A GRAPH BASED COLLABORATIVE AND CONTEXT AWARE RECOMMENDATION SYSTEM FOR TV PROGRAMS

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

BY

EMRAH ŞAMDAN

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

SEPTEMBER 2014

Approval of the thesis:

A GRAPH BASED COLLABORATIVE AND CONTEXT AWARE RECOMMENDATION SYSTEM FOR TV PROGRAMS

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

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

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

Prof. Dr. Nihan Kesim Çiçekli ______________
Supervisor, **Computer Engineering Dept, METU**

Examining Committee Members:

Prof. Dr. Ferda Nur Alpaslan ______________
Computer Engineering Dept., METU

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

Prof. Dr. Ali Doğru ______________
Computer Engineering Dept., METU

Prof. Dr. Ahmet Coşar ______________
Computer Engineering Dept., METU

M.Sc. Deniz Kaya ______________
Arçelik A.Ş.

Date: 05.09.2014

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: EMRAH ŞAMDAN

Signature:

ABSTRACT

A GRAPH BASED COLLABORATIVE AND CONTEXT AWARE RECOMMENDATION SYSTEM FOR TV PROGRAMS

ŞAMDAN, Emrah
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Nihan Kesim Çiçekli

September 2014, 74 pages

With the increasing amount of TV programs and the integration of broadcasting and the Internet with smart TV's, users suffer the difficulty of selecting the most appealing TV programs among various different programs available. User decisions are mostly affected by the contextual properties of programs such as the time of day, genre, actors and directors of program. This thesis proposes the design, development and evaluation of a graph based context-aware collaborative recommender system for TV programs. The proposed graph based algorithm is based on random walks performed on a tri-partite graph. The graph is constructed by using context aware pre-filtering in order to filter out programs which are irrelevant in the given context. The recommendation list generated by the graph based collaborative algorithm is updated by re-ranking the recommended items according to additional contextual variables. Thus, the proposed recommender system exploits both contextual pre-filtering and post-filtering to produce more effective recommendations. In order to measure the effectiveness of the context variables, we have implemented evaluation metrics on both context-free and contextual graph based methods. We have also tested the effects of parameters used in the graph based collaborative algorithm to the success of the recommender. The results indicate that context can provide better recommendations for TV programs.

Keywords: Recommender systems, collaborative filtering, context-aware recommendation, graph based recommendation, TV program recommendation.

ÖZ

TELEVİZYON PROGRAMLARI İÇİN ÇİZGE TABANLI İŞBİRLİKÇİ VE BAĞLAM DUYARLI ÖNERİ SİSTEMİ

Şamdan, Emrah
Yüksek Lisans, Bilgisayar Mühendisliği
Tez Yöneticisi: Prof. Dr. Nihan Kesim Çiçekli

Eylül 2014, 74 sayfa

Artan televizyon programı sayısı ve televizyon yayını ve internetin akıllı televizyonlar ile entegre edilmesiyle birlikte, televizyon izleyicileri mevcut programlar içinde en çekici televizyon programlarını seçmekte zorlanmaktadır. Televizyon izleyicilerinin kararları genelde programların yayın zamanı, türü, aktörleri ve yönetmeni gibi bağlamsal ögeler tarafından etkilenir. Bu tez, televizyon programları için çizge tabanlı işbirlikçi ve bağlam duyarlı bir öneri sistemini sunar. Önerilen çizge tabanlı algoritma, üç kısımlı bir çizge üzerinde yapılan rastlantısal yürüyüşler yapılması temeli üzerine kurulmuştur. Çizge, verilen bağlam ile alakalı olmayan programların bağlama duyarlı olarak önceden filtrelenmesi ile inşa edilir. Çizge tabanlı işbirlikçi algoritma tarafından üretilen öneri listesi ek bağlamsal değerler kullanılarak yeniden düzenlenir. Bu sayede, önerilen öneri sistemi hem bağlamsal ön filtreleme hem de bağlamsal geri filtrelemeyi daha etkili öneriler üretmek için kullanır. Bağlam değişkenlerinin etkinliğini ölçmek amacıyla, değerlendirme ölçütleri hem bağlama duyarlı hem de bağlama duyarsız yöntemler üzerinde uygulandı. Ayrıca, çizge tabanlı işbirlikçi algoritmada kullanılan parametrelerin öneri sisteminin başarısına etkisi test edildi. Sonuçlar bağlamın televizyon programları için daha iyi öneriler üretilmesini sağladığını göstermektedir.

Anahtar Kelimeler: Öneri sistemleri, işbirlikçi filtreleme, bağlam duyarlı öneri, çizge tabanlı öneri, televizyon programları önerisi

To my family

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. Nihan Kesim Çiçekli for the continuous support of this study, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. It was a big chance and privilege to work with such a friendly, intellectual and motivating supervisor.

I am deeply thankful to my family for their incentive belief and motivating words throughout this thesis. I especially thank to my mother Kadriye Şamdan who shares all emotions with me in this thesis.

I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing the financial means throughout this study with the project number of 112E11. Furthermore, I would like to thank the Ministry of Science, Industry and Technology of Turkey for the support with the project number of 1651.STZ.2012-2. My sincere thanks also go to Arçelik A.Ş. for their support with this research. My true-hearted thanks go to Burak Demirtaş, head of the METU-TECHNOPOLIS R&D Office of Arçelik, for his guidance, supports and friendship.

I am deeply indebted to my friends, Arda Taşcı, Yunus Emre Işıklar, İrem Gökçe Aydın, Mustafa İlhan, Ayşenur Durgut, Anıl Sevim, Ali Karakaya, Deniz Karatay, Başak Meral, Ruşen Aktaş for their encouragement and sharing experiences not only about their thesis studies but also with real life.

Finally, my special thanks go to my fiancé Duygu Çelik for her patience, love and endless support. Her love and friendship made everything including this research easier in my life. I hope to walk with her with this harmony all through my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

API	(Application Programming Interface)
PC	(Personal Computer)
TV	(Television)
BOW	(Bag of Words)
ORM	(Object Relational Mapping)
TF	(Term Frequency)
IDF	(Inverse Document Frequency)

CHAPTER 1

INTRODUCTION

Due to the vast improvements in broadcasting area, there are huge numbers of channels broadcasting TV programs at the same time. In Turkey, there are 311 channels broadcasting on satellite registered to The Radio and Television Supreme Council of Turkey [1]. Because of the density of the programs, TV users suffer from finding suitable programs for them and missing the programs that are available on channels they have never watched. For this reason, TV users need the systems that recommend programs which can be interesting for them.

The advent of Connected TV has provided the ability of recording the TV usage preferences for the manufacturers of consumer electronics and for the users. With the recorded past preferences, the profile of the user while watching TV can be constructed and the user preferences on TV programs can be calculated by using these user profiles.

TV usage is mostly affected by the context of the user who intends to watch TV and the content of the program which is broadcast at that time [2]. The age of the watcher, time of the day, genre of the program might be decisive for the preference of TV consumption. Therefore, utilization of contextual information brings significant advantages to the recommender systems. Contextual information is utilized for pre-filtering the target programs, for modeling the user with the context variables and for re-ranking or filtering the recommendation list [3].

Graph based recommender systems are generally used for connected data such as social media data or professional network for recommender systems [4]. By taking random walks from an initial node, algorithms can converge to the most relevant nodes in the graph. By restricting the type of the initial node to a user node and the type of the final node to an item node, random walks can yield the recommendation in the graph data [5].

This thesis proposes a graph based collaborative recommender algorithm which utilizes contextual attributes of TV programs in order to generate top-N recommendations. In this thesis, the proposed graph based collaborative recommender system uses the recorded past preferences gathered from Arçelik[1], Beko[2] and Grundig[3] users for a specific period of time in order to solve the information overload due to a large number of programs at the same time. This thesis uses a graph based collaborative approach which is a modification of the approach proposed by Bogers [6]. By exploiting context aware pre-filtering and post-filtering, it is intended to get more effective results compared to pure collaborative algorithm. In this thesis, we have exploited the Turkish program guide information of TV programs in order to determine the contextual variables and features of a TV program. The developed recommender system is aimed to be utilized in Arçelik connected TV's.

1.1. Contribution of Thesis

This thesis contributes to the literature of TV recommender systems in the following ways[7] :

- We have improved Phuong et al.'s graph based model[8] with edge weights between node types such as continuous ratings between users and programs, TF-IDF values between programs and terms instead of binary values.

[1]http://www.arcelik.com.tr/
[2]http://www.beko.com/
[3]http://www.grundig.com.tr/

- We have adopted Bogers' approach[6] by using the contextual pre-filtering and post-filtering according to the context attributes special to our data.
- We have used the real world TV usage data which is very sparse. As a result of this situation, we have figured out the need for pre-filtering of real world TV usage data.
- We have designed structural representation for program guide information in Turkish which can be utilized in future searches on Turkish program guide data.

1.2. Organization of Thesis

The rest of the thesis is organized as follows:

Chapter 2 presents organized information about the literature on recommendation systems. A detailed description is given about the notion of recommender systems, and the taxonomy of recommender systems. Formal description of the classified recommender systems is given and the existing work is explained according to the methods they exploit. Previous work on TV recommender systems is also presented. The evaluation methods of recommender systems are described and classified according to the approaches that they utilize.

Chapter 3 introduces our context aware collaborative graph based algorithm and gives the detailed description of our recommender system. First, it provides the general overview and the structure of the system. Design issues are elaborated and the main recommendation algorithm is presented in detail.

Chapter 4 presents the evaluation of the system. It compares the performance of the context aware collaborative algorithm with non-contextual collaborative algorithm. This chapter also summarizes the internal evaluation of our algorithm with respect to the parameters that are used in our algorithm.

Chapter 5 draws conclusions about this thesis work and discusses the advantages and disadvantages of our recommender. Possible improvements and future work to improve the performance of our recommendation algorithm are also discussed.

CHAPTER 2

RELATED WORK

In this chapter general information and terminology about recommender systems are presented. Different recommendation techniques are compared and explained. Previous works on TV and movie recommender systems are discussed considering the techniques they have used. Evaluation techniques and metrics are explained.

2.1. Recommender Systems

A recommender system is described as software tools and techniques providing suggestions for items to be of use to a user [9]. The term “item” is denoted for what the system recommends to its users. Recommender systems focus on the specific type of item such as a book for a digital library, a video for a video-on-demand system and a product for a shopping website according to the area that they are used in.

Recommender systems are primarily designed for generating finite set of items which are aimed to be preferred by the users of the system. With that point of view, recommender systems can be regarded as a mapping between users and items [10]. In their most basic form, recommender systems try to guess the most suitable items based on the user’s preferences and constraints of the system by generating ranked list of items [11]. In order to achieve this task, recommender systems might collect both past preferences on the current system and preferences that are not directly attached with the system such as social media usage information. The preferences are either explicitly expressed, e.g., as rating for a product, or are inferred by analyzing user actions, e.g., as watching time for a TV program.

2.2. Recommendation Methods

Recommender systems are divided into three main categories based on their methods in order to make the recommendations [12]:

- Content-based recommendations: Items that are similar to the items that user preferred in the past are recommended;
- Collaborative recommendations: Items that are liked by the users who have similar preferences are recommended.
- Hybrid approaches: These methods combine collaborative and content-based methods.

