# ENHANCING ACCURACY OF HYBRID RECOMMENDER SYSTEMS THROUGH ADAPTING THE DOMAIN TRENDS

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

BY

FATİH AKSEL

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

AUGUST 2010

Approval of the thesis:

**ENHANCING ACCURACY OF HYBRID RECOMMENDER SYSTEMS THROUGH ADAPTING THE DOMAIN TRENDS**

submitted by **FATİH AKSEL** 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 Department, METU** _____

**Examining Committee Members:**

Assoc. Prof. Dr. Ferda Nur Alpaslan
Computer Engineering, METU _____

Dr. Ayşenur Birtürk
Computer Engineering, METU _____

Asst. Prof. Dr. Pınar Şenkul
Computer Engineering, METU _____

Asst. Prof. Dr. Tolga Can
Computer Engineering, METU _____

Asst. Prof. Dr. Tuğba Taşkaya Temizel
Informatics Institute, METU _____

**Date:** _____

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last Name: FATİH AKSEL

Signature :

# ABSTRACT

ENHANCING ACCURACY OF HYBRID RECOMMENDER SYSTEMS THROUGH ADAPTING THE DOMAIN TRENDS

Aksel, Fatih

M.S., Department of Computer Engineering

Supervisor : Dr. Ayşenur Birtürk

August 2010, 75 pages

Traditional hybrid recommender systems typically follow a manually created fixed prediction strategy in their decision making process. Experts usually design these static strategies as fixed combinations of different techniques. However, people's tastes and desires are temporary and they gradually evolve. Moreover, each domain has unique characteristics, trends and unique user interests. Recent research has mostly focused on static hybridization schemes which do not change at runtime. In this thesis work, we describe an adaptive hybrid recommender system, called AdaRec that modifies its attached prediction strategy at runtime according to the performance of prediction techniques (user feedbacks). Our approach to this problem is to use adaptive prediction strategies. Experiment results with datasets show that our system outperforms naive hybrid recommender.

Keywords: Information filtering, Hybrid Recommender Systems, Adaptive Recommender Systems, Machine Learning, Switching Hybridization, Decision Tree Induction

# ÖZ

ALANDAKİ DEĞİŞİMLERE UYUM SAĞLAYARAK KARMA TAVSİYE SISTEMLERİNİN DOĞRULUĞUNUN ARTTIRILMASI

Aksel, Fatih
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Dr. Ayşenur Birtürk

Ağustos 2010, 75 sayfa

Geleneksel karma tavsiye sistemleri, karar verme süreçlerinde genel olarak elle oluşturulmuş, sabit tahmin stratejilerini uygularlar. Alan uzmanları bu statik tahmin stratejilerini farklı tekniklerin sabit birleşimiyle tasarlar. Ancak, insanların zevkleri ve ilgi alanları geçiçidir ve zamanla evrilir. Üstelik, her alanın kendine özgü karakteristiği, eğilimleri ve kullanıcı davranışları vardır. Son yıllarda yapılan araştırmalar daha çok, çalışma zamanında değişmeyen, statik karmalaştırma yöntemleri üzerine yoğunlaşmıştır. Bu tez çalışmasında, AdaRec olarak adlandırdığımız, kendi tahmin stratejisini çalışma zamanında tahmin tekniklerinin performansına göre değiştirebilen, 'uyarlanabilir karma tavsiye sistemini' anlattık. Bu probleme yönelik geliştirdiğimiz çözüm yaklaşımımızda, uyum sağlayabilen tahmin stratejileri kullandık. Veri kümeleri üzerine yaptığımız deneyler önerdiğimiz sistemin, geleneksel karma sistemlerden daha iyi çalıştığını göstermektedir.

Anahtar Kelimeler: Bilgi Filtreleme, Karma Tavsiye Sistemleri, Uyarlamalı Tavsiye Sistemleri, Otomatik Öğrenme, Anahtarlamalı Karma Sistemler, Karar Ağacı Oluşturma

*I would like to express my foremost and sincere gratitude to my family for their Love and Support throughout my life.*
*To Them I dedicate this Thesis.*

# ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Ayşenur Birtürk. I will always be grateful for her supervision, encouragement and support at all levels. I am also grateful to Asst. Prof. Dr. Pınar Şenkul, for her helpful comments and suggestions about this thesis work.

My deepest gratitude goes to my family for their unflagging love and support throughout my life.

I thank to all my colleagues for their effort and comments in conducting the research work in this thesis. I want to also thank to Mustafa Azak and Dr. Ruken Çakıcı, for taking the time to talk with me and constructive discussions at the formative, early stages of my thesis.

Additionally, I am grateful to TÜBİTAK-ULAKBİM for their encouragement, support and vision that sparked the writing of this thesis.

I would like to thank Mark van Setten and the creators of the Duine Framework for producing a high quality piece of software and making it open-source.

I would also like to thank my fiancee Hacer Yılmaz, for her unwavering support and fun-loving spirit.

Last, but not least, many thanks to anonymous reviewers of eChallenges 2009, ACM RecSys'10 Conferences and PL'10, HaCDAIS'10 and PRSAT'10 workshops.

# TABLE OF CONTENTS

# LIST OF TABLES

TABLES

# LIST OF FIGURES

FIGURES

# CHAPTER 1

# INTRODUCTION

With the rapid advances of the recommendation systems technology, a number of recommender engines have been developed for various application domains. *Content-based* and *collaborative filtering* are the two main paradigms that have come to dominate the current recommender systems. Content-based recommender system uses the descriptions about the content of the items (such as meta-data of the item), whereas collaborative filtering system tries to identify users whose tastes are similar and recommends items they have liked. The popular Amazon.com[1] item-to-item system [26] is one of the well-known recommender system that uses collaborative filtering techniques. NewsWeeder [37] and InfoFinder [36] are the pure content-based recommender systems that analyze the content of items, in their recommendation process.

In the literature other noteworthy paradigms have been studied [13]. Some of these as follows; *demographic recommenders*, which categorize the users based on their personal attributes and making recommendations according to the demographic classes; *knowledge-based systems*, which generates recommendations based on the knowledge about the item and the user; *hybrid recommender systems*, try to exploit the advantages of different recommendation techniques by combining them with a predefined hybridization scheme [42, 13, 2].

Previous research in this area, has shown that these techniques suffer from various potential problems-such as, sparsity, reduced coverage, scalability, and cold-start problems [2, 13, 73]. For example; collaborative filtering techniques depend on historical ratings of across users that have the drawback, called cold start problem- system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. The technique tends

[1] `http://www.amazon.com`

to offer poor results when there are not enough user ratings. Content-based techniques can overcome the cold start problem, because new items can be recommended based on item features about the content of the item compared with existing items. Unfortunately, content-based approaches require additional information about the content of item, which may be hard to extract (such as, movies, music, restaurants). Every recommendation approach has its own strengths and weaknesses. Hybrid recommender systems have been proposed to gain better results with fewer drawbacks.

Most of the recommender system implementations focuses on hybrid systems that use mixture of recommendation approaches [13]. In this way the known restrictions and drawbacks of content-based and collaborative filtering systems are minimized. Previous research on hybrid recommender systems has mostly focused on static hybridization approaches (strategy) that do not change their hybridization behavior at runtime. Fixed strategy may be suboptimal for dynamic domains&user behaviors. Moreover they are unable to adapt to domain drifts. Since people's tastes and desires are transient and subject to change, a good recommender engine should deal with changing consumer preferences.

In this thesis, we describe an adaptive hybrid recommender system that modifies its switching strategy according to the performance of prediction techniques. Our hybrid recommender approach uses adaptive *prediction strategies* that determine which *prediction techniques* (algorithms) should be used at the moment an actual prediction is required.

We apply an instance based learning scheme, that learns through user feedbacks in order to solve the problem of building adaptive hybrid recommendation systems. Hybrid recommendation systems combine multiple algorithms (prediction techniques) and define a switching behavior among these techniques. The main idea of our system is to re-design the hybrid system's switching behavior according to the performance results of the prediction algorithms. We calculate the performance results of the algorithms by using the user feedbacks. We introduce the novel idea of analyzing the prediction performance results and using them as run-time metrics. This method allows us to use characteristics of the input user/item when determining how to combine the prediction techniques at run-time. The run-time metrics are properties of the input user/item that are related to the nature of the application domain. For example, the number of items or item's similar user count that the input user has previously rated may indicate how well a collaborative filtering engine will perform. Learning algorithms

analyze performance results of previous predictions and then re-design the combination of prediction techniques.

The rest of the thesis is structured as follows.

**Chapter 2 - Background and Related Work**, gives an overview of recommender systems in literature. It introduces the main used paradigms and techniques of recommender systems, and identifies their approaches in detail. Later on other relevant studies in recommender systems and machine learning area are described. It also describes current recommendation systems, their approaches and structures.

**Chapter 3 - An Adaptive Recommender System**, after the introductory chapters have been explained, we will explain the the core concepts of our proposed hybrid recommender in more detail. The methodology and the framework that are used to develop the system is presented and discussed. The system architecture and components are presented and described in detail.

**Chapter 4 - Experiments & Evaluation**, gives an overview of; experimental setup, analysis of used datasets and evaluation metric. This chapter provides a summary of testing methods of the proposed adaptive system and presents experimental results. Also in this chapter the results of the experiments are discussed and the adaptive system is compared to the original Duine Framework.

**Chapter 5 - Conclusion & Future Work**, discusses the concluding remarks and explains the future work.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

The previous chapter provided the background and the problem definition. In this chapter, an overview of recommender systems and the recommendation algorithms will be presented. The aim of this chapter is to provide the relevant literature in the recommender systems field. It also includes the general concepts of the related machine learning literature, adaptive recommender systems and discusses some of the important recommendation systems framework examples.

## 2.1 Recommender System Types and Techniques

In the 'information age', one of the major problem is to deal with more information than we are able to process to make sensible decisions. We are bombarded with information whether or not we actively look for it. As the world moves into a new era of globalization an increasing number of people are connecting to the Internet. With the excessive use of the Internet users have to deal with a vast variety of information. A rapidly increasing rate of new information being produced The mass of content available has negative impact over its effective use. Recommender Systems (RSs) are designed to help individuals to deal with this *information overload* problem and enable them to make evaluative decisions [2].

The basic purpose of a RS is to recommend information items (restaurants, movies, films, news, songs, images, jokes, books, web pages, etc.) that will be of interest to a specific user. Recommender systems have been widely used both in academia and industry so far and many different recommendation approaches have been applied in order to make more accurate and more efficient recommender systems. Each approach has its own advantages and limitations.

Since RS are normally grounded to solve real world problems, the field is both exciting and rewarding to industry and academic society alike.

Based on their recommendation techniques, the RS systems are basically classified into the three basic categories, based on how recommendations are made [2].

- **Content-Based systems,** recommend an item to an user based on the description of the item's content and the user's preference.
- **Collaborative systems,** recommend items based on the the similarities of users or items.
- **Hybrid approaches,** combine collaborative and content-based methods.

RSs have been used in various different fields. These contexts range from different areas, such as web stores, online communities, music players, personalized media streams, social webs etc [57]. Broadly speaking it can be said that a RS can be used in every context where a user is to choose among several different options. The system assists the user to select one good-fitting item.

In the beginning of 1990s, Tapestry [23], is one of the first collaborative filtering-based systems, which was developed at the Research Center of Xerox. This system allowed users to annotate e-mail messages so that others could find documents based on previous comments. It was the first usage of combining human feedbacks with automated filtering which all of the systems users benefit from. Similar concepts and principles were successfully applied to the Internet discussion forums and movie filtering systems.

The initial success of RS leads the increase of e-commerce businesses that implement them. As stated by Resnick and Varian [57], e-commerce was also the one of the first application domains of RS. These sites were trying to offer products to their customers that might be interested in. The principle behind this approach was the abundance of products that made user choices and navigation very difficult. This generalizes information overload problem. Many systems that provides recommendations for e-commerce applications have been introduced in the literature [31, 57].

Recommender systems are not limited to offering what users can buy. Other applications for RS have been identified. One of them is posed by Ansari [3] that the proposed system

helps users for searching the Internet. In order to provide better results, search engines can be customized for each user. RS automate personalization on the Web by enabling individual personalization for each user. In generally with the power of RS the customization also can be applied to other types of websites because they help the site adapt itself to each customer. The evolution to a *semantic web* was seen as a method to enhance the way users browse the Internet [43]. Also the results of the search engines will not be limited to the keyword queries entered. However, as stated by Middleton [43], the real semantic web applications are still in their infancy as they dependent on the metadata that authors attach to their websites. The transition to the semantic web is something that will take a long time considering the amount of information that is available at the moment online and that needs to be transformed so that ontologies can be used to classify it.

There have been many collaborative systems developed in the academic society and many researches have been done in recommender systems with the various different fields. Recommender systems have been researched in the context of statistics, machine learning, distributed and mobile systems, social network analysis, agent-based artificial societies, human-computer interaction, computational trust and etc [73, 13, 61, 72].

In the content-based system, items are recommended to a user based upon description of the items and user's interest. Collaborative filtering recommenders use social knowledge, typically ratings of items by a community of users, to generate recommendations. Although hybrid recommender systems has focused into combining these two paradigms, it is useful to examine them separately to understand the fundamental concepts in their approach.

### 2.1.1 Content-Based Systems

Content-Based methods semantically analyse the descriptions of items. These representations of the items in the user profile are subsequently compared to possibly recommended items. If they matched with each other, the item is a candidate that would be recommended. Content-based recommender systems analyse item descriptions by using a similarity measure between the description of the item and user's preferences [2].

In content-based recommendation methods, system selects items based on the correlation between the content of the item (such as metadata) and the user preferences. The content-

based methods provides predictions $P(u, i)$ for an user $u$ and for an item $i$ based on the ratings $R(u, i')$ given by the same user $u$ to different items $i' \in Items$ that share *similar* content with to item $i$ [2]. For example, in a news recommendation application, in order to recommend news to user $u$, the content-based recommender system tries to understand user preferences by analyzing commonalities among the content of the news user $u$ has rated in the past (specific titles, authors, news classifications etc.). Then, the filtered news that have a high degree of similarity to whatever the customer's preferences are would get recommended.

More formally, let $Content(i)$ be the set of attributes characterizing item $i$. It is usually computed by extracting a set of features (meta-data) from item $i$ (its content) and is used to determine appropriateness of the item for recommendation purposes [1, 2].

Moreover, a knowledge base, also called user profile, for a user $u$ should be defined. Majority of the the content-based systems cope with recommending text-based items. It would be better to define the user profiles in terms of weights of important *keywords*. Also it can be explained such a way that, $ContentBasedProfile(u)$ for user $u$ can be defined as a vector of weights $(w_{u1}, \ldots, w_{uk})$, where each weight $w_{ui}$ denotes the importance of keyword $k_i$ to user $u$ and can also be specified using various information retrieval metrics [1].

Rating function $R(u, i)$ that is used by most of the content-based recommendation techniques, generally defined as;

$$R(u, i) = score(ContentBasedProfile(u), Content(i))$$

In the literature, one of the earliest content-based filters developed by H.P. Luhn at IBM in the 1950's. Luhn proposed a system that can be seen as a book recommendation system, which uses individual user profiles to produce reading recommendations by using an exact match system [8]. This system, which was designed for use in a library, uses the user's actual reading habits to update the user's profile. From that limited computations (and mostly manual) content-based filters are almost limited to text-based applications, since in majority of other areas of computers are not yet have such an efficient content analyzing process. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of several text-based applications, many current content-based systems focus on recommending items containing textual information,

such as documents, Web sites (URLs), and Usenet news messages.

Search engines were the one of the earliest and simplest content-based systems. These systems retrieve the list of web-sites or documents with respect to the keyword queries [39]. According to Oard' study [51], only four over 60 systems that were in operation in 1969, had implemented automatic profile updating, other systems builded upon on the users manually updating query terms. Therefore these systems use primitive approaches and usually they are not suitable for individual users. They return the same list to any user who uses the same terms in a search, and in this sense they really don't fit customized RS definitions.

In addition to the traditional information retrieval techniques, additional research (and computing power) brought new concepts. Various machine learning techniques can be applied for content-based recommendation methods; such as; Decision Trees [8], Bayesian classifiers [51], clustering [39], and artificial neural networks.

Content-based filtering techniques suffer from two major problems [2]. The first problem is about the contents of the items, some items suffer from a lack of easy-extractable content. In the literature, as mentioned at the above section, majority of the implemented content-based systems developed upon the text-based contents and other types of contents such as musics, movies, television programmes, hotels, have not been used very often. The second problem is about the content-based approach, since these systems are designed to recommend items that are similar to those already rated by the user, they are not suitable to recommend interesting items that some how the user may miss [5]. Moreover, content based recommender systems suffer from the cold-start problem since they require a sufficient number of items to be rated in order to construct a user profile. Therefore, for the new user the accuracy of recommendations are low.

### 2.1.2 Collaborative Systems

Collaborative Filtering (CF) systems on the other hand, recommend one certain item by correlating the user's profile with profiles of other users. Then the system recommends items similar users liked before. Depending on whether an item can be used several times, an item will also be recommended several times [2].

Collaborative filtering methods categorize information based on the user's conception of the

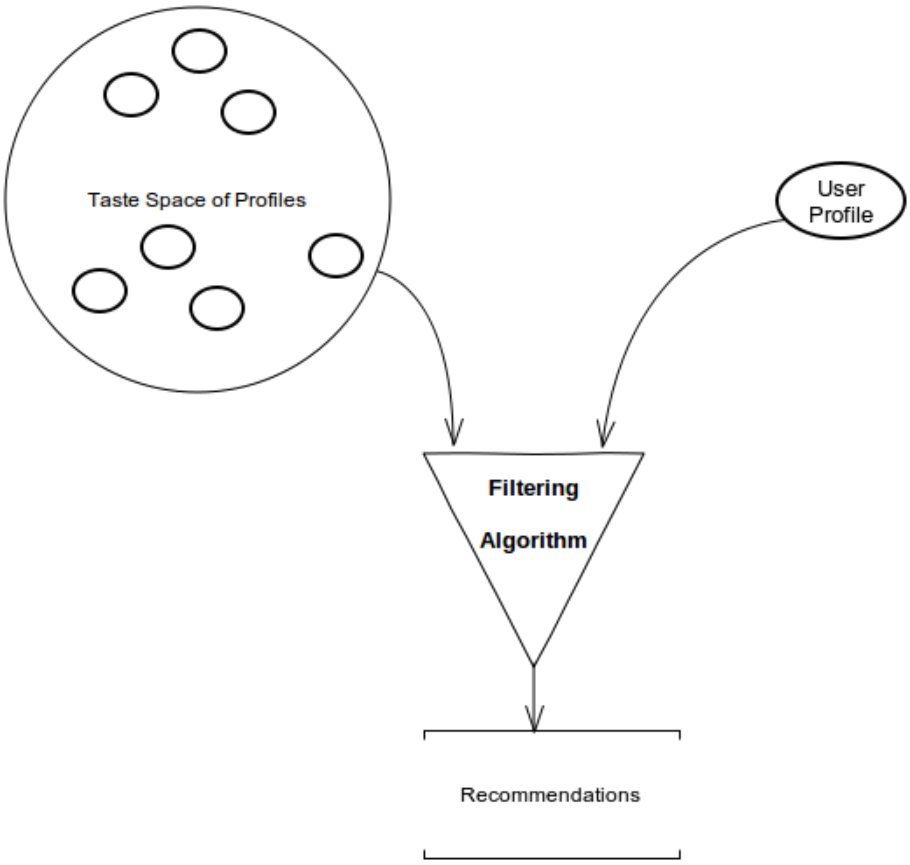


Figure 2.1: An abstract view of collaborative filtering process

data instead of the data itself. Therefore in the CF the data is filtered by using evaluation instead of analysis. Moreover, collaborative filtering methods stresses the tastes of user communities by calculating the recommendations by analyzing the opinions of the current user and community.

