CROSSING: A FRAMEWORK TO DEVELOP KNOWLEDGE-BASED RECOMMENDERS IN CROSS DOMAINS

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

BY

MUSTAFA AZAK

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

FEBRUARY 2010

Approval of the thesis:

CROSSING: A FRAMEWORK TO DEVELOP KNOWLEDGE-BASED RECOMMENDERS IN CROSS DOMAINS

submitted by **MUSTAFA AZAK** 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. Müslim Bozyiğit _____
Head of Department, **Computer Engineering**

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

Examining Committee Members:

Assoc. Prof. Dr. Nihan Kesim Cicekli _____
Computer Engineering Dept., METU

Dr. Ayşenur Birtürk _____
Computer Engineering Dept., METU

Prof. Dr. Mehmet R. Tolun _____
Computer Engineering Dept., Çankaya University

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

Assoc. Prof. Dr. Ali Doğru _____
Computer Engineering Dept., METU

Date: 03.02.2010

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 : Mustafa AZAK

Signature :

ABSTRACT

CROSSING: A FRAMEWORK TO DEVELOP KNOWLEDGE-BASED RECOMMENDERS IN CROSS DOMAINS

Azak, Mustafa
M.S., Department of Computer Engineering
Supervisor: Dr. Ayşenur Birtürk

February 2010, 97 pages

Over the last decade, excess amount of information is being provided on the web and information filtering systems such as recommender systems have become one of the most important technologies to overcome the 'Information Overload' problem by providing personalized services to users. Several researches have been made to improve quality of recommendations and provide maximum user satisfaction within a single domain based on the domain specific knowledge. However, the current infrastructures of the recommender systems cannot provide the complete mechanisms to meet user needs in several domains and recommender systems show poor performance in cross-domain item recommendations. Within this thesis work, a dynamic framework is proposed which differs from the previous works as it focuses on the easy development of knowledge-based recommenders and it proposes an intensive cross domain capability with the help of domain knowledge. The framework has a generic and flexible structure that data models and user interfaces are generated based on ontologies. New recommendation domains can be integrated to the framework easily in order to improve recommendation diversity. The cross-domain recommendation is accomplished via an abstraction in domain features if the direct matching of the domain features is not possible when the domains are not very close to each other.

Keywords: Web Personalization, Information filtering, Recommender Systems, Cross Domain Recommendation, Recommendation Frameworks, Recommender Engines.

ÖZ

CROSSING: ÇAPRAZ TAVSİYE ALANLARINDA BİLGİ TABANLI TAVSİYE SİSTEMİ GELİŞTİRMEK İÇİN BİR ÇATI

Azak, Mustafa

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Dr. Ayşenur Birtürk

Şubat 2010, 97 sayfa

Son on yıl içinde, çok fazla miktarda bilgi internete sağlanmakta ve bunun sonucunda "Aşırı Bilgi Yüklenmesi" sorunu ortaya çıkmaktadır. Tavsiye sistemleri gibi bilgi filtreleme sistemleri ise kullanıcılara kişiselleştirilmiş hizmetler sunarak bu sorunu aşmada en önemli teknolojileriden biri olmuştur. Belirli bir öneri alanı içinde bu alana ait bilgilere dayanılarak yapılan tavsiyelerin kalitelerini artırmak ve maksimum kullanıcı memnuniyetini sağlamak için bir çok çalışma gerçekleştirilmiştir. Ancak, mevcut tavsiye sistemlerinin altyapıları farklı öneri alanlarındaki kullanıcı ihtiyaçlarını karşılamak için tam bir mekanizma sağlayamamakta ve öneri alanları arasında gerçekleştirilen çapraz tavsiyelerde kötü bir performans göstermektedir. Bu tez çalışması içinde, önceki çalışmalardan farklı olarak bilgi tabanlı tavsiye sistemlerinin kolaylıkla geliştirilebilmesine odaklanılmış ve etkin çapraz tavsiye yeteneğine sahip dinamik bir tavsiye sistemi çatısı önerilmiştir. Çatı, jenerik ve esnek bir yapıya sahiptir, veri modelleri ile kullanıcı arayüzleri ontolojiler baz alınarak oluşturulmaktadır. Tavsiye çeşitliliği artırmak amacıyla yeni öneri alanları kolaylıkla sisteme eklenebilir. Eğer birbirlerine yakın olmayan öneri alanlarının özelliklerinin doğrudan eşleşmesi mümkün değilse çapraz tavsiyeler bu özelliklerin bir üst seviyede soyutlaması yoluyla gerçekleştirilir.

Anahtar Kelimeler: Web Kişiselleştirme, Bilgi Filtreleme, Tavsiye Sistemleri, Alanlar Arası Tavsiyeler, Tavsiye Sistemleri Çatıları, Tavsiye Motorları.

To My Family...

ACKNOWLEDGMENTS

I would like to present my deepest thanks to my thesis supervisor Dr. Ayşenur Birtürk for his valuable guidance, motivation and support throughout this thesis study.

I am very grateful to my family for all their patience and tolerance.

I would thank the Scientific and Technological Research Council of Turkey (TUBİTAK) for providing the financial means throughout this study.

My special thanks go to Fatih Aksel, Abdullah Çetin Çavdar and Damla Sivrioğlu for their help and support to complete this work and to all my friends who gave me support whenever I needed.

Lastly, thanks to anonymous reviewers of eChallenges 2009, WEBIST 2010, and ACM RecSys'09 Conferences and IJCAI 2009 7'th Workshop on Intelligent Techniques for Web Personalization and Recommendation.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CB	Content Based
CF	Collaborative Filtering
IF	Information Filtering
MAE	Mean Absolute Error
RS	Recommender Systems
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

With the increasing number of Web usage, the amount of information being provided on the Internet has become very difficult to handle and this causes the "Information Overload" problem [1]. Individuals have to spend so much time in searching in order to reach the valuable data. The need for providing filtered, relevant and useful data led to the development of information filtering and personalization techniques. Recommender Systems have become one of the most important technologies to overcome the 'Information Overload' problem.

Recommender Systems exploit the users' past experiences and preferences, build the user models and identify users' behaviors to predict their future needs and try to generate personalized recommendations [2]. Over the last decade, several researches have been made to improve the recommendation methods and user modeling techniques in order to extend quality of recommendations and provide maximum user satisfaction [3]. Therefore, exploiting the user profiles of social networking / Web 2.0 style sharing platforms [4] and applying advanced recommendation methods, recommender systems have achieved remarkable practical and commercial success. Recommender systems might be regarded as business tools for generating competitive advantages. Commercial sites use them

to suggest products to users and aim to increase their sells by providing the items that users might be most probably interested in among available products. Amazon [5], Last.fm [6], Netflix [7] and MovieLens [8] are the most successful and popular examples of the recommender systems.

Although web-based social and commercial networks provide variety of interests for users in various domains such as books, movies, music and games; most recommendation systems currently provide recommendations within a single domain based on the specific knowledge about the related domain. They suggest products based on the following criteria; popularity on the market, demographic information of users, modeling of previous preferences and buying behavior, product similarities, community critiques and evaluations. These techniques have been matured over the time and proved their success concerning single domains. However, the current infrastructures of the recommender systems cannot provide the complete mechanisms to meet user needs in different domains and recommender systems show poor performance in cross-domain item recommendations.

Cross-Domain recommendations play an important role for the success in the cross selling markets. Cross-Selling is the term for the practice of suggesting related products or services to a customer who is considering buying something [9]. In addition, cross-selling provides the user loyalty and reliance on the seller and decrease the customer switching behaviors to a competitor. Cross-Domain recommendations suggest the related items from different domains, provides strong capabilities to meet variety of interests and increase the user satisfaction which is the main goal of recommender systems. Current systems only consider the statistical analysis on the market and try to make use of relations between popular items without personalized recommendations.

Therefore, there is a lack of cross-domain recommenders which can make successful personalized recommendations in cross-domains. Moreover, there are few frameworks to develop recommender systems and very few of them support the cross-domain recommendations. In addition, the current infrastructures are very static that the systems are highly dependent on their recommendation domains and it is very difficult to improve their abilities to develop new features to adapt and cover changeable needs of users.

In this thesis, we propose a dynamic framework, which enables addition of new recommendation domains and provides the development of knowledge based cross domain recommenders as well as single domain recommenders.

The addition of a new recommendation domain is crucial because it enables to improve the recommender system's capability and evolve the system according to new user needs. In order to provide dynamic domain additions to framework, our system considers each domain as a pluggable component that provides the required domain data via well defined interfaces. The domain data includes user data, community data, items and domain knowledge. Domains may have different data structures and each domain has a specific ontology. Therefore, to be able to provide required interaction with the framework interfaces, each domain needs to have an adapter system to transform its data into a common vocabulary and structure. Considering the general ontologies and XML schema of the framework; an adapter system is employed to deal with ontology mapping and type conversions of the features. After creating an adapter system for the domain, it can be easily integrated with the framework.

Providing accurate predictions and useful recommendations among the integrated domains are the other important capabilities of the proposed framework. The framework provides knowledge based recommender development with a hybrid approach for generating recommendations. The recommendation engine has

different types of recommendation strategies which are applied depending on the available data about the user and the items in domains. Collaborative and content based strategies use feature-weighted models in similarity calculations. The initial feature weights for users and items are determined with domain knowledge for each domain. These initial weights play an important role for accurate recommendation especially for the cold start problems. While the user continues to receive recommendations, the weights are updated according to the reactions of the user for the recommendations which are captured by the feedback mechanism. Therefore, personalized feature weights are learned for each user individually and they increase the accuracy of the recommendations.

In addition, we have also high level abstractions and relations between domains. The relational information about domains can be dynamically added to the framework by defining the rule sets which provide inter-domain knowledge between two specific domains. The inter-domain knowledge is used for feature mapping between domains and it affects the weights of features in target domain. A knowledge base is constructed using these mappings and recommender engine benefits from this knowledge base by finding the relational rules between different domains to generate more accurate recommendations in cross-domains.

The remainder of the thesis is organized as follows.

Chapter 2 – Background and Related Work gives an overview of recommender systems in literature. It introduces the main paradigms of recommender system design, and identifies their approaches in detail. It also describes current recommendation system frameworks and their structures. Pros and cons of these frameworks are discussed.

Chapter 3 – CrosSing: A Framework to Develop Knowledge-Based Recommenders in Cross Domains presents our framework, CrosSing. The overview of the

framework is presented, the data representation scheme is explained, and the framework components are described in detail.

Chapter 4 – Evaluation of the system provides a summary of testing methods of CrosSing framework and presents experimental results.

Chapter 5 – Conclusions discuss the concluding remarks and explains the future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter aims to present an overview of recommender systems and their recommendation algorithms. It also includes the general concepts of the frameworks and discusses some of the important recommendation systems framework examples.

2.1 Recommender Systems

Recommender Systems (RS) are designed to help individuals to deal with information overload, incomplete information, and enable them to make evaluative decisions [1]. More general definition of the recommender systems is that RS represent a class of systems those have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options [10]. They use various personalization techniques and most common way to use of the opinions of a community of users which provides identifying the content of interest very effectively from a potentially overwhelming set of choices. Recommender systems also examine the model of the user, the user profile, and compare it against a description of available items to decide which ones should be recommended to the user and which ones must be filtered.

In the early 1990s, Tapestry [11], the first filtering system, was developed at the Xerox Palo Alto Research Center. 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 system's users benefit from. Similar concepts and principles were successfully applied to Internet discussion forums and movie filtering systems.
The initial success of recommender systems leads the increase of e-commerce businesses that implement them. In recent years, many E-commerce sites have used Recommender Systems in order to suggest products to their customers and to provide consumers with information to help them decide which products to purchase [12]. The recommendations are provided based on the top overall sellers on a site, on the demographic information of the user, or on an analysis of the previous buying behavior of the user. These e-commerce sites try to adapt themselves to each customer by providing personalized product information, summarizing community opinion, providing community critiques and most importantly predicting the future interests of the consumer. As a result, they have succeeded to build customer loyalty, increase profits, and boost item-cross selling [13].

The widespread commercial usage of RS has also mirrored in the academic society and many researches have been done in recommender systems with the various different fields. Recommender systems are researched in the context of statistics, machine learning, human-computer; social network analysis, distributed and mobile systems, agent-based artificial societies, computational trust and etc. More recently, user generated tag or label annotations on contents are being used to create recommendations, social networks and mobile networks are being explored to adapt changeable user needs in order to filter online news and improve search engine performance.

To conclude, the goal of a Recommender System is to that maximizes the user's utility by selecting a subset of items from a universal set, based on user preferences. The formal definition for a recommendation s'_c is follows:

$$\forall c \in C, \quad s'_c = \arg\max u(c,s), \quad s \in S \qquad (1)$$

where c indicates a user that belongs to the overall set of users C and s is an item of S, the set of all candidate for recommendation items. u(c,s) is the utility function that measures usefulness of item s to user c, u :C × S →R , where R is a totally ordered set [1].

2.1.1 Content-Based Recommender Systems

Content-based recommender systems recommend an item to a user based on the similarity between the content of the item and user's preferences [1]. The recommendation problem is converted to a search problem and tried to be solved by finding the most similar available items to those which user liked in the past. Therefore, content-based recommender systems (CS) have to represent and manage the descriptive contents of the items. Many recommender systems use "feature vectors" to represent the values for the descriptors of items [14]. Item features can be the title of a film, the genre of a song, or the writer of a book depending upon the type of recommendation domain that the item being recommended is belong to. Moreover, recommender systems should also represent the user's preferences and construct user profile in order to determine the user interests particularly. Eventually, the similarities between item features and user's preferences are determined and the most similar items would be recommended.

Machine learning and information retrieval techniques are applied to learn user models and construct user profiles. There are two ways to build a user profile [15]. The first one is the implicit technique that user's past actions and behaviors are examined and data about user profile is collected without placing any burden on the user. For instance, the user clicks on results of the recommendations or the time a user spends reading a document can be observed and the implicit feedback can be provided. Another technique is asking the feedbacks from the user explicitly. User may rate an item in a range to indicate how much he likes or make some comments on the item. In addition to those techniques, supervised learning algorithms can be applied to learn the user model implicitly.

As content based recommender systems recommend items similar to those that a user liked in the past, similarity measurements of items with user's preferences play an important role. Traditional systems use a utility function to predict an item's score by determining the similarity between item's features and generated user profile. Some systems define a vector of weights corresponding to feature vectors to denote the importance of each feature to users. These weights can be personalized and learned for each user to improve the recommendation quality. Along with these traditional information retrieval techniques, Bayesian classifiers [12], [13] and various machine learning techniques, including clustering, decision trees, and artificial neural networks can be applied for content based recommendation.

Although content based recommendation has valuable benefits, it has some certain drawbacks; the most important of them is user preferences have limited coverage. The coverage of a preference depends on the coverage of applicable feature(s) to recommendation domain. Another problem with the content-based recommendation systems is their tendency to overspecialize the item recommendations. As item selection methods are based on previously rated items,

recommendations become very similar to previous items seen by the user. Moreover, content based recommender systems require a sufficient number of items to be rated in order to construct a user profile and provide accurate recommendation. Therefore, their performances on accuracy are low for the new users.

2.1.2 Collaborative Filtering Systems

Collaborative filtering (CF) is the most widely used method in recommender systems [3]. The technique makes recommendations based on a set of user ratings on items. There exist two main approaches to collaborative filtering: memory based and model based CF. Memory based systems may also divided into two groups such as user-based and item-based CF.

2.1.2.1 User-based collaborative filtering

User-based Collaborative Filtering Systems recommend the items that have been rated highly by people sharing the similar preferences with the user [3]. In other words, collaborative filtering systems match the people with similar interests and make recommendations on this basis. They assume that user likes the items that are preferred by similar users. Therefore, the group of most similar users to active user is identified first. Generally, a similarity metric and a cluster algorithm like k-nearest neighbor classifier are used to find the most similar users. Then, the items rated by the group but have not been seen by user are selected. The rating prediction is performed for each selected items by aggregating the group's ratings. As a last step, the items with the highest predicted rating are recommended to active user.

A user based collaborative filtering system has to include methods to achieve the following requirements in order to provide accurate and useful recommendations. [3]. First, it should have a metric to determine the similarity between users the construct clusters (neighborhood) for similar users. Secondly, the system should provide a method for selecting the most similar users with the active user for rating prediction. Lastly, a method for predicting a rating for the items have not been rated by the active user is essential.

Various approaches including Pearson and Spearman Correlation [18], the cosine angle distance [19], Entropy, Mean-squared difference and constrained Pearson correlation have been used to compute the similarity between users in collaborative recommender systems. Pearson correlation and cosine-based similarity are the most commonly adapted methodologies. After measuring the similarities between users, the most similar users with the active user are selected based on the neighborhood of the active user. Therefore, a method is required to define the neighborhood of the active user. Threshold values for the user similarity and the number of the neighbors are the most commonly adapted approaches in the literature. These values directly affect the performance of the recommendations. If the neighborhood size is large, many of the selected neighbors become dissimilar with the active user and the accuracy of the recommendation decreases. On the other hand, selecting a small neighborhood can lead limited and similar recommendations. Finally, a method is required to determine the ratings of the each item that has not been rated by the active user. The most commonly used approach is to use the weighted sum of rank.

2.1.2.2 Item-based collaborative filtering

Item-based collaborative filtering systems apply the same idea with user-based collaborative filtering ones. The recommendation steps explained above also applicable to item-based collaborative filtering systems considering that the similarities are measured between pairs of items instead of users [3]. The rating of an item can be predicted using the ratings given to other similar items. Pearson correlation and cosine-based similarity can be also used to determine the similarity between items based on the ratings from all users. The neighborhood is also defined similar to user-based approach. The neighbor items are determined which are the most similar to items for which the prediction is calculated. Then, an item's rating is predicted by aggregating the ratings of similar items.