In addition, there are other types of recommender systems that are aimed to extend the two dimensional user-item space to multi-dimensional space with additional information related with either items or users. These are:

- Graph-based recommendations: The data is represented in the form of a graph where nodes can be items, users and additional node types like features of items. After building the graph model, items are used not only as output of the system but intermediate nodes to propagate the information in the graph [13].
- Context-aware recommendations: Contextual variables such as the mood of the user, type of the item are used to improve the success of recommender systems by either filtering or re-ranking the recommendation candidates or embedding the contextual information to the recommendation model [3].

2.2.1. Content Based Methods

Content based recommendation systems are designed for suggesting items to users according to the relationship between the content of the item and user's preferences. In order to recommend new items to users, content based recommenders use the past preferences of the users on the system [14].

Past preferences of a user about the items are generally specified by the feedback that users give either implicitly or explicitly. Explicit feedback has the advantage of simplicity since users provide their feedback willingly. In many cases, explicit feedbacks are gathered after a sign up process which enables researchers to learn demographic information about users such as age gender, education, occupation, and location and user interests. On the other hand, implicit feedback methods are based on assigning scores to the user actions on the system such as watch time for a video-on demand website, or hold time for digital library. Implicit feedback methods bring the advantage of not disturbing users for giving ratings to the items that are used [15]. In this thesis, implicit feedback mechanism is used for getting ratings from users. In order to get the ratings for TV programs, we have used the watch time of user for the program for calculating the rating for the program and weighted it by comparing it with the duration of the watched program.

For content based recommender systems, items are generally represented by a set of features. When each item is described by the same set of attributes, and there is a known set of values the attributes may take, the item is represented by means of structured data [16]. In most content-based filtering systems, item descriptions are textual features extracted from Web pages, product descriptions and so on, program or movie descriptions. While representing documents with the terms in them, the main problem is to weight the terms by using some methodology. In order to specify the term weights, the most commonly used method is the term frequency /inverse document frequency (TF-IDF) measure which is based on the idea that rare terms are more significant than frequent items for representing an item (IDF), and multiple occurrences of a term in a document is more important than single occurrences of a term in a document (TF) [17].

TF-IDF is calculated as a combination of two notions, which are term frequency (TF) and inverse document frequency (IDF). Term frequency is the number of occurrences of a term in a document. The notion inverse document frequency is a measure of how much information the word provides and calculated as follows:

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

In the formula of IDF, the number of all documents is divided by the number of documents containing the term t. Then, IDF value is calculated as the logarithm of the quotient. TF-IDF is calculated as the product of TF and IDF.

$$tfidf(t, d, D) = tf(t, d) \text{ x } idf(t, D)$$

In their recommender system PRES, Meteren and Someren used vector space model of articles by assigning TF-IDF weights for the words in the articles which were then used as a features in their feature vectors [14]. In this thesis, we have also used the TF-IDF measure to represent the weight of the terms in the program descriptions.

2.2.2. Collaborative Methods

Collaborative recommendation systems are based on the idea that if two users have similar preferences in the past, they might have similar preferences in the future. In other words, the preferences of similar users are observed in order to make correct suggestions for the target user [18]. This approach falls under the category of user based collaborative filtering techniques. In user based collaborative filtering, in order to predict the preference of a user on an item that he/she has not rated yet, other users whose past rating behavior is close to the current user are searched among the users of the system, and the ratings of those users are used on the item to predict the preference of the current user. User based collaborative filtering technique was first introduced by Resnick et al. in the GroupLens Usenet article recommender [19]. Other applications on different domains have also exploited the user based collaborative techniques; such as Ringo[20] on music domain and Video Recommender[21] on movie domain.

However, when the number of users increases, user-based collaborative filtering techniques become inadequate because of performance issues [22]. Instead, item-based collaborative filtering has been used in order to accord collaborative filtering algorithms to large user bases and facilitate deployment on large scale systems. Item based collaborative filtering focuses on the similarities between the rating patterns of items rather than the similarities between the rating patterns of users. In item-based collaborative filtering algorithms, if two items are preferred by similar ratings then those items are considered as similar. Although item based collaborative filtering techniques are similar to earlier content based approach, they exploit user preference patterns rather than extracted information from items [23]. Most well-known usage of item based collaborative filtering is Amazon.com which provides users with similar product recommendations [24]. Their algorithm produces recommendations in real-time, scales to massive data sets, and generates high quality recommendations.

In order to find the similar items or users in collaborative filtering techniques, several similarity metrics have been proposed and used in the literature. Pearson Correlation makes use of the commonly rated items in order to calculate the similarity between users[21] :

$$\mathrm{p_{uv}} = \frac{\sum_{c \epsilon c_{uv}} (\mathrm{r_{uc}} - \overline{\mathrm{r_u}})(\mathrm{r_{vc}} - \overline{\mathrm{r_v}})}{\sqrt{\sum_{c \epsilon c_{uv}} (\mathrm{r_{uc}} - \overline{\mathrm{r_u}})^2} \sqrt{\sum_{c \epsilon c_{uv}} (\mathrm{r_{vc}} - \overline{\mathrm{r_v}})^2}}$$

In the equation $\mathrm{r_{uc}}$ denotes the preference of user u for item c and $\mathrm{r_u}$ represents the mean of the preferences over items rated by user u. $\mathrm{C_{uv}}$ represents the set of items that users u and v rated together

Spearman Rank Correlation uses the same formula with Pearson correlation coefficient but instead of using the ratings of users directly, this method ranks the ratings by giving the rank of 1 to the highest ranked item and assigning higher ranks to lower rated items [25].

Unlike Pearson and Spearman Correlations, Cosine Similarity metric[26]uses all the ratings that are given to items by the users in order to get the rating vector. When the rating about an item is unknown, 0 is assigned as rating. After composing the feature vector, cosine distance is calculated between vectors of user pairs for calculating the similarity between them. Feature vectors of ratings from all users for an item are used in order to compose the rating vector for items in item based collaborative filtering [27].

Determining the number of similar users is also a challenging task for the success of the recommender. Hill et al., have randomly sampled the candidate users for neighborhood for decreasing the time required to find similar users by taking into consideration the expense of accuracy lose [21]. In their research, Herlocker et al. stated that the number of similar users, or items for item based collaborative recommender system, should be constrained to a limited neighborhood of k users [25]. In their research, they have found the $k = 20$ as the best performing value. However, in the research conducted by Lathia et al., it is stated that this number mainly depends on the domain that the recommender is used for. In the scope of their work, they have proposed dynamically adapting the neighborhood size used for each user [28].

Similar to their research, we have also restricted the neighborhood size for efficiency reasons. The details are explained in Chapter 3.

2.2.3. Hybrid Approaches

In some cases, single recommender system may not be appropriate for particular domains. For example, item based collaborative recommender algorithm is not suitable for the systems which do not include adequate ratings. In this case, a content based recommender system can be used instead of it. Furthermore, it is proved that hybrids of various recommender algorithms outperform individual algorithms in some applications[29].

In his survey, Burke[30] has analyzed hybrid recommender systems, and divided them into seven categories:

- Weighted recommenders combine the recommendation scores that are taken from several different recommendation algorithms to produce the final recommendation list for each user.
- Switching recommenders use different recommender approaches interchangeably to get best result according to specific conditions.
- Mixed recommenders present the results of several recommenders together similar to weighted recommenders. However, this type of hybrid recommenders does not necessarily combine the recommendation lists.
- Feature-combining recommenders make use of features from different recommendation data sources in order to build a single recommendation algorithm.
- Cascading recommenders are based on the idea of refining the results of one recommender in another recommender algorithm.
- Feature-augmenting recommenders use the output of one algorithm as one of the input features for another.
- Meta-level recommenders are similar to feature-augmenting algorithm but it gives the learned model by one algorithm to another algorithm instead of recommendation results.

Liu et al. have developed a recommender system for Google News that combines the content based recommendation algorithm that uses learned user profiles with existing collaborative filtering mechanism for composing news recommendation list for

Google users. As a result, it is concluded that hybrid approach improves the performance of the news recommendation [31].

In the scope of our work, we have used feature combining hybridization technique. We have used content based approaches by using TF-IDF similarity between TV programs and terms, while we have used the graph based collaborative recommender in order to find the similar users to the target users.

2.2.4. Context-aware Recommenders

Traditional recommender systems are designed by considering only the past preferences of a user on the items in the recommendation domain. Therefore, the input data for a traditional recommendation algorithm is in the form of <*user,item,rating*>. In contrast, context-aware recommender systems are based on the knowledge of partial contextual attributes that are somehow known by system in addition to the past preferences. That is, context-aware recommender system considers not only whether a given user liked a specific item, but also the contextual situation in which the item was preferred by the user [3]. Thus the input data is in the form of <*user, item, context, rating*>.

Context-aware recommender systems are divided into three main categories with respect to the usage of contextual information. These are contextual pre-filtering, contextual post-filtering and contextual modeling [3].

In contextual pre-filtering, the contextual information is used for selecting only relevant data as a candidate items for recommendations. After filtering the items which are not attached with the given context, any recommendation method can be used in order to predict the ratings of the user. Instead of using whole rating set for building an estimation model, contextual pre-filtering uses only ratings that are pertained by the given contextual variable. In their research, Adomavicius et al. have used various contextual attributes to filter out data and compared the results with non-filtered data. They have figured out that although filtering generates better results in general, sometimes it is unnecessary to use the filters [32]. In some researches, the idea of splitting the user profile instead of candidate data is used. In their work, Baltrunas and Amatriain present the idea of micro-profiling, which breaks the user profile into some possibly overlapping sub-profiles where each of those profiles represents the user in a given context. With the help of this approach, the recommendations are made by using the sub-profiles instead of one single user model [33].

In contextual post-filtering, the contextual information is used after the recommendation made by another algorithm to re-rank the recommendation results to provide better interaction or to filter out redundant recommendations like in contextual pre-filtering. This approach is particularly useful when the final recommended items are limited because of the domain specific reasons. Panniello et al. present an experimental comparison of pre-filtering algorithms with two different post-filtering algorithms where one of them is aimed to post-filter the resulting recommendation list and one of them is designed for reordering recommended items. As a result, reordering approach performs better than pre-filtering, while pre-filtering performs better than post-filtering. This result indicates that the best approach to use for a context aware recommender system can change according to the application domain [34].

The contextual modeling approach makes use of contextual information directly in the recommendation function as an explicit predictor of a user’s rating for an item. From the dimensionality point of view, it increases the dimensionality of the recommendation function by one by adding the contextual information to recommendation model. In their travel guide system named UbiquiTO, Cena et al. have used the contextual modeling by adapting their content based recommender with contextual features like device type that uses their system, location, time and so on [35].

In the scope of this thesis, we have developed a context-aware recommender system that uses context-aware pre-filtering by using the time of the day of a program and genre of a program to filter out the redundant candidates for recommendations, and context-aware post-filtering by using actors and directors in the program for possible genres to re-rank the recommendation list.

2.2.5. Graph-based Recommendations

While traditional recommender approaches focus only on properties of the dataset to make recommendations, graph-based recommender systems make use of not only the regular properties but also the connectivity properties of the dataset by representing the recommendation problem as graph projection [36]. Graph-based approaches are not invented to build a novel approach for recommender algorithms. Instead, it is aimed to take advantage of connectedness of the dataset in recommender systems.