Figure 2.1 shows an abstract view of the collaborative filtering process that all users contribute to the taste of the community based on their preferences. Recommendations for the current user are produced by analyzing the user's ratings with community ratings. By using the collaborative filtering approach, a community is formed with the combination of the similar users.

Collaborative based recommender systems, generate the predictions for a particular user based on the profiles of the users especially who have previously rated the same item. More pre-

cisely, the rating $R(u,i)$ of item $i$ for user $u$ is estimated based on the ratings $R(u',i)$ assigned to the same item $i$ by those users $u'$ who are *similar* to user $u$ [1, 2].

To date, several collaborative systems have been developed in the industry, since the outbreak of the early commercial collaborative systems. Some of these systems are; Video Recommender [29], GroupLens [56, 35], MovieLens [47] and Ringo [63], that gain popularity among the researchers. These systems use collaborative filtering algorithms for their recommendation process. Other well-kown examples of collaborative recommender systems include MovieCritic [65] that generate online movie recommendations, the popular book recommendation system from Amazon.com [40], the PHOAKS [66] system that assits users to find relevant information on the Web, and the Jester [24] system that recommends jokes.

In the literature Collaborative Filtering systems are divided in to two sub areas; these are *memory-based(user-based)* and *model-based(heuristic-based or item-based)* [11]. At the early stages of the collaborative filtering research, researchers used a memory-based approach that makes rating predictions based on the collection of previously rated items by the users [57]. Later on, due to some shortcomings and limitations with the memory-based approach, researchers developed model-based content filtering systems that use the collection of ratings to learn a model [11] instead of analyzing users past behaviors in the memory. The learned model then used to make predictions. Although the model-based approach deals with some of the limitations related to memory-based filtering, this approach also has its own drawbacks.

#### 2.1.2.1 Memory-based collaborative filtering

Memory-based algorithms [57, 63, 11] use entire collection of user rating data to compute similarity between users or items. According to the definition, the value of the unknown rating $r_{u,i}$ for user $u$ and item $i$ is usually computed as an aggregate of the ratings of some other (such as the $N$ most similar ) users for the same item $i$.

*Similarity measure*, uses the users previous ratings and calculates the similarity between them. The similarity measure between the users $u$ and $u'$, $sim(u,u')$, determines the distance between users $u$ and $u'$ and is used as a weight to ratings $r_{u',i}$. The more similar users $u$ and $u'$ are, the more weight rating $r_{u',i}$. Various approaches have been used to compute similarity measure $sim(u,u')$ between users in collaborative recommender systems. In most of these approaches,

$sim(u, u')$ is based on the ratings of items that both users $u$ and $u'$ have rated. The two most popular approaches are the correlation-based and the cosine-based [58, 57, 1]. These two equations are shown below.

$$sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r_x})(r_{y,s} - \bar{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r_x})^2 \sum_{s \in S_{xy}} (r_{y,s} - \bar{r_y})^2}}$$

$$sim(x, y) = cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \|_2 \times \| \vec{y} \|_2} = \frac{\sum_{s \in S_{xy}} r_{x,s}\, r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}}$$

where $r_{x,s}$ and $r_{y,s}$ are the ratings of item $s$ assigned by users $x$ and $y$ respectively, $S_{xy} = \{s \in Items | r_{x,s} \neq \varnothing \wedge r_{y,s} \neq \varnothing\}$ is the set of all items co-rated by both customers $x$ and $y$, and $x \cdot y$ denotes the dot-product between the vectors $x$ and $y$. In the equation, $r_{i,j} = \varnothing$ indicates that item $j$ has not been rated by user $i$.

In the literature many different memory-based collaborative filtering approaches have been developed so far [52]. The Tapestry system explained in the Section 2.1, GroupLens [35] and Ringo [63] projects which were developed by different research groups, were the first content filtering algorithms to automate prediction process.

Since memory-based approaches are relatively simple, easy to implement on a conceptual level and also these systems have no expensive model-building development difficulty, memory-based collaborative filtering methods began to gain popularity. Although these systems are sufficient enough to solve many real-world problems, they do have other limitations [5, 38, 46].

- **Sparsity:** Memory-based filtering systems are usually suitable for large datasets. Many real-world systems have large user and item database and available data are very insufficient (even active users rarely rate the items) to identity similar users. Accordingly, a memory-based CF system may be unable to make any item recommendation for a particular user. As a result, the similarity between two users cannot be defined, so the recommendation accuracy tends to be poor.

- **Learning:** In these systems no information is inferred from the user profile because the lack of explicit model. So no general insight is gained from the users profile.
- **Scalability:** The algorithms used by most memory-based CF systems require computations on the user-item matrix. This situation causes to computation times to grow according to the number of users and items. Because of this, a typical memory-based CF system with millions of users and items will suffer from serious scalability problems.

The problems of memory-based content filtering systems, issue have led to the exploration of an alternative model-based filtering approach.

#### 2.1.2.2 Model-based collaborative filtering

In contrast to memory-based methods, model-based collaborative filtering methods builds a model based on the user preferences [11, 10]. By using the compiled model, some of the limitations related to memory-based CF might be solved. This can be done by first building a descriptive model of users, items and ratings based on the complete data set. The model of the system should be built in an offline manner due to the models possible expensive compilation process. The recommender system then consult the model for its recommendations. Therefore, if we compare the the memory-based approach with the model-based one, it can be inferred for the memory-based algorithms that they apply *lazy learning* methods, since they do not compile a model and perform the heuristic computations at the time a recommendation required.

Breese [11] presented a model-based recommendation techniques, which was based on a probabilistic approach to collaborative filtering. In the system unknown ratings are calculated as follows;

$$r_{u,i} = E(r_{u,i}) = \sum_{x=0}^{n} x \times Pr(r_{u,i} = x | r_{u,i'}, i' \in S_u)$$

and it is assumed that rating values are integers between 0 and $n$, and the probability expression is the probability that the user $u$ will give a particular rating to the item $i$ given the previous ratings of items rated by the user $u$.

*Bayesian networks* and *Bayesian clustering* [27] are the two main probabilistic models that are evaluated at the early research on model-based collaborative filtering approaches. In the Bayesian network model, each node in the network corresponds to an item in the data set. The state of each node corresponds to the possible rating values for each item. Both the structure of the network, which encodes the dependencies between items, and the conditional probabilities, are learned from the data set. In the Bayesian clustering model, on the other hand, users with similar preferences are clustered together into classes. Given the user's class membership, the ratings are assumed to be independent. The number of classes and the model parameters are learned from the data set.

To date many different model-based collaborative recommendation approaches have been developed so far. Some of these are; a statistical model for collaborative filtering [67], a Bayesian model [14], a probabilistic relational model [22], and a linear regression [22].

Another approach in collaborative filtering research combines both model-based approaches and memory-based [52, 71]. Some of these approaches include model-based recommendations into the final prediction, and does not require clustering of the data set [68].

Model-based filtering approaches offer many advantages over memory-based filtering [11].

- Compiled models for model-based approach are normally require less memory than for storing the whole database.
- The model-based approach may offer added values beyond its predictive capabilities, by highlighting certain correlations in the data.
- Once the model is builded, predictions can be generated quickly by consulting to the model. However, it should be mentioned that the time complexity to compile the data into a model may be an expensive process, and moreover adding one new data point may require a full recompilation of the model.

The resulting model of model-based CF systems is usually very small, fast and essentially as accurate as memory-based methods [11]. Model-based methods may prove practical for environments in which user preferences and behaviors change slowly. Model-based methods, however, are not suitable for environments in which user preference models must be updated rapidly or frequently.

### 2.1.3 Hybrid Systems

In the recent years hybrid recommendations systems are developed to stand against the weakness of pure collaborative and pure content-based methods. A number of different recommendation systems both in academia and in industry use a hybrid approach by combining collaborative and content-based techniques [13]. This helps to avoid certain limitations of content-based and collaborative filtering systems [2]. Table 2.1 presents the different types of hybridization methodologies described by Burke [13, 12].

Table 2.1: Hybridization Techniques.

| **Method** | **Explanation** |
|---|---|
| Weighted | Scores of various recommendation techniques are grouped to produce a single recommendation. |
| Switching | The recommender system switches between several techniques, depending on the situation, to produce the recommendation. |
| Mixed | Several different techniques are used at the same time. |
| Feature Combination | Features from data sources of different techniques are combined and used as an input to one single recommendation technique. |
| Cascade | The recommender system uses one technique to generate a recommendation, and a second technique to break any ties. |
| Feature Augmentation | The recommender system uses one technique to generate an output, which in turn is used as an input to a second recommendation technique. |
| Meta-level | The recommender system uses one technique to generate a model, which in turn is used as an input to a second recommendation technique. |

There have been many hybrid approaches proposed in the literature. The Tapestry system, which is explained in the Section 2.1, Fab Recommender [5] are the earliest examples of the hybrid systems. However, these systems do not really use the full hybrid approaches.

The Research Assistant Agent Project (RAAP) [18] system can be considered as one of the first true hybrid recommendation system. RAAP system is developed to support collaborative research by classifying domain specific information retrieved from the Web, and recommend the gathered 'nugget of knowledge' called *bookmarks* to other researchers with similar research interests. EntreeC, which was proposed by Burke [12], is a cascade hybrid recom-

mender system that recommends restaurants to users using both content based filtering and social filtering as prediction techniques.

To date a number of hybrid recommender systems have been proposed that they combine individual systems to avoid certain aforementioned limitations of these systems [13]. This helps to avoid certain drawbacks of content-based and collaborative filtering systems. Previous research on hybrid recommender system has mostly focused on static hybridization approaches (strategy) that do not change their hybridization behavior at runtime. Fixed strategy may be suboptimal for dynamic domains and user behaviors. Moreover they are unable to adapt to domain drifts. Since people's tastes and desires are transient and subject to change, a good recommender engine should deal with changing consumer preferences.

## 2.2 Learning in Recommender Systems

After presenting background information about recommender systems and their approaches, in this section we introduce the machine-learning and profiling techniques commonly employed by recommender systems. We also present and describe the different learning techniques.

Many techniques have been published in the user-modeling [31] and machine-learning [70] literature. The techniques described here are the subset commonly used by the recommender systems community.

Data mining methods use *Machine Learning (ML)* techniques as their technical platform. ML algorithm is used to extract information from the raw data in databases-information that is expressed in a comprehensible form and can be used for a variety of purposes [70].

ML techniques have been used for generating individual user models according to the users interaction with a system and clustering the users into communities with common tastes. The model generation process is essential in order to have a useful and usable system that can modify its behavior over time and for different users.

In the recommender systems, the tasks of learning user profiles is often done by using machine learning techniques. A *classifier* [45] is usually applied to compile the user profile. Classifiers are general computational models for assigning a category to a given input.

*Classification* is a data mining task which labels or categorizes a set of cases in a *training set* into different classes according to a classification model [70]. For this task, a training (model) set, a set of cases whose class labels are known, is first analyzed and a classification model is constructed by a classifier, based on the features available in the data of the training set. Such a classification model is then used to categorize a score set (a set of cases whose class labels are unknown) [15]. Also the changes and drifts that might occur in the application domain or in the user behaviors can be learned by using the classifiers.

Several different ML strategies (probabilistic approaches, decision trees, Bayesian networks, association rules and neural networks) may be used for classification problems. In following parts we will provide a general overview of the ML techniques.

### 2.2.1 Decision Tree Learner

*Inductive learning methods*, as one type of machine learning technique, are used to infer rules of classification by analyzing examples from a domain [34]. Knowledge obtained by learning techniques has various representation forms, including parameters in algebraic expressions, decision trees, formal grammar, production rules, formal logic-based expressions, graphs, and networks [70].

The 'divide-and-conquer' approach commonly employed by ML algorithms to the problem of learning from instance sets. The problem of constructing a decision tree with this process naturally leads to a tree representation also called decision tree. A recursive process is used for this process. First, select an attribute to place at the root node and make one branch for each possible value. This process splits up the provided dataset into subsets, one for every value of the attribute [70]. The process can be repeated recursively for each branch, using only those instances that actually reach the branch. If at any time all instances at a node have the same classification, stop developing that part of the tree.

An abstract representation of a decision tree is shown in Figure 2.2. Every decision tree begins with what is termed a *root node*, considered to be the "parent" of every other node. Each node in the tree evaluates an attribute in the data and determines which path it should follow. Typically, the decision test is based on comparing a value against some constant. Classification using a decision tree is performed by routing from the root node until arriving

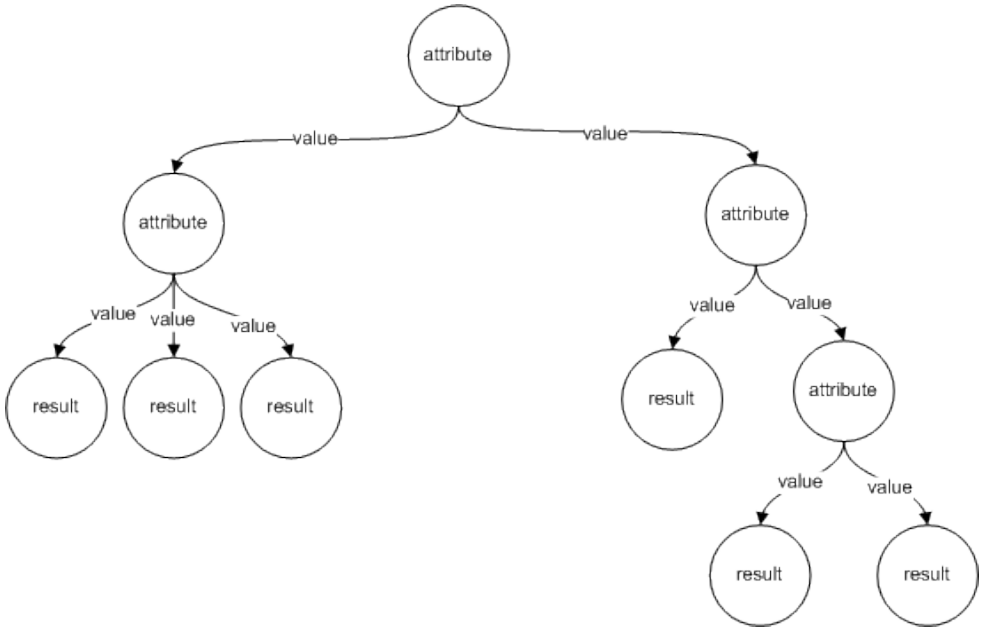


Figure 2.2: An example of a decision tree

at a leaf node.

Decision tree induction involves the recursive partitioning of a set of training data, which is split into increasingly homogeneous subsets on the basis of tests applied to one or more of the feature values. In the *Decision Tree Induction Process*, inductive learning is used in learning from the training set. Classification make it possible to build a decision tree (or classification rules) that classifies each instance. Each instance in the training set is used for classification, building a decision tree.

A well-known example of decision tree learner algorithm is Quinlan's[55] ID3 algorithm which is used to construct decision trees from structured data. The ID3 algorithm uses information theoretic precepts to create efficient decision trees. Given a structured data set, a list of attributes describing each data element, and a set of categories to partition the data into, the ID3 algorithm determines which attribute most accurately categorizes the data. A node is established and labeled by this attribute. The edges coming from this node are labeled with the possible values of the partitioning attribute. The dataset is then divided into subsets by the values of this attribute. If a subset is completely categorized, then the edge terminates in a leaf labeled by the categorization. Otherwise the subset is subdivided further by creating a new node and repeating this process recursively.

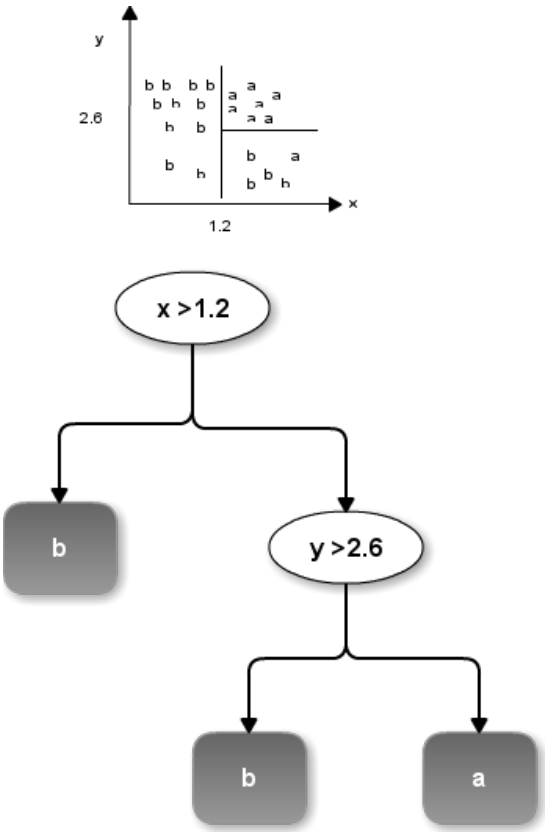


Figure 2.3: Covering algorithm: First covers the instances of a problem and then builds the decision tree for the same problem

### 2.2.2 Rule Construction

In Section 2.2.1 decision tree algorithms based on divide-and-conquer approach to the classification problem is described. Decision tree induction algorithms work from the top down, seeking at each stage an attribute to split on that best separates the classes; then recursively processing the subproblems that result from the split. This method generates a decision tree that can be converted into a set of classification rules [70].

An alternative approach is to take each class in turn and seek a way of covering all instances in it, at the same time excluding instances not in the class. This is called a *covering* approach [70] because at each stage a rule is identified that 'covers' some of the instances. This covering approach leads to a set of rules rather than to a decision tree.