Considering the relation between the number of the users and the number of the items in recommendation systems, item-based algorithms can both provide high quality recommendations and be more efficient than traditional user-based collaborative methods based on computational performance.

2.1.3 Demographic Filtering

Although demographic filtering uses user-based similarities like collaborative filtering, its approach differs from CF techniques as similarity measurement components are independent of the ratings that are given to items. Demographic Filtering techniques assumes that demographic attributes of an individual such as age, nationality, occupation etc. may carry discriminative information and this information can be used to identify the types of users in order to construct the user clusters [20]. The recommendations are produced based on the user classification among the clustered user types.

The first system with the demographic filtering was discussed by [21] which recommends web pages or sites using the user profiles and generalization of user-specified data along the patterns common across the population of 40 000 people from the USA. They try to identify one of 62 pre-existing clusters to which a user belongs and to produce recommendations to users based upon information about others in this cluster. With an internet-based experiment testing, they reached more than 20.000 users worldwide and conclude that 89 percent of users think that their application is successful.

Demographic filtering has an advantage over Content based and Collaborative based systems that it is not dependent to history of ratings. However, concerning the privacy issues, collecting high quality demographic information is a very difficult task. In addition, the accuracy and performance of the systems that produce recommendation purely based on demographic filtering have shown to be lower than those based on the item content and user behavior because of over-generalization of the user interest. Therefore, demographic filtering is typically used in hybrid recommenders to support the other recommendation techniques.

2.1.4 Knowledge Based

Knowledge based recommender systems try to exploit knowledge about users and products and reason about user's requirements and the products that meet those requirements to produce recommendations based on a knowledge-based approach [22].

In order to reason about what products meet the user's need, knowledge based recommender systems ask the user about the requirement of wanted products and use the user answers to exploit knowledge base of product domain. Therefore, knowledge base recommender systems need to have the product domain

knowledge which should be stored and organized in a way that enables inferring and reasoning. However, user's knowledge acquisition is very difficult process and a knowledge engineer is required to construct the knowledge-base which causes a bottle-neck for the knowledge based recommender systems. Therefore, researchers have given relatively little attention to knowledge-based recommender systems than other recommendation systems.

Knowledge-based systems have three different kinds of knowledge to infer recommendation [23].

- **Catalog knowledge**: The knowledge about the recommendation products and their features.

- **Functional knowledge**: The knowledge about the relationships between the user's needs and how the items might meet those requirements.
- **User's knowledge**: The knowledge about the user's preferences and necessities which are needed to find corresponding products.

2.1.5 Hybrid Systems

Hybrid recommender systems combine two or more recommendation techniques to improve performance with fewer of the drawbacks of any individual one. Most commonly, collaborative filtering is combined with some other technique in an attempt to avoid the new user and new item problems. The following are the common hybridization methodologies [10]:

- **Weighted**: The score for a recommended item is computed by weighted sum of the results from available recommendation techniques.

- **Switching**: The system decides to use a recommendation techniques based on the current available information.

- **Mixed**: Different recommendation techniques are used simultaneously and their recommendations are presented at the same time.

- **Feature combination**: Feature data from different recommendation techniques such as content and collaborative information are combined into a single recommendation algorithm.

- **Cascade**: Recommendation techniques are used with a defined order. Recommendations which are given on the previous step are refined by the current recommender.

- **Feature augmentation**: One technique produces discriminative features and they are used as inputs to another technique.
- **Meta-level**: The model generated by one recommender is used as input for another.

2.1.6 Cross-Domain Recommendations

Generating recommendations across different domains is a very new area of research and it requires more complicated processes than single domain recommendations. Therefore, it is rarely studied in the research community. In addition, majority of current cross-domain recommendation systems consider the statistical analysis on the market and try to make use of relations between popular items without personalized recommendations [24].

One of the approaches that have introduced so far is creating generic user profiles for cross-domain recommendations. In [25], authors try to build a single user model instead of having several versions of the same user spread throughout various services. Regarding objective, subjective and emotional features of users, they define Smart User Model (SUM) which is domain-independent and consists of a collection of attribute-value pairs which represents objective (O), subjective (S) and emotional (E) features of the user. In addition, they create a User Model (UM) for each domain to model domain specific characteristics, user interests and socio-demographic features. Then, they use a weighted graph to connect the features and establish a connection between two models. Therefore, instead of making the user fill out the UM of each domain, they extract information from SUM according to the graphs that are defined by each application.

The other approach [26] is to form a unified user profile framework built upon the multiple information obtained about a user in different resources. They assumed that different services would benefit from enriching their user models (UMs) through importing and integrating partial UMs created by other services. Therefore, richness of the user model in increase and more accurate personalization can be possible. They define this technique as Cross-Domain User Modeling (CDUM). However, the interoperability of different services may not be possible in many cases. Due to the competition between commercial services, heterogeneity in data structures, privacy issues and lack of standard representations prevent this model to be applied on real applications.

2.2 Recommendation System Frameworks

After presenting background information about recommender systems and their recommendation algorithms, we introduce the general concepts of the software frameworks and investigate the important recommendation systems framework

examples in order to gain knowledge about their approaches, strengths and weaknesses.

Development of a framework requires understanding a specific domain and solving the common practical problems related to this domain. Therefore, the related framework examples help us to define the road map of our framework design by describing the potentials, limitations and necessary requirements about recommendation systems.

2.2.1 Frameworks Overview

With the simple definition, “A framework is a basic conceptual structure used to solve or address complex issues” [27]. In the context of software, a framework consists of libraries, conventions, a set of tools and best practices that promote the reuse of design and source code in order to facilitate software development. Frameworks try to abstract routine tasks into re-usable generic modules in order to allow designers or developers to focus on the specific problems related to their applications without reinventing the wheel each time around. In addition, frameworks reduce the overall development time.

The ability of addressing all of the tedious and low-level details of application development in reusable packages, the frameworks gains acceptance rapidly in specific domains. For example, a researcher can quickly and easily create an online recommender system using our framework in order to test an algorithm’s performance, rather than having to write all of the code required to accomplish this task. Therefore, the researcher can spend his/her time worrying about specific problems related to his/her algorithm, and not the actual building of the code behind it.

There are major artifacts and activities relating with artifacts in the development of a framework. In Figure 1, the artifacts are shown in large fonts and the related activities are shown in small font [28, 29].

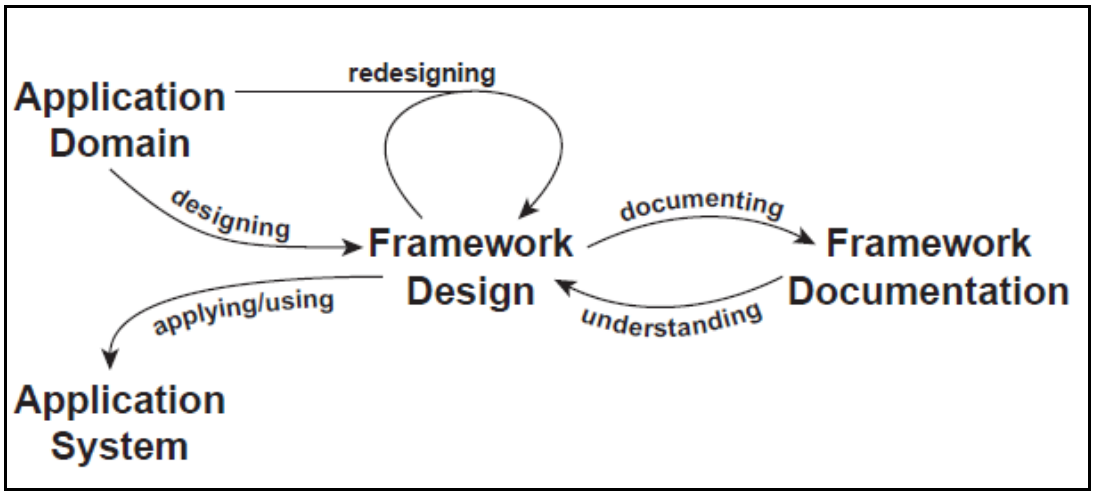

Figure 1 – Artifacts and Activities of a framework development

Framework development begins with the analysis of application domain. During the domain analysis, the domain's requirements and possible future requirements are tried to be discovered. In order to determine these requirements, previous works, existing similar software systems, personal experiences, and standards are considered.

After determining the requirements, designing activity is performed. All the ideas and the concepts involved in the domain analysis are mapped onto modular components. These components define framework's abstractions and constitute the framework design. The overall architecture of a software system is modeled and the extensibility and the flexibility features are outlined. Briefly, framework design is the detailed description of how the system is to be built.

According to [30], software frameworks consist of frozen spots and hot spots which are also modeled in the design phase. Frozen spots constitute the kernel of a framework. They define the basic components and the relationships between them. These remain unchanged and always present part of each instance of the framework. Hot spots constitute those parts where the developers using the framework add their own code to extend the functionalities in order to develop capabilities specific to their own project.

In order to describe the aspects of the framework design, documentation is needed. The documentation tries to explain the ideas behind the design as detailed as possible, but it is never able to fully cover them. Rather, the documentation helps the developers who want use the framework to understand the most important features and enables the application development.

As a final step, necessary hot spots are implemented in order to develop an application using the framework.

The overall process can be summarized as follows:

- Designing a system in an application domain results in a design.
- Documenting the design leads to documentation.
- Reading documentation and working with a framework enables understanding the framework design.
- Applying the framework produces an application.

2.2.2 Recommendation Systems Frameworks Examples

2.2.2.1 The Duine Framework

The Duine Framework is a set of software libraries written in JAVA which enables developers to create prediction engines for their own applications. It has been developed by Telematica Instituut/Novay [31]. Duine allows developers to customize and extend the prediction engine by choosing among a set of prediction strategies or creating their own. In order to improve the quality of prediction, developers can combine two or more prediction techniques which built-in algorithms and define prediction strategies. The prediction strategies can be defined with a set of rules to select prediction techniques and their weights in the prediction. Therefore, prediction algorithms performances can be optimized for different kinds of data. The output of a prediction strategy is normalized with a value between -1 (absolutely not interesting) and +1 (definitely interesting).

The Duine framework recommender is a hybrid recommender which includes both collaborative-based algorithms (User Average, TopN, Social Filtering and Already Known) and content-based ones (GenreLMS, Case-based Reasoning and Information Filtering). Social Filtering is similar to classical collaborative filtering algorithm and it uses the similar users to generate prediction. TopN algorithm looks for the items popularity. GenreLMS reasons on the items genres and Case-based Reasoning make use of similar items rated previously. Information filtering extracts information from an item and behaves in the same manner as the classical content-based recommendation algorithm. Developers can add new algorithms to these existing ones by creating a Java class which extends the base recommendation algorithm class.

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 also slightly adapts itself to these interests after each given rating by its learning capabilities.

Being a hybrid recommender, the duine recommender provides more accurate predictions and reduces the cold-start problems. The interesting feature of the recommender is that it checks which conditions hold based on the current available data about the users and dynamically selects the most suitable prediction techniques in order to provide best results.

In addition to its extensibility, the duine framework has also two important features. First, it provides tools for the validation of the frameworks which can measure the accuracy of predictions. It has also an explanation API which enables to explain how the prediction results were determined.

However, the duine framework supplies only a prediction engine, the database containing the data about items and users should be managed by the application that uses this recommendation system. The framework also needs to an information wrapper to be developed for converting the stored data in Java objects in order to process this information and apply the predictions strategies.

2.2.2.2 CoFE (the COllaborative Filtering Engine)

CoFE stands for "COllaborative Filtering Engine" which is an engine for collaborative filtering recommendation systems. It is developed by the Intelligent Information Systems research group of Oregon State University [32].

CoFE is developed in Java and it is implemented to run as a server to generate recommendations. The clients can invoke the functionalities to receive recommendations for individual items, top-N recommendations over all items, or

top-N recommendations limited to one item type. CoFE uses the nearest-neighbor algorithm which is user-to-user based Pearson Correlation algorithm in order to compute recommendations. Developers can write and use new collaborative filtering algorithms by implementing an available JAVA interface provided by CoFE.

Similar to Duine Framework, CoFE is only a prediction engine and the processes about data storing and management are not provided. The official web site of the CoFE is out-of-date and the details of the framework can not be reached and this project does not seem to be maintained anymore.

2.2.2.3 Apache Mahout - Taste

Taste [33] is an open source, Apache Software Foundation project to create recommendations engines for mainly JAVA applications. It provides flexible and fast collaborative engine and supports user-based, item-based and slope-one recommenders. It has also includes some other experimental algorithm implementations. Currently, model-based recommenders are not supported. The recommendation engine processes users' preferences for items which are also called as “tastes” and predict the preferences for other items.

Taste also allows creating customized recommender systems by combining a set of recommendation algorithms. The followings are the key abstractions that define the frameworks design.

- DataModel
- UserSimilarity and ItemSimilarity
- UserNeighborhood
- Recommender

Recommender is the main component of Taste; it processes the data which is provided by DataModel which provides an interface to get the data from the storage system. It applies an algorithm to find "User Similarities" and "Item Similarities". Finally, recommender searches for finding a "neighborhood" of similar users near a given user in order to produce recommendations.

The advantage of Taste is that it is designed to be scalable and flexible. It has also a good performance for working with high amount of data. Taste can deployed as an external server which exposes recommendation logic by communicating directly over HTTP or as a SOAP Web Service. Therefore, it can also be integrated with applications which are not developed with JAVA.

2.2.2.4 RACOFI: A Rule-Applying Collaborative Filtering System

RACOFI is a rule-applying collaborative filtering system which can also provide multi-dimensional rating system [34].

The aim of the RACOFI is to make use of the meta-data about objects in the prediction process. It assumes that meta-data describes everything relevant and interesting about the objects. Meta-data is divided into two groups: objective and subjective. Objective meta-data represents the general features of the objects like title, genre or the name of objects. Subjective meta-data depends on user preferences; how the users rated the object in different features. In order to handle these different kinds of meta-data, RACOFI is developed with a modular architecture based on two software agents. The objective meta-data is processed by a rule engine called RACOLA which adjusts the predictions with a set of rules defined based on RuleML. RuleML is a markup language which allows expressing both bottom-up and top-down rules in XML for reasoning tasks. The subjective meta-data is handled by a collaborative filtering library called COFI.

RACOFI can be considered as a hybrid system because COFI generates predictions with collaborative filtering algorithms but RACOLA combines these predictions with its own content-based processing. As a result, recommendation quality is improved with help of the rules which allow considering the relationships that exist among objects and between objects and users.

RACOFI is used in a Canadian Music recommender site called RACOFI Music in order to test and validate its model. It is found that a rule-based approach made it easy to adapt the system to user expectations.

One of the advantages of the RACOFI is that it enables developers to change the algorithms used in predictions easily. The framework has also a flexible structure because the collaborative filtering algorithm does not depend on the contents of the objects and content-based filtering is performed based on domain specific rules which are allowed to be loaded to rule engine.

2.2.3 Conclusion

We have described a set of recommendation system frameworks above. Table 1 summarizes the features of those frameworks.

The Duine framework and Apache Mahout – Taste seem better than the other framework as they offer different features. The Duine is a hybrid systems and it allows combining different recommendation techniques dynamically whereas Apache Mahout – Taste generally focuses on collaborative filtering. On the other hand Apache Mahout – Taste has an express component to manage data, whereas in The Duine does not support data management and the application which uses the framework should provide components to make data ready to be processed.

In CoFE, although it is possible to extend and implement new recommendation algorithms, it just allows collaborative filtering. Similar The Duine Framework, CoFE does not have an express component to manage data. In additions, it is probably out-of-date and is not developed any more.

RACOFI is a hybrid system that it processes collaborative filtering predictions with mechanisms provided by content-based approaches. The main drawback of RECOFI is that it is not completely available, only collaborative filtering component (COFI) is publicly available.

Table 1 – Existing Recommendation System Frameworks

Framework	**Advantages**	**Disadvantages**
Duine Framework	• Hybrid Recommender (Content / Collaborative) • Dynamic combination of algorithms • Algorithms can be expanded	• Only recommendation engine, no data management support • Wrapper components needed to process data • Learning Curve is high
CoFE	• Algorithms can be expanded • RMI/Corba interface	• Only collaborative algorithms supported • Only recommendation engine, no data management support • Out-of-date

Table 1 – Existing Recommendation System Frameworks (Continued)

Apache Mahout - Taste	• Could be integrated in different ways (service, EJB, application) • Scalable (high performance) and Flexible	• High memory and resource consumption
RACOFI	• Hybrid Recommender (Content / Collaborative / Rule based)	• Only the sources of Collaborative Filtering algorithms is available

CHAPTER 3

CROSSING: A FRAMEWORK TO DEVELOP SINGLE AND CROSS DOMAIN RECOMMENDERS

In this chapter, our framework, named CrosSing, which provides a dynamic infrastructure for developing knowledge based recommender systems that can also support recommendations in cross-domains is presented. First, we present an overview of the system. Then the architecture of the system is explained, and the data representation scheme is described. After that, each component of the system is discussed and detail. Finally, the implementation details and the system interaction interface with both the developers and end-users are explained.

3.1 Overview of the Framework

The objective of CrosSing is to provide an architecture for recommender systems that is capable of integrating new domains dynamically, creating semantic relationships between existing domains and extending traditional recommendation approaches to provide more accurate and useful recommendations in cross domains.

We propose a framework for developing knowledge-based recommender systems, which consists of a set of pre-implemented recommendation algorithms and is

designed to support cross-domain recommendations by means of a centralized profile management component and the possibility to define ontological mappings for item features of different domains.