In the scope of graph based approaches, users and items are represented as the nodes in the graph and the similarity metrics between users and items are represented with edge weights. If the connection does not imply a weight, an unweighted edge is used. An example graph representation used in the research of Huang et al. can be seen in Figure 2-1 [37]. They have used similarity metrics between users and books according to purchase history, among users according to demographic connections and among books vector based similarities in their recommender system designed for a digital library.

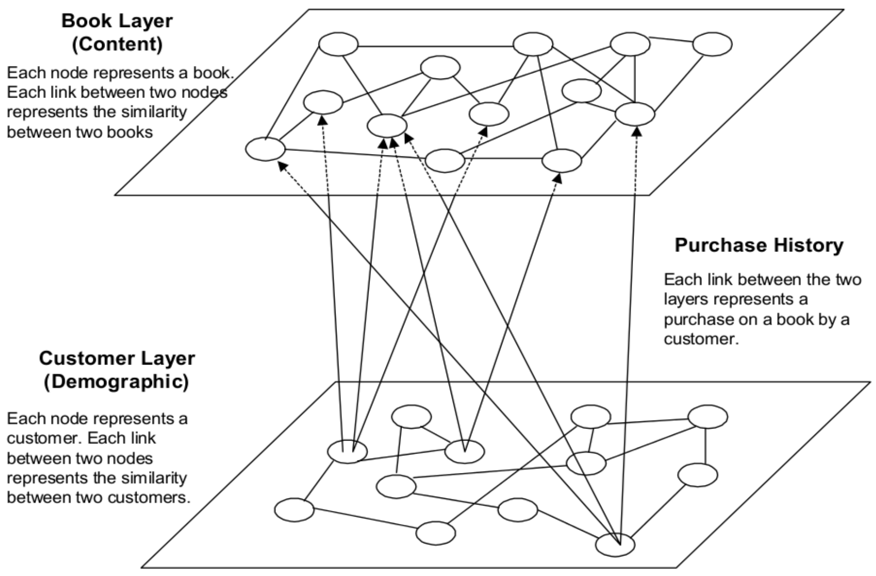

Book Layer
(Content)
Each node represents a book. Each link between two nodes represents the similarity between two books
Purchase History
Each link between the two layers represents a purchase on a book by a customer.
Customer Layer
(Demographic)
Each node represents a customer. Each link between two nodes represents the similarity between two customers.

Figure 2-1: Bipartite Graph in Huang's Research

Graph-based approaches are generally used for collaborative filtering by using the connectivity between users and items of the system. In the research conducted by Baluja et al., a graph-based algorithm named Adsorption is developed for collaborative filtering which aims to be successful in the graphs where both labeled and unlabelled nodes reside by propagating information in the graph for finding labels for unlabelled nodes via random walks [38]. In their research, Öztürk and Çiçekli have extended the Adsorption algorithm by enriching it with content based results. They have improved the success of Adsorption with their hybridization [39]. Moreover, Phuong et al. use the graph based approach for combining the content based and collaborative recommendation approach. They have proved that their graph based approach outperforms a pure collaborative filtering, a pure content-based filtering, and a hybrid method [8]. In our research, we have improved Phuong et al.'s graph based model with edge weights between node types such as continuous ratings between users and programs, TF-IDF values between programs and terms instead of binary values.

Graph-based approaches are used for context-aware recommendations by embedding the contextual features to the graph projection of dataset. Bogers have built a contextual graph (see in Figure 2-2) by embedding context attributes as node types to the graph. In this research, algorithm named ContextWalk, which is a modification of random walk algorithm, is proposed for calculating the similarity between each node in the graph[6]. Since, ContextWalk can be used for calculating the similarity between any pair of node types, his algorithm is capable of achieving other recommendation tasks such as actor-to-movie recommendation, without the need for retraining or changing the recommendation model.

This thesis is based on the idea of random walks on the contextual graph which is proposed in Bogers' research. In the scope of our research, we have constructed our own graph and used Bogers' approach to calculate similarities between user nodes for collaborative filtering. However, because of the performance issues, we did not embed the context variables. Instead, we have used our context variables for pre-filtering purposes by eliminating the redundant nodes from the graph.

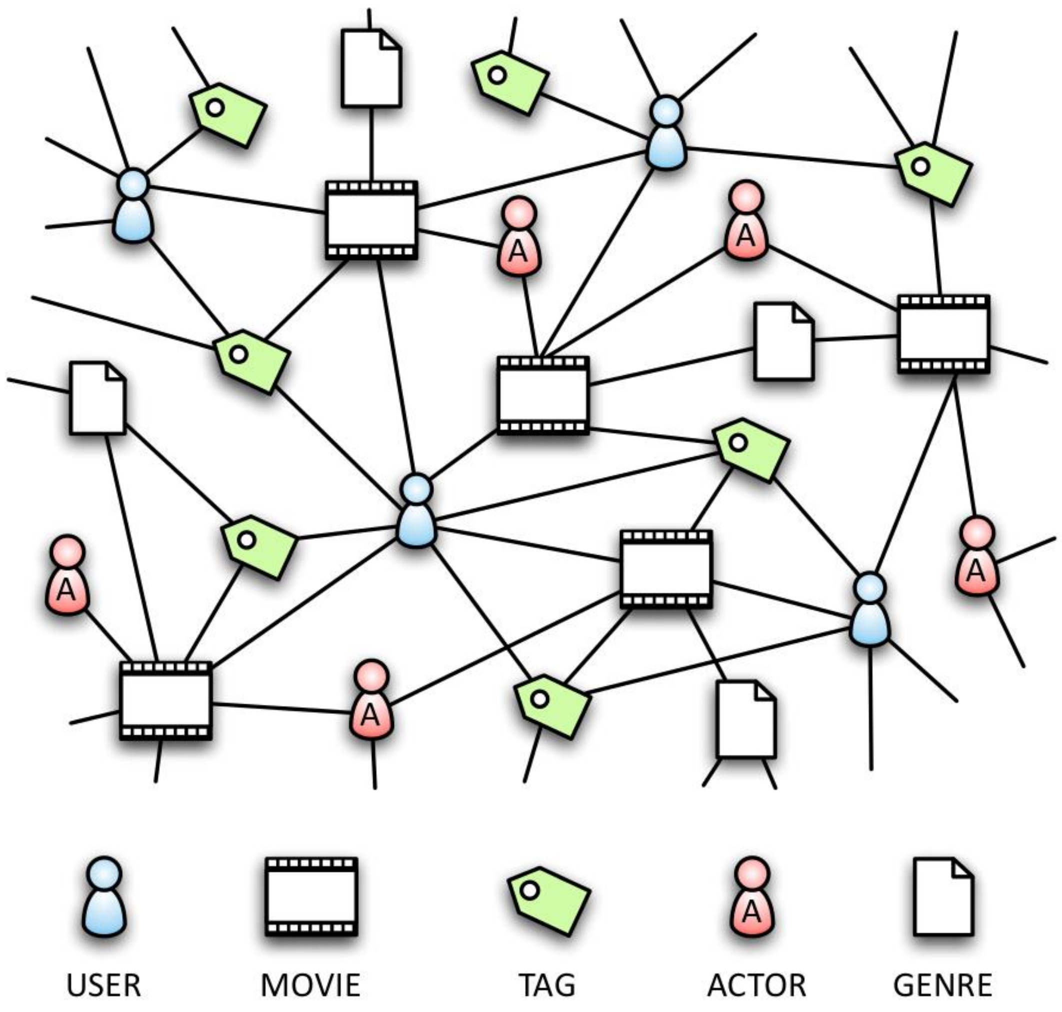


Figure 2-2: Contextual Graph in Bogers' Research

2.3. Recommender Systems in TV Domain

Digital TV's have been evolved in such an expeditious way that they can provide not only telecasts the existing air, satellite or cable TV, but also contents such as video-on-demand, smart applications and so on. Users can select the contents they prefer, but they can face the problem to find the contents they are interested in. Therefore, demand for the recommendation systems for TV users is increasing in recent times [40].

As a specialized domain of recommender systems, TV recommender systems are mainly divided in three categories with respect to the method they have used; content based recommender systems, collaborative recommender systems and hybrid systems.

In content based recommendation, the items are represented by the features inferred from their content. For television environment, it is hard to get the features from the video and audio streams because it requires semantic interpretation of the video and audio streams [41]. That’s why the mostly used source for feature extraction is the textual sources like EPG (Electronic Program Guide) which is supposed to include brief but explanatory information about TV programs. The most usual way to represent a TV program with EPG data is BOW (bag-of-words) approach, where frequencies of words are retained, discarding any grammar/semantic connection. Usually the words are pre-processed by means of tokenization, stop-words removal and stemming. In the research and implementation conducted by Bambini et al., Latent Semantic Analysis (LSA) is used together with the bag of words approach for automatic indexing and searching of the EPG document [42]. In our research, we utilize the BOW approach to represent the content. However, we have placed the words as terms in the graph connected to program nodes, while Bambini et al. used them as attributes in the feature vector.

Collaborative recommender systems have been also applied to the TV domain widely. In the research conducted by Kim et al., user profiles are built by using a scoring technique called CF-IUF (category frequency-inverse user frequency, which is a modification of a well-known information retrieval concept TF-ID[43]. The method aims to handle the bias towards best-selling content, and series contents containing many episodes [44]. They have defined a term called category with the combination of the content genre, provider, etc. and calculated the frequency of each category for a user by using the CF-IUF formula they have invented. All category frequencies for a user constitute the feature vector for a user. Since the number of clusters is not well defined in the recommender systems for televisions, it is better to use algorithms other than k-means. That’s why, they have used the ISOData algorithm, which minimizes the sum of squared errors between data points and their closest cluster centers and automatically determines optimum number of clusters. For collaborative filtering in the clusters of users, they have used Spearman correlation coefficient. In another research by Kwon and Hong, a collaborative recommender system which aims to relieve the cold start problem of collaborative recommender systems are designed for TV programs [45]. In the scope of their work, in order to cope with the cold start problem, they have used raw moment-based similarity which is based on the idea of detecting expected difference between two linear variables.

There are also hybrid recommendation systems that use the content based and collaborative filtering algorithms together in the TV domain. Martinez et al. introduce queveo.tv: a recommendation system for personalized television program. It proposes a hybrid approach (combining content filtering based techniques with collaborative filtering) and also offers to exploit the social network usage of users such as comments, tags, notes, etc.[46]. They use the post hybridization method that uses mixed recommenders approach for combining recommendation results. Their proposed recommendation algorithm is described in the Figure 2-3.