In Figure 2.3 the covering method is visualized in two dimensional space of instances. Rule covering algorithm splits the space vertically and horizontally and tries to cover all instances

of $a$'s and $b$'s correctly.

### 2.2.3 Naive Bayes

Naive Bayes is the most used probabilistic algorithm and belongs to the general class of Bayesian classifiers. These approaches generate a probabilistic model based on previously observed data. It is usually used in content-based systems where the items are represented by textual documents.

For example in a document classification process the Naive Bayes model work as follows; it estimates the posteriori probability, $P(c|d)$, of document $d$ belonging to class $c$. This estimation is based on the a priori probability,$P(c)$, the probability of observing a document in class $c$, $P(d|c)$, the probability of observing the document $d$ given $c$ and, $P(d)$, the probability of observing the instance $d$. Using these probabilities, the Bayes theorem is applied to calculate $P(c|d)$ as follows [17]:

$$P(c|d) = \frac{P(c)\,P(d|c)}{P(d)}$$

Although Naive Bayes performances are not as good as some other statistical learning methods such as nearest-neighbor classifiers or support vector machines, it has been shown that it can perform surprisingly well in the classification tasks where the computed probability is not important [19]. Another advantage of the Naive Bayes approach is that it is very efficient and easy to implement compared to other machine learning methods.

### 2.2.4 Nearest Neighbor

Nearest neighbor methods are also common techniques to solve classification tasks, by defining the $n$ most similar (just labeled) items to a given unlabeled item $I$. In succession, Item $I$ is associated with the classes of the most similar items [25]. Depending on the kind of data a variety of similarity functions can be applied. While Euclidean distance metrics are often used in the case of structured data, the vector space model (based on cosine similarity of vectors, see Section 2.1.2.1) is often used in the context of more complex descriptions, e.g., feature vectors based on texts.

The nearest neighbor algorithm simply stores all of its training data in memory. In order to classify a new, unlabeled item, the algorithm compares it to all stored items using a similarity function and determines the 'nearest neighbor' or the $k$ nearest neighbors. The class label or numeric score for a previously unseen item can then be derived from the class labels of the nearest neighbors.

Nearest neighbor algorithms are quite effective, the most important drawback is their inefficiency at classification time, since they do not have a true training phase and thus defer all the computation to classification time [70].

### 2.2.5 Other Machine Learning Techniques

Some of the ML techniques, not employed in the previous section, include neural networks, inductive logic learning and reinforcement learning. These techniques can be found not only in commercial but also academic recommendation systems and other related technologies [42]. Neural networks learn patterns within the term vectors via a network of nodes and connections. Reinforcement learning adjusts weights based on the successful actions and is used to learn patterns of behavior.

## 2.3 Adaptive Recommender System Examples

The recommender systems researchers have recently focused on the use of ML algorithms for user modeling purposes. Many successful recommender systems have been developed and used by people in many websites.

In this section, we present and explain the recommender system examples that use machine learning techniques and adaptation methods. Instead of examining well-known examples of commercial recommender systems (such as; Amazon.com[40], del.icio.us[16], Last.fm[53] ), we will describe novel and the similar ones to our system.

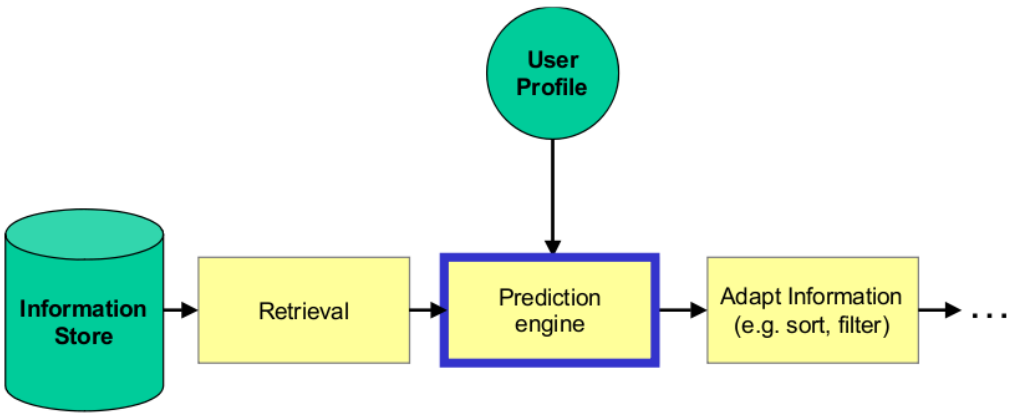


Figure 2.4: The abstract view of the Duine Framework prediction engine component.

### 2.3.1 Duine Framework

The Duine Framework [61] is a domain independent hybrid recommendation system that was presented at the PhD thesis completed by Mark van Setten. It has been developed by Telematica Instituut/Novay [20].

The Duine Framework [61, 50] is a software framework developed by Java. The framework allows developers to create prediction engines for their own applications. Predictions are generated by prediction engines, which are the components of the framework. These predictions can be used for personalization. Figure 2.4 presents the abstract view of components.

The framework creates a generic model for prediction techniques, that calculates a predicted interest value for a piece of information for a given user, based on knowledge stored in the user profile.

The *prediction techniques* use user profiles and information items as input for their calculation. Therefore, the Duine recommender stores data provided by users in user profiles. The data are extracted from the ratings that are given to items and interests of user are tried to be discovered. The Duine Recommender's prediction techniques also have adaptation capabilities that they have learned from explicit user feedbacks.

Duine allows developers to customize and extend the prediction engine by choosing among a set of prediction strategies or creating their own. The *prediction strategies* can be defined

with a set of rules to select prediction techniques and their weights in the prediction.Prediction strategies are a way of easily combining prediction techniques dynamically and intelligently in an attempt to provide better and more reliable prediction results. Prediction strategies are the core part of the framework which are used for hybridization.

Figure 2.5 shows the visual representation of the default prediction strategy of the framework. The main idea behind a prediction strategy is to select the prediction techniques based on the most up-to-date information. Duine uses hard decision rules (if ...then ...else ...) to decide what prediction techniques to use and combine. The framework allows the prediction strategies to be nested in a hierarchy when necessary.

Table 2.2: Duine Framework Prediction Techniques

| **Prediction Technique** | **Description** |
|---|---|
| User Average | Returns the average rating of the user for all content items the user has rated in the past. |
| TopN Deviation | Returns a prediction based on how all other users have rated this content item in the past. |
| Social Filtering | Bases its prediction on how other users that have similar interests with the current users rated the content item. |
| Already Known | When the user has already rated the item, that rate is returned, otherwise no prediction is returned. |
| Genre Least Mean Square (LMS) | Bases its prediction on the categories or genres of piece of information and learns how interested the user is in information of certain categories or genres. |
| Case-Based Reasoning | Bases its predictions on similar items that the user has already rated in the past. |
| Information Filtering | Bases its prediction on a large piece of text describing an item and the known and learned interests of the user by comparing the text with the interests of the user. |

The Duine Framework has six built in recommendation techniques (including both social filtering and content-based techniques) that can be used to create prediction strategies. These techniques are listed in Table 2.2 [60]. Moreover, developers can add new algorithms to these existing ones.

As seen in the Figure 2.5, the strategy contains some attributes, called *tuning parameters*, that are used for technique selection criteria. In the Figure gray nodes represent prediction

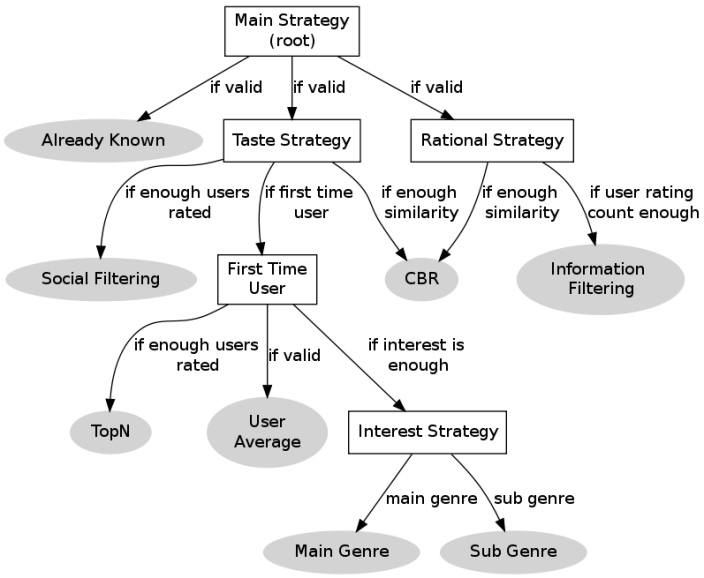


Figure 2.5: The Duine Recommender's default prediction strategy depicted as decision tree.

algorithms and arrows represent the attributes and threshold values. For example in the default strategy 'social filtering' technique is used if enough users have rated the same item. The optimal value of this parameter (the number of enough user) differs per system or domain. This rule-based prediction strategy in the framework uses this threshold values. The tuning parameters, makes the framework more flexible and easier to implement in various domains. However, they are fixed at the startup and never adapted to the changes.

Prediction explanation is one of the important topics in the transparency of recommender systems research [64]. Transparency allows users to meaningfully revise the predictions or suggestions in order to improve recommendations. Duine was designed with explanations in mind each recommendation that is created using the framework can have an explanation object attached to it, which describes how exactly that prediction was produced. This feature enables to explain the recommendations in detail.

Duine prediction framework is open-source and freely available, which can be used to create a hybrid recommender system, allows for the incorporation of prediction techniques that can predict the interests of users for an item with limited knowledge about either the user or an item.

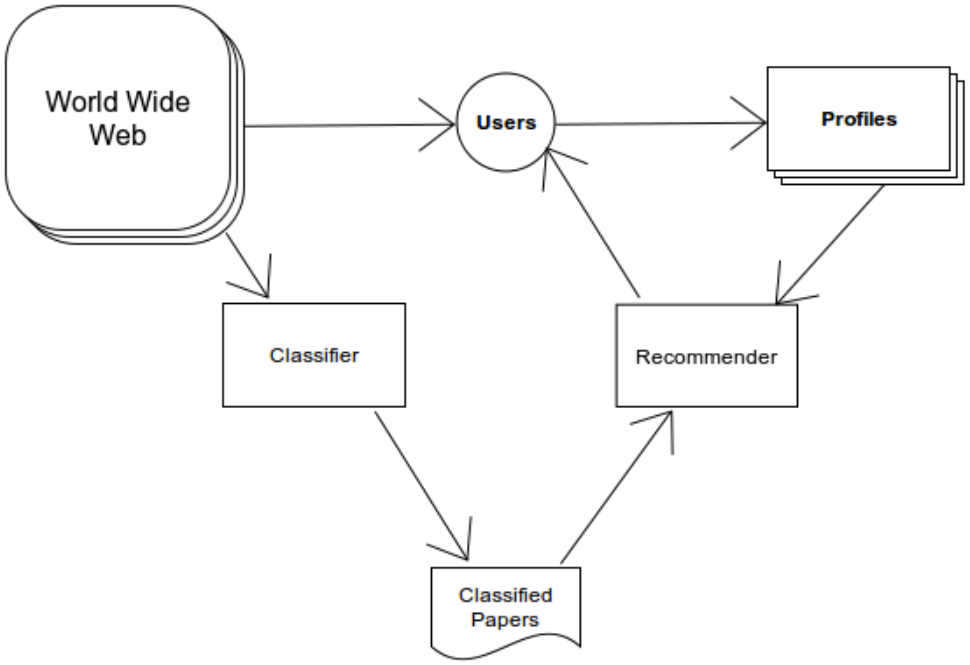


Figure 2.6: The Quickstep recommender system

### 2.3.2 Quickstep Recommender

Quickstep [42] is a hybrid recommender system, which recommends on-line research papers to researchers. It is an experimental recommender system. User browsing behavior is unobtrusively monitored via a proxy server, logging each URL browsed during normal work activity.

The recommender use a nearest-neighbor classification algorithm that classifies browsed URL's based on a training set of labeled example papers. The system stores each new paper in a central database. The database of known papers grows over time, building a shared pool of knowledge. Explicit feedback and browsed URL's form the basis of the interest profile for each user. Figure 2.6 shows an overview of the Quickstep system.

Quickstep bases its user interest profiles on an ontology of research paper topics. This allows inferences from the ontology to assist profile generation.

Quickstep uses a term frequency vector in its central database that is used to represent re-

search papers. Terms are single words within the document, so term frequency vectors are computed by counting the number of times words appear within the paper. Each dimension within a vector represents a term. Dimensionality reduction on vectors is achieved by removing common words found in a stop-list and stemming words using the Porter[54] stemming algorithm. Quickstep uses vectors with 10-15,000 dimensions.

However, cold-starts and interest acquisition (for ontologies) are serious problems in Quickstep Recommender. If initial recommendations are inaccurate, user confidence in the recommender system may drop with the result that not enough usage data is gathered to overcome the cold-start. In regards to ontologies, up-to-date interests are not generally available from periodically updated information sources such as web pages, personal records or publication databases.

### 2.3.3 BellKor System

The BellKor system [7] won the first annual progress prize of the Netflix competition [48, 49]. This system, statically combines weighted linear combination of more than hundred collaborative filtering engines. The system uses model based approach that first learns a statistical model in an offline fashion, and then uses it to make predictions and generate recommendations. The weights are learned by using a linear regression on outputs of the engine.

BellKor uses 'Singular Value Decomposition' approach that conceptualizes the Netflix data [48] as a sparsely populated $m-by-n$ matrix $M$ of movies, customers, and ratings. The Netflix dataset contains more than 100 million date-stamped movie ratings performed by anonymous Netflix customers between Dec 31, 1999 and Dec 31, 2005 [9]. This dataset gives ratings about $m = 480,189$ users and $n = 17,770$ movies (items).

Singular Value Decomposition can be formulated as[69];

$$M = U \sum V^*$$

where $U$ is an $m\ by\ m : m \times m$ unitary matrix, $\sum$ is an $m - by - n$ diagonal matrix with non-negative entries, and $V^*$ is the conjugate transpose of a $n - by - n$ unitary matrix. By sampling the eigenvalues of $\sum$, a compressed representation of the matrix is obtained. The resulting

compression causes movies to be grouped by common features. These feature groups allow higher predictive accuracy.

BellKor's 'Singular Value Decomposition' approach is more efficient then the other Netflix proposed systems, because[7];

- It incorporates the concepts of the Nearest Neighbor model described in Section 2.2.4. Essentially, the BellKor algorithm performs k-NN at the local level and 'Singular Value Decomposition' at the global level.
- Second, the BellKor approach is iterative, allowing scalability and increased performance.

The BellKor solution is exclusively aimed at building a system that would predict subscriber ratings with the highest possible accuracy. Although highly accurate predictions are an important part of a recommender system, these alone cannot ensure good recommendations. Moreover, the solution is based on a huge amount of models, combines more than hundred engines which would not be practical as part of a commercial recommender system.

### 2.3.4 Daily Learner System

The Daily Learner System [10] uses a multi-strategy machine learning algorithm designed to acquire detailed user models, based on explicit and implicit user feedbacks. 'Adaptive Information Server' is the core component of the system, which handles a variety of functions ranging from downloading and storing news content to maintaining user profiles and recommending stories with respect to individual users' personal interests.

The Daily Learner System periodically accesses various news on the Internet, downloads news stories and stores them. The system users then access these resources and give feedbacks. The gathered feedbacks allow to maintain the user profile based on preference information. The system personalize the news, such as re-ordering the current news, based on the personal user models.

A hybrid user model is used for news story classification. The system uses multi-strategy learning approach that learns two separate user-models: one represents the user's short-term

interests, the other represents the user's long-term interests. This learning approach is intended to adjust more rapidly to the user's changing interests. Short term interests are modeled by using the nearest neighbor algorithm (described in Section 2.2.4, on the other hand long term interests are modeled by using naive bayesian classifier (described in Section 2.2.3).

Multi-strategy learning approach predict relevance scores for stories by incorporating the short-term and long-term models into one unifying algorithm. The approach uses the short-term model first, if a story cannot be classified with the short-term model, the long-term model is used. The system selects the recommender engine with the highest confidence level. However, this method is not generally applicable for two reasons. First, not all engines generate output confidence scores for their predictions. Second, confidence scores from different engines are not comparable. Scores from different recommendation engines typically have different meanings and may be difficult to normalize.

### 2.3.5 STREAM

STREAM (Stacking Recommendation Engines with Additional Meta-Features) [6] applies an ensemble learning method, called *stacking*, to solve the problem of building hybrid recommender system. In this system runtime metrics, which represent properties of the input users/items as additional meta-features, are used in order to combine component recommendation engines at runtime based on user/item characteristics.

The STREAM recommender system classifies the recommender engines in two levels, called level-1 and level-2 predictors. The hybrid STREAM system uses runtime metrics to learn next level predictors by a linear regression. The runtime metrics can be both application domain specific and component engine specific.

The main idea behind the STREAM system is to first learn multiple level-1 (base) classifiers from the set of original training examples using different learning algorithms, then learn a level-2 (meta) classifier using the predictions of the level-1 classifiers as input features. The final prediction of the ensemble is the prediction of the level-2 classifier.

STREAM framework is presented in Figure 2.7. The system uses user-item pair $(u, i)$, the rating of the input user $u$ on the input item $i$. Historical ratings of users and additional information (such as item's meta-data) are stored in the system's background data. The framework

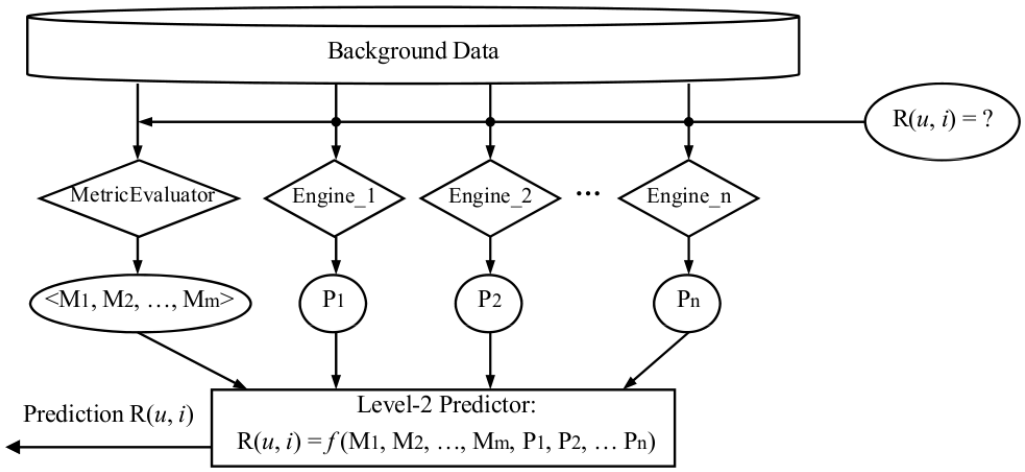


Figure 2.7: Architecture of the STREAM Framework

consists of attached engines that are required to predict rating for given an input user-item pair and a set of background data.