Our recommendation engine chooses the recommendation method to be used depending on the currently available data at run time. In order to avoid recommendation systems problems such as cold-start problem; recommendation engine selects the right hybridization strategy and reuse the user' profile/preference information across different domains to exploit the available information about a user as far as possible.

In addition, the capability of adapting new domains and creating useful cross domain recommendations result in offering a bundle of related items from different domains and thus improve the possibility of user satisfaction. The adaptive and flexible infrastructure also provides the capability to integrate variety of new and existing systems to framework easily. The dynamically provided inter-domain knowledge helps us to determine the domains' relationships and features mapping in different domains. Moreover, new domain integration problems can be minimized by providing domain knowledge and assigning features' weights as accurate as possible in user models.

Finally, our framework provides a user-friendly interface to simplify the system interaction with user. Framework adapts itself to the integrated domains and all dynamic graphical user interfaces are generated automatically by code generation module based on the data models defined in the ontology and configuration files of recommendation domains. In addition, the framework interface provides a feedback mechanism for user to evaluate the generated recommendations. The user feedbacks are used to adjust the feature weights on profile and item similarity measurements. The framework has the ability to learn the user profiles

individually and personalize the feature weights in order to improve the recommendation quality. Therefore, the system evolves with the user experience and increases its performance. The feedback mechanism is also used to monitor the performance of the inter-domain relationships.

In order to achieve the above goals, we designed our framework in a modular way which is described in the following section. The details of the framework and its components are explained in the rest of the chapter.

3.2 The Framework Architecture

The framework consists of the following components. The logical relations between components are shown in the Figure 2.

1. **Profile Management**: Constructs and maintenances the user profile and preferences, receives feedbacks from users about recommendations and updates the feature weights.

2. **Recommender Engine:** Generates recommendations using the integrated recommendation algorithms. It selects the hybridization strategy based on the currently available data at run time.

3. **Domain Management:** Manages the integrated domains and their relationships. It also construct the knowledge based using the domain knowledge and defined rules between domains.

4. **Items Module:** Deals with retrieving and maintaining items from integrated domain's item information.

5. **Code Generation Module:** Generates the code for graphical user interfaces based on the data models.

6. **Common Vocabulary Adapters:** Transforms the target domains information to common data structures in order to integrate the domains to framework. It also deals with ontology mapping and type conversions of the features.

7. **Target Domains:** Consist of the specific fields of interests and constitute of knowledge about users, items, concepts and relationships in a field such as books, movies, music and games.

8. **Test Suite:** Provides an environment to test, evaluate and verify the algorithms of recommender engine.

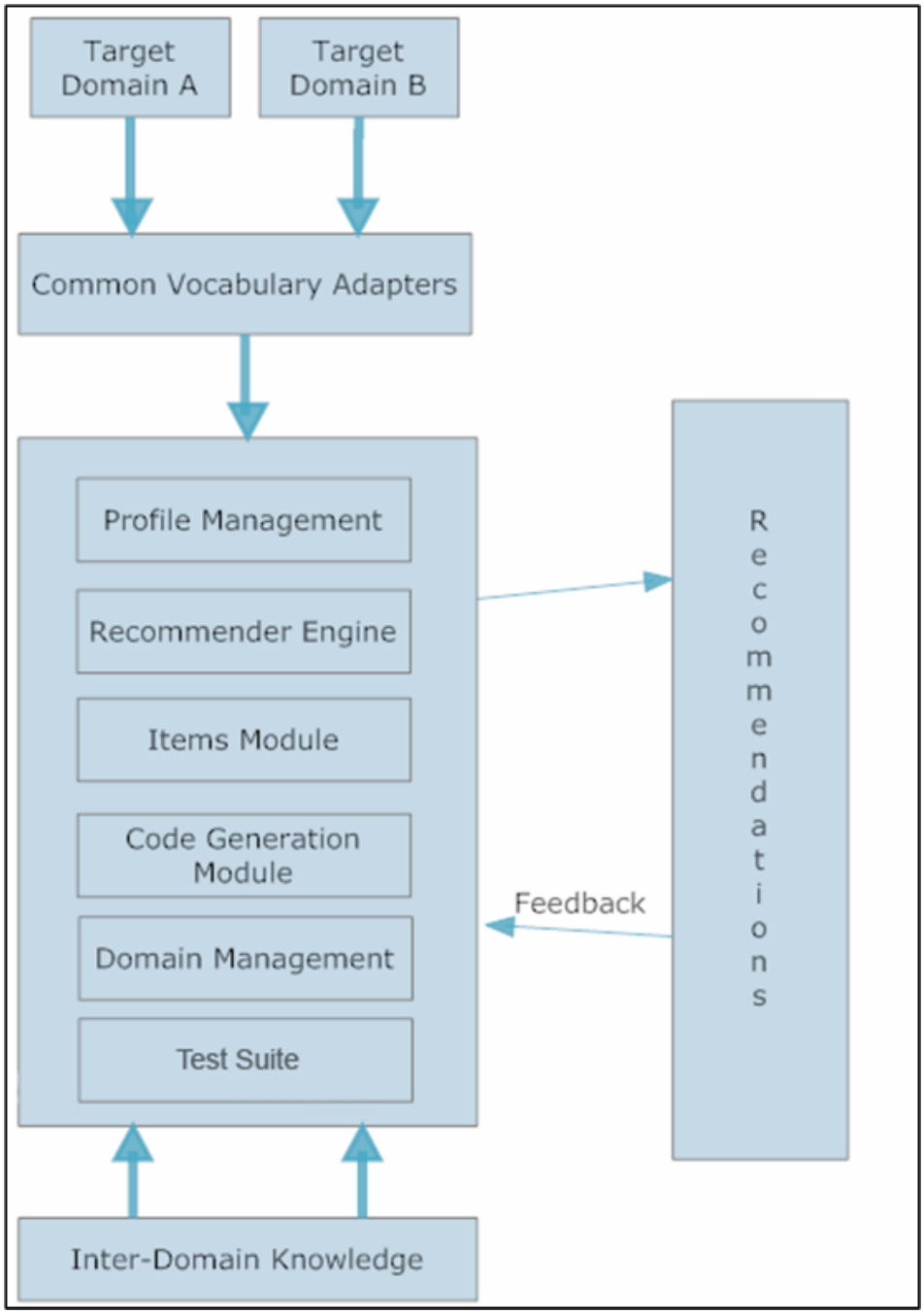

Figure 2 – Overview of the Framework

3.2.1 Profile Management

Creating a comprehensive and detailed user model is very important to analyze user interests and needs correctly. Our profile management component is responsible for creating and maintaining the user profiles. It has two main parts to support the recommenders in the system. “Data Store” part provides required knowledge about user preferences while “Actions” part enables users to edit their profiles.

3.2.1.1 Data Store

This subcomponent is responsible for storing the user preference structures. The structure of user data includes user profile information, transaction history with item ratings and for each domain feature weights for both item similarity and user similarity measurements. User profile information features are determined within the framework ontology files and currently the following features are defined: name, surname, username, age, gender, occupation, zip code, address, city, and country. When the demographic information exists, it helps to find the similar user. When a new user profile is generated, features weight vectors are created for each domain. If a feature in the framework ontology is not applicable for a specific domain, it can simply be ignored by assigning zero weight to it.

3.2.1.2 Actions

The actions component enables users to create, modify and delete their individual profiles. User profiles are updated when a new domain is added, a profile related ontology is updated or a recommendation feedback is received.

3.2.2 Recommender Engine

Our framework provides knowledge-based recommender development and provides a hybrid approach for generating recommendations. The recommendation engine has different types of recommendation strategies which are applied depending on the available data about the user and the items in domains. The system starts with the simple recommendation techniques and move to more complex algorithms with user's experience. Before describing the recommendation process, we introduce the recommendation techniques which are used.

3.2.2.1 Recommendation Strategies

3.2.2.1.1 Most Popular Items

This strategy is the simplest technique in the system but it helps us to generate useful prediction when no data available about users. The algorithm finds the most popular items in integrated domains by sorting the item scores which is calculated using the following formula:

$$\text{Score}(\text{I}_\text{k}) = \frac{\sum_{i \in C} (r_{i,k} - \bar{r}_i)}{|C|} \qquad \textbf{(2)}$$

where I is the set of items, C is set of user that rated item I_k, $|C|$ is the number of users in set C, $r_{i,k}$ is the rating by user i on item k, and $\bar{r}_i$ is the mean rating by user i to all items. Top 5 items in each domain are recommended to user. This strategy also used with other strategies to find the popular items for a specific set of users.

3.2.2.1.2 Knowledge Based

A knowledge base is a centralized repository for dynamic collection of information which is used by our system to retrieve rules about the users, features, domain relations and feature mappings. For each domain, features rules are provided by domain knowledge.

Group Rules help us to create user groups in domains and define the relations between user groups and item features. The rules for these relations are defined as 6-tuples $GroupRule < d, uf, ufv, if, ifv, status >$ where d represents the domain, uf defines the user feature name and ufv is the value of the given feature in uf, if indicates the item feature name, ifv is the value of that item feature and status determines whether this rule affects rating prediction positively or negatively.

For instance, we can define a group rule such as "Observer people like Animation movies". It is based on "personality" feature of user and it uses "genre" feature of items in "movie" domain. It also states that if this rule holds for a user, it affects the rating positively.

GroupRules < Movie, Personality, Observer, Genre, Animation, Like >

The group rules directly affect the predicted ratings of a user on items by using the formula below:

$$\bar{r}_{i,k} = r_{i,k} \pm \begin{cases} \text{knowlege effect (if conditions holds)} \\ \text{0 (otherwise)} \end{cases} \qquad \textbf{(3)}$$

$\bar{r}_{i,k}$ is the final rating calculated where $r_{i,k}$ is the rating by user i on item k determined by Collaborative Filtering or Content-Based filtering algorithm. The

value of knowledge effect is between 0 and 1. It is calculated experimentally and can be configured.

The feature rules can allow features to have multiple values. These are defined as 4-tuples $FeatureRule\langle r, d, f, list\rangle$ where r defines the rule name, d represents the domain, f is the feature name and $list$ refers the possible values for feature.

For instance, while calculating the user similarity based on demographic information in movie domain, the "age" features of two users need to be compared. In most systems, they should be equal in order to decide that the feature of the users matches. In our framework, demographic filtering strategy checks the knowledge base to find similarity rules about this feature in the movie domain. Assume we have the following rules in the knowledge base.

FeatureRule < AgeGroup1, Movie, Age, (18,19,20,21) >

FeatureRule < AgeGroup2, Movie, Age, (22,23,24) >

The system assumes the "age" features of users' have the same value if they belong to the same group. Ranges are defined for each group for the value abstraction.

Domain relations and feature mappings are provided by "inter-domain knowledge" which represents the relationships between two specific domains and provides the abstraction between domain features. Especially in cross domains, they play an important role for creating the accurate recommendations. Inter-domain rules are also defined as 4-tuples $InterDomainRule\ \langle s, t, sf, tf\rangle$ where s denotes the source domain, t denotes the target domain, sf defines the related feature rule in source domain and tf represents the feature rule in target domain.

For instance, we can define a relationship such as "People who like movies in romance drama group also like books in dramatic poetry with the following rules between Movie and Book domains.

InterDomainRule < Movie, Book, Romance Drama, Dramatic Poetry >

FeatureRule < Romance Drama, Movie, Genre, (Romance, Drama) >

FeatureRule < Dramatic Poetry, Book, Genre, (Drama, Poetry, Epic)>

The performance of the rules and relations can be monitored by a feedback mechanism. Existing rules can be updated; deleted or new rules can be added dynamically to the knowledge base.

3.2.2.1.3 Demographic Filtering

Demographic information of user can also carry important discriminative information while finding the similar user profiles in collaborative filtering. User's demographic information is represented as feature vector and each feature has weights for each domain. As we mention before, when a user profile is generated, weights are initialized for each domain separately. Therefore, users' demographic similarity can vary from domain to domain. The formal description of the total similarity calculation is below:

Assume that we have a target user u, other users in the system $C = c_1, c_2, ...$ such that u is a member of C. Then, similarity between u and c_i is given by:

$$\text{sim}(u, c_i) = \sum_{k \in F} W_k X \text{ compare}(fu_k, fc_k) \qquad \textbf{(4)}$$

where F is the feature set, W is set of feature weights for user u for target domain, fu_k and fc_k are the feature values of users for feature k.

The 'compare' function checks the knowledge base if there is any related rule about feature k, return 1 if feature values can be assumed to be same, or return 0 otherwise. If the similarity value is greater than the threshold value T, users are assumed to similar and user c_i is added to the neighboring users list of user u.

With the rules defined in knowledge base, the user clusters can be created according to the demographic information.

The demographic filtering strategy is used in two ways in the framework. First, if a user is a newcomer and s/he does not have enough rating history to apply "content" and "collaborative" based strategies, the demographic filtering is used to find similar users and system applies "most popular item" strategy for these users and the recommendations are generated. Second, the demographic filtering is used to filter similar users if the system finds many users at the end of collaborative filtering.

3.2.2.1.4 Content Based

It recommends similar items to the ones that user preferred in the past. The candidate items compared with the previously rated items and best matching items are recommended as described in [1].

In order to find these similar items, first, the system retrieves the items that user has highly rated before. Then, using the Inverse Document Frequency [16] theory, common features values among highly rated items are found. Then, the items which have similarity above the threshold are found with a similar calculation defined in "Demographic Filtering". However, "compare" function in content based filtering is different. Although there is only one possible value for the target user feature in "Demographic Filtering", there may be more than one feature values which are ordered according to TF-IDF (term frequency–inverse document frequency). Comparisons for each possible feature value are performed between target item and similar items beginning with the highest TF-IDF score. When a feature value is matched, "compare" function returns its TF-IDF score. If no

features match with similar items feature values, “compare” function returns zero. This compare function is also checks the knowledge base in order to find rules about features in comparison.

Content based recommendation strategy is very important for cross-domain recommendations when common rated item count is not enough between target user and other users.

3.2.2.1.5 Collaborative Based

It recommends the items that people sharing the similar preferences with the user prefer. It assumes that user likes the items that are highly rated by similar users. The similarity between users is calculated based on commonly rated items. Our system uses Pearson correlation coefficient to measure the user similarity [1]:

$$\text{sim}(x,y) = \frac{\sum_{s \in S} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S} (r_{x,s} - \bar{r}_x)^2 \sum_{s \in S} (r_{y,s} - \bar{r}_y)^2}} \quad \textbf{(5)}$$

where S is the set of items co-rated by users x and y, $r_{i,k}$ is the rating by user i on item k, and $\bar{r}_i$ is the mean rating by user i to all items.

After the user similarities found, the users, which have greater similarity value than the threshold, assume to be similar users with the target user. If many users are found similar, “demographic filtering” strategy is applied. Then, recommendations are generated by applying the most popular items for similar users.

3.2.2.1.6 Surprise Strategy

There are two well known problems in recommender systems: "Overspecialization Problem" which refers to always recommending items that are very similar to the ones that user already rated and "Sparsity Problem in Rating" which happens when few users have rated the same items resulting in never recommending this item. To overcome these problems, our framework generates one surprising recommendation randomly among the newly added items, popular items or few rated items.

3.2.2.2 Recommendation Generation

Recommendation generation evolves with the user experience in the system. Strategies are applied according to the if-then-else rules. "Knowledge based" recommendation strategy is not used standalone but it helps other recommendation strategies. "Surprise" Strategy is always used but it only generates one random recommendation at a time. Other recommendation procedures are as follows:

- When a new user logs in to system and there is no information available about user, recommendations are generated using the "most popular item" strategy.

- If the user has no or very few transaction history in all domains but the demographic information is available, "demographic filtering" strategy is applied. Then, similar users are found and their favorite items are determined using "most popular items" for these users. Therefore, recommendations can be generated.

- If the user has enough rating history for a domain but does not have required number of co-rated items with other users in target domain, “content based” recommendation strategy is used. For instance, the user may rate many movies in movie domain but s/he may not rate any books. The system can not find commonly rated books using user history but it can generate recommendation by finding similar books to movies that the user rated.

- If the user has enough number of co-rated items with other users in target domains, “collaborative based” recommendation strategy is used to generate recommendations. Depending on the similar user count, “demographic filtering” can be used before predicting recommendations.

3.2.3 Domain Management

The domain management component mainly deals with the integrated domains and their relationships. It includes the “Domain Knowledge” of all domains which is required by knowledge-based recommender strategy. In addition, domain relations which are determined by “Inter-Domain Knowledge” are also kept in this component. “Inter-Domain Knowledge” consists of the relationship descriptions between two domains in a rule-based fashion. These rules can be added, updated or deleted dynamically. Domain management component also contains the general ontology interface with XML schema of the framework.

General ontologies of framework have the specifications of features and relationships between features of different domains. They provide the uniformity and knowledge exchange between different domains. For instance, “release date” for movie domain means the date when a movie is made available to the public

while “the year of publication” has the same meaning in book domain. The framework ontology about items has “item year” to define that feature which is meaningful for all integrated domains. The feature names and their types for common meanings can vary across the domains. Therefore, domain data is needed to be transformed to the common vocabulary and structure of the framework.

3.2.4 Items Module

Items are the elements of domains such as books, movies, or songs that are recommended to user according to the user preferences. Our items module is responsible for retrieving and maintaining items from integrated domains. Items’ data structures are similar to the user data structure as they represented as feature vectors. New items can also be added to the specific integrated domain by using the administrator user perspective.

3.2.5 Code Generation Module

Code generation module generates the code for all dynamic graphical user interfaces based on the data models defined in ontology files. When new domains are integrated to framework or data models are updated, the changes can easily be applied to system by this module.