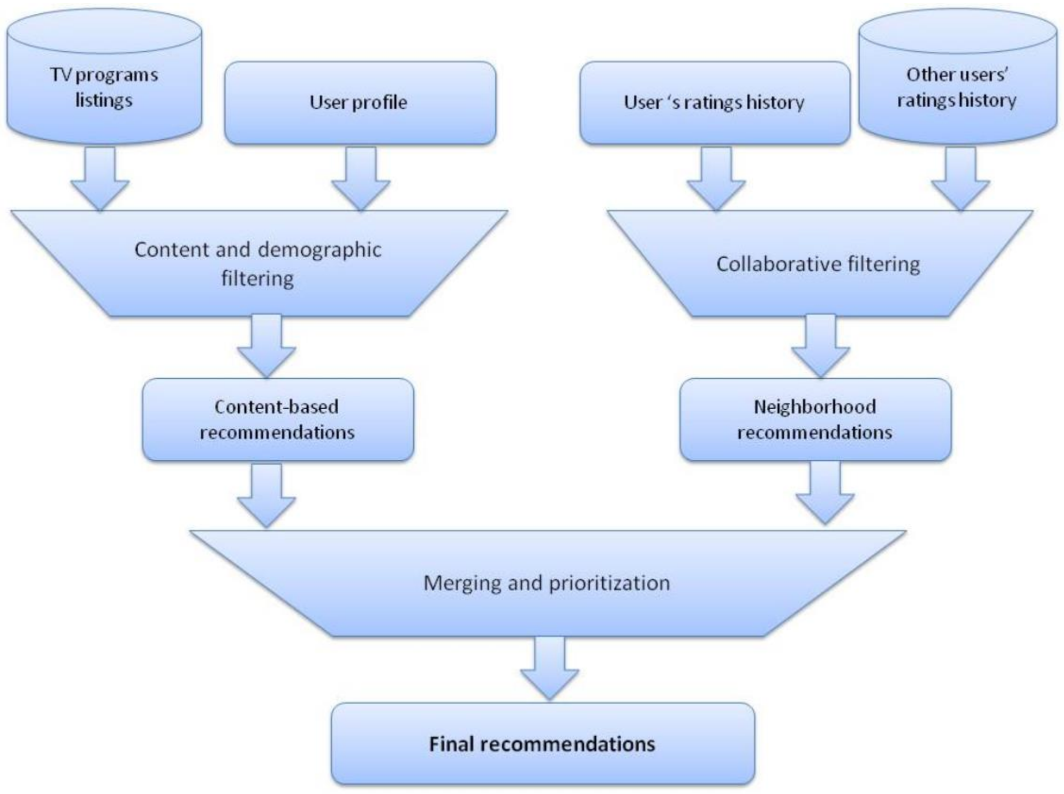

TV programs listings
User profile
User 's ratings history
Other users' ratings history
Content and demographic filtering
Collaborative filtering
Content-based recommendations
Neighborhood recommendations
Merging and prioritization
Final recommendations

Figure 2-3: Hybrid Recommender System of Martinez et al.

Most of the current recommender systems designed for TV programs does not take the notion of context into consideration. They only run on two dimensional *User*x*Item* space. In the research conducted by da Silva et al., a contextual user profile which is the result of the aggregation of the user contextual information like user personal data profile, and the genre of TV program is used for filtering. They have implemented a contextual filtering method similar to the content based filtering but using the contextual information such as date, time and place of the origin of the TV program and the user. Their work shows that the context notion improves the performance of the recommender system. Furthermore, they propose to extend contextual features by adding contexts such as the room of television in the house, domain of TV usage [2]. Similar to their work, we have used the genre and the time of day of the program for pre-filtering.

TV is usually viewed by multiple members of the groups sitting together. So a TV recommendation system should not only provide personalized programs for individuals, but also be able to recommend programs to multiple viewers taking care of the preferences of the majority of viewers, in the case where the viewers are watching TV at the same time, and in the same spot. In order to overcome this problem, Yu et al. propose a group recommender system for multiple viewers using profile merging [47]. There are also other methods to provide group recommendations which are merging recommendations and group agent approach. Merging recommendations method first creates the recommendation list for each individual in the group then, merges the program recommendation lists of those users in the group, while group agent method forces users to register a common account for them and input their original preferences to that account. Group agent method is not so applicable because the common agent fails with the absence of one of the group members or additional members to the group. At this point, it is good to mention that there are also some research to understand who is watching the TV at a specific time called hidden eye technology[48]. Yu et al. take the feature vector of each user as input and make use of the total distance minimization to make a common profile from the profiles of the users in the group. To be specific, they do not include every feature in the feature vector of all users but they use only the positive or negative valued features for making computation less complicated and easier.

In a more recent research conducted by Shin and Woo, not only the independent individual user profiles but also the group characteristics are taken into consideration. They call their system socially-aware because they allow users to harmoniously decide which of the programs of interest to watch [49]. According to the social situation in the group, they have developed three ways to recommend an item.

In the case in which all users are interested in the same program, the assistant automatically selects it as the best program and recommends it, though alternative programs would also be indicated.

In the case where users have similar preferences, age, and/or interests, the assistant determines the program to view by sifting through the feedback obtained regarding the list of preferences for a set of possible programs.

If the situation does not fit in any case, the recommender selects a program by asking for both category and programs of interest.

They have used a graphical user interface to let the user give his/her preferences regarding the metadata of the programs; i.e., preferred genres, actors and keywords explicitly in order to later use in profile merging and combination. They have found that they made further improvement to the previously proposed group recommendation systems by understanding the group characteristics. The interesting point they have showed is that the participants of their evaluations had different preferences when they have watched TV with other people. Although their preferred programs spread in every category when they were alone, their choices changed to other categories when in a group. In the case of the family group, participants preferred programs in the entertainment category that might be acceptable for all members of the family; consequently, their interests in educational programs decreased.

In the early recommender systems for televisions developed in the 90's, the only data source used was the agenda of the TV channels and the recommendations were mostly channel based. In more recent times, with the including of the EPG as a data source, recommender systems made use of richer information such as actor names, genres, program descriptions suitable for bag of words approach and so on. However, with the emergence of the social networks, there are vast amount of personal data on web that can solve the cold start problem. Cold start problem means that in order to make recommendations to a specific user, we need to know more about him/here and this requires some training time for any recommender system to operate.

In the Notube project founded by European Union, Aroyo et al., have built a recommender system that uses existing web services and shared background knowledge to collect, enrich and recommend TV data. This work can be regarded an aspect of TV-Web convergence that can open a door to new types of TV application in the future[50].

In Notube, they analyze the information that user generates on some social network sites with the permission of the users. Their system currently supports Facebook, Twitter and LastFm. As they have said, usage of the cross platform data for an individual may yield better results with further research.

2.4. Evaluating Recommender Systems

In order to measure the success of a recommender system, there are various methods which are classified according to the evaluation strategy that they use. Those are offline experiments, user studies and online experiments.

2.4.1. Offline Experiments

Offline experiments are conducted to measure the success of the recommender system with an existing data and without further interaction with the users of the system. The aim in this approach is to simulate the users' behavior while using the system [51]. Offline evaluations are easier to conduct and more economical comparing to other evaluation methods with multiple algorithms, since it does not include any interaction with users.

For offline experiments, the data set is divided into two parts: A test and a training set. The ratings in the training set are used by the recommendation algorithm to predict the ratings in the test set, which can then be compared to the actual ratings in the test set. Most commonly used offline experiment method is k-fold cross validation, in which the data set is partitioned into K subsets. From these subsets, one is separated for using as the test set; the other subsets are used as training set. This process is repeated K times, each time with a different test set [52].

In this thesis, we have used 3-fold cross validation method in order to measure the success of our recommender system. Because of the time issues, offline experiments are well fitted to our needs to evaluate our algorithm.

2.4.2. User Studies

Experiments with user studies are generally conducted by asking users to interact with system. Users are usually requested to ask to fill forms or questionnaires before starting the experiments. Such experiments are not preferable economically since it may require some payments to participants of these studies [51].

2.4.3. Online Experiments

In this kind of experiments, the experiments are done with real users while system is running commercially. However, this brings the risk of system crash because of the experimental setups [51]. Online experiments are regarded as more accurate since they show the behavior of the recommender system in reality.

CHAPTER 3

A GRAPH BASED COLLABORATIVE AND CONTEXT AWARE RECOMMENDATION SYSTEM FOR TV PROGRAMS

This chapter presents the conceptual description of a graph-based collaborative context-aware recommendation system that is designed to operate on real usage data provided by Arçelik. First, the system architecture is described from modular perspective and brief information about the modules of system is given. Before elaborating the proposed system from algorithmic point of view, we give background knowledge about the design issues and decisions we take in the scope of this work.

3.1. General System Overview

The graph-based collaborative recommender system that is developed in this thesis is an application which aims to select the preference of the user according to given contextual variables.

The developed recommendation system is aimed to be embedded in the Arçelik Connected TVs. The TV usage data used in this thesis is provided by Arçelik.

Recommendations are done through the collaborative filtering approach according to the random walks that are performed on a contextually tailored tri-partite *<User, Program, Term>* graph. Results are re-ordered according to context variables in order to get the final recommendation list.

3.2. System Architecture

The proposed system compromises two different data sources which are Arçelik TV usage data that are gathered from Arçelik TVs and Radikal TV Guide data. The system aggregates these data in order to create a dataset to be used by the recommendation algorithm. Three separate modules deal with graph construction according to the contextual attributes, recommendation and post evaluation of recommendations according to context, respectively. General structure of this system is shown in Figure 3-1.

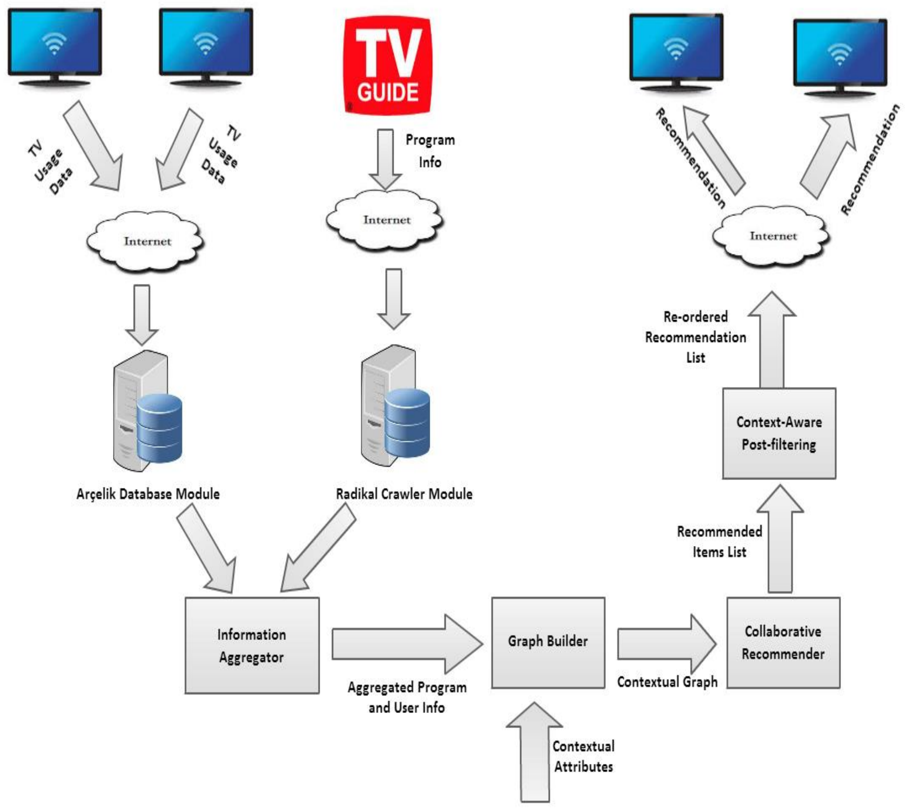


Figure 3-1: General Structure of Proposed System

Our proposed system consists of seven components that are responsible from different tasks. These are: Arçelik TV reporter module, Arçelik database module, Radikal TV guide crawler module, information aggregator module, context aware graph builder module, collaborative recommender module, context aware post-filtering module.