*MetricEvaluator* is a component for computing the runtime metrics. The engines' predictions $< P_1, P_2, \ldots, P_n >$ and the values of the runtime metrics $< M_1, M_2, \ldots, M_m >$ are passed to the level-2 predictor. Level-2 predictor generates the final prediction.

However like The BellKor solution (described in Section 2.3.3), combining many engines level by level results performance problems at run-time. Our approach combines different algorithms on a single hybrid engine with an adaptive strategy. In the system eight meta-features are tested, but the results showed that most of the benefit came from using the number of user ratings and the number of item ratings, which were also two of the most commonly used meta-features by BellKor's solution. Linear regression, model trees, and bagged model trees are used as blending algorithms with bagged model trees yielding the best results. Linear regression was the least successful of the approaches.

# CHAPTER 3

# AdaRec: AN ADAPTIVE HYBRID RECOMMENDER SYSTEM

In this chapter, we present our proposed Adaptive Hybrid Recommender System, named AdaRec, which supports a dynamic adaptation of the domain trends based on the explicit user feedbacks. This chapter organized as follows; First, we present the motivation behind our approach. Then the architecture of the system is explained, and the learning scheme is described. After that, implementation details and the Duine Framework integration of our approach is explained.

## 3.1 Research Background & Motivation

With the rapid development of the recommender system technology, a number of recommender engines have been developed for various application domains. Content-based and collaborative filtering are the two major recommendation techniques that have come to dominate the current recommender system. Content-based recommender system (described in Section 2.1.1) uses the descriptions about the content of the items (such as meta-data of the item), whereas collaborative filtering system (described in Section 2.1.2) tries to identify users whose tastes are similar and recommends items they like. For example, Netflix.com [7] recommends movies based on the user's scoring of previously watched movies (both the user and others); Amazon.com [26], on the other hand recommends goods to the online shopper based on the previously viewed/purchased items and the purchases of other customers. Other recommendation technologies include hybrid techniques, knowledge-based approaches etc [13].

Previous research in this area, has shown that these techniques suffer from various potential

problems-such as, sparsity, reduced coverage, scalability, and cold-start problems [2, 13, 73]. For example; collaborative filtering techniques depend on historical ratings of across users that have the drawback, called *cold start problem* - an item cannot be recommended until it has been rated by a number of existing users. The technique tends to offer poor results when there are not enough user ratings. Content-based techniques can overcome the cold start problem, because new items can be recommended based on item features about the content of the item with existing items. Unfortunately, content-based approaches require additional information about the content of item, which may be hard to extract (such as, movies, music, restaurants). Every recommendation approach has its own strengths and weaknesses. Hybrid recommender systems have been proposed to gain better results with fewer drawbacks.

To date a number of hybrid recommender systems have been proposed that combine individual systems to avoid certain aforementioned limitations of these systems [13]. In order to improve predictions, real-world recommendation systems are typically use hybrid approaches that combine multiple prediction techniques. This helps to avoid certain drawbacks of content-based and collaborative filtering systems.

Core part of the hybrid recommender systems is its prediction technique (algorithm) selection strategy (described in Section 2.3.1). Recommender system's algorithm switching procedure is done by using its attached prediction strategy. This strategy decides which technique to choose under what circumstances. Recommender system's behavior is directly influenced by the prediction strategy. The construction of accurate strategy that suits all circumstances is a difficult process. A well-designed adaptive prediction strategy offers advantages over the traditional static one.

Previous research efforts on hybrid recommender system has mostly focused on static hybridization approaches (strategy) that do not change their hybridization behavior at runtime [57]. However, people's tastes and desires are temporary and the gradually evolve. Moreover, each domain has unique characteristics, trends and unique user interests. Since user's interests might change dynamically over time, it does not seem possible to adapt trends by using the static approach. Therefore, in this chapter, we focus on building hybrid recommendation systems that dynamically adapt itself to the changes in the domain and users.

We apply an instance based learning scheme, that learns through user feedbacks in order to solve the problem of building adaptive hybrid recommendation systems. Hybrid recommen-

dation systems combine multiple algorithms (prediction techniques) and define a switching behavior among these techniques. The main idea of our system is to re-design the hybrid system's switching behavior according to the performance results of the algorithms. We also introduce the novel idea of analyzing the prediction performance results and using them as runtime metrics. This method allows us to use characteristics of the input user/item when determining how to combine the prediction techniques at runtime. These runtime metrics are properties of the input user/item that are related to the nature of the domain. For example, the number of items or item's similar user count that the input user has previously rated may indicate how well a collaborative filtering engine will perform. Learning algorithms analyze performance results of previous predictions and then re-design the combination of prediction techniques. We combine prediction techniques with runtime metrics and up-to-date threshold values according the performance analysis of the techniques.

## 3.2 Our Approach

In this section, we first introduce our learning scheme, instance-based learning. We then describe how we apply it to solve the hybridization problem with runtime metrics. Finally we demonstrate our AdaRec Framework.

### 3.2.1 Instance-Based Learning Scheme

Techniques that hold the training set in memory are called instance-based methods [70]. Instead of performing explicit generalization, instance-based methods compare new problem instances with instances seen in the training set. Instance-based methods, which are kind of lazy learning, construct hypotheses directly from the training instances themselves. The most common instance-based method in the recommender systems literature is the k-nearest neighbor (kNN) algorithm [41].

The learning scheme that is used in our work is *supervised learning*-learning from examples. A fixed set of $n$ attribute with their value pairs represent the instances. The attributes are fixed in the system. A learning algorithm is attached to the framework that analysis previously stored instances. Algorithm determines the attribute-value pairs and build a model with respect to previously recorded instances called *training set*.

Previous predictions and *user feedbacks* (especially the explicit ones) are stored. Each prediction together with its attributes are recorded. Learning module analyses the previous predictions and classifies the instance to the best performed algorithm. In other words; every instance is classified to an algorithm that predicts more closest to the user's feedback. The collection of classified instances are called training set.

After classifications are completed, best performed algorithms of the instances in the training set is identified. Learning algorithm builds a model, such as decision tree, rule sets or decision list, according to the training set. This model defines the hybrid recommender system's algorithm switching behavior. Nature of the domain and the characteristics of users can be learned implicitly by an automatic learning method, using attributes as input to a supervised learning algorithm, and producing a model as output.

### 3.2.2 Static Prediction Strategy

A *prediction strategy* as described in Section 2.3.1 defines the switching behavior of the hybrid recommender system. Prediction strategies are the core part of the recommenders which are used for hybridization. A prediction strategy uses attribute-value pairs called *threshold values* in order to make decisions about which algorithm (case-based reasoning, collaborative filtering, content-based etc.) to use and how to combine them. However in the naive hybrid recommender systems the supplied threshold values are static and do not change during the execution of recommender engine. Figure 3.1 shows a sample static prediction strategy, that defines a sample combination of its available algorithms (prediction techniques).

Like most of the available open source hybrid recommender systems, the Duine Framework [61], which was described in Section 2.3.1, uses a *static pre-defined* decision tree approach that employs the prediction strategy of the framework. The prediction strategy visualizes what steps are taken to arrive at a prediction technique in the hybrid recommender systems [50, 62].

Another way to represent a decision tree is using decision rules. Decision rules (described in Section 2.2.1) generally take the form of IF ... THEN ... rules, i.e.

Experts often describe their knowledge about a decision process using decision trees and decision rules, as they are easy to interpret by other people. In the Duine Framework prediction algorithm selection strategy is defined by a static decision tree.

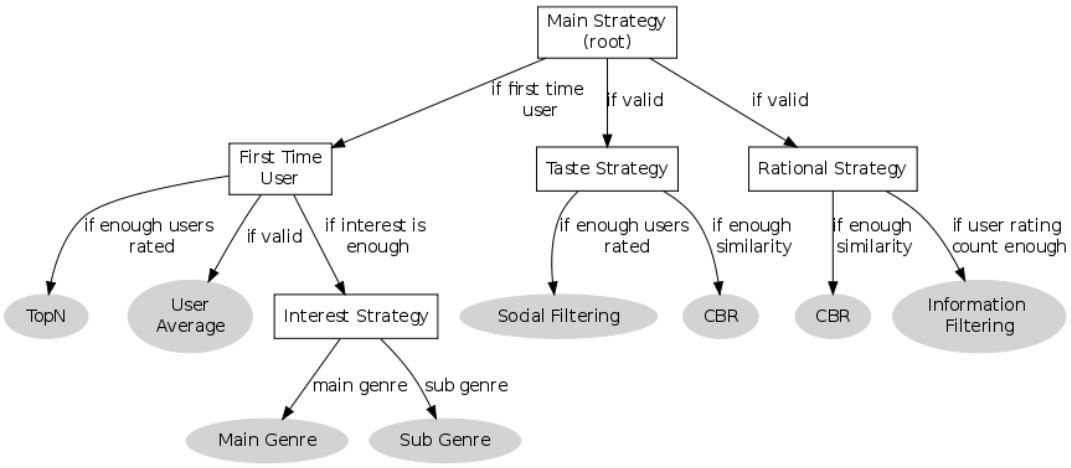


Figure 3.1: A sample static prediction strategy, depicted as decision tree.

In Section 2.3.1, Figure 2.5 represents a sample static decision tree (default prediction strategy) in the Duine framework. In this static prediction strategy (tree) nodes represent validity indicators and each branch from a node represents a value or range of values for that validity indicator. The leaf nodes of the decision tree represent prediction algorithms [50, 61].

Most of the currently available personalized information systems focus on the use of a single selection technique or a fixed combination of techniques [61, 42]. However, fixed strategy may be suboptimal for dynamic domains and user behaviors. Moreover they are unable to adapt to domain drifts. Since people's tastes and desires are transient and subject to change, a good recommender engine should deal with changing consumer preferences. It does not seem possible to adapt trends by using a static approach (static prediction strategy).

In this section we present our adaptive prediction strategy method, that learns the domain dynamics by analyzing the previous performance results. Instead of using static methods, dynamic methods can handle changes in domain.

### 3.2.3 Prediction Strategy Learning

In our approach, we describe an adaptive hybrid recommender system, called AdaRec, that modifies its switching strategy according to the performance of prediction techniques. Our hybrid recommender approach uses adaptive *prediction strategies* (described in Section 2.3.1)

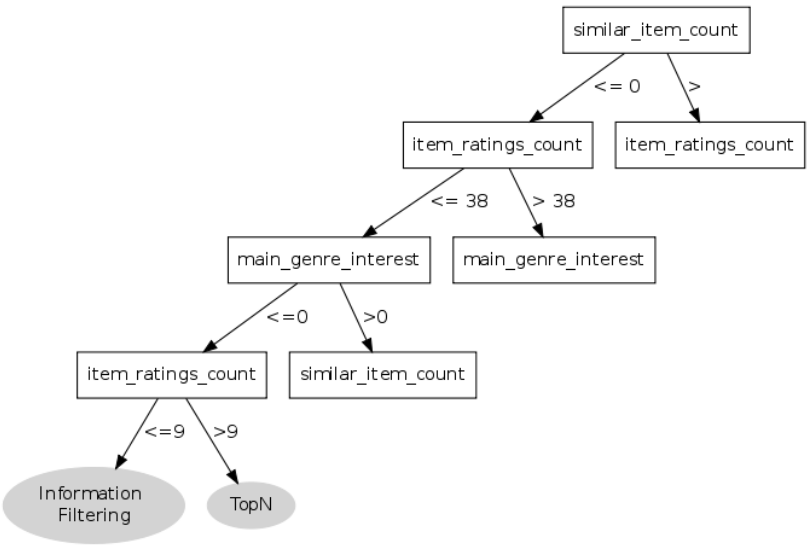


Figure 3.2: A sub-tree of a sample prediction strategy that uses domain attributes, depicted as decision tree.

that determine which *prediction techniques* (algorithms) should be used at the moment an actual prediction is required.

Initially we used manually created rule-based strategies which are static, as described in the previous section. These static hybridization schemes have drawbacks. They require expert knowledge and they are unable to adapt to emerging trends in the domain. We now focus on prediction strategies that learn by themselves.

Mostly the wide range of research areas are focused on the use of static decision approach at the switching hybridization methods. A *switching hybrid* is one that selects a single recommender from among its constituents based on the recommendation situation [13]. For a different user profile or domain, a different switching technique might be chosen. The switching hybridization method assumes that some reliable criterion is available on which to base the switching decision. The choice of this switching criterion is important for hybrid recommender systems [12, 62].

A key observation of our work is that when user behaviors begin to change, the recommender system's prediction accuracy tends to detoriate. Old, static, pre-defined switching behavior unable to adapt to the changes. As users interact with the domain, some trends are rising, some are falling. And even more, the users themselves are changing over time. An adaptive

hybrid recommender system should monitor these social trends and changes.

In our *Adaptive Hybrid Recommender System (AdaRec)* Framework, the switching behavior is not static, instead the framework able to learn the trends and re-design its switching behavior according to the captured trends. The system learns to adapt to the changing trends by re-designing its prediction strategy. For example, in a simple way, the system can change the early defined threshold values. Figure 3.2 presents a small subtree of an adaptive strategy which assigns the most context sensitive values to the thresholds. These adapted threshold values provide up-to-date knowledge about the relevancy of each algorithm for the current prediction request. In other words, system adapts its prediction strategy either by simply modifying the threshold values or completely re-designing the prediction strategy.

Our system, employs switching hybridization by deciding which algorithm is most suitable to provide a recommendation. The decision is based on the most up-to-date knowledge about the current user, other users, the information for which a prediction is requested, other information items and the system itself. In AdaRec we propose the use of Machine Learning (ML) techniques (described in Section 2.2) to analyze and learn which user and item features in a personalized recommender system are more adequate for correct recommendations. User feedback and MAE (Mean Absolute Error) are the main concepts, which describe the trends in the domain. Adaptive prediction strategy learns its domain trends over time via unobtrusive monitoring and relevance feedback.

## 3.3 AdaRec Framework

In this section, we first introduce a high level general overview of our AdaRec architecture. We then describe the implementation details of the architecture and its components. Finally we demonstrate the integration with the Duine Framework.

### 3.3.1 Overview of AdaRec

The architecture of the adaptive hybrid recommender system (AdaRec) is composed of the following modular components. Figure 3.3 illustrates these logical components and relations among them.

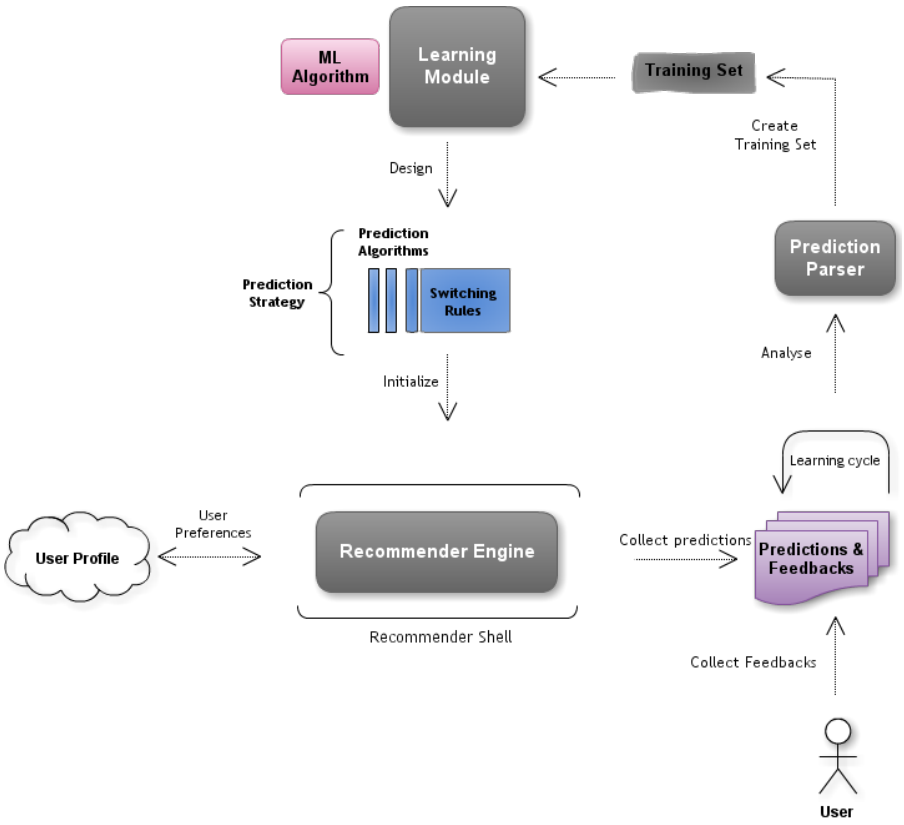


Figure 3.3: A High Level Overview of AdaRec

- **User Profile**, is the representation of the user in the system. For each active user a *user model* is stored in the user profile. User profile holds the knowledge (such as preferences, feedbacks, feature weights etc.) about users in a structured way.

- **Recommender Engine**, which is the active part of the framework, produces recommendations to a user based upon the user profile. Recommender Engine builds and exploits profiles (models) of the users interacting with the system. Recommender Engine can both use the current user and other users profile when making a recommendation depending on recommendation algorithm's approach. Also updates the user profile based on the feedbacks. Recommender Shell, encapsulates the Recommender Engine's interaction with other modules. The shell serves the created prediction strategies to the engine. Also the first initialization of the recommender engine is the responsibility of the shell.

- **Prediction Parser/Analyzer**, produces the performance results of the prediction algorithms based on the analysis of the collected predictions and feedbacks. This module handles the decomposition of the prediction results and generates the training set of sample instances with current attributes.

- **Training Set**, serves as background data for the current context. The set contains nugget of knowledge, which reflects the characteristics of the current context, that is used by the learning module.

- **Learning Module**, handles the new prediction strategy creation upon the previous instances and performance results of the prediction techniques on each learning cycle. It allows the building of new decision trees/decision rules based on the previous recorded instances.

- **ML Algorithm**, uses the learning methods in order to extract information from the training set. The algorithm learns the nature of the domain and re-design the prediction strategy based on its results. The ML Algorithm is attached to the learning module.

- **Prediction Strategy**, consists of prediction algorithms and switching rules associated with them. The prediction strategy defines the switching behavior of the recommender engine. Prediction strategy is created by Learning Module according to the training set of instances using the attached ML algorithm.

Our proposed AdaRec System, which has adaptation capabilities, basically comprises two entities, namely *Learning Module (LM)* and *Recommender Engine (RE)*.