3.2.6 Common Vocabulary Adapters

Each domain data has its own data structures and ontology but in order to integrate a domain to framework, we have to transform its domain information to a common vocabulary and structure. Therefore, for each domain, a common vocabulary adapter is needed to be developed. It is not very complicated to

develop an adapter as we define our interfaces and data structures as a framework API. Considering the general ontologies and XML schema of the framework; a common vocabulary adapter deals with ontology mapping and type conversions of the features.

3.2.7 Target Domains

Recommendation domains are the specific fields of interests and constitute of knowledge about users, items, concepts and relationships in a field.

Initially movie and book domains are addressed in our experimental framework implementation, because required knowledge and information of these domains are currently available. In addition, these domains have similar features, common users and close relationship to each other.

We divided domain information into the following categories.

3.2.7.1 User Data

In many data sets, user data includes the generated user profiles by recommender systems and generally they have a feature vector that represents the user preferences. User data has also transaction history and item ratings which constitute the knowledge about user actions history that help us predict user attitude to new items and new domains.

3.2.7.2 Items

Items are the elements of the domains which have certain features based on the domain structures and specifications. Features can be shared by different domains as well as they can be applicable to only one domain.

3.2.7.3 Domain Knowledge

Domain knowledge stores the structured knowledge about the domains which is retrieved and used to determine weights of factors which have effects on items during the recommendation process. Domain knowledge may not be available for all domains. It increases the accuracy and quality of recommendations in available domains. If domain knowledge does not exist, recommenders consider the weights of all features to be equal at the beginning. The weights are adjusted according to the user feedback on the generated recommendations. Domain knowledge also contains the feature rules which provide to create user groups or items groups in domains by allowing features to have multiple values.

3.2.7.4 Domain Ontology

Domain ontology represents the meanings of the terms in user data, community data and item features. The concepts and relationships are defined by ontological categories. “Artists”, “composition”, “musical work”, “sound”, “genre”, and “release date” are the example categories of music domain. Domain ontology is required for developing common vocabulary adapters for domains.

3.2.8 Test Suite

It is one of the most important components of the framework. Test Suite provides an environment to test, evaluate and verify the algorithms of recommender engine. It also allows us to observe the effects of the knowledge base on the integrated domains. We also used Test Suite module to obtain our first experimental results given in the “Evaluation” section.

3.3 IMPLEMENTATION DETAILS

A prototype of the framework has been developed to evaluate the proposed structure in the Figure 3. It is an online cross domain recommender system which is available at www.crossingframework.org.

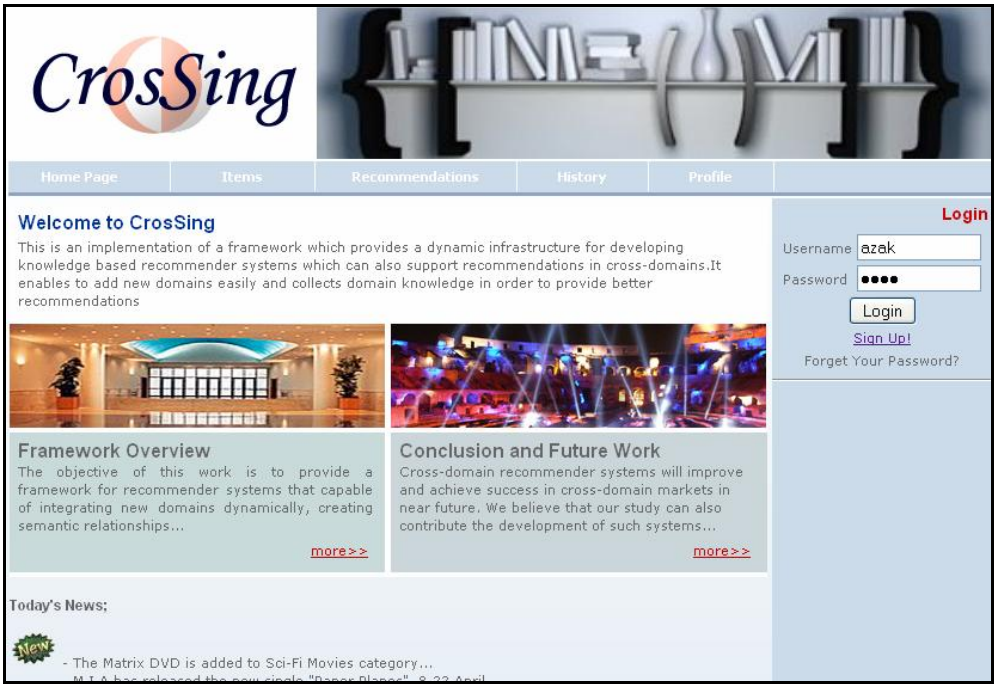

Figure 3 – The Prototype Implementation of the Framework

The main purposes of the prototype are to test applicability of the framework, provide interaction with real world users, observe the performance of inter-domain knowledge rules and prepare an environment to evaluate the recommendation strategies.

The core of framework was developed with Java programming language [35]. Dynamic and flexible data interaction is provided over the ontology files with XML [36] interfaces. Framework's web infrastructure was created with Java Servlets [37] and JSP [38] technologies using Sun Java Studio Enterprise IDE

[39]. MySQL database [40] is used to store required information about users, items, ratings etc.

The system deployed on Apache-Tomcat Application Server [41] over Linux operating system [42]. Common vocabulary adapters are developed for target domains such as movie and book domains.

3.3.1 Ontology

Ontology files are required for the generation of the data structures in the system and creation of tables in the database.

```
<?xml version="1.0" encoding="UTF-8"?>
<Items>
   <Feature>
     <name>ItemId</name>
     <type>INTEGER(11)</type>
     <primaryKey>yes</primaryKey>
     <inputtype>text</inputtype>
     <description>Unique item identification
     </description>
   </Feature>
   <Feature>
     <name>Title</name>
     <type>VARCHAR(200)</type>
     <inputtype>text</inputtype>
     <required>yes</required>
     <description>Specifies the title of item
     </description>
   </Feature>
</Items>
```

Figure 4 – Ontology Structure

These files are also used for developing adapter systems to transform different domain data into common structure so that the framework can load the required information to the database.

As it can be seen in Figure 4, ontology files consist of features with the description and other required attributes for system. All dynamic graphical user interfaces of the framework are generated automatically based on these features.

When a new domain is needed to integrate with the framework, the corresponding data files should be created according to the following ontology files: Domain Ontology, Item Ontology, User Ontology, Rating Ontology, User Profile Similarity Ontology and Item Similarity Ontology.

Important ontology files are explained in details as follows:

3.3.1.1 Domain Ontology

Domain ontology represents the identification of domains. It has only two features: “Domain Id” is the unique identification number in the system and “Domain Name” specifies the domains name.

3.3.1.2 Item Ontology

Item ontology represents the features of the items. It provides the uniformity and knowledge exchange between different domains. Currently, item ontology has following features.

- *ItemId: Unique item identification in the system.*
- *DomainId: Domain id which item belongs to.*
- *Title: Specifies the title of item.*
- *ItemYear: Date when an item is made available to the public*
- *Performer: People who perform the action in the item.*

- *(Ex. Author, Artist, Singer)*
- *Producer: People or companies who creates the item.*
- *Genre: Item's type.*
- *Price: Price of the item.*
- *Size: Item's size or duration.*
- *Tags: Item's keywords, tags.*
- *Country: Country name where item produced.*
- *Language: Language used in item.*
- *Locations: Places where mentioned in item content.*

3.3.1.3 User Ontology

User ontology represents the demographic features of the user. User profile features are below:

- *UserId: Unique user identification in the system.*
- *Name: The name of the user.*
- *Surname: The surname of the user.*
- *Username: Unique user name for the user.*
- *Age: Age of the user.*
- *Gender: The gender of the user.*
- *Occupation: User's profession.*
- *ZipCode: The zip code of the user.*
- *Address: The address of the user.*
- *City: The city where user lives.*

- *Country: The country of the user*.

3.3.2 User Interfaces and Perspectives

The framework has different interfaces and perspectives for end users and administrators.

3.3.2.1 End User Perspective

This perspective is the simple view that the end user can only see his/her basic profile, transaction history, navigate between integrated domains' items, receives recommendations and give feedback about quality of recommendations. User's interaction with the framework begins with the sign up procedure. The user is asked to enter his/her profile information. Then, the user logs in and selects a recommendation domain. The system immediately begins to generate a recommendation. If user's profile information is adequate, the demographic filtering and the most popular item strategies are applied. Otherwise, only most popular items are recommended from the selected domain. As the user rates the items, strategies change to content based and collaborative based ones.

The cross-domain recommendations are performed in two ways. First, if the user wants to receive recommendations from a target domain where s/he does not have enough rating history, the system checks user transaction history in other domains and generates cross-domain recommendations in target domain, based on user preferences in other domains. Second, when the user views an item details page, Figure 5; cross-domain recommendations are generated for the selected item.

Figure 5 – Item Details Page

The user can rate or update the rating by simply clicking the "Rate It" button of the item. The user can also rate the recommendations positively or negatively; then the system updates the weights of the used features and logs the performance of the rules according to reaction of the user. With this feedback mechanism, personalization in feature weights is provided and performances of the rules are monitored.

A recommendation from "Surprise Strategy" is always generated at the end of other recommendations.

3.3.2.2 Administrator User Perspective

Administrator users have full control of the framework. They can add new domains by providing domain data and common vocabulary adapters. They can also track the performance of the inter-domain rule and define new inter-domain relations by just defining rule-sets. In addition, they determine, observe and update the feature weights in user and item similarity measures in order to improve the recommendation qualities.

CHAPTER 4

EVALUATION

This chapter presents the methodologies used to test, validate and evaluate our proposed approach. In order to examine the various aspects of the framework, we divided our experimental work into the following categories. First of all, we performed “functionality tests” to show the applicability, flexibility and usability of the framework. After that, “algorithms performance tests” are carried out to test the performance of the built-in recommendation algorithms and our recommendation engine. Automated testing is not applicable for cross-domain recommendations because there is no dataset available for our target domains. Therefore, we focus on the single domain testing. For each single target domain, recommender engine is tested with framework’s test suite which is also explained deeply. These two tests led us to develop an application and we performed alpha testing with real end users and obtained “end-user evaluation” about the overall system and cross-domain recommendations.

4.1 Functionality Tests

As we discussed in the previous chapter, a prototype has been developed to prove the applicability of proposed structures and approach. The components of the framework and the implementation details are explained so far. We will introduce

the required steps and the sequence of process to develop a real application using our framework so that we can show the flexibility and usability of the framework and demonstrate how our framework enables easy development. We will also present the critics of an external developer who helps us to evaluate our framework with the third person point of view.

4.1.1 Application Development Testing

In this section, we will demonstrate developing a movie recommender system using MovieLens dataset by the GroupLens Research group at University of Minnesota [8]. As discussed in the Frameworks Overview section, in order to develop an application using a framework requires reading the documentation and understanding the framework design. Therefore, after presenting the conceptual design of the framework given in the previous chapter, we have to examine the deployed package view of the framework which is seen in Figure 6 so that we can explain what exactly needs to be done to develop an application.

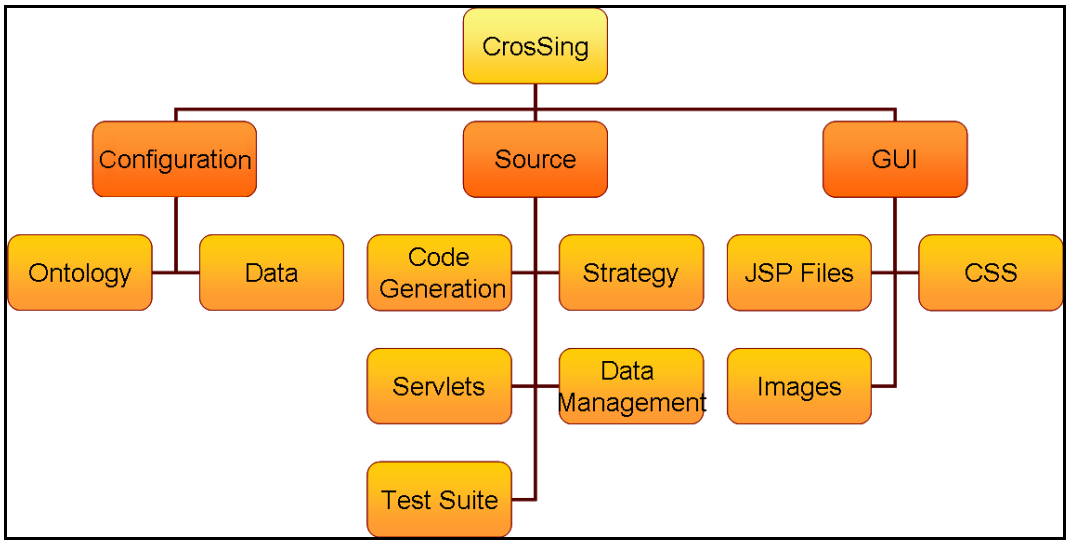

Figure 6 – Package structure

Configuration, source and GUI packages constitute the main parts of the framework file structure. Source and GUI components do not need to be modified for developing a recommender system. Those components can be changed by advanced users so we are omitting them for now.

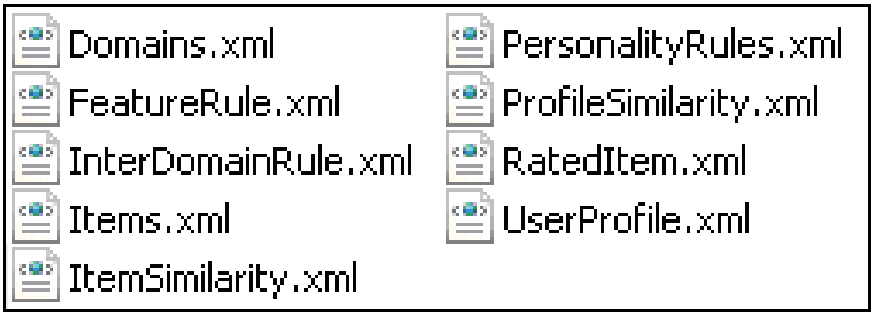

Figure 7 – Ontology Files

The important point is to understand the ontology files because generation of the data structures in the system and creation of tables in the database are based on these files. The details of the ontology file structures are also given in the previous section. These files, in Figure 7, can be modified in order to meet different user needs. The data structures of system in the database can be easily created from the Administrative panel by selecting the menu item seen in Figure 8. This capability covers the flexibility requirement of the framework.

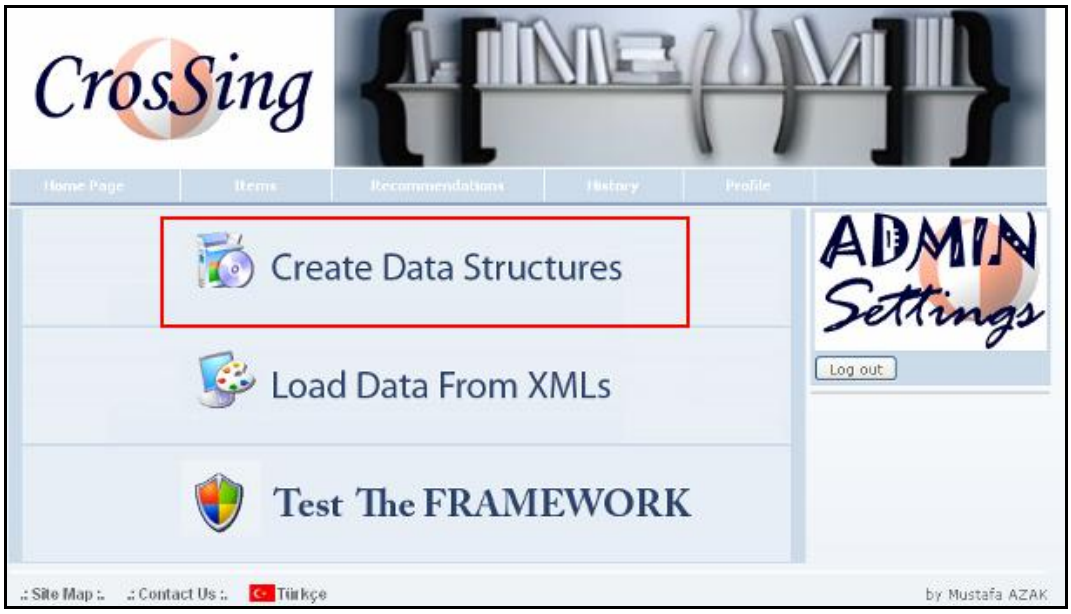

Figure 8 – Creating Data Structures

After deciding the data formats in the system, we have to provide the data corresponding to these ontology files in desired format. Domains, UserProfile, RatedItem and Items files are basic needs to create the application. Others are auxiliary files that provide optimization and improvement in the accuracy of recommendations.

First, we create an xml file for movie domain with the content seen in Figure 9. We gave an identification number to domain and state its name.

```
<Domains>
        <Feature>
            <name>DomainId</name>
            <value>1</value>
        </Feature>
        <Feature>
            <name>Name</name>
            <value>Movie</value>
        </Feature>
</Domains>
```

Figure 9 – Movie.xml

For user profiles, movies and ratings, we will use the Movie Lens dataset however the dataset provides these information in the raw data type as shown in example below.