3.2.1. Arçelik TV Reporter Module

Arçelik TV reporter module is an embedded module that resides on every connected TV that Arçelik manufactures. This module is responsible of recording channel usage behavior of TVs and sending them to the Arçelik database module when it has an internet connection. Arçelik TVs are programmed to send the TV usage data in every 5 minutes. If there is no internet connection, this module collects the TV usage information on their memory and sends the collected TV usage data as the internet connection is provided.

3.2.2. Arçelik Database Module

This module is responsible of keeping the TV usage information and other essential information in an organized manner. This module is also responsible of taking the backup of Arçelik data monthly. This module returns adequate information when a query is posed to it.

3.2.3. Radikal TV Guide Crawler Module

In our implementation, we need the structured information of TV programs which is not provided by Arçelik data. In order to gather this data, we have examined several TV guide websites such as Digiturk TV Guide[53], Teledünya TV Guide[54]and Radikal TV Guide[55], and we preferred to use Radikal TV Guide since it provides more structured and more accurate information compared to other sources. This module is programmed to crawl the website of Radikal TV guide in a daily manner at midnight. In the scope of this thesis, this module have collected the TV guide information for 3 months (between October 2013 and January 2014) in order to create the dataset for our research.

3.2.4. Information Aggregator Module

In our research, Arçelik data does not include the program information but contains only the channel name and start time and end time of the TV usage. In order to find out which program is watched at that time we have used EPG information from Radikal TV Guide. This module is responsible of combining these two data sources and transforming the channel usage data taken from Arçelik to the program usage data matched with users according to the channel name, start time and end time of the usage.

3.2.5. Context Aware Graph Builder Module

In our implementation we have developed a context-aware graph based recommendation algorithm which uses the graph tailored according to the given context as input. This module is responsible of constructing the graph by using the data that is aggregated by the information aggregator module. While achieving this task, this module takes the contextual variable as input and filters the irrelevant data and constructs the graph only with the data that is in the context. For example; if the given context is in the form of *Time of Day = PRIME_TIME* and *Genre = TV Series*, this module takes the TV series that are broadcast on prime time and their usage information while constructing the graph.

3.2.6. Collaborative Recommender Module

This module is the core module that carries out the recommendation task by taking random walks on the contextually adjusted graph. It gets the contextual graph, length of the path for the random walk, number of users to be determined as similar, number of recommendations to be produced as input and produces the output as a ranked list of recommendations which are subject to change by context aware post-filtering (see in Figure 3-2). For making recommendations, this module does finite length traversals on the graph in order to find similar users. Then, it produces recommendation results again by taking random walks that use those similar users as a start point. In order to find the best path length, the number of users, and the length of the recommendation list, this module has been re-run many times by setting the contextual variables for every possible value.

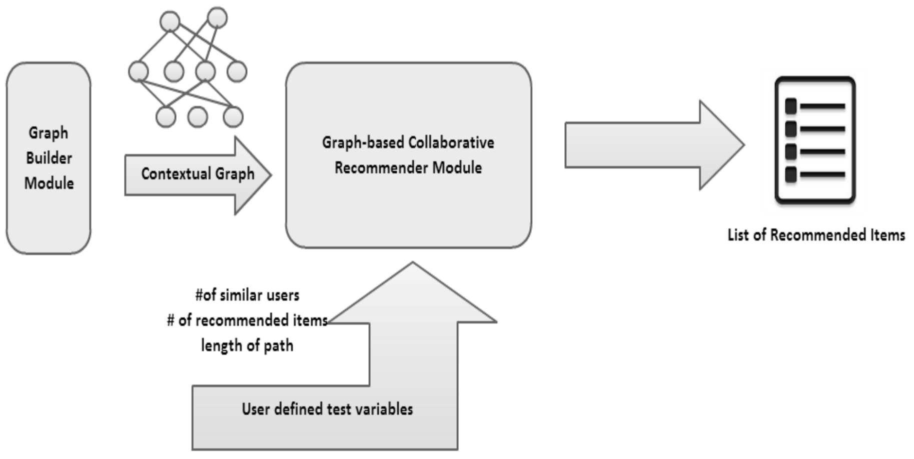

Graph Builder Module
Contextual Graph
Graph-based Collaborative Recommender Module
List of Recommended Items
#of similar users
# of recommended items
length of path
User defined test variables

Figure 3-2: Input Output Flow of the Recommender Module

3.2.7. Context-Aware Post-filtering Module

In our implementation, the recommendation list that is produced by the recommender module is re-evaluated according to the context variables. This module is responsible of re-evaluating the recommended item list according to the preference of the context variables. This module uses different context variables for post-filtering from the context variables used in the pre-filtering module. To be specific, this module makes use of the preference of target user on actors and directors, while pre-filtering uses more general contextual variables such as genre of the program and time of day of the program. This module puts the items in the input recommendation list in an order according to the contextual preferences of the user and produces a new re-ranked recommendation list.

3.3. Design Issues

Throughout this thesis, some design decisions have been taken by discussing several technologies and approaches. In order to achieve the task of this thesis, several tools and programming languages and already developed libraries are exploited. External data sources are selected with respect to some attributes. In this section, we will elaborate the design decisions that are taken in the scope of this project.

3.3.1. Database

In the scope of this thesis, we have kept our dataset in a structured way in a database. For this need, we have used MySQL[56]which is the second widely used relational database management system. In order to map our database tables to the programmatically meaningful classes we have used the Active Record[57] modeling technology that is developed in the scope of a well-known web development framework Ruby on Rails[58].

3.3.2. Arçelik TV Usage Dataset

In the scope of this thesis, Arçelik provided access to their channel usage data for 3 months (October, November and December 2013) in order to be used by our research. Their channel usage data contains only the device id as the user information, which should be joined with *customer_devices* table and this table is not available to us. From this point of view, the channel usage data is automatically anonymized with the cost of regarding each device that is possibly used by a group of users as one user in our system. Channel usage data taken from Arçelik only includes channel name, start time of TV usage and end time of TV usage which are useful for our research. Apart from those, it includes some other attributes such as signal frequency, source type and channel type. The structure of the channel usage model with useful attributes is presented in Table 3-1.

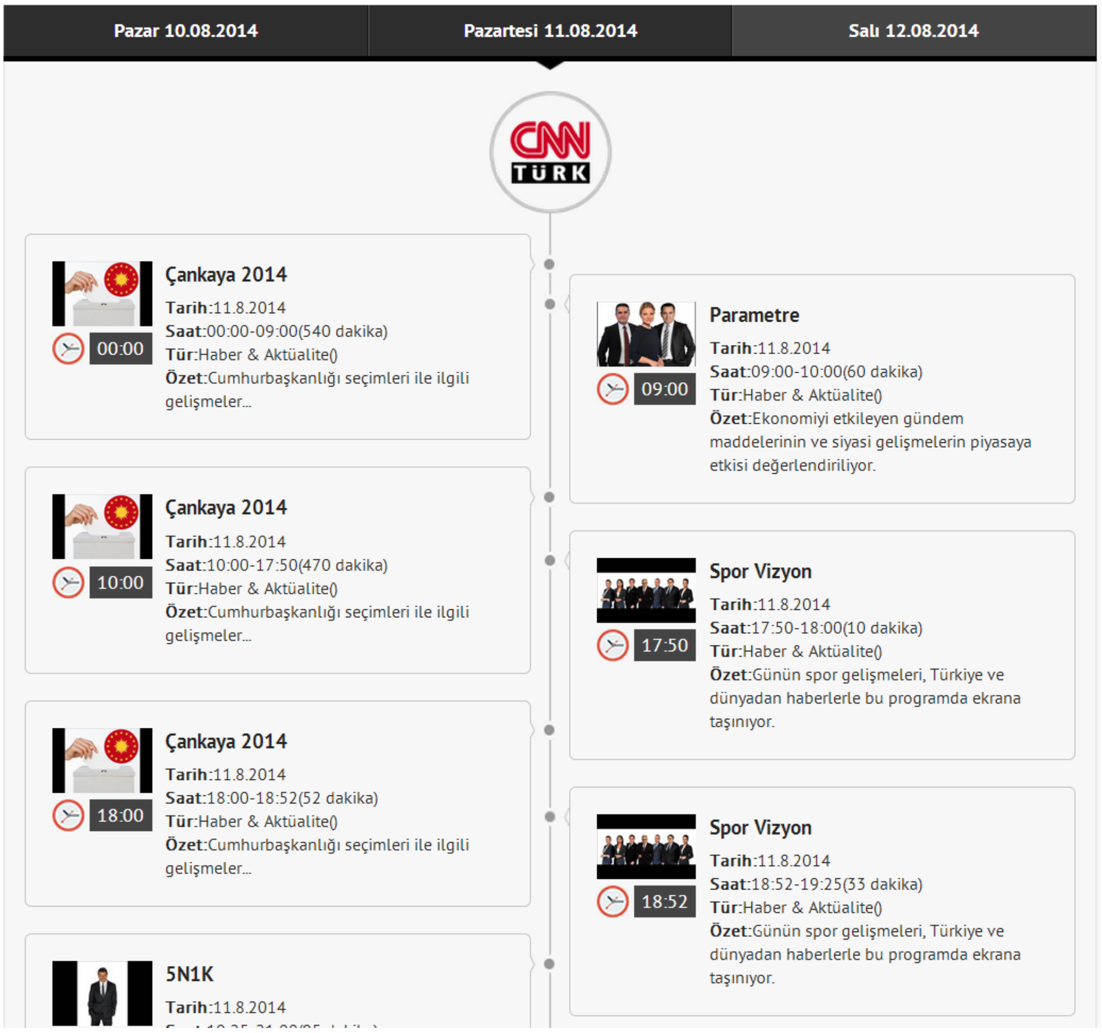
The channel usage data includes the TV usage records that are longer than ten seconds. In the dataset retrieved for 3 months, there are 3,865,821 channel usage records which belong to 5,466 users.

Table 3-1: Table Structure of Channel Usage Object

Attributes	Summary
id	Id attribute is used to ensure uniqueness of channel usage object.
device_id	Id of the device that this channel usage belongs to.
start_time	Start time of the channel usage in the form of UNIX timestamp.
end_time	End time of the channel usage in the form of UNIX timestamp.
name	Name of the channel that channel usage belongs to.
frequency	Signal frequency of the channel on satellite
source_type	Source used when the channel usage occurs. For example, terrestrial or satellite
channel_type	Type of the channel used such as HD or FullHD.

3.3.3. Radikal TV Guide Dataset

TV channels periodically update their schedule of broadcasts and provide information about their programs on their websites or by using some other sources. Some general TV guide applications gather this program information from the channels and present them in a structured form named EPG (Electronic Program Guide). As we mentioned, we have retrieved the EPG information from the website of Radikal which was a former newspaper that has switched to only online broadcasting recently. The TV guide resided in Radikal web site provides EPG of thirty nine TV channels broadcasting in Turkey for every three days. It presents the broadcast stream of each channel in separate websites. For example in order to get the broadcast stream for channel CNN_TURK on 11/08/2014, we should parse the website that is presented in *http://www.radikal.com.tr/tvrehberi/cnn_turk/#!11.08.2014*. In order to get the broadcast stream of all channels on Radikal TV Guide, we have put all channel names into a configuration array, and iterated over this array in order to crawl the website for EPG information for all channels. The example broadcast stream can be seen in Figure 3-3.