Recommender Engine, as mentioned above, is responsible for generating the predictions of items based on the previous user profiles and item contents. It attempts to recommend information items, such as movies, music, books, that are likely to be of interest to the user. The recommender generate the predictions by using its attached prediction strategy. The implementation here uses the open source Duine Framework for the recommender engine which will be described in Section 3.4.1.

Learning Module, as mentioned above, handles the new prediction strategy creation upon the previous instances and performance results of the prediction techniques on each learning cycle. It allows the building of new decision trees/decision rules based on the previous recorded instances.

*Learning cycle* is a logical concept that represents the re-design frequency of the prediction strategy. Each instance, based on the indicated count, and prediction algorithm's performance results are collected between two learning cycles. The learning module modifies the values of decision rules, which is also called threshold values according to the gathered results of the prediction techniques performance. The old prediction strategy is modified by using recommender engine's machine learning algorithm (rule tuning, rule adaptation, decision tree induction etc.). The modification of the threshold values allows recommender system to analyze and adapt the nature of the users and the domain.

### 3.3.2 Learning Module

The learning module first tests the accuracy of each predictor in the system. Then the prediction strategy is re-designed by the learning module in order to improve proper use of predictors. Adaptive prediction strategy improves the prediction accuracy by learning when to use which predictors. The learning module adapts the hybrid recommender system to the current characteristics of the domain.

Inductive learning is used in learning from the training set. In our experiments we tested different (1K, 1.5K, 2K and 3K) instance sizes for training sets. The training set contains the instances from the previous learning cycle results. Neural networks (e.g. back-propagation),

rule learners (e.g. RIPPER), instance-based methods (e.g. nearest neighbour), decision trees (e.g. C4.5) and probabilistic classifiers are some of the popular inductive learning techniques [42].

Different design approaches might be used for the prediction strategy adaptation. Rule based, case based, artificial neural networks or Bayesian are some of the learning techniques. Each technique has its own strengths and weaknesses. In AdaRec, we introduce self adaptive prediction strategy learning module which employs a strategy, based on its attached learning technique. Learning module initializes prediction engine according to the specified machine learning technique. This prediction strategy adapts itself to the current context by using the previous performance results of the techniques.

User feedbacks and MAE (Mean Absolute Error) are the main concepts, which describe the trends in the domain. Learning module adjusts the prediction strategy and re-initializes the recommender system with modified strategy. The module both collects the predictions, which are predicted by the recommender, and the feedbacks, which are harvested from the users. On each prediction request, the strategy selects a prediction algorithm that is suitable for the context. The algorithm then produces a prediction. The MAE score of the algorithm, which is calculated by the difference between the user feedback and the prediction, together with the application domain attributes are stored as instances by the learning module. The average MAE results is treated as performance scores of the algorithms. A sample monitored instance for a prediction request is shown below.

```
% For each instance, current attributes, used prediction algorithm and MAE score.

%attributes
item_ratings_count numeric, item_similar_user_count numeric, similar_item_count numeric
main_genre_interest numeric, sub_genre_interest numeric,

%selected prediction algorithm
technique {informationFiltering,socialFiltering,
                    topN,mainGenreLMS,userAverage,cbr,subGenreLMS}
%MAE
<0,1>
```

ML Algorithm that is attached to the learning module re-designs the prediction strategy in order to maximize the prediction accuracy (or minimize the MAE scores). The learning module is the dynamic aspect of our hybrid recommender system.

In AdaRec System, we focused self adaptive prediction strategy that classifies according to its attached machine learning technique. This prediction strategy adapts itself to the current context by using the previous performance results of the techniques. Different machine learning algorithms that induce decision trees or decision rule sets could be attached to our experimental design. The architecture is open and flexible enough to attach different machine learning algorithms.

Adaptive prediction strategy learns its domain trends over time via unobtrusive monitoring and relevance feedback. Learning cycle of the framework is started with the initial design of the prediction strategy. The adaptive strategy is at first manually created by using expert knowledge. Later on the recommender system is initialized with this manually created rule-based strategy.

The induction of the prediction strategies by using the previous user feedback allows recommender system to analyze the nature of the users and the domain. To sum up, the learning module supports the recommender systems, in the following ways;

- It adapts the recommender system to the current nature of the domain and the users,
- It reduces the impact of incorrectly designed decision rules

## 3.4 Integration with Duine & WEKA

In order to demonstrate our proposed AdaRec Framework, we integrated the system with the open-source Duine Framework[1] [50] and WEKA[2] [70]. AdaRec Framework's core part was developed with Java and Spring Framework[3] [32] technologies using Netbeans IDE[4]. MySQL[5] database is used the user profile (items, ratings etc.).

In this section, we explain the implementation details of the Duine Framework and WEKA integration. At the early stages of this study, we aimed a set of implementation goals. Some of these are; simplicity, provide fast recommendations, extensible system architecture, enough

[1] http://duineframework.org/
[2] http://www.cs.waikato.ac.nz/ ml/weka
[3] http://www.springsource.org/
[4] http://netbeans.org/
[5] http://www.mysql.com/

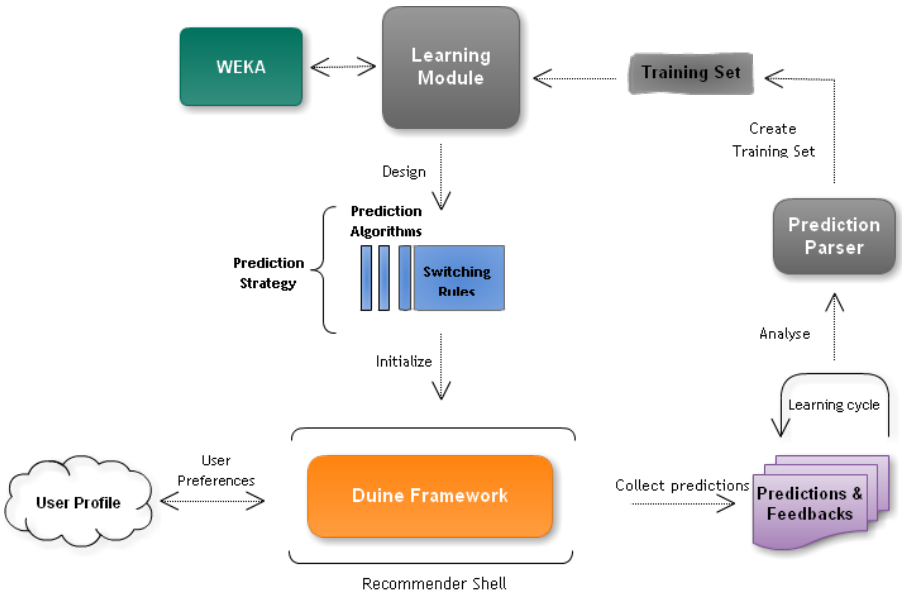


Figure 3.4: A High Level General Overview of the Duine and WEKA Integration with AdaRec System

abstraction for pluggable ML algorithms. To achieve these goals, we select the Java[6] platform supported with Spring Framework for. The AdaRec System is implemented to run in the latest version of the Sun Microsystems JVM (Java Virtual Machine) which is the most popular and widely used Java solution.

### 3.4.1 Integrating with the Duine Framework

We chose to integrate with Duine Framework rather than to develop a plug-in because, frameworks offer many advantages over plug-ins. For example, an advantage of the recommender frameworks is the possibility to adapt the recommendation task to specific requirements of your experiment or your domain. This is not possible with the plug-ins, because they offer less flexibility for further development.

We introduce an adapter to the recommender shell component that maps our system's requirements with Duine and vice versa. Figure 3.4 presents the high level architectural overview of Duine and WEKA integration with AdaRec system.

In the Duine Recommender the knowledge that designs the selection strategy is provided

[6] `http://java.sun.com`

manually by experts [28, 62]. However, the combination of different techniques in a dynamic, intelligent and adaptive way can provide better prediction results. The combination of techniques should not be fixed within a system and that the combination ought to be based on knowledge about strengths and weaknesses of each technique and that the choice of techniques should be made at the moment a prediction is required. With the integration, the Duine Framework gained new adaptive features. The framework's static prediction strategy is continuously re-designed by AdaRec. AdaRec uses Duine Framework as the built-in Recommender Engine. In the integration, AdaRec System works as previously described in Section 3.3.

The Recommender Shell component for the Duine integration has extra mapping capacities. The shell maps produced prediction strategy to the Duine's working format. Also the system is integrated with the Duine Framework prediction processes. As shown in the Figure 3.5, Duine Framework processes consists of four main phases [61].

- **The prediction phase:** A prediction is generated by the recommender engine.
- **The usage phase:** The user and/or other parts of the recommender system use the generated predictions.
- **The feedback phase:** After the user has used one or more items, he can provide feedback to the system about his actual interest in the items.
- **The learning phase:** The system learns from the feedback provided by the user, which is used to increase the accuracy of future predictions.

We quickly realized the integration with Duine Framework, through the flexible plug-in based architecture. Figure 3.6 presents the general overview of the plug-in based architecture. The Duine Framework is developed with Java programming language[4] and provides a very clean division between controllers, JavaBean models, and views with Spring Framework[32].

Duine has built-in prediction techniques as described in Section 2.3.1 at Table 2.2, some of them are supplied for movie domain. In our system, we used and tested different prediction techniques. These are; *topNDeviation, userAverage, socialFiltering, CBR (Case-Based Reasoning), mainGenreLMS (Least-Mean Square), subGenreLMS, informationFiltering*.

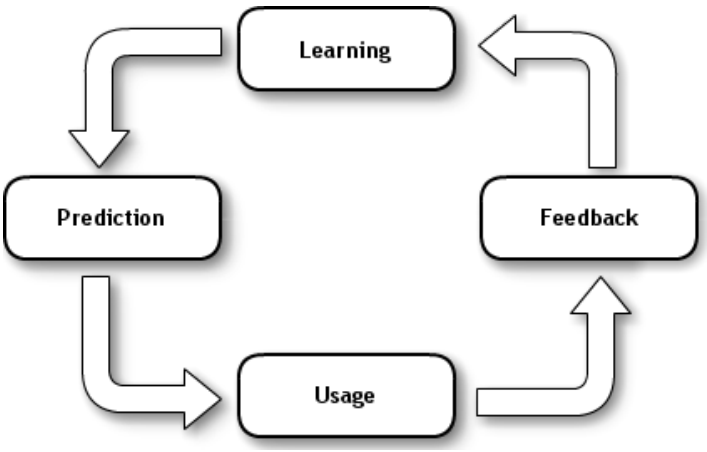


Figure 3.5: Prediction Process of the Duine Framework.

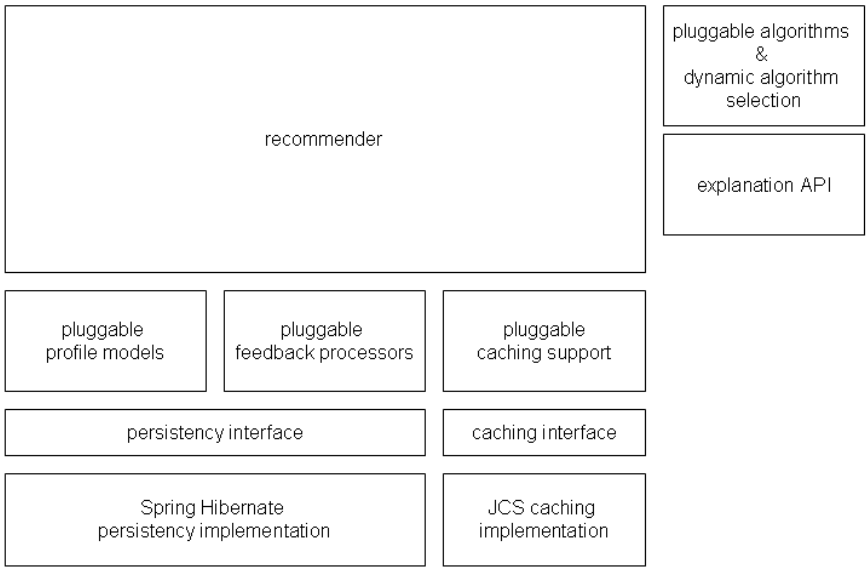


Figure 3.6: General Overview of the Duine Framework.

Depending on the nature of the domains (movie, music, book etc.) different attribute-value combinations can be used for prediction strategy design. Adaptive strategy, as described in Section 3.2.3, is designed with the combination of attributes by using rule sets or trees (depending on the used ML technique) that contain the knowledge on decisions. In AdaRec, since we conduct experiments on MovieLens dataset, we chose these specific attributes that have meaningful correlations between movie domain and prediction techniques. We believe that, by measuring the changes on these attributes, we can capture the domain drifts and trends.

In AdaRec, since we prefer to conduct experiments on the movie domain, we chose these specific attributes that have meaningful correlations between movie domain and prediction techniques. Sub-genre and main-genre are domain-dependent attributes. Genre certainty is measured by using the movies metadata information which is interpreted according to MovieLens and IMDB (Internet Movie Database) datasets [30]. We believe that, by measuring the changes on these attributes, we can capture the domain drifts and trends.

1. **item ratings count**, the number of ratings that the current item has.

2. **item similar user count**, similar users count that have already rated the current item.

3. **similar item count**, the number of similar items count according to similarity measures.

4. **main genre interest**, main genre interest certainty of the current item among the users items.

5. **sub genre interest**, sub genre interest certainty of the current item among the users items.

Decision trees/decision rules are constructed using the combination of the above five attributes. As shown in Figure 3.2, at each node of the decision tree a attribute is compared with a value, which is called the *threshold value*. These five attributes are used to classify the prediction techniques.

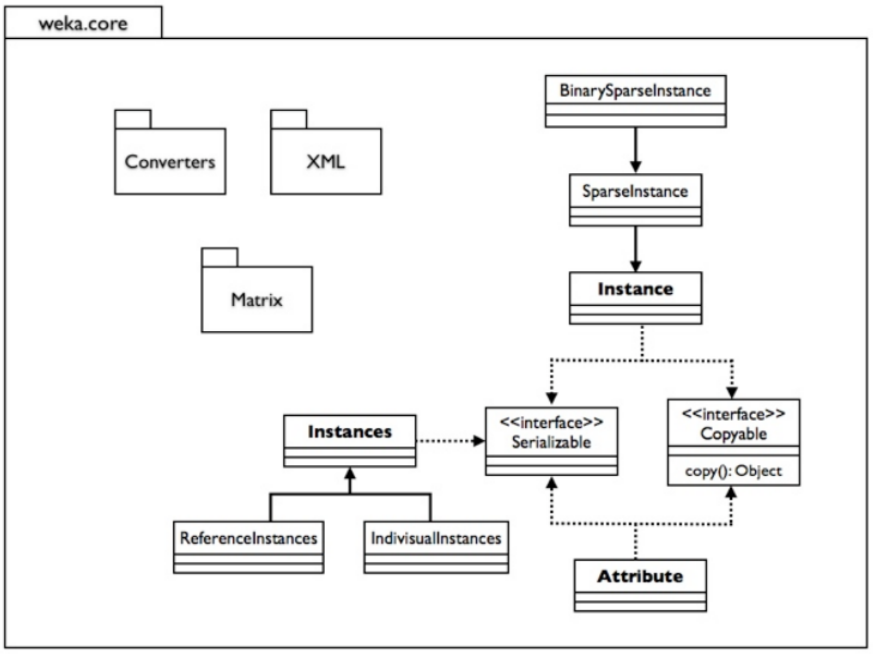


Figure 3.7: Class Diagram for weka.core package

### 3.4.2 Integrating with the WEKA

As shown in the Figure 3.4, AdaRec's Learning Module is integrated with WEKA tool. The *WEKA-Machine Learning Framework* has been developed at the University of Waikato in New Zealand [70]. Several versions of WEKA are developed since it has started. Prototype for the experimental study is built within WEKA-3.6.2[7].

A number of data mining methods and machine learning algorithms are implemented in the WEKA software. WEKA provides different experimentation platforms for running training and test different ML algorithms, These are a graphical user interface (GUI) and a simple command line interface (CLI). Also WEKA provides an API (Application Programming Interface) with lots of descriptive API sources (documents, books, web sites etc.). This gives the system developer the opportunity of experimenting with different learning algorithms using using the Java API and the provided GUI tool. In addition, WEKA supports different kind of pre-processing, association rules, feature validation and more.

Collection of related classes are grouped together in WEKA, called packages. WEKA has

[7] `http://sourceforge.net/projects/weka/files/weka-3-6/3.6.2/`

three main packages for core operations, classifiers and filters. Figure 3.7 shows the class diagram of WEKA's core package.

The data that is used for WEKA should be made into the ARFF (attribute-relation file format) file. An ARFF file is simple text file that describes a list of instances sharing a set of attributes. Typical ARFF file contains a header section that describes the relation and type of the data. WEKA refers to keys being *attributes*, categories being *classes* and atoms of training being *instances*. A sample ARFF file that is used in our experiments is shown below. In AdaRec learning module is responsible to convert the instances to ARFF file format.

```
%A sample ARFF file
@relation sample_result_set

@attribute item_ratings_count numeric
@attribute item_similar_user_count numeric
@attribute similar_item_count numeric
@attribute main_genre_interest numeric
@attribute sub_genre_interest numeric
@attribute technique {informationFiltering,socialFiltering,
                      topN,mainGenreLMS,userAverage,cbr,subGenreLMS}
%Instances
@data
195,3,2,1.7,4,subGenreLMS
196,4,2,3.3,4,topN
197,7,1,0.8,4,subGenreLMS
198,9,2,0.6,7,mainGenreLMS
199,2,1,3.2,6,subGenreLMS
200,3,0,2.7,8,topN
110,1,5,0,5,topN
```

In our proposed Adaptive Recommender System Framework we use different instance-based learning algorithms that are supported by WEKA [70]. These are; J48 (a C4.5 variant), BF-Tree (Breath First Tree) and Conjunctive Rules [70].

- **J48:** WEKA's implementation of the decision tree learner based on the C4.5 [55] decision tree algorithm, which selects the attribute that minimizes entropy in the split. C4.5, which is the most widely used decision tree learning approach in ML research [33], uses gain ratio as the feature selection measure. C4.5 prunes by using the upper bound of a confidence interval on the re-substitution error as the error estimate; since

nodes with fewer instances have a wider confidence interval, they are removed if the difference in error between them and their parents is not significant.

- **BF-Tree:** A decision tree learner that uses a best first method of determining its branches. This method adds the *best* split node to the tree in each step. The best node is the node that maximally reduces impurity among all nodes available for splitting (i.e. not labeled as terminal nodes).

- **Conjunctive Rules:** A simple rule learner that learns a set of simple conjunctive rules that can predict for numeric and nominal class labels. It uses a technique called Reduced Error Pruning [21] to trim an initial set of rules to the smallest and simplest subset of rules that provide highest discrimination.