- **User Profile**: It contains simple demographic info for the users. User id, age, gender, occupation and zipcode are provided.
  98|24|M|technician|85711
- **Movie Info**: It contains the information about movies. Movie id, movie title, release date, video release date, IMDb url and genre information is provided. The last 19 fields are the genres; a 1 indicates the movie is of that genre, a 0 indicates it is not.
  109|Toy Story (1995)|01-Jan-1995|http://us.imdb.com/M/titleexact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0|0

- Rating Info: It presents ratings given by users on movies. User id, movie id rating and timestamp are provided. Ratings are scaled between 1 and 5. The time stamps are seconds since 1/1/1970.
  98|109|4|881250949

In order to integrate these dataset with the framework, we have to transform raw data to our common vocabulary and structure based on the ontology files. We need to develop a common vocabulary adapter to perform ontology mapping and type conversions of the features. Therefore, we implement a java program that read raw data files, parse them, find appropriate feature correspondences and store output files in the xml format as shown in Figure 10.

```
<user>
    <Feature>
        <name>UserId</name>
            <value>98</value>
    </Feature>
    <Feature>
        <name>Age</name>
            <value>24</value>
    </Feature>
    <Feature>
        <name>Gender</name>
            <value>Male</value>
    </Feature>
    <Feature>
    <name>Occupation</name>
        <value>technician</value>
    </Feature>
    <Feature>
        <name>ZipCode</name>
            <value>85711</value>
    </Feature>
</user>
```

```
<items>
    <Feature>
        <name>DomainId</name>
            <value>1</value>
    </Feature>
    <Feature>
        <name>ItemId</name>
            <value>109</value>
    </Feature>
    <Feature>
        <name>Title</name>
          <value>Toy Story (1995)</value>
    </Feature>
    <Feature>
        <name>ItemYear</name>
            <value>1995-01-01</value>
    </Feature>
    <Feature>
        <name>Genre</name>
            <value>Animation|Adventure|
            Comedy|Family|Fantasy</value>
    </Feature>
</items>
```

```
<ratedItem>
    <Feature>
        <name>UserId</name>
            <value>98</value>
    </Feature>
    <Feature>
        <name>ItemId</name>
            <value>109</value>
        </Feature>
    <Feature>
        <name>Rating</name>
            <value>3</value>
    </Feature>
    <Feature>
        <name>TimeStamp</name>
          <value>1997-12-04</value>
    </Feature>
</ratedItem>
```

Figure 10 – User, Item and Rating XML Files Example

After creating the XML files (movies, users and ratings) and placing them under the data folder, selecting the "Load Data From XMLs" menu item in the administrative panel, Figure 11, finishes the application development process. Additional data about users, movies or ratings can be also loaded to the system later by using the same menu.

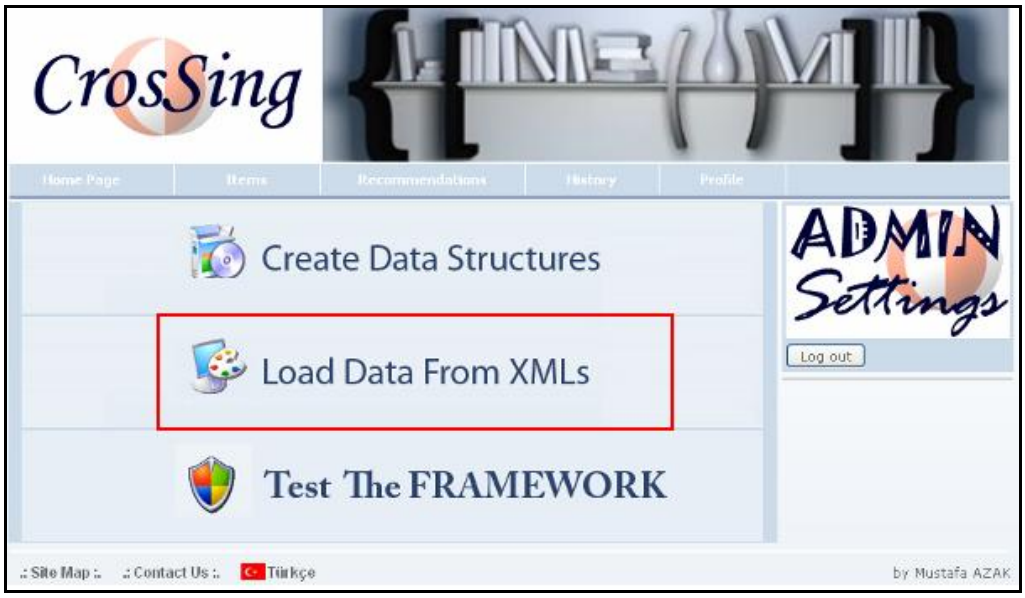

Figure 11 – Data Loading from XML files

We can evaluate the functionality testing as follow:

- Following the steps defined above, we easily create a ready to use online movie recommender system.
- We did not have to implement any recommendation algorithm, did not deal with data management or database related problems, and did not worry about any WEB technologies or GUIs development activities.
- If we do not consider the common vocabulary adapter implementation, we succeed to develop this online movie recommender system in a short time with a minimum effort without writing any code.
- The ability to modify ontology files and create data structures provides the flexibility. In addition, while converting raw data to XML files by adapter systems, additional information can be added about items automatically in order to extend available features. The example of the adding new features can be seen in the data processing section of Algorithm Performance testing.

In addition those, if we add another recommendation domain to the system such as books or music domains, the framework provides cross-domain recommendations automatically.

The benefits of other ontology files are as follows. Advance users can define feature rules and inter-domain rules to improve single and cross domain recommendation quality and accuracy. The rules should also be provided in the XML format based on their ontology files. Users can also develop knowledge based recommendation systems by providing the Personality rules and constituting the knowledge base. In addition, UserSimilarity and ItemsSimilary files are used to define which features are used for the similarity measurements in Content based Filtering and Demographic Filtering. Users can give weights to selected features in order to optimize the distance calculations. This functionality also provides framework to adapt each recommendation domain separately. The

effects of these additional rules can be tested, monitored and validated with build-in testing component called Test Suite. The details of Test Suite are given in the next section.

To mention GUI and Source package components, the user of the framework can modify the general structure of the Graphical User Interface. CSS files and images can be changed to create a custom look according to the user needs.

Source package is very important part of the system. We are expected that academic researchers can also benefit from our framework as it allows extending recommendation algorithms and enables changing the recommendation strategy. Users can extend the existing algorithms or create new recommendation algorithms by implementing the base recommendation interface. The recommendation strategy can be adjusted by modifying the strategy manager class. The effects of these changes can be also tested with Test Suite component.

4.1.2 Developer Testing

In order to improve our framework's usability and capabilities, we performed test with a developer. Fatih Aksel is a graduate student at Computer Engineering department of Middle East Technical University. He is also studying recommender systems and he has hand-on experience with Duine Framework previously.

We provided him the development environment, an example adapter system and Book Crossing dataset [43]. Fatih followed the fundamental steps explained in the application development testing to develop a recommendation system for Books domain. He also tries to modify a recommendation algorithm as he has experience

in recommendation systems. His development time was not too much. The followings are his comments on our framework CrosSing:

"Cross-Domain capability and feature mappings are good. The procedure to develop an application is not complicated and Graphical User Interface seems successful. The accuracy of recommendation algorithms is fair. About modifying the recommendation strategy, it can be managed using a configuration file or a menu item on the administrative panel without need for changing a java class and end-users may have the ability to select which recommendation algorithm they prefer. In addition, knowledge effect can be determined dynamically in run-time using a learning algorithm. The framework may have some performance and memory problems while generating recommendations. Based on my knowledge about another framework (Duine Framework), I can conclude that my overall opinion about the CrosSing Framework is good, it may need some improvements but it is useful. I may also use the framework in my thesis work."

4.2 Algorithm Performance Test

In this section, "Test Suite" component is explained and the performance of the build-in recommendation algorithms of the framework is tested.

4.2.1 Test Suite

It is one of the most important components of the framework. Test Suite provides an environment to test, evaluate and verify the algorithms of recommender engine. The recommendation algorithms are generally tested with the same manner based on some parameters such as neighborhood size, ratings count for a user, training and test users. In addition, Mean Absolute Error (MAE) (see 4.2.2.)

is the most commonly used metrics to evaluate the accuracy of a recommendation algorithm. Therefore, we make use of these commonalities to create a generic test tool which can be applicable to many recommendation algorithms. The test suite also allows us to observe the effects of the knowledge base on the integrated domains and monitor the performance of the personality and feature rules.

There are static and dynamic parts at the interface of Test Suite seen in Figure 12. Recommendation Techniques and the definition of parameters are the static part and in order to change them, the Test Suite code needs to be modified. The other parts are dynamic and they are automatically generated according to the data available on database.

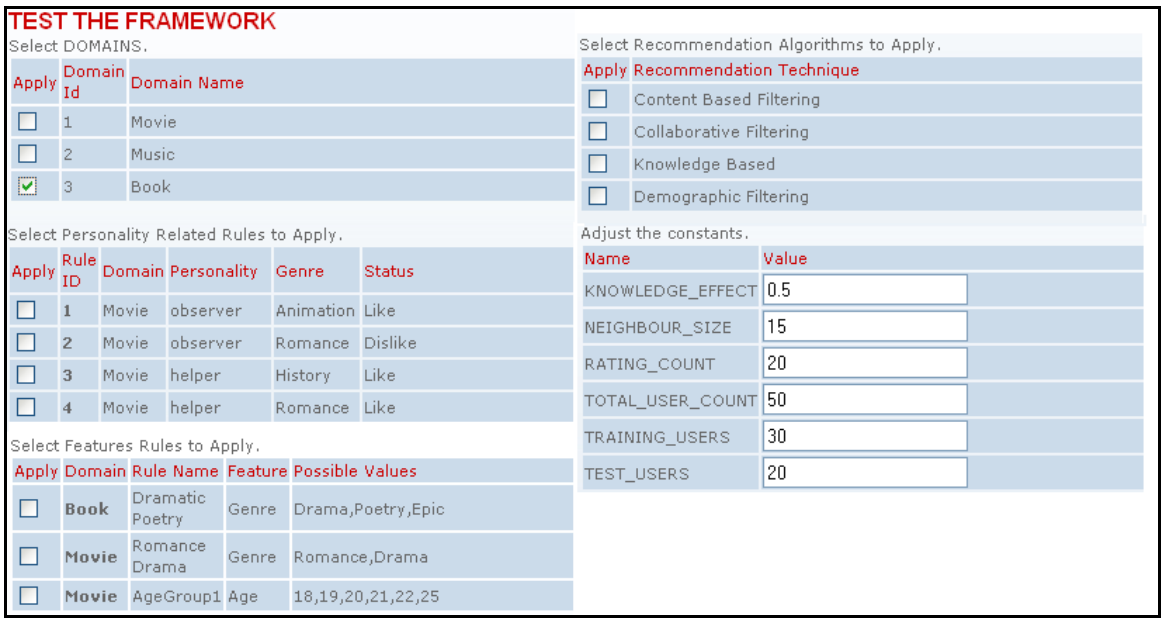

Figure 12 – Test Suite

Test Suite fields are the following:

- Domains: User selects the domain in which the test will run.
- Personality Rules: User selects personality rules to create a knowledge base on the selected domain.

- Feature Rules: User selects feature rules to create a group of values which assumed to be equal. It is only applicable to Content based filtering.
- Recommendation Algorithm: User selects which recommendation algorithm is tested. “Knowledge based” selection is not applicable for its own. It is used with other algorithms and enables the personality rules to be applied on the recommendation generation.
- Parameters: User can adjust the parameters and create different test configurations.

After the required selections are made, the test configuration is run, MAE is calculated and the feedback like in the Figure 13 is provided to user.

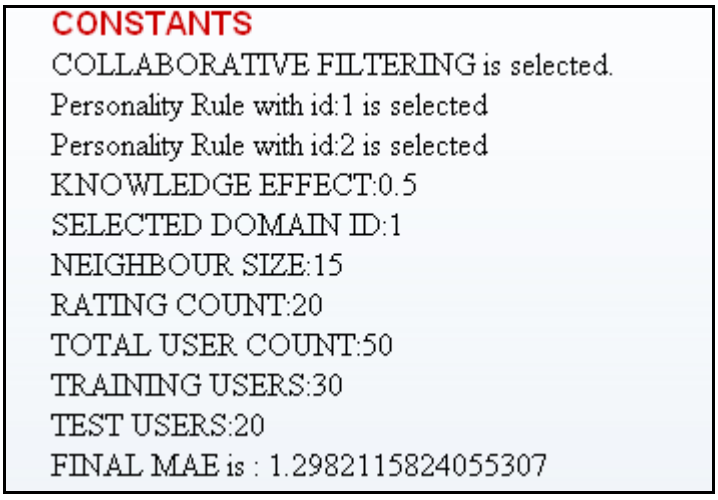

Figure 13 – Feedback message

The test suite allows running only a test configuration. Enabling to run a test scenario with different configurations and to generate graphical test results are the planned future work for this component.

The experimental results given in 4.2.6 are also obtained by using the Test Suite.

4.2.2 Knowledge Acquisition

In order to form a knowledge base about the target domains ‘movies’, ‘music’ and ‘books’, we prepared an **online** survey and 100 people from 10 different countries were asked to complete a short questionnaire. The complete survey can be found in **Appendix A**.

The purpose of the questionnaire was to learn users’ preferences and needs in target domains and their personality features. We also collected the demographic information of users such as age, gender and occupation. As it can be seen on the Figure 14, participants had a nearly equal gender distribution. There were 56 (56 %) women and 44 (44 %) men in the sample. The distribution for participants’ age was as follows: 66% Age 25-30, 18% Age 31 – 42 and %16 Age 19-24. Most of them aged between 20 and 30 which can be the target marketing segment for most of the commercial recommendation systems. The distribution for participants’ occupation was as follows: 34% Grad Student, 25% Software Engineer, 18% Academic Personnel and %23 other professions.

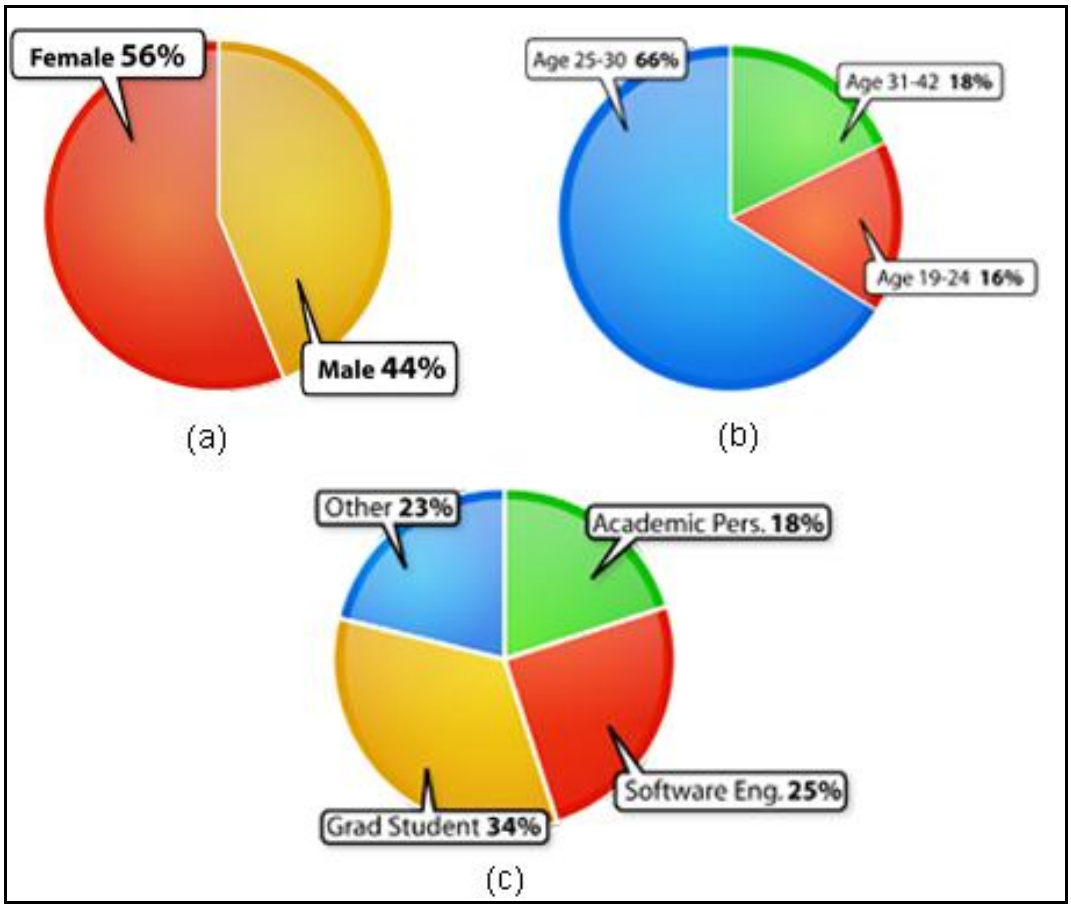

Figure 14 – Gender (a), Age (b) and Occupation Distribution (c)

First, we try to learn participants' prior ranks for the features of items in each target recommendation domain. The participants are asked to sort the features according to the importance for them. The ranks are estimated as follows;

- The first 10 features are taken from the ordered list.
- The points from 10 to 1 are assigned based on their ordering. (First listed feature gained 10)

Therefore, regarding 100 participants, the maximum score for a feature is 1000 if it is listed at top by each participant. The details are given in the Figure 15. The precedence of features carries very important information for the assignments of the initial feature weights which are used in similarity measurements in the

content based filtering. The most important features for the movies are ordered as follows: ***genre, actor, actress*** and ***director***. People are given relative less attention to languages, producers or years of the movies. ***Writer*** is the most important factor that affects participants' opinions about a book. ***Genre, title*** and ***language*** are also listed as the critique features for books. Similar to writers in the book domain, ***singers*** are selected as the most important feature that determines participant's taste on music. ***Genre***, ***composer*** and ***live performance*** of the songs are also selected as important features.