Pazar 10.08.2014 | Pazartesi 11.08.2014 | Salı 12.08.2014

CNN TÜRK

Çankaya 2014
00:00
Tarih:11.8.2014
Saat:00:00-09:00(540 dakika)
Tür:Haber & Aktüalite()
Özet:Cumhurbaşkanlığı seçimleri ile ilgili gelişmeler...

Parametre
09:00
Tarih:11.8.2014
Saat:09:00-10:00(60 dakika)
Tür:Haber & Aktüalite()
Özet:Ekonomiyi etkileyen gündem maddelerinin ve siyasi gelişmelerin piyasaya etkisi değerlendiriliyor.

Çankaya 2014
10:00
Tarih:11.8.2014
Saat:10:00-17:50(470 dakika)
Tür:Haber & Aktüalite()
Özet:Cumhurbaşkanlığı seçimleri ile ilgili gelişmeler...

Spor Vizyon
17:50
Tarih:11.8.2014
Saat:17:50-18:00(10 dakika)
Tür:Haber & Aktüalite()
Özet:Günün spor gelişmeleri, Türkiye ve dünyadan haberlerle bu programda ekrana taşınıyor.

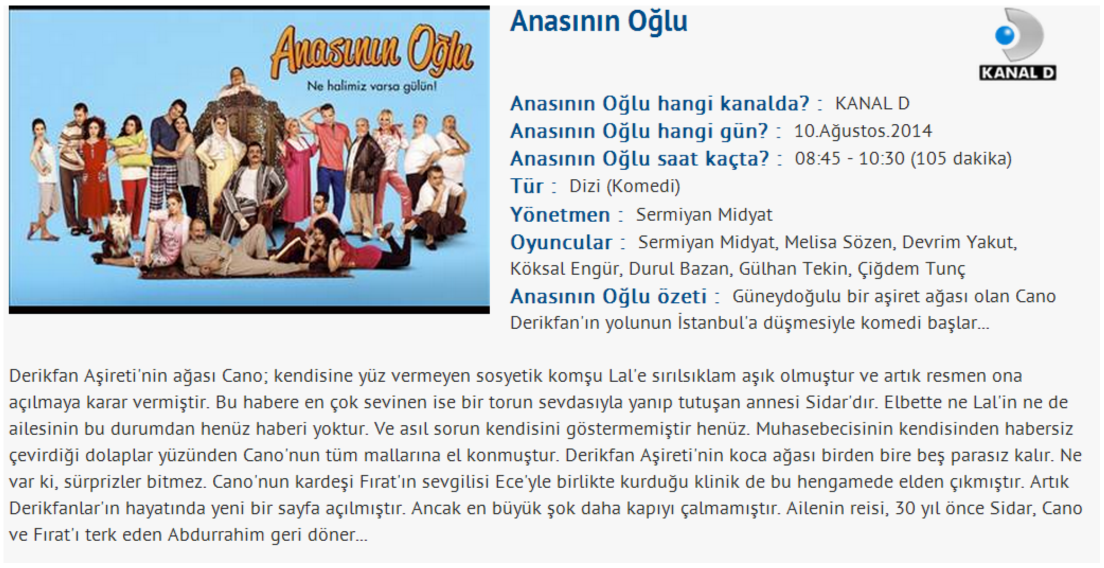
Çankaya 2014
18:00
Tarih:11.8.2014
Saat:18:00-18:52(52 dakika)
Tür:Haber & Aktüalite()
Özet:Cumhurbaşkanlığı seçimleri ile ilgili gelişmeler...

Spor Vizyon
18:52
Tarih:11.8.2014
Saat:18:52-19:25(33 dakika)
Tür:Haber & Aktüalite()
Özet:Günün spor gelişmeleri, Türkiye ve dünyadan haberlerle bu programda ekrana taşınıyor.

5N1K
Tarih:11.8.2014

Figure 3-3: Broadcast stream of CNN TURK on 11/08/2014

While parsing the broadcast stream, we take the program id in the broadcast stream in order to retrieve the details of the program by combining the program id with the program name. For example; the details of the program "Anasının Oğlu" is found at the URL*http://www.radikal.com.tr/tvrehberi/kanald/anasinin_oglu/502301/*which is constructed by combining the channel name, program name and program id. As it can be seen in Figure 3-4, Radikal TV guide provides sufficient attributes for a program which are channel name, day and time of program, genre, director, cast information, summary and long description of the program.



Anasının Oğlu

Anasının Oğlu hangi kanalda? : KANAL D
Anasının Oğlu hangi gün? : 10.Ağustos.2014
Anasının Oğlu saat kaçta? : 08:45 - 10:30 (105 dakika)
Tür : Dizi (Komedi)
Yönetmen : Sermiyan Midyat
Oyuncular : Sermiyan Midyat, Melisa Sözen, Devrim Yakut, Köksal Engür, Durul Bazan, Gülhan Tekin, Çiğdem Tunç
Anasının Oğlu özeti : Güneydoğulu bir aşiret ağası olan Cano Derikfan'ın yolunun İstanbul'a düşmesiyle komedi başlar...

Derikfan Aşireti'nin ağası Cano; kendisine yüz vermeyen sosyetik komşu Lal'e sırılsıklam aşık olmuştur ve artık resmen ona açılmaya karar vermiştir. Bu habere en çok sevinen ise bir torun sevdasıyla yanıp tutuşan annesi Sidar'dır. Elbette ne Lal'in ne de ailesinin bu durumdan henüz haberi yoktur. Ve asıl sorun kendisini göstermemiştir henüz. Muhasebecisinin kendisinden habersiz çevirdiği dolaplar yüzünden Cano'nun tüm mallarına el konmuştur. Derikfan Aşireti'nin koca ağası birden bire beş parasız kalır. Ne var ki, sürprizler bitmez. Cano'nun kardeşi Fırat'ın sevgilisi Ece'yle birlikte kurduğu klinik de bu hengamede elden çıkmıştır. Artık Derikfanlar'ın hayatında yeni bir sayfa açılmıştır. Ancak en büyük şok daha kapıyı çalmamıştır. Ailenin reisi, 30 yıl önce Sidar, Cano ve Fırat'ı terk eden Abdurrahim geri döner...

Figure 3-4: Details of a Program Description on Radikal TV Guide

In order to parse the program details and extract the necessary information, we use the jsoup[59] Java library. This library enables programmers to crawl on the retrieved HTML page by using the DOM traversals and CSS selectors.

Before inserting the program information to the database, we make some pre-processing on the retrieved program data. In order to make the program data compatible with ORM, the following operations are done:

- Using the human readable date and time of the program, we have calculated the UNIX timestamp of the program which is in terms of seconds. This operation is done in order to make the start and end time of the program compatible with Arçelik channel usage data which is in terms of UNIX timestamp.
- By using the time of the program, we have classified the program into one of the classes that are composed with respect to time of day. While deciding for intervals of time of day slots, we have used the dayparting article on Wikipedia[60] and merged some of the day parts which are too short for our purposes. The resulting day partitions can be seen in Table 3-2. We have

permitted multiple times of day for programs that are broadcast in more than one time of day.

- By splitting the genre information with the character “(“, we have revealed hidden genres in the parenthesis which are actually quite important. In the program in Figure 3-4, *comedy* genre is more decisive compared to *series* genre. With the help of this information, we permit multiple genres for programs.
- By splitting the actors and directors with the character “,”, we represent multiple actors and directors that can belong to a program.
- In order to represent the programs as BOW, we have stemmed all words that are in the program description and summary and got the stemmed words which are called as terms in our work. After stemming all words in the program description and summary, we have excluded the verbs in order to avoid the ambiguity problem of verb stems in Turkish. In order to achieve the stemming task, we have exploited Zemberek[61] which is an open-source natural language processing framework developed for Turkish language mostly. Beyond stemming, Zemberek is capable of many tasks such as spell checking, morphological parsing and word construction. It is also used in real world applications such as OpenOffice.org.

Table 3-2: Day partitions

Time Slot	Time Of Day
00:00-04:00	NIGHT
04:00-07:00	EARLY MORNING
07:00-09:00	BREAKFEAST
09:00-13:00	LATE MORNING
13:00-18:00	DAYTIME
18:00-20:30	EVENING
20:30-24:00	PRIME TIME

While inserting the program data, we have pursued a structured representation of the data by taking the opportunity of relational database management system. Instead of creating a programs table which includes all necessary information tucked-in its columns, we have created separate tables for every logical model in the program data. For example, we have created a separate *actors* table that keeps all actors that are encountered in one table and a *programs_actors* table in order to keep the relation between programs and actors. The structured tables that are used to store our models can be seen in Table 3-3, and the structured junction tables that are used to store the relations between our models can be seen in Table 3-4.

Table 3-3: Storage tables to keep the record of our models.

Table Name	Fields in Table	Summary
programs	id, channel_id, start_time, end_time, name, dataset	This table is used to store the programs object in our database. id field is used to keep the uniqueness of object. channel_id points to the channel that the program is broadcast in. start_time and end_time is used to present the broadcast time of a program. dataset is a boolean value that indicates that the program is in the dataset when it is set to 1.
terms	id, name, idf	This table is used to store all terms (stemmed nouns in program descriptions and program summary) that are in our dataset. idf value is the number of documents that this term takes place.
actors	id, name	This table is used to keep the record of actors in our dataset. The tables directors, genres and time_of_days are in the same form.
channels	id, name, icon	This table is used to keep the records of channels in our dataset.

Table 3-4: Junction tables for representing relationships between models

Table Name	Fields in Table	Summary
programs_actors	program_id, actor_id	This table is a junction table that connects programs and actors. There is a many-to-many relationship between programs and actors. In other words, a program may have many actors and an actor can take part in several programs. Junction tables with directors, genres and time_of_days are in the same form with this table.
program_terms	program_id, term_id, term_frequency, tf_idf, normalized_tf_idf	This table is a junction table between programs and terms (stemmed words in program description). Since we have a relationship weight between program and term there are metrics called term_frequency which represents the frequency of a term in a program, tf_idf which represents tf_idf value of term in a program and normalized_tf_idf which is the normalized form of tf_idf value according to the maximum tf_idf value in dataset.
term_connections	term_a_id, term_b_id, co_occurence	This table is a junction table between each term in dataset. This table is aimed to keep the number of documents that term a and term b are in a document together, in order to create a weighted relation between terms.

In the scope of this thesis, we have retrieved the TV guide data corresponding to the data which is supplied by Arçelik. Therefore, we have retrieved the EPG information for October, November and December 2013 for 39 channels. In this data, there are

- 45107 different programs
- 42 different genres
- 4338 actors
- 1092 directors
- 4697 terms.