# CHAPTER 4

# EVALUATION AND EXPERIMENTS

In this chapter, we present our experiments on the MovieLens dataset in order to assess the impact of our proposed system and different machine learning algorithms. we conduct experiments on the MovieLens dataset. We evaluate the accuracy of predictions (by using MAE) of the system using different configurations of the machine learning algorithms.

This chapter is organized as follows; In the first section we examine the MovieLens dataset and its conversion process. Next in Section 4.2 we describe the evaluation metrics of the experiments. After that we explain our experimentation environment and carried out experiments. Finally we will evaluate our proposed system and discuss the results.

## 4.1 Dataset

Predicting users' movie ratings is one of the most popular benchmark tasks for recommender systems. There are several publicly available data sets for this problem. GroupLens Research Project[1] has set up a web site called MovieLens[2], which is based on the collaborative filtering [44]. Everyone who logs on the system is asked to rate a list of movies with 1 to 5 scale, where 1 means *certainly not the movie for me* and 5 means *absolutely love it*.

GroupLens Research Group released three datasets[47] with different properties; the MovieLens $100,000$ rating dataset, the MovieLens 1 Million rating dataset, and the MovieLens 10 Million rating dataset. These datasets became the standard datasets for recommender research, and have been used more than 300 research papers [59]. The dataset is also being used

[1] http://www.grouplens.org/
[2] http://www.movielens.umn.edu

for educational purposes by researchers.

In our experiments we used the MovieLens 1 Million dataset[3] [47] in its full extent with 6040 users and 3900 movies pertaining to 19 genres. MovieLens dataset contains explicit ratings about movies. The dataset has sparse user-item matrix. Both MovieLens users and MovieLens movies were declared as AdaRec users AdaRec movies. The original ratings scale $< 1 .. 5 >$ was linearly scaled into the AdaRec's internal scale $< 0 .. 1 >$. This way transformed dataset is further referred to as the MovieLens database.

MovieLens $1M$ dataset main characteristics (value, sparsity, size etc.) are presented at Table 4.1. An *active user* is a user who have rated more than once. And passive users are users that have rated at least once. The last three rows contain values that are scaled to the AdaRec's ratings scale 0..1.

Table 4.1: An overview of the MovieLens 1M dataset basic characteristics.

| **Feature** | **MovieLens 1M Dataset** |
|---|---|
| Inception date | February 2003 |
| Total number of users | 9,746 |
| Number of active users | 6,040 |
| Number of passive users | 3,706 |
| Total number of ratings | 1,000,209 |
| Max. ratings from 1 user | 2,314 |
| Mean rating value | 0.822 |
| Median rating value | 0.95 |
| Standard deviation | 0.3505 |

In our experiments; the dataset was not used in its pure form, instead we passed through some transformations on the dataset. Firstly we imported the dataset files into structured relational database. And then in order to handle learning cycles we divided the dataset in to temporal partitions. These processes will be described in Section 4.3 together with the experimentation setup.

[3] `http://www.grouplens.org/node/12`

## 4.2 Evaluation Metrics

A wide variety of different evaluation metrics have been used for evaluating the success of the prediction algorithms by recommender system researchers [58, 63]. We use Mean Absolute Error (MAE) metric, which is a popular statistical accuracy metric among research community, for our experiments. MAE is computed by the absolute difference between the predicted rating and real user rating and then computing the average. For each ratings-prediction pair $< p_i, q_i >$, this metric treats the absolute error between them i.e., $|p_i - q_i|$ equally.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

Since MAE can be interpreted easily, we choice it as our primary metric in our experiments. Recommender researchers in the related field have also suggested the use of MAE for prediction evaluation metric [27].

Prediction accuracy is calculated according to MAE, which is scaled to AdaRec's internal scale $< 0 .. 1 >$. The Duine generates prediction ratings in unipolar range goes from 0 (not interesting) to +1 (interesting) with 0.5 representing a neutral interest. This system predictions are scaled to the $< 0 .. 1 >$ range and referred as baseline in the experiments. MAE compares the actual rating a user gave to an item with the predictions made by an algorithm. The lower the mean absolute error, the better an actual predictor can be classified.

## 4.3 Experimental Setup

In our experiments, we evaluate the performance of our AdaRec system on the MovieLens dataset and compare with the performance of naive (static) hybrid system called baseline. In the experiments each prediction provided by the two different systems (AdaRec and baseline) are examined. The prediction accuracy and the prediction error (MAE) are recorded.

MovieLens pure dataset is converted & mapped in to a relational database called *MovieLens Database* for further processings. MovieLens dataset is imported into related tables (movies, users and ratings) and the relations among them are constructed. Figure 4.1 shows the logical database diagram of the converted MovieLens dataset. MovieLens database is used as


Figure 4.1: Logical view of the MovieLens database.

information source in our experiments. By using this database we can repeat experiments as many times as we want. With the relational database, it is possible to manipulate the data by using the SQL (Structured Query Language). For example, we can sort the data according to timestamp values and divide the big dataset into subsets.

The aim of the experiments is to examine how the recommendation quality is affected by our proposed learning module which was described in Section 3.3. The present model of the Duine Framework is non adaptive but it supports predictor level learning [61]. This original state of the framework is referred to *baseline*.

As shown in the Figure 4.2, Duine recommender uses a static prediction strategy as hybridization scheme, which does not change at runtime. We want to compare the prediction quality obtained from the framework's baseline (non adaptive) to the quality obtained by our proposed experimental framework (adaptive). The approach will be considered useful if the prediction accuracy is better than the baseline.

The validation process is handled using the following procedure:

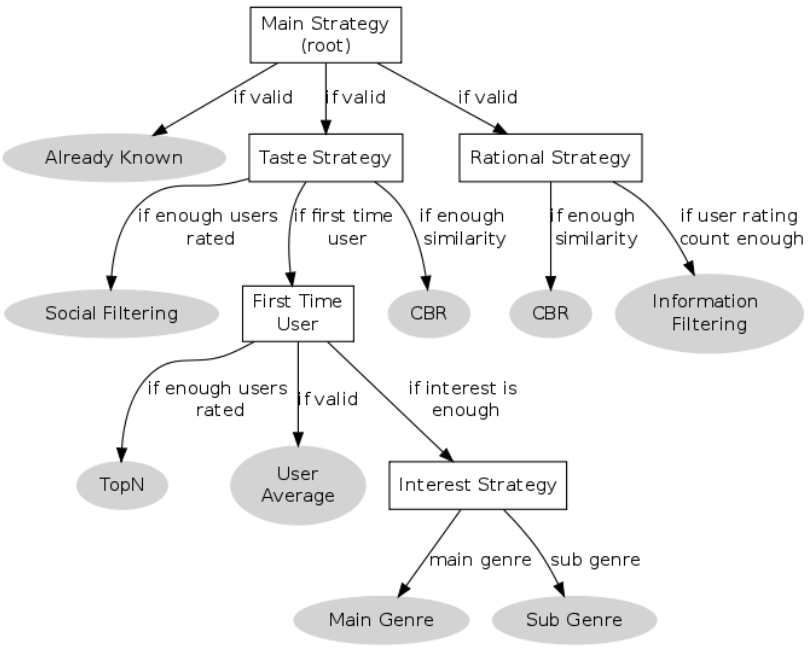


Figure 4.2: Duine Framework's static prediction strategy.

- Both the adaptive and baseline systems are initialized with the Duine's default prediction strategy. However, baseline system's prediction strategy stays static, the adaptive system's, on the other hand changes over time.
- The ratings provided by the dataset are fed to the system one by one, in the logical order of the system (ordered by timestamps).
- When a rating is provided during validation, prediction strategy is invoked to provide a prediction for the current user and the current item. The average prediction error can be used a performance indicator of the attached prediction strategy.
- After the error has been calculated, the given rating is provided as feedback to the recommender system. The adaptive system collects the feedback as well as the current attributes of the system as instances.
- Whenever the collected instances reached the learning cycle's instance count (1000 instances for example), the prediction strategy of the system will be redesigned by the adaptive system according to the instances.

This way, when the next learning cycle is processed, the adaptive system has learned user

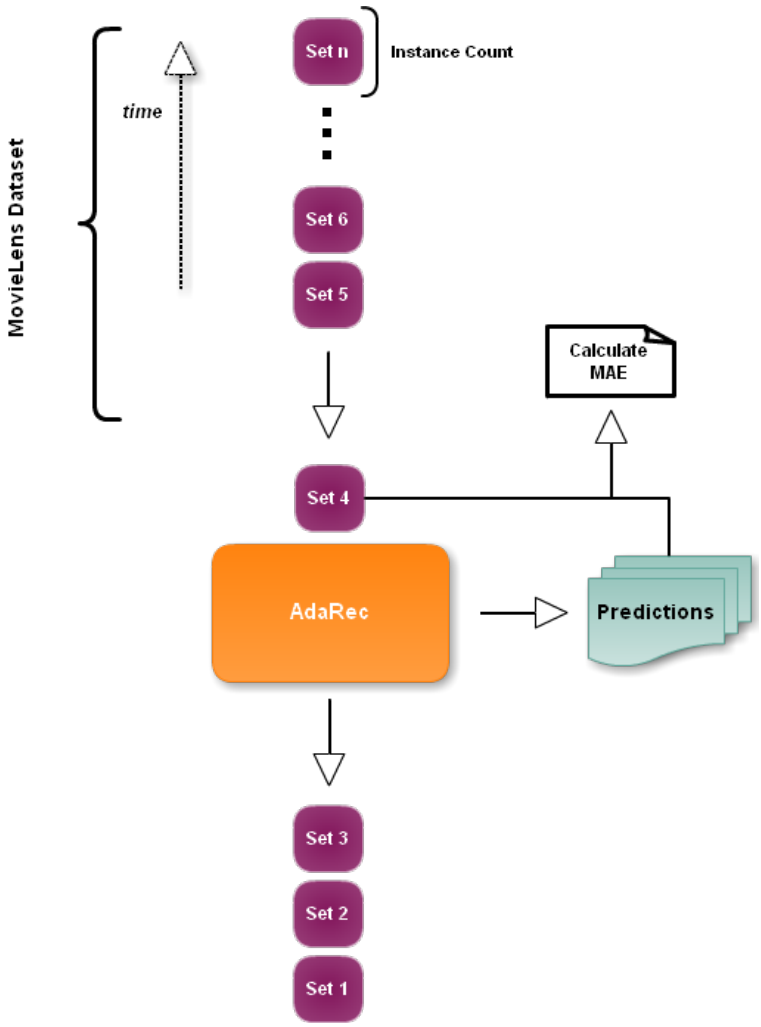


Figure 4.3: The MovieLens dataset splitting process & experimental setup.

preferences from all the previously processed ratings. This process is repeated for all ratings at both adaptive and non-adaptive (baseline) systems in the test set. At the end MAE is calculated by averaging absolute errors within the baseline and the adaptive system, as described above.

In order to train the recommender system, the MovieLens dataset is also divided into different *temporal sets* based on their distribution in time. Figure 4.3 shows the temporally split subsets of the MovieLens dataset. During testing the ratings of first sets are used for recommender engine training [30, 48].

The split of the dataset is performed by ordering the ratings table records according to their

timestamp values. The sorted whole ratings table is divided into different logical sets based on the learning cycle instance count. This clustered data is fed into the system as if real users provide feedbacks for the movies. Each set is used for learning for the next step.

The previous set serves as background data for the next set. The splitting process and collecting the MAE results of the predictions are plotted in Figure 4.3. After each set's processing is finished, AdaRec Framework redesigns itself. In our experiments we vary the instance count of the learning cycle (1K, 1.5K, 2K and 3K). The provided set is sparse when the instance count is small.

In the AdaRec, the decision tree induction is one of the fundamental classification method that is intensively used in our initial experiments. The decision tree induction method C4.5 (described in Section 2.2.1) which builds a decision tree and performs classification on the given data, is the first tested method during the initial experiments.

We conduct experiments on different instance-based learning algorithms that are supported by WEKA. These are; J48 (a C4.5 variant), BF-Tree (Breath First Tree) and Conjunctive Rules. These learning algorithms are described in Section 3.4.2.

- **J48:** WEKA's implementation of the decision tree learner based on the C4.5 decision tree algorithm.
- **BF-Tree:** A decision tree learner that uses a best first method of determining its branches.
- **Conjunctive Rules:** A simple rule learner that learns a set of simple conjunctive rules that can predict for numeric and nominal class labels.

In the adaptive system, C4.5, BF-Tree and Conjunctive Rules classifier algorithms (described in Section 4.3) are attached to the learning module and their results are recorded. We tuned the algorithms to optimize and configured to deliver the highest quality prediction without concerning for performance. The configurations of the learning algorithms are listed in Table 4.2. Since we used WEKA as our ML tool we also list the WEKA configuration of the algorithms in Table 4.2. In our experiment results we referred these algorithms as J48, BF-Tree and Conjunctive Rules.

As seen in the Table 4.2, in order to compare the performance of decision tree learners over

Table 4.2: Configurations of the used ML algorithms.

| ML Algorithm | Configuration Details | WEKA Configuration |
|---|---|---|
| J48 | Decision tree learner is used with, prunning strategy with no Laplace method used and sub-tree raising operation is considered while prunning. Minimum number of instances at per leaf is 2. | J48 -C 0.25 -M 2 |
| BF-Tree | Decision tree learner is used with, post-prunning strategy. Heuristic search is used for binary splitting for nominal attributes. Minimum number of instances at per leaf is 2. Gini index, which is a measure of statistical dispersion, is used for splitting criteria. | BF-Tree -C 0.25 -M 2 |
| Conjunctive Rules | Conjunctive rule learner is used with no attribute splits. Also pre-prunning is not used. Minimum total weight of the instances is 2.0. | ConjuctiveRule -N 3 -M 2.0 -P -1 -S 1 |

rule learner algorithm, we used very simple options for 'Conjunctive Rule' learner algorithm.

In our experiments we also collect the produced prediction strategies on each redesign. Some of the produced prediction strategies will be described in Section 4.4.

## 4.4 Experiment Results & Discussion

The purpose of this section is to compare the prediction accuracy of the implemented AdaRec Framework and the baseline system, which were described in previous chapters. Both of the two systems shall go through the same set of test cases with the same input conditions as mentioned above. Experiment environment and test cases are described in Section 4.3.

As mentioned in Section 4.2, prediction accuracy is calculated according to MAE, which is scaled to AdaRec's internal scale < 0 .. 1 >. MAE compares the actual rating a user gave to an item with the predictions made by a recommender. The lower the mean absolute error, the better an actual predictor can be classified.

In experimenting with the MovieLens dataset, we considered both the proposed method, called *AdaRec- adaptive system*, and the existing method, called *baseline*. In the experi-

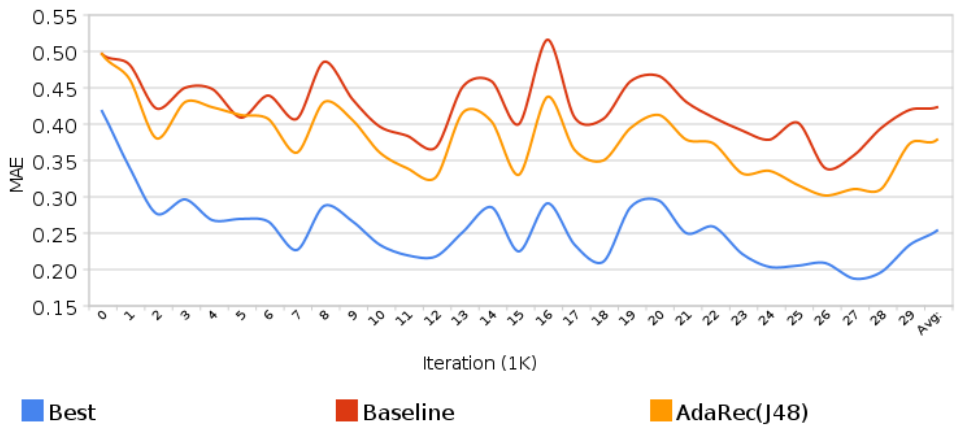


Figure 4.4: Quality of prediction (MAE) using AdaRec (attached J48 with 1000 instances) vs Baseline & Best.

ments as described above, different number of instances, such as 1K, 2K and 3K, are used. The purpose of different number of instances was to compare the influence of the instance size on algorithms at the same domain.

In order to compare our system against the best possible outcome, we also plot the result of the optimal MAE of the hybrid recommender in the current context at each run. The best MAE is referred to *best* in the charts, which is calculated by looping over each prediction algorithm. Naturally, this situation leads to linearly degraded performance ($O(N)$). *Best*, can be thought as the *globally optimal* accuracy of the prediction algorithms, which choses the best possible algorithm for each prediction. Therefore it is possible to compare the performance of the ML algorithms over the best possible result.

### 4.4.1 Comparison of ML Algorithms & Baseline

In this section, we present the experiment results of our proposed system AdaRec and baseline with the perspective of prediction accuracy. All of the results obtained during the experiments are in a form of the MAE, which is described in Section 4.2. The lower the MAE, the more accurately the recommendation system predicts user ratings.

Figure 4.4 presents the prediction quality (average MAE) results of our experiments, which used 1000 instances as learning cycle count, for the adaptive system as well as the original

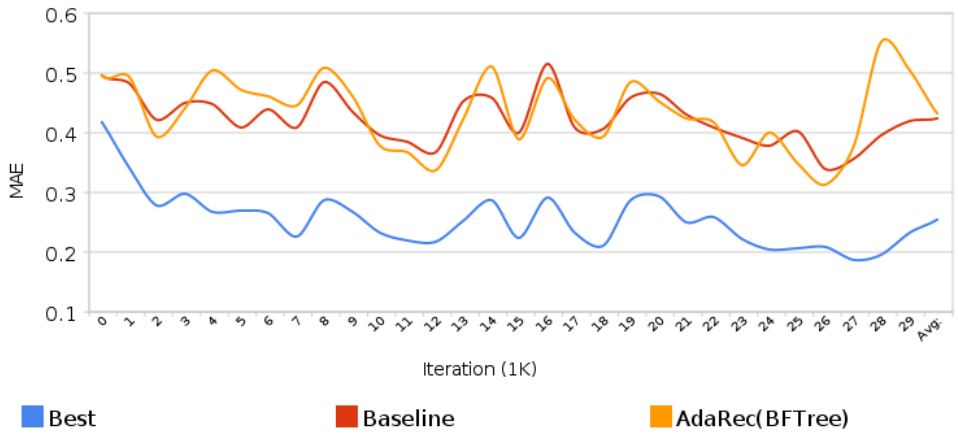


Figure 4.5: Quality of prediction (MAE) using AdaRec (attached BF-Tree with 1000 instances) vs Baseline & Best.

system referred to baseline. J48 algorithm is used in these experiments. In this chart, prediction quality is plotted for each of the runs. On each run adaptive system re-designs its prediction strategy according to the previous runs instances (feedbacks and results).