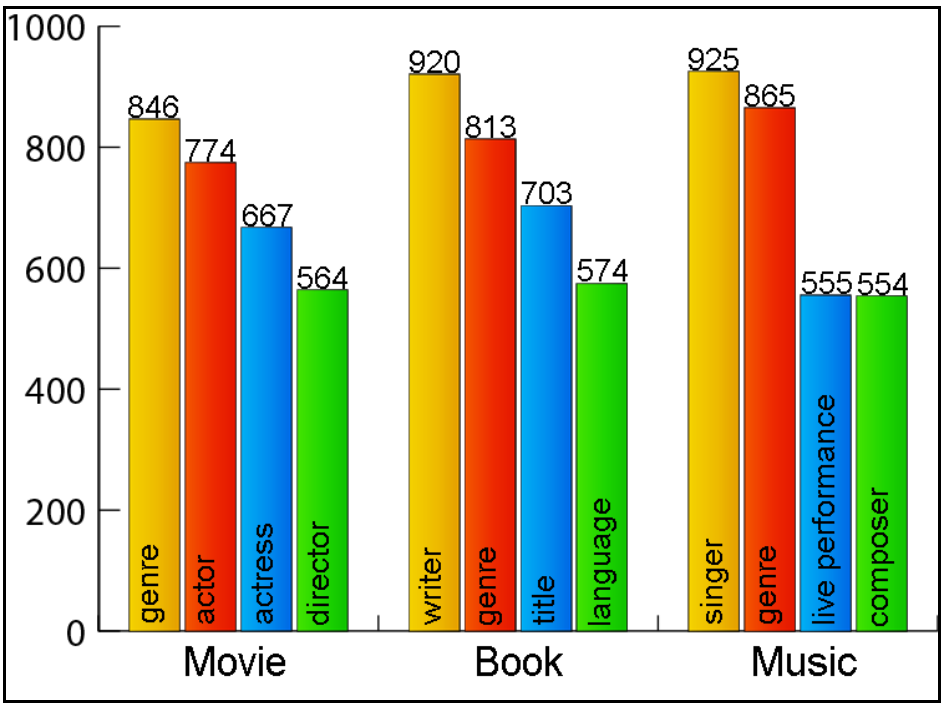

Figure 15 – The precedence of features in movie, book and music domains.

If we make an overall evaluation of the importance of the features on determining participants' tastes, "***genre***" is the most vital feature for recommender systems in "movie", "book" and "music" domains. Therefore, we created our knowledge base rules based on **"genre"** information of the items.

We also asked the participants which types of movies, books and music that they like. This information gained more importance as we had found that “genre” was the most discriminative feature for all domains. Participants could select more than one type. The types of items are ordered in the results according to how many users selected them. The results are given in the Figure 16.

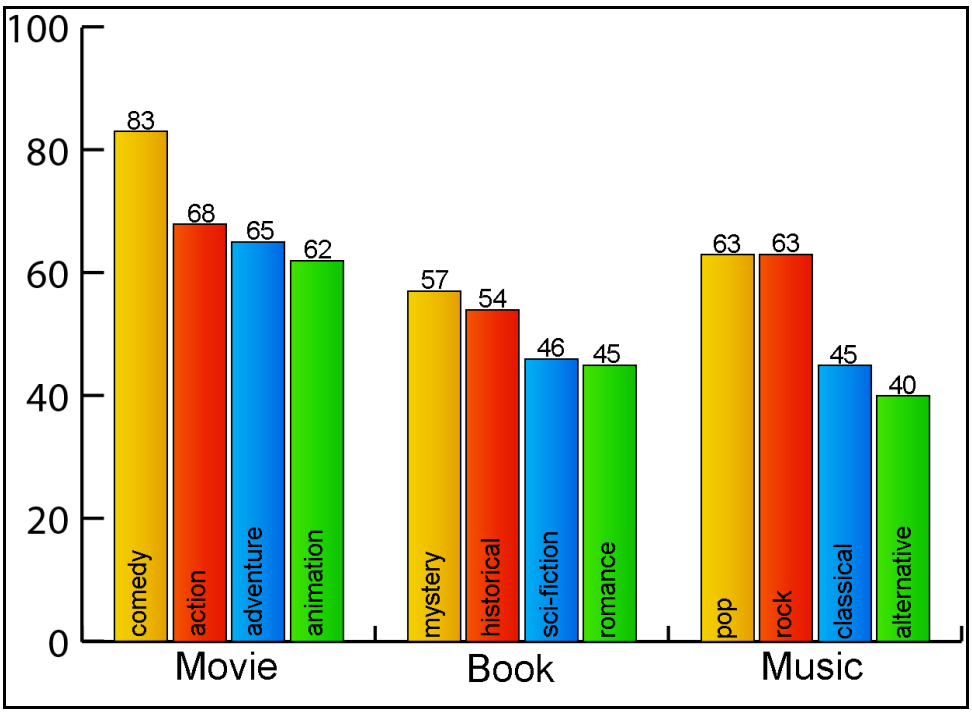

Figure 16 – The popular items types in each domain

Comedy, action, adventure and animation movies are selected by more than half of the participants. Mystery, historical, science-fiction and romance books became the most popular book genres. Pop and Rock music types are top listed in music domain.

The last question required the participants to describe their personality. For personality types we chose the nine types of the Enneagram of personality [44, 45] given above which are useful in classifying characters. The personalities are given with brief information in order to make it easy to understand for

participants. The distribution given in Figure 17 for personality types was as follows: 18% Helper, 17% Observer, 14% Performer, 13% Perfectionist, 12% Adventurer, 10% Peacemaker, 6% Romantic and 4% Boss. The personalities of the participants are used to define the relation with item genres such as like or dislike.

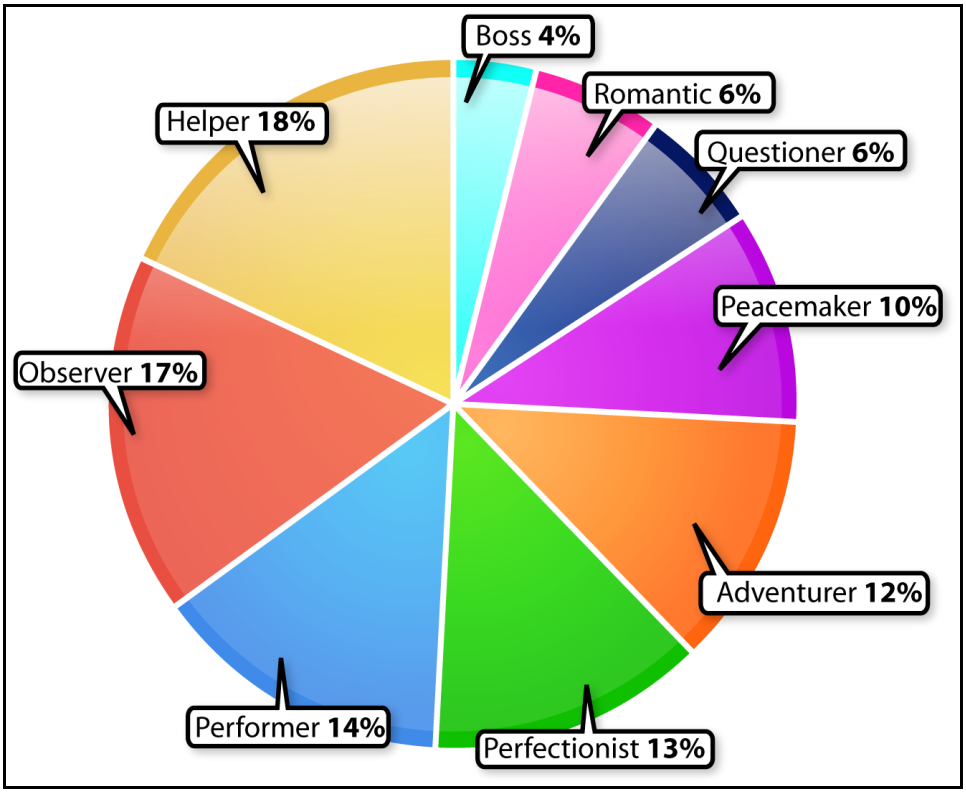

Figure 17 – The distribution of Personality Types

The results of the survey are analyzed statistically with the SPSS (Statistical Package for the Social Sciences) software by the help of two professional statisticians. The statistical results for questions are also calculated separately for each kind of personality type in order to obtain preferences of each personality type independent of others. The variances and differences in the attitudes of

people in a certain personality type regarded as discriminative information and the relations between personality features and movie/music/book types are extracted. After that, "**Chi-square**" tests are applied to these relations in order to make sure that our sample size is large enough to infer that our rules can be generalized. Finally, the feature relations and rules about users' preferences and tendencies are obtained for target domains. The extracted rules for each domain are below.

Table 2 – Movie Domain Rules

- "Observers like Animation movies"
- "Observers **dislike** Romance movies"
- "Helpers like History movies"
- "Helpers like Romance movies"
- "Performers like History movies"
- "Adventurers like Adventure movies"
- "Perfectionists like Animation movies"
- "Perfectionists like Fantasy movies"
- "Perfectionists dislike Adventure movies"
- "Peacemaker like Adventure movies"
- "Peacemaker like Animation movies"
- "Romantics like Animation movies"
- "Questioner like Adventure movies"
- "Questioner like History movies"
- "Bosses **dislike** Adventure movies"

Table 3 – Book Domain Rules

- "Observers like Science-Fiction books"
- "Observers like Fantasy books"
- "Helpers like Romance books"
- "Performers like Historical books"
- "Performers like Romance books"
- "Adventurers like Comics books"
- "Adventurers **dislike** Historical books"
- "Perfectionists like Fantasy books"
- "Perfectionists **dislike** Romance books"
- "Peacemaker like Historical books"
- "Peacemaker like Science-Fiction books"
- "Peacemaker **dislike** Mystery books"
- "Romantics like Romance books"
- "Romantics like Historical books"
- "Questioner like Historical books"
- "Questioner like Romance books"
- "Bosses like Historical books"
- "Bosses like Biography books"

Table 4 – Music Domain Rules

- "Observers like Rock music"
- "Adventurers like Alternative music"
- "Romantics like Latin music"
- "Questioner like Classical music"
- "Questioner like Latin music"
- "Questioner **dislike** Alternative music"
- "Bosses like Classical music"

We tested these rule's effects on the collaborative and content-based recommendation algorithms with different configurations in order to find out which of them could be help us to improve the recommendation quality.

4.2.3 Metrics

In order to determine the prediction quality of our knowledge-based approach which extends collaborative and content-based algorithms, Mean Absolute Error (MAE) metrics [46] was used.

$$MAE = \frac{\sum_{i=1}^{N} |r_i - \hat{r}_i|}{N}. \quad \textbf{(6)}$$

The MAE is computed by first summing the absolute errors of the N corresponding ratings-prediction pairs and then averaging the sum. A smaller value of MAE indicates a better accuracy.

4.2.4 Data Sets, Common Vocabulary Adapters and Data Preprocessing

In order to test our approach we developed common vocabulary adapters for the movie, music and book domains using the datasets available datasets.

For movie domain, we used a popular database, the MovieLens Dataset [8] by the GroupLens Research group at University of Minnesota. The MovieLens data set contains 1682 movies, 943 users and 100,000 ratings (1–5 scales), where each user has rated at least 20. However, MovieLens provides only title, release date and genre information about the movies. Therefore, we matched the movie's information with the IMDb dataset to extract other features required by the item ontology. In addition, we implemented a web crawler using Google Ajax Search API [47] and captured movie posters for our interface.

For book domain, we used Book-Crossing Dataset [43] by Cai-Nicolas Ziegler. The dataset was containing 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. Because of the uncontrollable large data size, we have picked out a smaller dataset, which is constructed with randomly selected users who rated more than 20 books and items which are rated by at least five users and reduced the number of books to 2000. Then, we found ratings of these books and the users who rated them and obtain 7363 users and 19664 ratings. Book-Crossing Dataset also does not contain all the features for books. It contains ISBN, Title, Author, Year-of-Publication, Publisher and the cover image URL. We used Amazon Web Services [48] and LibraryThing *Services* API [49] to complete the necessary features for the 2000 books.

To compare our approach with the state of art collaborative and content-based algorithms, we chose the cross validation technique with holdout method and performed the experiments under the following configurations. A subset of 500 users is selected from each data set. 300, 200 and 100 of them were selected as the training users respectively [50]. And the rest 200, 300, 400 were selected as the active users. These sets were named Movie/Book300, Movie/Book200 and Movie/Book100. As for the ratings from the active users, we varied the number of ratings provided by the active users from 5, 10, and 20, naming them Given5, Given10 and Given20, respectively. As a result, we obtained 9 different configurations in each domain.

As our knowledge base rules make use of user's personality features, some preprocessing is required in order to determine the active user's personalities in these configurations. We used Weka (Waikato Environment for Knowledge Analysis) which is a popular suite of machine learning software in order to classify users via Decision Trees. Using j48 decision tree algorithm with our

survey results about preferred movies, music, books and demographic information, we obtained the decision trees for each personality feature and succeed to classify active users. Figure 18 shows the decision tree to classify user's personality as a "Peacemaker".

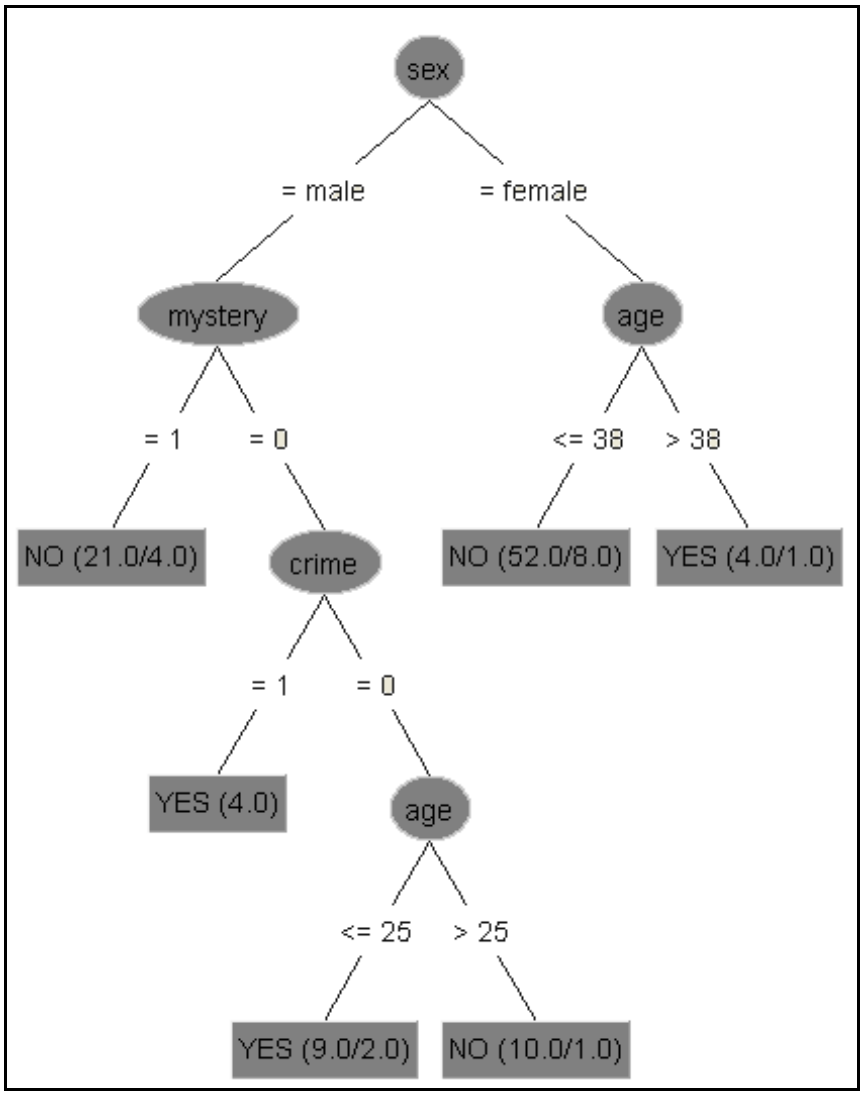

Figure 18 – Decision Tree for Personality of "Peacemaker"

4.2.5 Evaluation Details

In this section, we present our built-in recommendation algorithms with and without knowledge base against a state-of art [1] collaborative filtering technique (CF) and content based filtering technique (CB) on each data set with configurations defined previous section. Our aim is not to find best algorithm but provide a reasonable performance with the framework and demonstrate the effect of knowledge base. Therefore, we did not include the other collaborative filtering and content based filtering algorithms in evaluation details. We performed several tests with different rule combinations and present the most successful ones.

4.2.5.1 Movie Domain

A. Collaborative Filtering:

We prepare 4 different knowledge bases with the following below. The last knowledge base is the combination of others.

Knowledge Base 1 (KB1)

- "Helpers like History movies"
- "Helpers like Romance movies"

Knowledge Base 2 (KB2)

- "Perfectionists like Animation movies"
- "Perfectionists like Fantasy movies"

Knowledge Base 3 (KB3)

- "Peacemakers like Adventure movies"

Knowledge Base 4 (KB4)

- "Helpers like History movies"
- "Helpers like Romance movies"
- "Perfectionists like Animation movies"

- “Perfectionists like Fantasy movies”
- “Peacemakers like Adventure movies”

Comparative Results:

The number of nearest neighbors in collaborative filtering is set as 35 and the knowledge effect variable is set to 0.4 in all configurations since they are best values shown in Figure 19 and 20.