3.3.4. Matching Channel Usages with TV Guide Information

As we stated, we have dealt with the channel usage data that is retrieved from Arçelik, Beko and Grundig TVs between October 2013 and January 2014 which contains 3,865,821 records. Although, channel usage data shows the information about the watched channel for a certain period of time, it does not provide any information about which programs are watched in that time period. In order to find the programs that are watched during the channel usage, we match the channel usages by querying the programs with channel name, start time and end time. However, name of the channel might be recorded differently for the same channel by different devices. For example, the channel names EUROSPORT 2, EUROSPORT2 and EUROSPORT 2 HD which are all present in our channel usage data correspond to one channel named eurosport2 in the program information on the website. In order to match the channel usages with programs correctly, we have examined all different channel names in the channel usage data and created a look-up table that keeps the corresponding channel names in the channel usage data for the channel names in program data. We have found out 63 different channel names in channel usage data, for 26 channels in our program data. A small fraction of the look-up table for some channels can be seen in Table 3-5. For 13 channels in the program dataset, there is no corresponding channel usage. The reason is that those channels are not recorded by their name because they are specific to a Turkish satellite provider.

Table 3-5: Channel Name Matching between Program and Channel Usage Data

Name in Program Data	Name in Channel Usage Data
trt1	TRT 1
	TRT 1 HD
	TRT1 HD
	TRT1
	TRT-1 HD
cnbc_e	CNBC-e
	CNBCE
	CNBC e
kanald	Kanald
	KANAL D HD
	KANAL D

As a result of matching of channel usages with programs, we have matched 1,171,533 i.e., one third of all channel usages, channel usage data with 41,357 programs in programs data which are watched by 5466 users. The average number of programs watched by a user is 307.

While matching channel usages with programs, users are also stored in our database in a structured way. In addition, the need for junction tables for keeping relationships between users and programs and between users and programs-related tables such as *terms* arises. The structure of *users* table and conjunction tables can be seen in Table 3-6.

.

Table 3-6: Structures of Tables Related with User

Table Name	Fields in Table	Summary
users	id, user_id, dataset	This table is used to store the users that are in our dataset. The id column grants the uniqueness of our user. The user_id is the id of the user in Arçelik dataset. The dataset is a boolean value that indicates that the user is in the dataset when it is set to 1.
program_users	program_id, user_id, watch_time, rating	This table is a junction table between programs and terms (stemmed words in program description). Since we have a relationship weight between program and user there are metrics called watch_time which represents the watched time of a program by user in terms of seconds, rating which is a calculated value about the preference of the user on the program by using watch time of user and duration of the program.
users_terms	term_id, user_id, rating	This table is a junction table between each user and term in dataset. Rating value is calculated by using the ratings that user gives the programs in which term takes place. The junction tables with other programs related models users_actors and users_directors are in the same form with this table. The ratings in those tables are calculated similarly.

3.4. Context-Aware Pre-Filtering

As we stated, every program in our dataset has some contextual features such as genre, channel, and time of day. There are even more specific context variables like actors and directors which are special to only specific genres like movie and TV series. In our implementation, we use context aware pre-filtering in order to shrink the set of candidate programs. In order to select the contextual attributes to use for context aware pre-filtering, we have analyzed the contextual variables according to the size of the filtered dataset. We have concluded that genre and time of day is suitable for context aware pre-filtering. In the scope of pre-filtering, we have filtered out the programs which do not have the selected context variable as an attribute. For example, we have 6500 programs for the time of day "PRIME_TIME", within the dataset of 41357 programs. We have done our experiments by filtering our data with 42 different genres and 7 different times of day both separately and in conjunction with each other. Experimental results are shared in Chapter 4.

3.5. Graph Construction

In our implementation, we have modified the approach defined by Bogers[6]by pruning the graph according to contextual attributes instead of putting the contextual attributes into the graph as different node types. In our work, we have constructed a tri-partite graph which includes three node types, namely User, Program and Term. We denote users by U = {User_1, User_2, User_3, User_4... $\text{User}_{|U|}$}, programs by P = {Program_1, Program_2, Program_3, Program_4... $\text{Program}_{|P|}$} and terms by T = {Term_1, Term_2, Term_3, Term_4... $\text{Term}_{|T|}$}. The set T is used to represent the set of stemmed nouns used in the description of a TV program. There is a weighted edge between every node type in the graph. The weights are determined by using similarity functions between our node types. In order to represent the similarities between node types we have used matrices whose cells contain the similarity value between each node. For example, the matrix UP= (up_{ij}) with size of |U| x |P| is composed in order to keep user ratings over programs. The detailed information about similarity functions is given in sub-sections of this section.

3.5.1. Similarity Functions

In our implementation, we have defined similarity functions between node types in our tri-partite graph. As a result, we have composed matrices that keep the relationships between our node types.

3.5.1.1. User-User Relation

In order to define similarity among users, we need to have former information about the relationship between users. In our implementation, we used an identity matrix UU= (uu_{ij}) to represent user similarities since we do not have former information about user similarities.

3.5.1.2. User-Program Relation

In our implementation, we define a similarity metric called rating between users and programs, which is in the range [0, 1]. The rating metric is calculated by using the watch time of the users with following formula:

$$rating = \frac{\text{\# of minutes of watch time}}{\text{\# of minutes of program}}$$

With this calculation, the rating that the user gives a program is subject to not only the watch time of the user but also to the duration of the program. This approach is aimed to eliminate bias towards shorter watch times which can be indeed important for a short program. When we calculate the rating for all <*User,Program*>tuples, the average rating is calculated as 0.62.

After calculating the rating between users and programs, we construct a matrix UP= (up_{ij}) with size of |U| x |P| in order to keep user ratings over programs.

3.5.1.3. Program-Program Relation

In the scope of our implementation, we do not define any prior similarity between programs in our dataset. Therefore we use an identity matrix PP = (pp_{ij}) for representing program similarities.

3.5.1.4. Program-Term Relation

In this thesis, the edge weight between program and term nodes is defined by a well-known information retrieval concept named TF-IDF. TF-IDF is calculated as a combination of two notions, which are term frequency (TF) and inverse document frequency (IDF). TF is the number of occurrences of a term for its basic form in a document which is the program description for our implementation. The notion inverse document frequency is a measure of how much information the word provides about the document in which it takes place. In the formula of IDF, the number of all documents is divided by the number of documents containing the term t. Then, IDF value is calculated as the logarithm of the quotient. TF-IDF is calculated as the product of TF and IDF. Program-term relation is represented by the matrix PT= (pt_{ij}) with the size of |P| x |T|, where each cell pt_{ij} takes the TF-IDF value between program p_i and term t_j. We have normalized all TF-IDF values to the range [0, 1].

3.5.1.5. Term-Term Relation

In our dataset, the average number of terms in a program description is calculated as 12.7. We exploit this situation for defining co-occurrence similarity between terms. Co-occurrence metric is calculated as follows:

$$tt_{ab} = \frac{\#\text{number of documents a and b together}}{\#\text{number of all documents}}$$

Term-term relation is represented by matrix TT= (tt_{ij}) with the size of |T x |T|, where each cell tt_{ij} takes the co-occurrence value between terms t_i and t_j.

3.5.1.6. User-Term Relation

In our implementation, there is no direct relationship between users and terms. For this reason, we have defined a metric between users and terms which is calculated by the summation of all multiplications of program rating of user and TF-IDF of the term. For example, the user-term relationship between $User_2$ and $Term_4$ in Figure 3-5 is calculated as:

$$ut_{24} = up_{22} \times pt_{24} + up_{23} \times pt_{34}$$

User-term relation is represented by the matrix UT= (ut_{ij}) with the size of |U| x |T|, where each cell ut_{ij} takes the calculated similarity value between user u_i and term t_j.

3.5.2. Graph Structure

In our work, we define a three layered graph whose layers are *User*, *Program* and *Term*. An example graph that is constructed with the help of the similarity metrics defined between and in the layers can be seen in Figure 3-5.

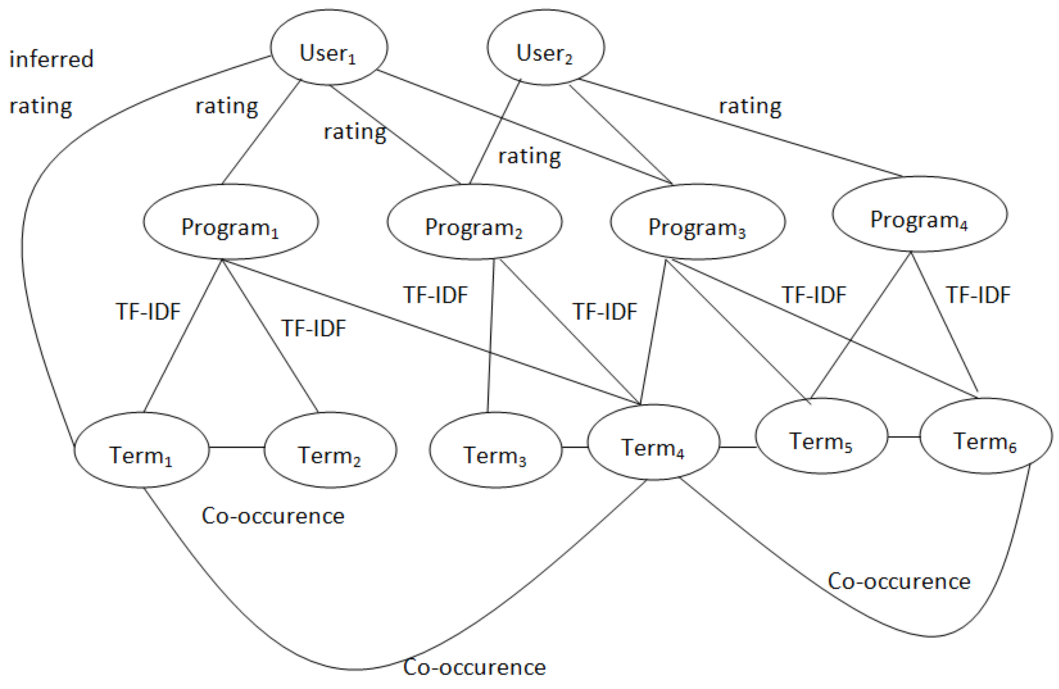


Figure 3-5: An Example Tri-partite Graph

In our implementation, we have used a transition probability matrix X that is composed of sub-matrices which contain similarity values between different node types. The transition probability matrix X with size (|U| + |P| +|T|) x (|U| + |P| +|T|) is constructed as follows:

$$X = \begin{bmatrix} U \times U & U \times P & U \times T \\ P \times U & P \times P & P \times T \\ T \times U & T \times P & T \times T \end{bmatrix}$$

3.6. Recommender Algorithm via Random Walk

We use k-nearest neighbors algorithm [62]by means of similar users for collaborative filtering. In order to find the k-nearest neighbors, we exploit the random walk algorithm as in Bogers' work [6]. In order to begin the random walk over our tri-partite graph, we need to define the initial state vector s_0 in which only the value for the initial user node is 1 and all the values are set to zero. We can find the state probabilities at the next step by multiplying the vector s_0 with the matrix X. In general, we can calculate the transition probabilities after *n* steps using the following formula:

$$s_{n+1} = s_n X$$

After making *n* steps of random walk, the transition probabilities to jump on another user are sorted in order to grab the first *k* users as similar to the target user. In the scope of our implementation, we have tested the value of *k*in the range of $[\sqrt{numberofusers} - \sqrt[4]{numberofusers}, \sqrt{numberofusers} + \sqrt[4]{numberofusers}]$ by using the rule of thumb mentioned in [63].

Because of performance issues, we had to restrict the length of the random walks on our contextual graph. Therefore, we made random walks with finite length whose path length varies from 1 to 6 for finding similar users.