At first glance, we make two important observations from the Figure 4.4.

1. The prediction quality of the adaptive system performs better than the baseline. It can also observed from the chart that, even if both systems start with the same strategy, the adaptive system has adapt to the changing user behaviors.

2. It can be inferred from the figure that, AdaRec MAE decreases over time. This is the natural result of the learning algorithms. This shows that the used ML algorithm has the ability to improve with experience, to get better over time.

Figure 4.5 presents the average MAE results of AdaRec System with BF-Tree attached vs. Baseline & Best. The AdaRec system attached with BF-Tree as it's ML algorithm performs as good as the baseline but not better. This situation may occur due to the inadequateness of the chosen ML algorithm. In larger experiments or in different datasets this algorithm may produce more accurate results. Because the nature of the domain (movie, music, book etc.) has a major impact on the performance of the algorithms.

Figure 4.6 presents the average MAE of hundred runs for 1K instance size. In this experiment we evaluate the impact of more runs for 1K instance size. It can be observed from the chart

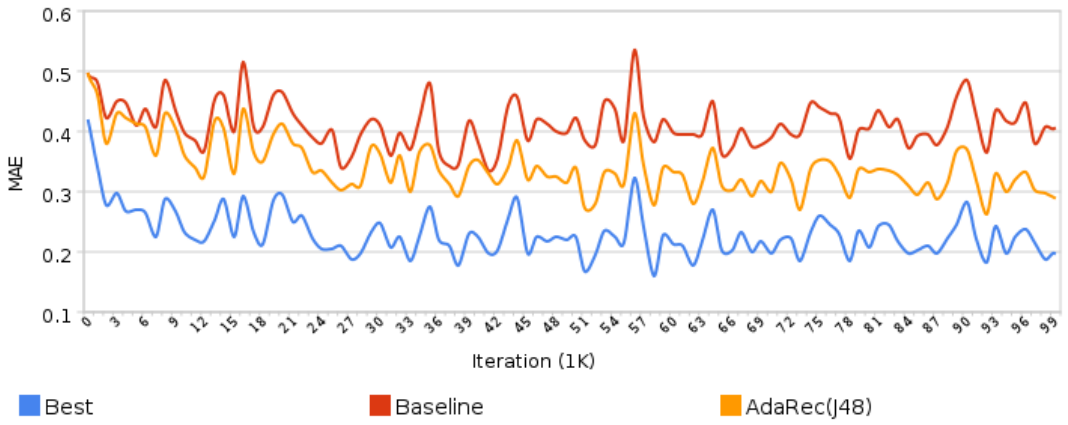


Figure 4.6: Quality of prediction (MAE) using AdaRec (attached BF-Tree with 1000 instances) vs Baseline & Best.

that changes in the MAE show the similar trends for both J48 and best. A harmony is achieved through time between the adaptive system and the best one. The curves are similar in such a way that if one of them has a good prediction accuracy in one run, the others also have the good accuracy for that run. Fluctuations in the chart shows similar trends.

In Figure 4.7, we draw the conclusion of the accuracy results of all used ML techniques for 1K instance size. Observed from the Figure 4.7 that the J48 attached AdaRec system performs better than the other systems. Also BF-Tree seems good enough to compete against the naive hybrid system. But BF-Tree algorithm needs some domain depended configuration adjustments. From the Figure 4.7, we also observe that the Conjunctive Rules algorithm under performs among other ML algorithms. The rule learner algorithm seems not stable as the decision tree learners.

Figure 4.8 presents the prediction accuracy of the AdaRec system and baseline for 2K instances. Also the best possible results of the predictions are plotted in the chart. Instance size of the learning cycle plays a very important role in our AdaRec system. As seen in Figure 4.8 the difference between the AdaRec and baseline are more sharp than the 1K instance which is plotted in the Figure 4.4. It can be inferred from the chart that J48 algorithm performs better with big instance sizes.

We carried out the same experiments with both 2K and 3K instance size. Figure 4.9 and Figure 4.10 shows the accuracy results of all used ML techniques for 2K and 3K instance

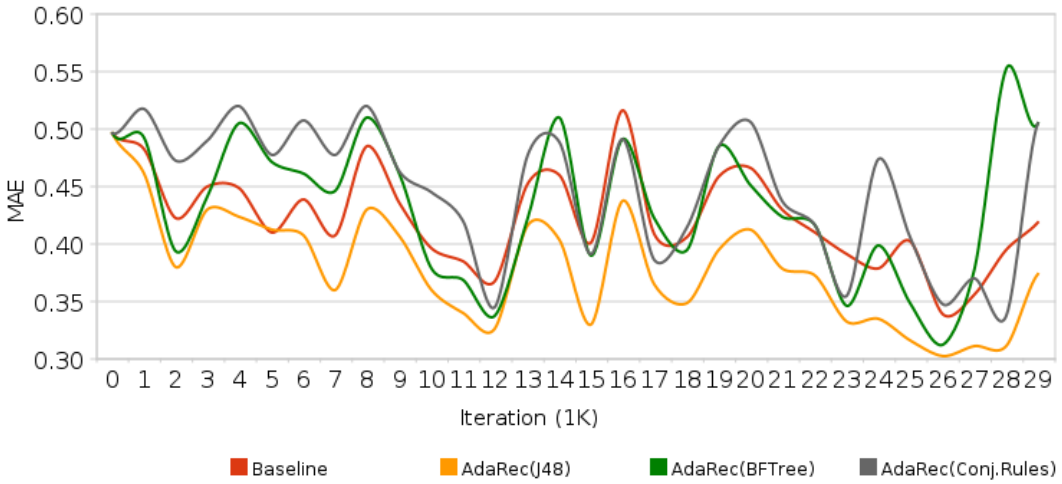


Figure 4.7: Quality of prediction (MAE) using AdaRec (attached J48, BF-Tree and Conjunctive Rules with 1000 instances) vs Baseline & Best.

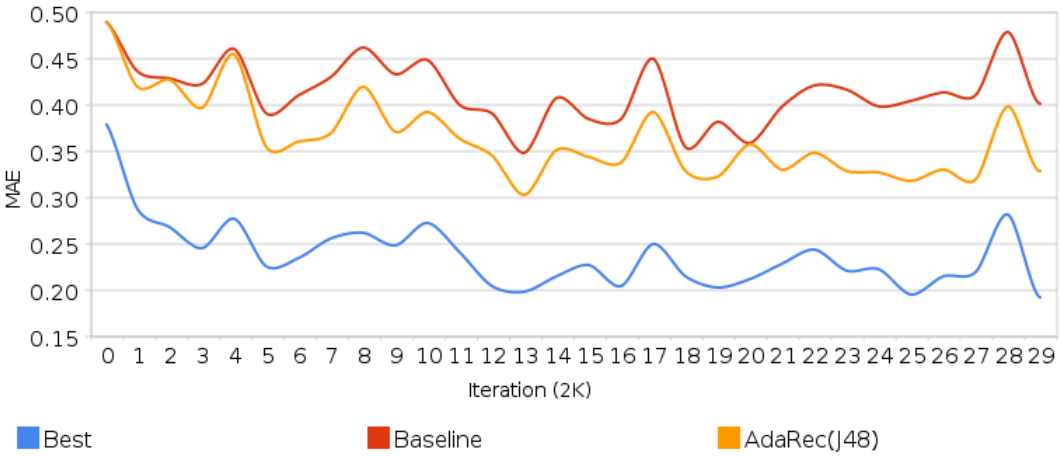


Figure 4.8: Quality of prediction (MAE) using AdaRec (attached J48 with 2000 instances) vs Baseline & Best.

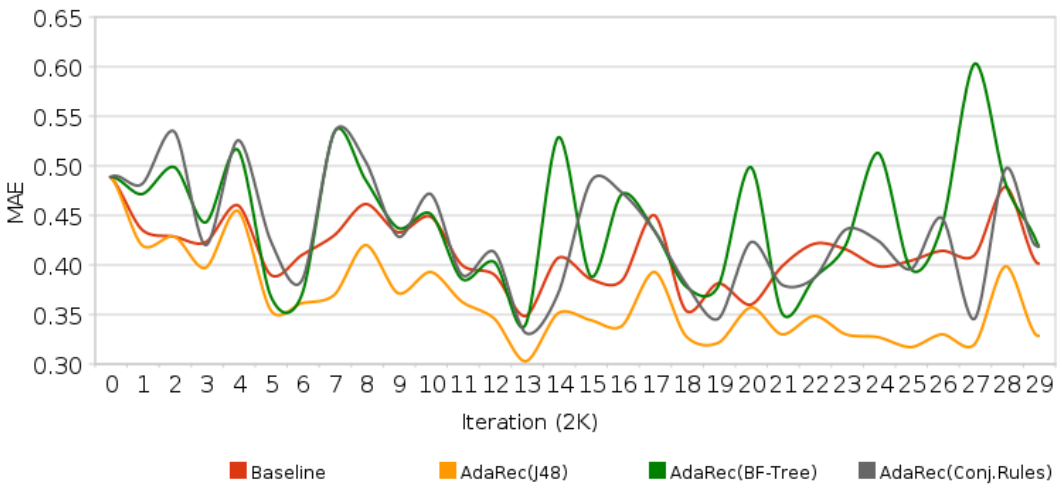


Figure 4.9: Quality of prediction (MAE) using AdaRec (attached J48, BF-Tree and Conjunctive Rules with 2000 instances) vs Baseline & Best.

sizes respectively. From these figures, we observe that AdaRec with well configured ML algorithm, outperforms naive hybrid system. This observation again proves the advantage of our proposed system.

In order to determine the impact of the instance size, we carried out experiments where we varied the value of instance size (1K, 1.5K, 2K and 3K). For each of these training set/instance size values we ran our experiments. Figure 4.11 presents the whole picture of the adaptive system's performance results. From the plot we observe that the quality of MAE increases as we increase the instance size.

Figure 4.11 summarizes the performance results of different instance counts. We plot the average MAE values of different instance sizes. From the plot we see that using the *J48* as the plugged learning algorithm, the MAE is substantially better than the baseline. This is due to the reason that with the adaptive approach the recommender system adapts itself to the current context. Another important observation from the chart that as we increase the instance size of algorithms the quality tends to be superior (decreased MAE). In case of other algorithms it is expected that increasing the number of instances would mean small MAE values. The same trend is observed in the case of 2K and 3K instances.

The average MAE results of the experiments for different instance sizes are given in Table 4.3. From the results, we observe that all ML algorithms performs better with 3K instance

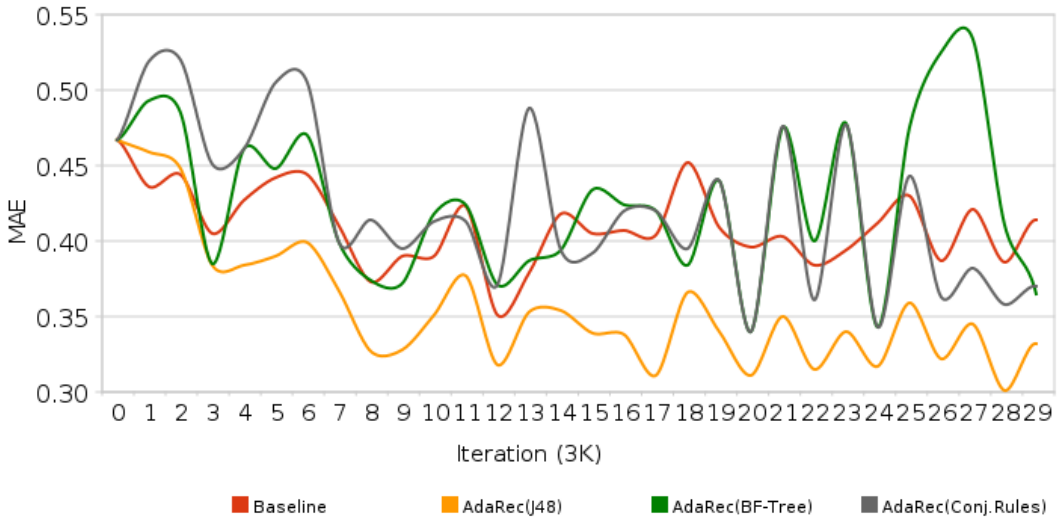


Figure 4.10: Quality of prediction (MAE) using AdaRec (attached J48, BF-Tree and Conjunctive Rules with 3000 instances) vs Baseline & Best.

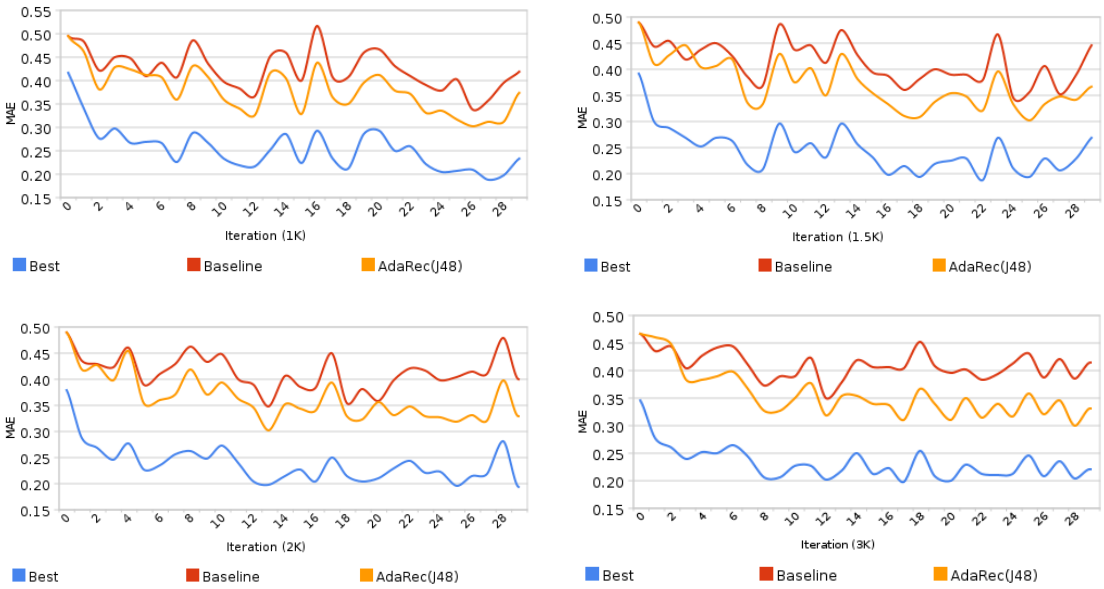


Figure 4.11: Comparison of the different instance sizes. 1K, 1.5K, 2K and 3K instances are used for learning cycle.

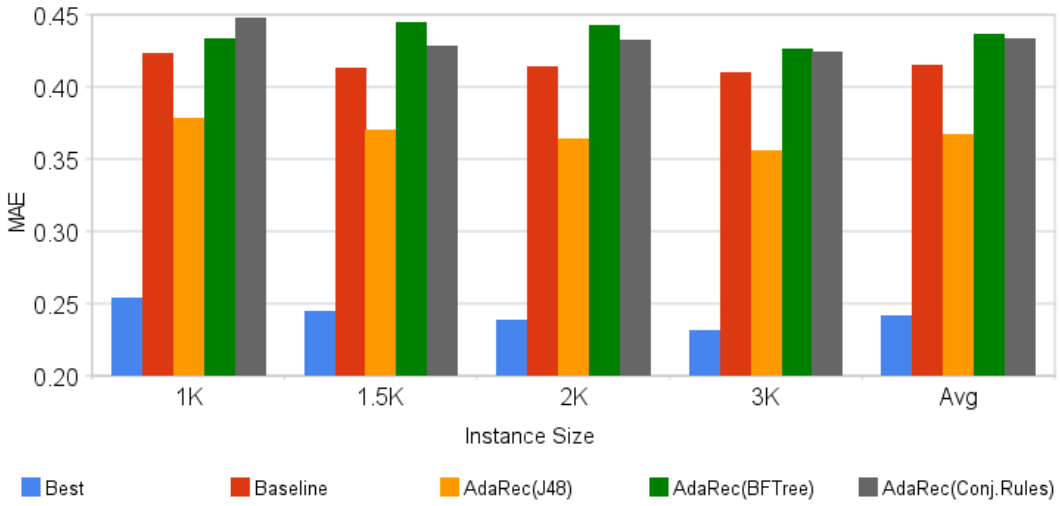


Figure 4.12: Prediction accuracy comparison of the AdaRec (attached J48, BF-Tree and Conjunctive Rules) vs Baseline & Best for different instance sizes.

size. This observation again proves that as we increase the instance size of algorithms the quality tends to be superior. As showed in the table J48 algorithm performs better results than the baseline and the other algorithms. We observe that a good configured algorithm achieves better accuracy than the static one.

Table 4.3: Average MAE of the used ML algorithms for different instance sizes.

| **ML Algorithm** | **1K** | **2K** | **3K** |
|---|---|---|---|
| Best | 0.254437 | 0.238296 | 0.231473 |
| Baseline | 0.423804 | 0.413809 | 0.410117 |
| AdaRec(J48) | 0.378762 | 0.364287 | 0.356275 |
| AdaRec(BFTree) | 0.433188 | 0.443327 | 0.426553 |
| AdaRec(Conj.Rules) | 0.447756 | 0.432602 | 0.424115 |

Figure 4.12 also presents the prediction accuracy results of the different instance sizes. In the figure, we also plot the overall performance of ML algorithms as average.

The results also show that, when using well tuned algorithms, the adaptive system is stable (better than the baseline) in obtaining the average prediction accuracy. This durability, which may be called as the impact of learning. This improvement is a natural result of the adaptation. In order to adapt recommender engine to the current trends, the learning module re-designs

the prediction strategy. The learning ability supports the recommender system adaptation to the changes. The more recommender system adapts to the domain & the users, it produces better results.

### 4.4.2 Created Prediction Strategies

In this section, we present some of the produced prediction strategies that are created by different ML algorithms (J48, BF-Tree and Conjunctive Rules) during the experiments. ML algorithms, which were tested in our experiments, basically produce two type of prediction strategies; decision trees and decision rule sets.

Learning Module in AdaRec creates prediction strategy for each learning cycle, by using the attached ML algorithm, according to the training set of instances. The whole process of the Learning Module was described in Section 3.3 in detail. Prediction strategy, as described in Section 3.2.3 consists of prediction algorithms & switching rules associated with them. The prediction strategy defines the switching behavior of the recommender engine.