Table 5 – MAE comparison of methods for Movies (CF)

Training Users	Methods	Given5	Given10	Given20
Movie100	CF (State-of-Art)	0.8377	0.8044	0.7934
	CF (CrosSing)	0.8227	0.7984	0.7907
	KB1	0.8217	0.7975	**0.7898**
	KB2	0.8225	**0.7981**	0.7905
	KB3	0.8224	0.7996	0.7913
	KB4	**0.8212**	0.7984	0.7904
Movie200	CF (State-of-Art)	0.8185	0.8067	0.7960
	CF (CrosSing)	0.8001	0.7875	0.7818
	KB1	0.7995	**0.7869**	**0.7808**
	KB2	0.7998	0.7872	0.7817
	KB3	0.7986	0.7878	0.7824
	KB4	**0.7979**	0.7870	0.7813
Movie300	CF (State-of-Art)	0.8055	0.7910	0.7805
	CF (CrosSing)	0.7901	0.7852	0.7769
	KB1	0.7906	**0.7847**	**0.7765**
	KB2	0.7897	0.7848	0.7769
	KB3	**0.7892**	0.7867	0.7770
	KB4	**0.7892**	0.7857	0.7767

In Table 5, we can observe that our prediction approach slightly improve the quality of the state-of-art collaborative filtering algorithm in all configurations.
The improvement is not so significant but this performance is necessary enough to use this algorithm in the framework. Collaborative filtering algorithm also performed better with the knowledge base. Comparing the knowledge bases; KB1 is more useful than the KB2 and KB3 in many configurations. We can sort the effectiveness of knowledge bases as follows: K1 > KB3 > KB2.
The combination of other knowledge bases (KB4) can also be considered as successful. It shows that the results can be better with different rule combinations.
The different configurations results also show that our algorithm is working as we expect. The MAE is decrease with increase number of training users and ratings that each user gives.

We can prove that our framework infrastructure and testing suite is working and personality rules have effects on the predictions. Additionally, we had some disadvantages about determining the users' personality in the survey and the dataset. The participants might make mistakes about deciding their real personalities in the survey and there is an error rate at the decision trees used in WEKA.

Impact of Parameters (Knowledge Effect, Neighborhood Size)

In order to examine the sensitivity of the knowledge effect variable, we performed an experiment where we varied the value of knowledge effect variable that were used and computed the MAE for each variation. The Figure 20 shows the impact of Knowledge Effect variable on MAE for Movie300Given20 with configuration with KB4. It is observed that 0.4 is the optimum value for the knowledge effect variable.

Next, in order to examine the sensitivity of the neighborhood size variable, we varied the number of nearest neighbors that were used and computed the MAE for each variation. Figure 19 show Movie300Given20 with configuration with KB4, however, the other configurations yield similar results. The size of neighborhood does affect the performance but the effect decrease after 30. Therefore, 35 is suitable for the size of neighborhood regarding both performance and computational issues.

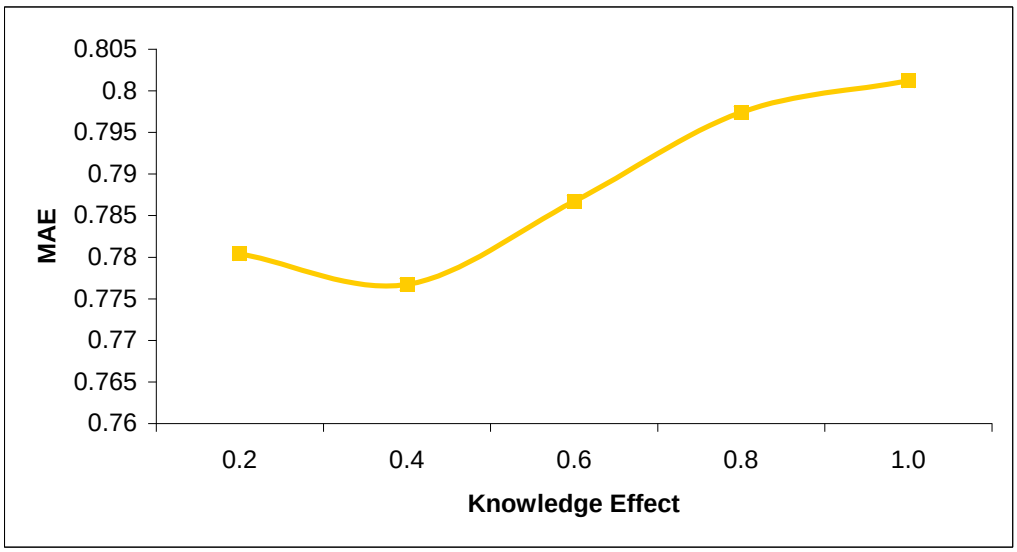

Figure 19 – Impact of Knowledge Effect on MAE (Movie300 Given20 KB4)

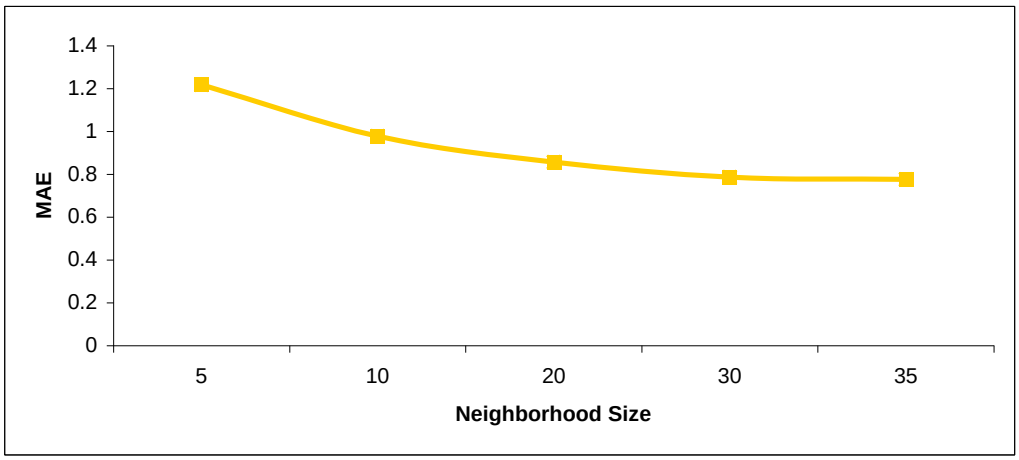

Figure 20 – Impact of Neighborhood Size on MAE (Movie300Given20 KB4)

B. Content Based Filtering:

To compare the content based filtering with the knowledge base support, we chose the first two knowledge bases from the previous part.

Knowledge Base 1 (KB1)

- "Helpers like History movies"
- "Helpers like Romance movies"

Knowledge Base 2 (KB2)

- "Perfectionists like Animation movies"
- "Perfectionists like Fantasy movies"

Comparative Results:

The configurations for content based filtering differ from the collaborative filtering test. As the results can be seen in Table 7, it is expected that different number of test and training users does not affect the MAE results significantly on content based filtering. However, the number of elements in the user's taste list

according to TF-IDF scores for each item feature has an important effect on the results. Therefore, we performed our test on set Movie100 which contains 100 training and 400 active users. We also varied the number of elements to be considered in the user's taste list and constitute the test sets Movie100Sim5, Movie100Sim10 and Movie100Sim20. The performance results of the state of art content based algorithm could not be found but the overall MAE of MovieLens dataset is 0.73 given in [51].

Table 6 – MAE comparison of methods for Movies (CB)

Training Users	Methods	Given5	Given10	Given20
Movie100 (Sim5)	CB (CrosSing)	1.0820	0.9630	0.8946
	KB1	**1.0811**	**0.9623**	**0.8944**
	KB2	1.0814	0.9629	0.8946
Movie100 (Sim10)	CB (CrosSing)	1.0510	0.9457	0.8880
	KB1	**1.0500**	**0.9451**	**0.8878**
	KB2	1.0502	0.9457	0.8880
Movie100 (Sim20)	CB (CrosSing)	1.0328	0.9368	0.8836
	KB1	1.0315	**0.9361**	**0.8834**
	KB2	**1.0314**	0.9367	0.8836

The MAE results of the evaluations for 27 runs are given in Table 6. It is shown that our algorithms performance worse than the state-of-art algorithm. However, we observe the positive effects of the knowledge base support again. When we examine Table 6 further, it can be observed that, as the number of ratings provided from training users to the recommender increase, the accuracy increases; which may be due to the fact that, the content based recommenders accuracy increase with increased clue from the user about his preferences. From the results, we can observe that, the main benefit of the knowledge bases is its power to

produce recommendations even if there is not much clue about the user preferences.

In addition to this, the heterogeneity of the training dataset in terms of genres of movies has a very important effect on the accuracy of the content-based with knowledge bases because the rules are applied only if the movie's genre matches with the rule.

Table 7 – Effect of different number of users

Training Users	Methods	Given5	Given10	Given20
Movie100 (Sim20)	CB (CrosSing)	1.0328	0.9368	0.8836
Movie200 (Sim20)	CB (CrosSing)	1.0348	0.9409	0.8855

4.2.5.2 Book Domain

A. Collaborative Filtering:

We prepare 3 different knowledge bases with the following below. The last knowledge base is the combination of others.

Knowledge Base 1 (KB1)

- "Helpers like Romance books"

Knowledge Base 2 (KB2)

- "Romantics like Romance books"
- "Romantics like Historical books"

Knowledge Base 3 (KB3)

- "Helpers like Romance books"

- “Romantics like Romance books”
- “Romantics like Historical books”

Comparative Results:

The number of nearest neighbors in collaborative filtering is set as 35 and the knowledge effect variable is set to 0.2 in all configurations since they are best value shown in Figure 20 and 21. The performance results of the state of art collaborative filtering algorithm could not be found but the overall MAE of this algorithm on BookCrossing dataset is 1.53 given in [51].

Table 8 – MAE comparison of methods for Books (CF)

Training Users	Methods	Given5	Given10	Given20
Book100	CF (CrosSing)	1.8180	1.7168	1.6134
	KB1	1.8149	1.7141	1.6116
	KB2	1.8140	1.7140	1.6105
	KB3	**1.8109**	**1.7114**	**1.6087**
Book200	CF (CrosSing)	1.5797	1.3932	1.2710
	KB1	1.5769	1.3913	1.2692
	KB2	1.5764	1.3908	1.2683
	KB3	**1.5736**	**1.3888**	**1.2665**
Book300	CF (CrosSing)	1.3828	1.1616	1.0932
	KB1	1.3800	1.1595	1.0916
	KB2	1.3802	1.1602	1.0910
	KB3	**1.3774**	**1.1580**	**1.0894**

We applied the steps that are applied in movie domain. The results in Table 8 are parallel with the results found in previous section. The base performance for our collaborative filtering in book domain seems worse than movie domain however these results are expected because BookCrossing is very sparse data set. In

addition, our algorithms performance is considerable better compared to overall MAE result given in [51]. The MAE is also decrease with increase number of training users and ratings that each user gives.

Comparing the knowledge bases; KB2 is better than KB1 in general. KB3 (Combination of KB1 and KB2) performs best in all configurations.

We can conclude that collaborative filtering algorithms show consistent performance on two different domains.

Impact of Parameters (Training Size, Rating, Knowledge Effect, Neighborhood Size)

We also examine the sensitivity of the knowledge effect variable and the sensitivity of the neighborhood size variable. The Figure 21 shows the impact of Knowledge Effect variable on MAE for Book300Given20 configuration with KB3. It differs from movie domain and found that 0.2 is the optimum value for the knowledge effect variable.

The behavior of the neighborhood size in Figure 22 is same as the behavior in movie domains. Therefore, 35 is selected as optimum value for the neighborhood size variable.

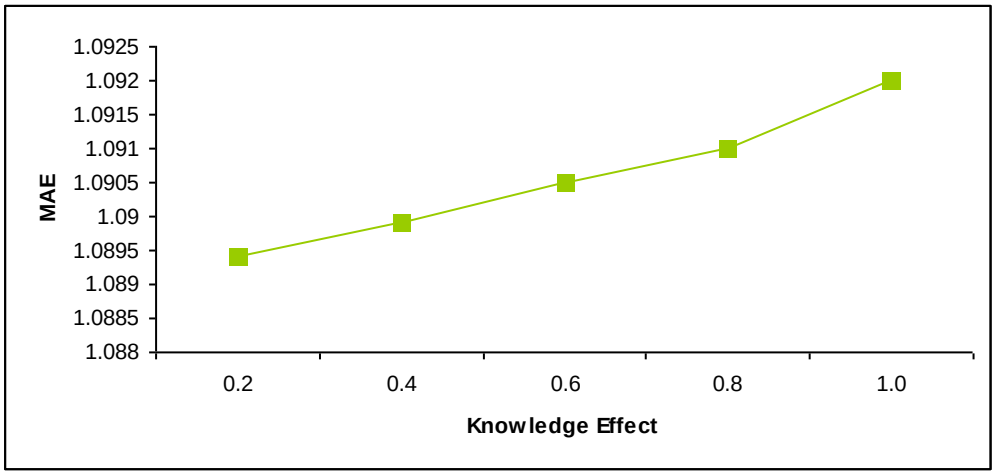

Figure 21 – Impact of Knowledge Effect on MAE (Book300 KB4)

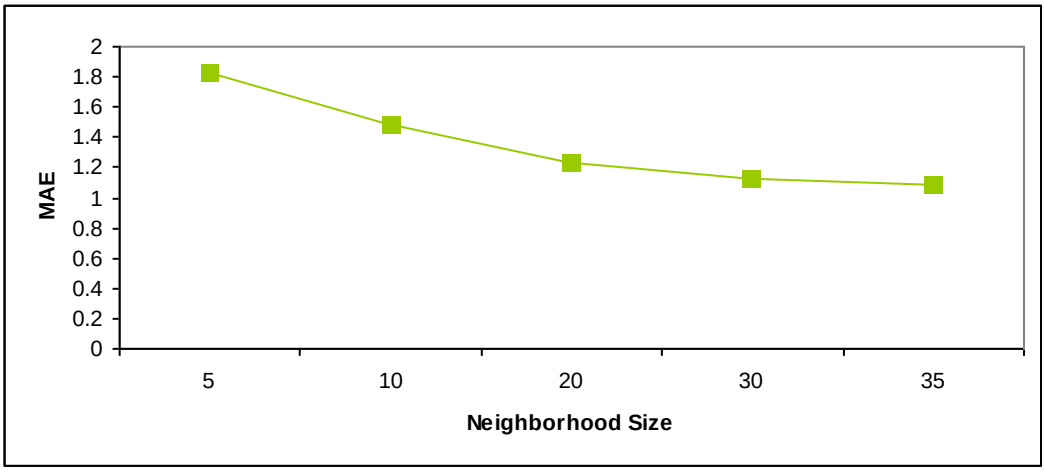

Figure 22 – Impact of Neighborhood Size on MAE (Book300 KB4)

B. Content Based Filtering:

We chose the first two knowledge bases from the previous part.

Knowledge Base 1 (KB1)

- "Helpers like Romance books"

Knowledge Base 2 (KB2)

- "Romantics like Romance books"
- "Romantics like Historical books"

Comparative Results:

The configurations are the same as the content based filtering test on Movie dataset. We also checked the effect of different number of test and training users on MAE results which can be seen in Table 10. We performed our test on set Book100 which contains 100 training and 400 active users. We varied the number of elements to be considered in the user's taste list and constitute the test sets Book100Sim5, Book100Sim10 and Book100Sim20. The performance results of

the state of art content based algorithm could not be found but the overall MAE of MovieLens data is 1.34 given in [51].

Table 9 – MAE comparison of methods for Books (CB)

Training Users	Methods	Given5	Given10	Given20
Book 100 (Sim5)	CB (CrosSing)	**1.6007**	**1.5690**	**1.5555**
	KB1	1.6020	1.5712	1.5566
	KB2	1.6015	1.5692	1.5556
Book 100 (Sim10)	CB (CrosSing)	**1.5416**	**1.5394**	**1.5390**
	KB1	1.5429	1.5417	1.5401
	KB2	1.5424	1.5397	1.5393
Book 100 (Sim20)	CB (CrosSing)	**1.5283**	**1.5347**	**1.5355**
	KB1	1.5296	1.5371	1.5366
	KB2	1.5291	1.5350	1.5358

The results in Table 9 are also consistent with the results found for movie dataset. It is shown that our algorithms performance also worse than the state-of-art algorithm in BookCrossing dataset. In addition, for the first time, we observed the knowledge base rules negatively affect the performance of the recommendation algorithm. Although heterogeneity of the training dataset might be the reason for the decrease in accuracy, the rules need to be revised.

Table 10 – Effect of different number of users

Training Users	Methods	Given5	Given10	Given20
Book 100 (Sim20)	CB (CrosSing)	1.5283	1.5347	1.5355
Book 200 (Sim20)	CB (CrosSing)	1.5211	1.5300	1.5319

4.3 End-User Evaluation

In order to test the application developed in the previous section and gather the people's opinions about overall systems and cross-domain recommendations, we performed alpha testing with real end users. Alpha testing means that testing the software in the developer site by the customer after 80% - 90% completion of developing the application. Therefore, tests are performed on the development machine and the numbers of participants had to be lower than the online survey previously presented. 12 users (6 male and 6 female) are involved in the alpha testing. The average age of users was 28.3. The average duration of a test was 45-60 minutes.

The tests are performed in four steps:

- First two questions of survey were answered by the user before testing began.
- The application is presented and recommendation systems concepts are introduced.
- The user used the system and received both single and cross-domain recommendations.
- The remaining parts of the survey were completed.

The survey was composed of 16 closed-ended questions. The complete survey can be found in **Appendix B**. The questions in the survey were grouped as follows:

Background: These questions are aimed to learn about the participants' profile in terms of familiarity with the recommender systems and their first thoughts before they use the application.

The distribution for participants' familiarity with the recommender systems was as follows: 50% not familiar, 20% not at all familiar and 15% somewhat familiar,

% 15 very familiar. As it can be seen, recommender systems were very new concept to most of the participants and most of them used a recommender system for the first time. Half of the participants thought that the system can accurately predict the items they like while the remaining was neutral about possibility of accurate prediction.

