content-based and demographic filtering,” *Artificial Intelligence Review*, vol. 13, pp. 393–408, December 1999.

[38] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, pp. 331–370, November 2002.

[39] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, “Combining content-based and collaborative filters in an online newspaper,” in *Proceedings of ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, August 1999.

[40] C. Basu, H. Hirsh, and W. Cohen, “Recommendation as classification: using social and content-based information in recommendation,” in *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI ’98/IAAI ’98, (Menlo Park, CA, USA), pp. 714–720, American Association for Artificial Intelligence, 1998.

[41] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[42] E. Peis, J. Morales-Del-Castillo, and J. Delgado-López, “Semantic recommender systems - analysis of the state of the topic,” *Hipertext.net*, vol. 6, pp. 1–5, 2008.

[43] “Ontology (information science).” http://en.wikipedia.org/wiki/Ontology_(information_science), last visited on August 2011.

[44] R.-Q. Wang and F.-S. Kong, “Semantic-enhanced personalized recommender system,” in *Proceedings of the 2007 International Conference on Machine Learning and Cybernetics*, vol. 7, pp. 4069–4074, 2007.

[45] I. Cantador and P. Castells, "Multilayered semantic social network modeling by ontology-based user profiles clustering: Application to collaborative filtering," in *Managing Knowledge in a World of Networks* (S. Staab and V. Svátek, eds.), vol. 4248 of *Lecture Notes in Computer Science*, pp. 334–349, Springer Berlin / Heidelberg, 2006.

[46] C.-N. Ziegler, G. Lausen, and S.-T. Lars, "Taxonomy-driven computation of product recommendations," in *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, CIKM '04, (New York, NY, USA), pp. 406–415, ACM, 2004.

[47] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, pp. 5–53, January 2004.

[48] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*, pp. 257–297, Springer, 2011.

[49] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *Journal of Machine Learning Research*, vol. 10, pp. 2935–2962, December 2009.

[50] "Movielens data sets." http://www.grouplens.org/mldata, last visited on August 2011.

[51] "The internet movie database(imdb)." http://www.imdb.com/, last visited on August 2011.

[52] A. Maedche and V. Zacharias, "Clustering ontology-based metadata in the semantic web," in *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '02, (London, UK, UK), pp. 348–360, Springer-Verlag, 2002.

[53] P. Lula and G. Paliwoda-Pekosz, "An ontology-based cluster analysis framework," in *Proceedings of the 7th International Semantic Web Conference (ISWC2008)*, October 2008.

[54] "The protégé ontology editor and knowledge acquisition system." http://protege.stanford.edu/, last visited on August 2011.

[55] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, (Stroudsburg, PA, USA), pp. 133–138, Association for Computational Linguistics, 1994.

[56] D. Lin, "An information-theoretic definition of similarity," in *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, (San Francisco, CA, USA), pp. 296–304, Morgan Kaufmann Publishers Inc., 1998.

[57] Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, pp. 871–882, 2003.

[58] N. Tintarev and J. Masthoff, "Similarity for news recommender systems," in *Proceedings of the AH'06 Workshop on Recommender Systems and Intelligent User Interfaces*, 2006.

[59] T. L. Bach and R. D. Kuntz, "Measuring similarity of elements in owl dl ontologies," in *Proceedings of the AAAI'05 Workshop on Contexts and Ontologies: Theory, Practice and Applications*, (Pittsburg, Pennsylvania, USA), pp. 96–99, 2005.

[60] O. Maimon, *Decomposition Methodology For Knowledge Discovery And Data Mining: Theory And Applications (Machine Perception and Artificial Intelligence)*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2005.

[61] P. Somol, J. Novovicova, and P. Pudil, *Efficient Feature Subset Selection and Subset Size Optimization*, pp. 75–97. InTech, 2010.

[62] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[63] A. Whitney, "A direct method of nonparametric measurement selection," *IEEE Transactions on Computers*, vol. C-20, no. 9, pp. 1100–1103, 1971.

[64] K. Chapphannarungsri and S. Maneeroj, "Combining multiple criteria and multidimension for movie recommender system," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Volume I, IMECS '09, March 18 - 20, 2009, Hong Kong*, Lecture Notes in Engineering and Computer Science, pp. 698–703, International Association of Engineers, Newswood Limited, 2009.

[65] O. Kirmemis and A. Birturk, "A content-based user model generation and optimization approach for movie recommendation," in *Proceedings of the AAAI'08 Workshop on Intelligent Techniques for Web Personalization & Recommender Systems*, (Chicago, Illinois, USA), pp. 78–88, 2008.

[66] "Weka 3 - data mining with open source machine learning software in java." http://www.cs.waikato.ac.nz/ml/weka/, last visited on August 2011.

[67] S. Grant and G. I. McCalla, "A hybrid approach to making recommendations and its application to the movie domain," in *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '01, (London, UK), pp. 257–266, Springer-Verlag, 2001.

[68] H. Karaman, "Content based movie recommendation system empowered by collaborative missing data prediction." M.Sc. Thesis in Computer Engineering Department of Middle East Technical University, 2010.

APPENDIX A

PUBLICATIONS

Ugur Ceylan and Aysenur Birturk, Content-boosted Collaborative Filtering Using Semantic Similarity Measure, 7th International Conference on Web Information Systems and Technologies (WEBIST'11), 6-9 May 2011, Noordwijkerhout, The Netherlands.

Ugur Ceylan and Aysenur Birturk, Combining Feature Weighting and Semantic Similarity Measure for a Hybrid Movie Recommender System, The fifth International Workshop on Social Network Mining and Analysis (SNAKDD 2011), In conjunction with 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-2011), 21-24 August 2011, San Diego, CA, USA.