Figure 4.13 shows a sub-tree of the created prediction strategy. This sample prediction strategy is created by J48 ML algorithm at the $15^{th}$ run of the 1K instance experiments. In our experiments we observed that J48 ML algorithm produces bigger trees than the BF-Tree.

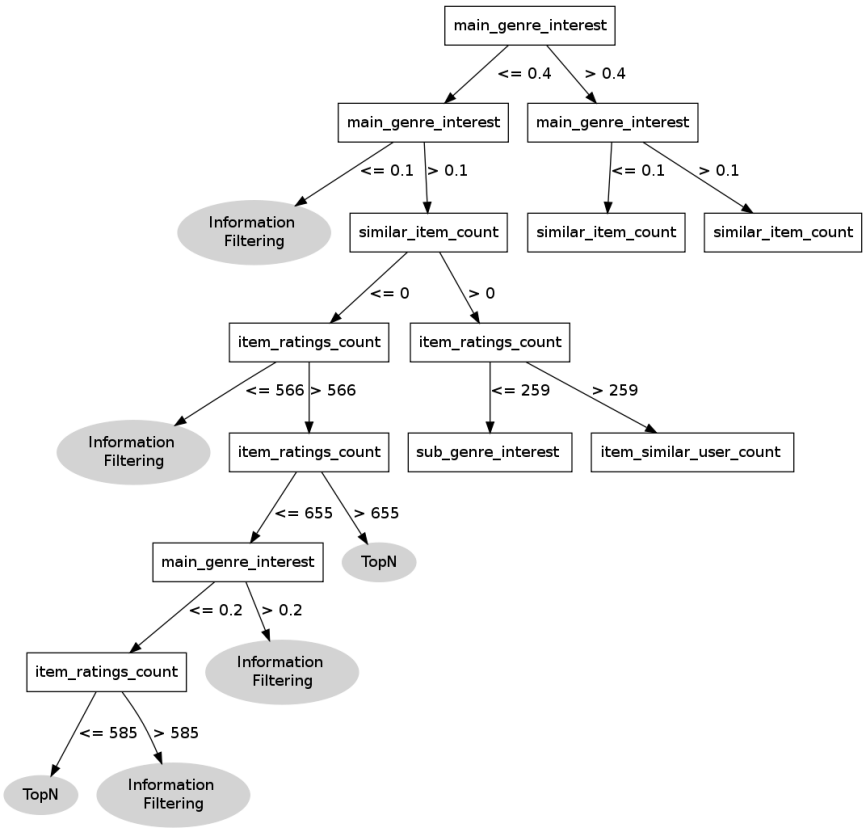


Figure 4.13: A sub-tree of a prediction strategy that is created by J48 ML algorithm.

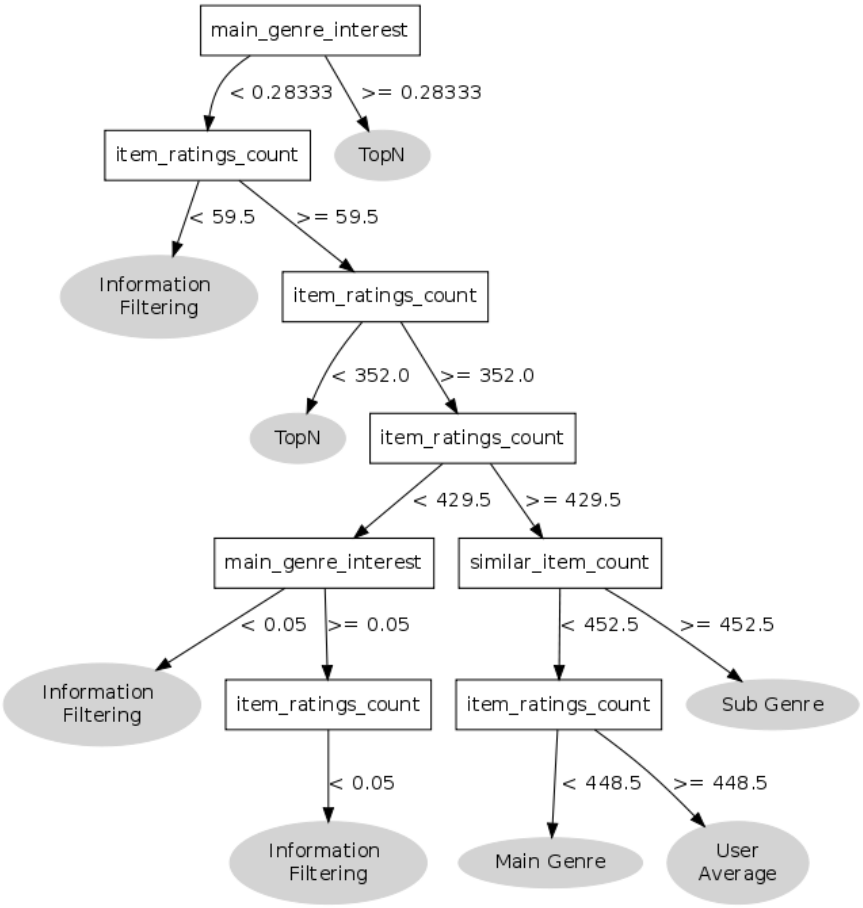


Figure 4.14: A prediction strategy that is created by BF-Tree ML algorithm.

Figure 4.14 shows the prediction strategy produced by the BF-Tree. This sample prediction strategy is created by BF-Tree ML algorithm at the $7^{th}$ run of the 1K instance experiments. It can be observed from the tree that BF-Tree produced smaller prediction strategies than the J48.

Conjunctive Rule learner, as described in Section 2.2, produce simple single rules for prediction strategies. The strategy that is created by Conjunctive Rule learner ML algorithm at the $10^{th}$ run of the 3K instance experiments is given below.

```
Single conjunctive rule learner:
--------------------------------
(main_genre_interest <= 0.283334) and (similar_item_count <= 4.5) => technique = informationFiltering

Class distributions:
Covered by the rule:
informationFiltering socialFiltering topN mainGenreLMS userAverage cbr subGenreLMS
0.374502 0.10093 0.354582 0.09429 0.037185 0 0.038513

Not covered by the rule:
informationFiltering socialFiltering topN mainGenreLMS userAverage cbr subGenreLMS
0.249399 0.105052 0.35846 0.186848 0.056937 0.001604 0.0417
```

# CHAPTER 5

# CONCLUSION & FUTURE WORK

This concluding chapter summarizes the contributions of the thesis and proposes possible directions for future work.

## 5.1 Conclusions

Traditional hybrid recommender systems typically follow a manually created fixed prediction strategy in their decision making process. In this thesis, we introduced an adaptive hybrid recommender system, called AdaRec, that to combines several recommender techniques (algorithms) in which the combination parameters are learn and dynamically updated from the results of previous predictions. Research study shows that traditional static hybrid recommender systems suffer from changing user preferences. In order to improve the recommendation performance, we handle domain drifts in our approach.

In AdaRec architecture, which is built upon an open-source recommender system called Duine, we aim to improve prediction accuracy through adapting the domain trends. The Learning Module re-designs its prediction (switching) strategy according to the performance of prediction techniques. As a result, the system becomes adaptive to the application domain, and the performance of recommendation increases as more data are accumulated. We believe that with this adaptive learning module, a traditional hybrid recommender should have higher chance to allow its users to efficiently obtain an accurate and confident decision.

We have used and tested several ML algorithms in the Learning Module component. Initially, we used simple decision tree induction methods such as Breath First Tree (BF-Tree). Later on we used WEKA implementation of the well-known C4.5 decision tree induction algo-

rithm, called J48. We also include a rule learner, called Conjunctive Rule Learner, in order to compare the results.

In the experiments we used MovieLens 1M dataset. At these experiments, the proposed adaptive system adapts outperforms the baseline (naive hybrid system). By using well-configured ML algorithms with MovieLens dataset, AdaRec provides significantly more accurate predictions than the static hybrid recommender system.

## 5.2 Future work

During the research for this thesis, new ideas have arisen and new research lines have been investigated with potential interest for the continuation of this research work. There has several future work directions we can follow, some of them are listed below.

- **Different Datasets:** As described in Chapter 4, initial experimental results with MovieLens dataset show potential impacts of AdaRec Framework. Therefore, for the next step, we plan to deeply test the learning module with various heterogeneous datasets. It would be also interesting to examine the different domains other than movie, such as music, book, etc. As the first trial publicly available datasets would be used in the experiments. Some of these are; Jester Joke dataset[1], ChceteMe dataset[2], LibimSeti dataset[3]. We believe that additional testing over a longer period and with real system & real users will quietly improve the system.

- **Different ML Algorithms:** Learning Module is developed on pluggable architecture. In AdaRec Framework it is possible to attach different ML algorithms through a common interface. We plan to test different learning methods, such as support vector machines, artificial neural networks etc., in our experimental framework and compare their results with other ML techniques. Also it would be interesting to compare the different ML algorithms on different configurations.

- **More than one metric:** In our experiments we used Mean Absolute Error (MAE) metric for prediction accuracy. However it would be beneficial to use more than one metric

[1] http://goldberg.berkeley.edu/jester-data/
[2] http://chceteme.volny.cz/
[3] http://www.libimseti.cz/

in the experiments. *Coverage metric* may be used in conjunction with MAE. Also in our system in order to compare the ML algorithm's efficiency, produced prediction strategies may be compared by using a common metric, such as tree size, leaf size, rule count etc.

- **Improving the performance:** Performance is the another big issue for recommender systems. During this thesis work, we noted that one of the key aspects in successful practical recommender systems research is providing fast predictions. Recommender Systems are usually used in websites such as Amazon[4], Last.fm[5],del.icio.us[6], . Since most of these system's recommendations are time dependent, a reasonable performance improvement should be made on our proposed AdaRec Framework.

- **Analyzing domain trends:** In AdaRec Framework we analyze the dynamics and changing trends of the domain in order to adapt the changing user preferences. It would be interesting to analyze the changes in the domains. For example publicly available datasets, such as MovieLens dataset, may be used to show the changes in the users preferences.

- **Contribution to open-source:** As the future work, the AdaRec Framework may be an open-source project. Also it is possible to introduce a plug-in (such as adaptation plugin) to open-source Duine Framework in order to get more feedbacks from the community.

[4] `http://www.amazon.com`
[5] `http://www.last.fm`
[6] `http://del.icio.us`

# REFERENCES


[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[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*, 17(6):734–749, 2005.

[3] A. Ansari, S. Essegaier, and R. Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, 2000.

[4] K. Arnold, J. Gosling, and D. Holmes. The Java programming language. *Reading, Massachusetts*.

[5] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):72, 1997.

[6] X. Bao, L. Bergman, and R. Thompson. Stacking recommendation engines with additional meta-features. In *Proceedings of the third ACM conference on Recommender systems*, pages 109–116. ACM, 2009.

[7] R. Bell, Y. Koren, and C. Volinsky. The bellkor solution to the netflix prize. *KorBell Team's Report to Netflix*, 2007.

[8] J. Bennett. Independent study report: A survey of som and recommender techniques. *A Research Report,, Golisano College of Computing and Information Science Rochester Institute of Technology*, pages 5–53.

[9] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007. Citeseer, 2007.

[10] D. Billsus and M. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2):147–180, 2000.

[11] J. Breese, D. Heckerman, C. Kadie, et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[12] R. Burke. Hybrid recommender systems: A comparative study. Technical report, CTI Technical Report 06-012. 2006.(Available at http://www. cs. depaul. edu/research/technical. asp.

[13] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[14] Y. Chien and E. George. A bayesian model for collaborative filtering. In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*, 1999.

[15] Y. Cho, J. Kim, and S. Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329–342, 2002.

[16] K. Clink. Del. icio. us. *Reference Reviews*, 22(5), 2008.

[17] M. de Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Preference Learning in Recommender Systems. *PREFERENCE LEARNING*, page 41, 2009.

[18] J. Delgado, N. Ishii, and T. Ura. Content-based collaborative information filtering: Actively learning to classify and recommend documents. *Cooperative Information Agents II Learning, Mobility and Electronic Commerce for Information Discovery on the Internet*, page 206, 1998.

[19] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, 29(2):103–130, 1997.

[20] Duine recommender - telematica instituut-novay. Last accessed 01 Aug 2010.

[21] J. Furnkranz and G. Widmer. Incremental reduced error pruning. In *Proceedings of the Eleventh International Conference on Machine Learning*, volume 11, 1994.

[22] L. Getoor and M. Sahami. Using probabilistic relational models for collaborative filtering. In *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*. Citeseer, 1999.

[23] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):70, 1992.

[24] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[25] E. Gstrein, F. Kleedorfer, R. Mayer, C. Schmotzer, G. Widmer, O. Holle, and S. Miksch. Adaptive personalization: A multi-dimensional approach to boosting a large scale mobile music portal. In *Fifth Open Workshop on MUSICNETWORK: Integration of Music in Multimedia Applications*. Citeseer.

[26] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 117–124, New York, NY, USA, 2009. ACM.

[27] J. Herlocker. *Understanding and improving automated collaborative filtering systems*. PhD thesis, Citeseer, 2000.

[28] J. Herlocker and J. Konstan. Content-independent task-focused recommendation. *IEEE Internet Computing*, pages 40–47, 2001.

[29] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[30] Imdb (internet movie database). http://www.imdb.com/. Last accessed 06 June 2010.

[31] D. Jensen and P. R. Cohen. Multiple comparisons in induction algorithms. In *Machine Learning*, pages 309–338, 1998.

[32] R. Johnson, J. Hoeller, A. Arendsen, and R. Thomas. *Professional Java Development with the Spring Framework*. Wiley-India, 2009.

[33] K. Karimi and H. Hamilton. Finding temporal relations: Causal bayesian networks vs. C4. 5. *Foundations of Intelligent Systems*, pages 266–273, 2009.

[34] J. Kim, B. Lee, M. Shaw, H. Chang, and M. Nelson. Application of decision-tree induction techniques to personalized advertisements on Internet storefronts. *International Journal of Electronic Commerce*, 5(3):45–62, 2001.

[35] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):87, 1997.

[36] B. Krulwich and C. Burkey. Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the AAAI spring symposium on machine learning in information access*, pages 100–112, 1996.

[37] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[38] W. Lee. Collaborative learning for recommender systems. In *Machine Learning-International Workshop*, pages 314–321. Citeseer, 2001.

[39] Q. Li and B. Kim. Clustering approach for hybrid recommender system. 2003.

[40] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[41] R. Michalski, J. Carbonell, and T. Mitchell. *Machine learning: An artificial intelligence approach*. Morgan Kaufmann Pub, 1986.

[42] S. E. Middleton. *Capturing knowledge of user preferences with recommender systems*. PhD thesis, University of Southampton, May 2003.

[43] S. E. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):88, 2004.

[44] B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl. MovieLens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, page 266. ACM, 2003.

[45] M. Montaner, B. López, and J. D. L. Rosa. A taxonomy of recommender agents on the internet. *Artificial intelligence review*, 19(4):285–330, 2003.

[46] M. Mortensen. Design and evaluation of a recommender system. 2007.

[47] Movielens dataset. http://www.grouplens.org/node/73. Last accessed 01 July 2010.

[48] Netflix dataset. http://www.netflixprize.com/download/. Last accessed 10 May 2010.

[49] The Netflix prize. http://www.netflixprize.com/. Last accessed 10 May 2010.

[50] Novay. Duine recommender — telematica instituut-novay, 2010. [Online; accessed 1-July-2010].

[51] D. Oard. The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178, 1997.

[52] D. Pennock, E. Horvitz, S. Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the 16th conference on uncertainty in artificial intelligence*, pages 473–480. Citeseer, 2000.

[53] E. Perik, B. De Ruyter, P. Markopoulos, and B. Eggen. The sensitivities of user profile information in music recommender systems. *Proceedings of Private, Security, Trust*, pages 137–141, 2004.

[54] M. Porter. An algorithm for suffix stripping. 1997.

[55] J. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[56] 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*, pages 175–186. ACM, 1994.

[57] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[58] B. M. Sarwar, J. A. Konstan, A. Borchers, J. L. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. pages 345–354. ACM Press, 1998.

[59] T. Segaran. *Programming collective intelligence: Building smart Web 2.0 applications*. O'Reilly Media, Inc., 2007.

[60] M. Setten, J. Reitsma, and P. Ebben. Duine Toolkit–user manual, 2003.

[61] M. V. Setten. *Supporting People in Finding Information- Hybrid Recommender Systems and Goal Based Structuring*. Telematica instituut fundamental research series no:016, Telematica Instituut, November 2005.

[62] M. V. Setten, M. Veenstra, and A. Nijholt. Prediction strategies: Combining prediction techniques to optimize personalization. In *Proceedings of the workshop Personalization in Future TV*, volume 2. Citeseer, 2002.

[63] U. Shardanand and P. Maes. Social information filtering: algorithms for automating ẅord of mouth.̈. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[64] R. Sinha and K. Swearingen. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*, page 831. ACM, 2002.

[65] K. Swearingen and R. Sinha. Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems*. Citeseer, 2001.

[66] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):62, 1997.

[67] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, pages 112–125, 1998.

[68] J. Wang, A. D. Vries, and M. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, page 508. ACM, 2006.

[69] Wikipedia. Singular value decomposition — wikipedia, the free encyclopedia, 2010. Last accessed 06 July 2010.

[70] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005.

[71] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[72] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model.

[73] C. Ziegler. *Towards Decentralized Recommender Systems*. Freiburg i. br, Albert-Ludwigs-Universität Freiburg,Germany, June 2005.

# APPENDIX A

# PUBLICATIONS

- Fatih Aksel and Ayşenur Birtürk, Enhancing Accuracy of Hybrid Recommender Systems through Adapting the Domain Trends. *The International Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies (PRSAT'2010) in conjunction with the 4th ACM Conference on Recommender Systems (ACM RecSys 2010)*, 26-30 September 2010, Barcelona, Spain.

- Fatih Aksel and Ayşenur Birtürk, An Adaptive Hybrid Recommender System that Learns Domain Dynamics.*International Workshop on Handling Concept Drift in Adaptive Information Systems: Importance, Challenges and Solutions (HaCDAIS'2010) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2010 (ECML PKDD 2010)*, 20-24 September 2010, Barcelona, Spain.

- Fatih Aksel and Ayşenur Birtürk, An Improved Switching Hybrid Recommender System that Adapts Domain Dynamics. *accepted to Preference Learning Workshop (PL'10) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2010 (ECML PKDD 2010)*, 20-24 September 2010, Barcelona, Spain (withdrawn paper).

- Fatih Aksel and Ayşenur Birtürk, A Hybrid Movie Recommendation System. *eChallenges e-2009 Conference Proceedings, Paul Cunningham and Miriam Cunningham (Eds), IIMC International Information Management Corporation,* , 2009, ISBN: 978-1-905824-13-7.