Human–computer interaction (HCI): The main concern was the evaluation of the graphical user interface and feedbacks about usability of the application.

Nearly 85% of the users are satisfied with the system interface and interactions with them. The registration process and the feedback mechanism are found easy by more than half of the participants. The adaptation of the systems was positive and the durations to receive useful recommendation were reasonable. However, receiving similar recommendations was the drawback of the system.

Algorithms: The performance of algorithms and cross-domain recommendations are tried to be evaluated.

The recommendations are evaluated according to the criteria such as “Accuracy”, “Novelty”, ”Enjoyability”, “Diversity” and “Serendipity”. Accuracy and novelty of the recommendations are regarded as very good or excellent. However, diversity and serendipity of the recommendations get relatively poor feedbacks. The enjoyability of the recommendations was also evaluated as good.

All of the users agreed on receiving more personalized recommendations as they provide more feedback to systems. 80% of them found the system recommendation good compared to recommendations which may receive from their friends.

The users were able to observe by which recommendation algorithm the recommendations are generated. They are asked to choose which type of algorithm the like most. The distribution of popularity of algorithms was as follows: 26% Collaborative, 23% Content-Based, 21% Demographic Filtering, 16% Most Popular and 14% Surprise. We can conclude that Collaborative and Content-Based filtering methods are the most liked algorithms as we expected. The interesting one was the popularity of Surprise algorithm was nearly same as Most Popular algorithm. It shows that random generated items can also be liked as much as the items found popular by all users.

The most important question of the survey was how the user found the cross-domain recommendations. The results were not so excellent but they can be determined promising as half of the participant found the cross-domain recommendations good or very good. %25 of the participants found fair and the remaining %25 found poor.

Future: The opinions about the possible further development of the system are asked. 85% of the participant evaluated the systems as promising and all of them would like to use the application as a real system in the future.

To conclude to alpha testing process, the number of the participants is very limited as the test took so long time and the system was only available on the development machine. However, the system can be regarded as successful because all of the participants' evaluations were positive and the results are very promising for the initial release of the system.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this work, we proposed a dynamic framework for developing knowledge-based cross-domain recommender systems. The framework has a generic and flexible structure that data models and user interfaces are generated based on ontologies. New recommendation domains can be integrated with the framework easily in order to improve recommendation diversity. In addition, knowledge base helps to generate useful recommendations in cross-domains and maximize user satisfaction.

Several methods are used to evaluate the various components of framework. Initially, “Functionality Tests” are performed to prove the applicability, flexibility and usability. The accuracy and the performance of the algorithms are appraised through analytical approaches. Finally, the whole system is evaluated through user experiments and questionnaire analysis.

Regarding the design of the framework, the evaluation process and the prototype developed; we can conclude the advantages and disadvantages of our framework among other recommendation frameworks as follows:

The advantages of the CrosSing framework:

- Enables cross-domain functionality and mapping between different domains.
- Has a hybrid recommendation engine.
- Enables development of knowledge based recommender systems.
- Generic and Flexible. Data structures can be managed via XML files.
- Algorithms can be expanded.
- Test suite is included to monitor integrated algorithms performances and rules.
- Provided user friendly interface and can be deployed to web easily.

The disadvantages of the CrosSing framework:

- It's developed as a research study. It should be improved and become mature in order to be used commercially.
- Knowledge acquisition process can be a bottleneck while developing applications.
- Frameworks need community support. It may be open-source or be developed by a group.

As the future work, the framework will be an open-source project and the prototype application will be deployed on the web. The built-in algorithms will be improved to provide better recommendation and beta testing with more users will be conducted to provide further evaluation on cross-domain recommendations.

We can conclude that cross-domain recommendation approach will gain more attention in near future and our framework can be used to develop successful products or experiments by researchers.

REFERENCES

[1] Adomavicius, G. and Tuzhilin A. Toward the Next Generation of Recommender Systems: a Survey of the State-of-the-art and Possible Extension. IEEE Trans. Know. And Data Eng. 17, 6: 734-749, 2005.

[2] Chung, R., Sundaram, D. and Srinivasan, A. ,Integrated Personal Recommender Systems. ICEC'07, 2007.

[3] Anand, S. and Mobasher, B. Intelligent Techniques for Web Personalization. IJCAI 2003 Workshop, 2003.

[4] Szomszor, M., Cantador, I. and Alani, H. Correlating User Profiles from Multiple Folksonomies. ACM Confrence on Hypertext and Hypermedia, 2008.

[5] Amazon.com, www.amazon.com, Last accessed on 31 January 2010.

[6] last.fm, www.last.fm, Last accessed on 31 January 2010.

[7] NetFlix, www.netflix.com, Last accessed on 31 January 2010.

[8] MovieLens, www.movielens.umn.edu, Last accessed on 31 January 2010.

[9] WIKIPEDIA, Cross-selling, http://en.wikipedia.org/wiki/Cross-selling, Last accessed on 31 January 2010.

[10] Burke R., Hybrid Recommender Systems: Survey and Experiments, User Modeling and User-Adapted Interaction, v.12 n.4, p.331-370, 2002.

[11] Terry, D., Goldberg, D., Nichols, D. and Oki, B., Continuous queries over append-only databases, Proceedings of the 1992 ACM SIGMOD international conference on Management of data, p.321-330, June 02-05, 1992.

[12] Kim, Y.S., Yum, B.-J. and Kim, S.M., Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites, Exp. Syst. Appl, 28, 381–393, 2005.

[13] Schafer, J. B., Konstan, J and Riedl, J., 'Recommender Systems in E-Commerce', In: EC '99, Proceedings of the First ACM Conference on Electronic Commerce, Denver, CO, pp. 158-166, 1999.

[14] Debnath, S., Ganguly, N. and Mitra, P., Feature weighting in content based recommendation system using social network analysis, Proceeding of the 17th international conference on World Wide Web, 2008.

[15] Candillier, L., Jack, K., Fessant, F. and Meyer, F., State-of-the-Art Recommender Systems, In Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling, chapter 1, 2008.

[16] Mooney, R.J., Bennett, P.N. and Roy, L., "Book Recommending Using Text Categorization with Extracted Information," Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08, 1998.

[17] Pazzani, M. and Billsus, D., "Learning and Revising User Profiles: The Identification of Interesting Web Sites," Machine Learning, vol. 27, pp. 313-331, 1997.

[18] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P. and Riedl, J., Grouplens: An open architecture for collaborative filtering of netnews, In: Proceedings of the 1994 Computer Supported Collaborative Work Conference, 1994.

[19] Sarwar, B.M., Karypis, G., Konstan, J.A. and Riedl, J., Application of dimensionality reduction in recommender system - a case study, In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop, 2000.

[20] Pazzani, P., A Framework for Collaborative, Content-Based and Demographic Filtering, AI Review 13, 5-6, pp. 393-408, 1999.

[21] Krulwich, B., LIFESTYLE FINDER: Intelligent User Profiling Using Large-Scale Demographic Data, Artificial Intelligence Magazine 18(2), 37-45, 1997.

[22] Burke, R., Knowledge-based recommender systems. Encyclopedia of Library & Information Systems, 69, 32, 2000.

[23] Martinez, L., Perez, L.G., Barranco, M.J. and Espinilla, M., A Knowledge Based Recommender System Based on Consistent Preference Relations, Intelligent Decision and Policy making Support Systems, Springer. pp 93-111, 2008.

[24] Winoto, P and Tang, T., If You Like the Devil Wears Prada the Book, Will You also Enjoy the Devil Wears Prada the Movie? A Study of Cross-Domain Recommendations, New Generation Computing, 26 Ohmsha, Ltd. and Springer: 209-225, 2008.

[25] Gonzalez, G., Lluıs de la Rosa. J, Dugdale, J., Pavard, B., El Jed, M., Angulo, C. and Klann, M., Towards Ambient Recommender Systems: Results of New Cross-disciplinary Trends. In Proc. of the EU. Conf. on A.I., 2006.

[26] Berkovsky, S., Kuflik, T. and Ricci, F. ,Entertainment Personalization Mechanism through Cross-domain User Modeling. 1st Int'l Conf. Intelligent Technologies for Interactive Entertainment, LNAI 3814: 215-219, 2005.

[27] WIKIPEDIA, Framework, http://en.wikipedia.org/wiki/ Framework, Last accessed on 31 January 2010.

[28] Riehle D., Framework Design and Integration: A Role Model Based Approach, Work in progress.

[29] Riehle D. and Gross, T., Role model based framework design and integration, Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, p.117-133, 1998.

[30] Pree, W., Meta patterns - a means for capturing the essentials of reusable object-oriented design. in M. Tokoro and R. Pareschi (eds), Springer-Verlag, proceedings of the ECOOP, Bologna, Italy: 150-162, 1994.

[31] Novay, Telematica Instituut, www.telin.nl/index.cfm, Last accessed on 31 January 2010.

[32] Intelligent Information Systems (IIS) Research Group, Oregon State University, eecs.oregonstate.edu/iis, Last accessed on 31 January 2010.

[33] Apache Mahout - Taste, lucene.apache.org/mahout/taste.html, Last accessed on 31 January 2010.

[34] Anderson, M., Ball, M., Boley H., Greene, S., Howse, N., Lemire, D. and McGrath, S., RACOFI: A Rule-Applying Collaborative Filtering System, In Proc. IEEE/WIC COLA'03, Halifax, Canada, 2003.

[35] Java programming language, java.sun.com, Last accessed on 31 January 2010.

[36] XML - Extensible Markup Language, www.w3.org/XML, Last accessed on 31 January 2010.

[37] Java Servlets, java.sun.com/products/servlet, Last accessed on 31 January 2010.

[38] JSP - JavaServer Pages, java.sun.com/products/jsp, Last accessed on 31 January 2010.

[39] Sun Java Studio Enterprise IDE, developers.sun.com/jsenterprise/, Last accessed on 31 January 2010.

[40] MySQL, www.mysql.com/, Last accessed on 31 January 2010.

[41] Apache-Tomcat Application Server, tomcat.apache.org, Last accessed on 31 January 2010.

[42] Linux operating system, www.linux.org, Last accessed on 31 January 2010.

[43] Ziegler, C.N., McNee, S., Konstan, J. and Lausen, G., Improving Recommendation Lists Through Topic Diversification, Proceedings of the 14th International World Wide Web Conference (WWW '05), 2005.

[44] WIKIPEDIA, Enneagram of Personality, http://en.wikipedia.org/wiki/Enneagram_of_Personality, Last accessed on January 2010.

[45] Personality Types: the Enneagram, Seven Valleys Software, www.svsoft.com, Last accessed on 31 January 2010.

[46] Sarwar, B., Karypis, G., Konstan, J. and Reidl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285–295), 2001.

[47] Google Ajax Search API, code.google.com/apis/ajaxsearch, Last accessed on 31 January 2010.

[48] Amazon Web Services (AWS), aws.amazon.com, Last accessed on 31 January 2010.

[49] LibraryThing Web Services API, www.librarything.com/services, Last accessed on 31 January 2010.

[50] Luo, H., Niu, C., Shen, R. and Ullrich, C., A collaborative filtering framework based on both local user similarity and global user similarity, Machine Learning, v.72 n.3, p.231-245, 2008.

[51] Rafter, R., O'Mahony, M., Hurley, N. and Smyth, B., What Have the Neighbours Ever Done for Us? A Collaborative Filtering Perspective, Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization: formerly UM and AH, 2009.

APPENDIX A

KNOWLEDGE ACQUISITION SURVEY

1. What are your gender (female/male) and age?

Ex) Male, 25

2. What is your occupation?

Ex) Software Engineer, Grad Student

3. Would you sort the following MOVIE features considering the importance for you?

"Title, Actor, Actress, Producer, Director, Year, Genre, Tags, Language,

Country, Length, Locations"

Ex) 1.Genre, 2. Actor, 3. Actress, 4. Title, 5. Producer, 6. Director, 7. Tags, 8. Locations, 9. Country, 10. Language, 11. Year

4. Would you sort the following MUSIC features considering the importance for you?

"Singer, Genre, Song Name, Song Duration, Composer, Year, Performance

(Live/Studio), Producer, Art Work, Video-Clip, Album Name"

Ex) 1. Singer, 2. Genre, 3. Composer, 4.Performance(Live/Studio), 5. Producer, 6.Video-Clip, 7. Art Work, 8. Song Duration, 9. Song Name, 10. Album Name, 11. Year

5. Would you sort the following BOOK features considering the importance for you?

"Title, Writer, Publisher, Year, Genre, Tags, Language, Price, Size"

Ex) 1. Genre, 2. Writer, 3. Title, 4. Tags, 5. Language, 6. Size, 7. Year, 8.Price 9. Publisher.

6. Which types of MOVIES do you like?

☐ Action
☐ Adventure
☐ Animation
☐ Biography
☐ Comedy
☐ Crime
☐ Documentary
☐ Drama
☐ Family
☐ Fantasy/Fiction
☐ History
☐ Musical
☐ Mystery
☐ Romance
☐ Sport
☐ Thriller
☐ War
☐ Western
☐ Other

Ex) Action, Adventure, Fantasy/Fiction, Crime

7. Which types of MUSIC do you like?

☐ Metal
☐ Punk
☐ Rap
☐ Jazz
☐ Hip Hop
☐ Pop
☐ Hard Rock
☐ Alternative
☐ Latin
☐ Rock
☐ Country
☐ Disco
☐ Classical
☐ Blues
☐ Progress
☐ Techno
☐ Reggea
☐ Death
☐ Other

Ex) Rock, Alternative, Techno, Pop

8. Which types of BOOKS you like?

☐ Romance	☐ Thriller
☐ Westerns	☐ Horror
☐ Mystery	☐ Historical
☐ Science-Fiction	☐ Poetry
☐ Comics	☐ Biography
☐ Fantasy	☐ Other

Ex) Mystery, Science-Fiction, Thriller

9. How do you define yourself?

☐ **Perfectionist** - Perfectionists are idealistic. Hold themselves to high standards, and tend to be reliable and organized. Very productive, achieve a lot, and tend to get involved in public service.
☐ **Helper** - Helpers want to be loved or liked. They tend to react to others, and put great emphasis on relationships. Helpers can be attentive, protective, charitable, and warm.
☐ **Performer** - The Performer wants to make a good impression, and to always meet or exceed expectations. Performers are organized, busy, no-nonsense people and are hard workers who get results.
☐ **Romantic** - Romantics are very good at expressing their feelings, and value being in touch with their feelings and the feelings of others. They are free thinkers and don't follow the herd.
☐ **Observer** - The Observer lives a world of the mind, is analytical and likes to figure things out. Observers value being independent, and are comfortable being alone.
☐ **Questioner** - Questioners are motivated by a need for security. They can be anxious and full of doubts, take things too seriously, and can sometimes be suspicious, withdrawn, and sarcastic.
☐ **Adventurer** - Adventurers believe that life's full of interesting things to do. Pleasurable and optimistic. They enjoy planning, but are flexible. If a plan doesn't work, they'll try something else.
☐ **Boss** - Bosses are assertive and self-reliant. They are concerned with the dynamics of power: of getting it, having it and keeping it. Bosses approach situations by attempting to dominate them.
☐ **Peacemaker** - Peacemakers want to avoid conflict and love harmony. Peacemakers are usually generous and open-minded, and see both sides of every issue.

Ex) Romantic, Peacemaker

APPENDIX B

END-USER EVALUATION SURVEY

1. Are you familiar with the Recommender Systems?

☐ Not at all familiar ☐ Not too familiar ☐ Somewhat Familiar ☐ Very Familiar

2. Do you think that the system can accurately predict the items you like?

☐ Strongly Disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly Agree

3. How do you define the registration process of the system?

☐ Very Complicated ☐ Difficult ☐ Fair ☐ Easy ☐ Very Easy

4. How do you define your adaptation to the system in terms of usability?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

5. How long does it take to get useful recommendations from the systems?

☐ Never ☐ Very Long ☐ Reasonable ☐ In Short Time ☐ Immediately

6. Are you satisfied with system interface and interaction with you?

☐ Extremely Dissatisfied ☐ Dissatisfied ☐ Neutral

☐ Satisfied ☐ Extremely Satisfied

7. What had worsened your satisfaction most?

☐ Interface ☐ Poor Recommendation ☐ Receiving Similar Rec.

☐ Feedback Mechanism ☐ Registration Process

8. How do you define the feedback mechanism for recommendations?

☐ Very Complicated ☐ Difficult ☐ Fair ☐ Easy ☐ Very Easy

9. How do you find recommendations in terms of the following criteria?

a. Accuracy ☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

b. Novelty ☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

c. Enjoyability ☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

d. Diversity ☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

e. Serendipity ☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

10. Did the system give more personalized recommendations based on your feedback?

☐ No ☐ Yes

11. Was the system good compared to recommendation you may receive from a friend?

☐ No ☐ Yes

12. Which recommendation type do you like most? (Give points from 5 to 1)

☐ Collaborative ☐ Content Based ☐ Demographic Filtering

☐ Most Popular ☐ Surprise

13. How do you find cross domain recommendations?

☐ Poor ☐ Fair ☐ Good ☐ Very Good ☐ Excellent

14. Do you find the system promising?

☐ No ☐ Yes

15. Would you like to use it as a real system?

☐ No ☐ Yes

APPENDIX C

PUBLICATIONS

Mustafa AZAK, Aysenur BIRTURK, CrosSing Framework: A Dynamic Infrastructure to Develop Knowledge-Based Recommenders in Cross Domains, In 6th International Conference on Web Information System and Technologies (WEBIST 2010), 07-10 April 2010, Valencia, Spain.

Mustafa AZAK, Aysenur BIRTURK, CrosSing: A Framework to Develop Single and Cross Domain Recommenders, The eChallenges e-2009 Conference, 21-23 October 2009, Istanbul, Turkey.