After finding the *k* nearest neighbors, we have used the same technique for finding top-N program recommendations for each neighbor with a path length of four. After getting top-N recommendations for each similar user, we sum up all weights that belong to a program from *k* users in order to find the final weight of it. As a result, we have ended up with top-N recommendation list that is subject to re-rank in the context aware post-filtering step. In our experiments, we have tested the effect of the length of recommendation list with 10, 20 and 50 recommendations in it.

3.7. Context Aware Post-Filtering

Context aware post-filtering is used to re-rank the recommendation list that is constructed by the graph based collaborative algorithm. In our work, we have used actors and directors for context aware post-filtering, which are valid for only some of the program genres. Therefore, we have examined the effect of post-filtering only for several genres.

We have used the ratings that users give to programs in which actors and directors take place in order to infer the ratings of users on actors and directors. For example, if Robin Williams takes part in 4 programs that the user has watched, the ratings of those 4 programs are summed up for calculating the rating given to Robin Williams by the user.

In order to do the context aware re-rank, the score of the program calculated by the recommender algorithm is multiplied by the ratings given by the target user to the contextual attributes that take place in the program. After updating the scores by using contextual attributes, we sort the scores of programs in the recommended items list in order to construct new top-N recommendation list. With the help of context aware re-rank, a program which is not placed in the top-10 recommendation by the recommender algorithm can be put in the list which can result with a more successful recommender algorithm. For this reason, we believe that context aware post-filtering is particularly helpful when the recommendation list should be restricted to a small number of items.

CHAPTER 4

EXPERIMENTS AND RESULTS

In this chapter, experiments that were carried out for the evaluation of our recommender system are presented. First, the pre-processing of dataset is explained. Then, evaluation metrics that are used to measure the success of the recommender system are described. After defining the parameters of our recommender system, we discuss the effects of them to the success of the recommender. Finally we describe the experiments that are performed to measure the success of our algorithm, and discuss their results.

4.1. Data Preprocessing

Because of performance issues, we had to shrink our dataset. For this reason, we have excluded some of the users from our dataset by taking some statistical values into consideration.

The average number of programs watched by a user in our dataset is 307. However, nearly half of our users watch less than 100 programs and one third of the users watch less than the average. Nearly 350 users watch more than 1500 programs in our dataset which is considered as outliers. Therefore, we have selected 1081 candidate users whose number of watched programs is between 300 and 1500.

Average rating for a program is 0.62 in our dataset. Similar to the research conducted by Bambini et al.[42], we assume that a rating given above average should be considered as positive, and we have picked the users whose average rating is bigger than the average rating among 1081 users. Finally, we have come up with 198 users whose ratings are used in our experiments.

4.2. Evaluation Metrics

In the evaluation of recommender systems, different metrics are used according to the motivation of the recommender systems. Researchers can focus on time and space efficiency of the recommender system or the efficiency of recommender system or the satisfaction of user with the help of the recommender system [64]. In this thesis, we have focused on measuring the effectiveness of our recommender algorithm.

For the measurement of the effectiveness of a recommender algorithm, precision and recall are the most common metrics. Since precision and recall are set-based metrics, they are particularly useful for the evaluation of recommender systems which usually generate a list of items to be recommended.

Precision is the ratio of the number of relevant items which are recommended by the recommendation system to the total number of recommended items. Recall is the ratio of the number of relevant items which are recommended to the total number of relevant items [65].

Precision can be formulated as follows according to Figure 4-1

$$precision = \frac{Relevant\ and\ Retrieved}{Retrieved + Relevant\ and\ Retrieved}$$

Recall can be formulated as follows according to Figure 4-1

$$recall = \frac{Relevant\ and\ Retrieved}{Relevant + Relevant\ and\ Retrieved}$$

F-measure is the harmonic mean of precision and recall and used to measure the success of the recommender algorithm. It is calculated as follows:

$$f - measure = 2 * \frac{(precision * recall)}{precision + recall}$$

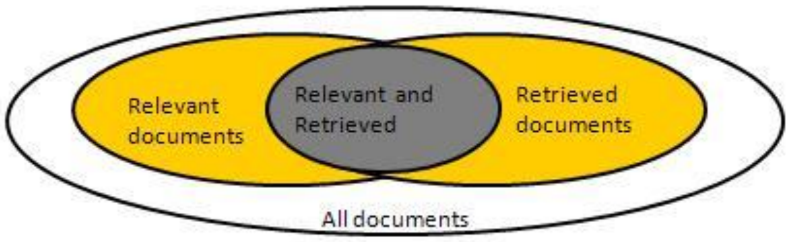


Figure 4-1: Precision and Recall

4.3. Parameters

In the scope of this research, we have used several parameters that can affect the performance of the proposed recommendation algorithm. These parameters are examined in order to find the condition under which our recommendation algorithm performs best. Parameters that we use in our experiments are β, γ and α, and their explanations are given in the following.

4.3.1. Parameter β

Parameter β denotes the number of jumps in the random walk. It represents the length of the path for the random walk. Due to the performance issues, the maximum number of steps in our graph is determined to be6. In our experiments, we have tested the path length from 1 to 6.

4.3.2. Parameter γ

Parameter γ is the number of similar users to be selected for collaborative filtering. In the scope of our research, we have tested γ in the range of;

$$\sqrt{numberofusers} - \sqrt[4]{numberofusers} < \gamma < \sqrt{numberofusers} + \sqrt[4]{numberofusers}$$

Since we selected 198 users after preprocessing of our data, we tested the number of users between 11 and 17.

4.3.3. Parameter α

Parameter α represents the length of the list of items that are recommended by our algorithm. Since we are making top-N recommendations, precision and recall values are affected by the length of the recommendation list. During experiments, optimal value for this parameter is searched. We have tested the values 10, 20 and 50 in our experiments.

4.4. Experiments

This section presents the results of the experiments that are carried out for this thesis. We have tested the effectiveness of our graph based collaborative algorithm both with any possible combination of our contextual attributes and without using any context. For our experiments, we have used 3-fold cross validation technique that segments the dataset into 3 slices and uses one of them as test set and others as training set.

4.4.1. Pure Collaborative Algorithm

In order to show the effect of contextual attributes to the performance of the recommender system, pure collaborative form of our graph based algorithm is considered as a baseline. In the scope of the experiments, we have used all of our parameters for testing the performance of our recommendation method.

As a result of the experiments conducted for pure collaborative graph based algorithm, we have fixed two of our parameters β and γ to the specific values at which our algorithm performs best. Since it takes about 8 hours for one experiment to finish by testing all parameters, we had to keep the number of similar users and path length of random walk fixed. For the experiments that are aimed to fix the β and γ, α is set to 10.

In order to find the best γ, we have checked all γ values from 11 to 17. It can be seen in Figure 4-2 that when we use 12 as γ, we get the best values for our metrics. Therefore, we have selected the optimal k-value as 12 during our experiments with context variables.

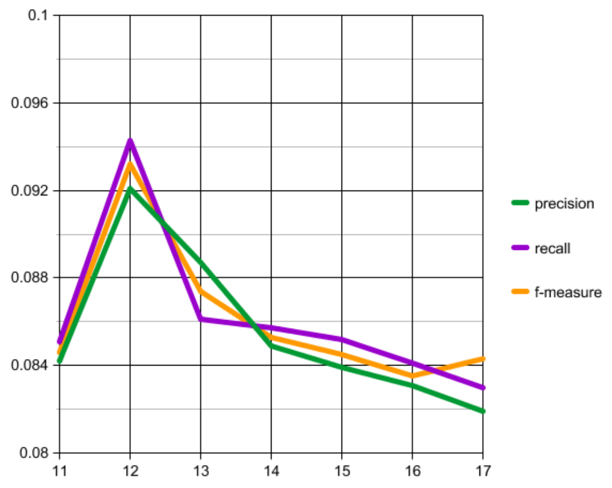


Figure 4-2: Performance of Collaborative Algorithm w.r.t. γ

In our experiments, we have tested the values from 1 to 6 for the length of path in our graph in order to find the best β value. As it can be seen in Figure 4-3, best results are reached with condition β= 6.

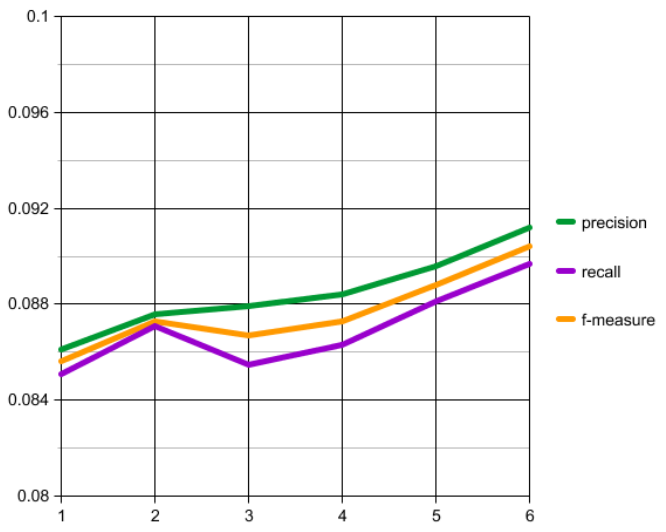


Figure 4-3: Performance of Collaborative Algorithm w.r.t. β

After fixing β and γ, we have tested the effectiveness of our algorithm with α values that is set to 10, 20 and 50. Results are presented in Table 4-1.

Table 4-1: Evaluation Metrics for Pure Collaborative Algorithm

Pure collaborative algorithm			
α value	**10**	**20**	**50**
Precision	0.0889	0.0726	0.0638
Recall	0.0701	0.1287	0.1533
F-Measure	0.0783	0.0928	0.0901

Figure 4-4shows the precision, recall and f-measure values for pure collaborative graph based algorithm.

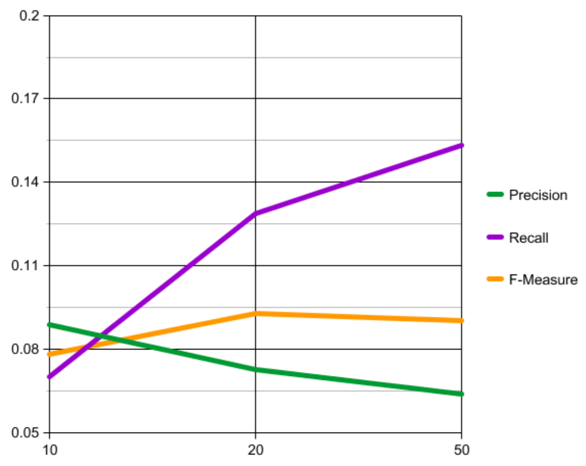


Figure 4-4: Performance of Collaborative Algorithm w.r.t. α

While the set of recommended items grows to a bigger size, precision value worsens because the rate of adding relevant items to the recommendation list is smaller than the rate of change of α, and recall value increases because more relevant items are added to the recommendation list.

4.4.2. Collaborative Algorithm with Context Aware Pre-filtering

In the scope of our experiments, we aim to get better results by filtering irrelevant programs out according to the contextual variables. As we stated, we have used time of day of program and genre of program as context for pre-filtering. We have used those attributes both separately and together in our experiments.

In Figure 4-5, the performance of our graph based collaborative is presented in different time slots with α=10. Our algorithm is most successful in daytime and early morning and least successful in prime time. The reason for this is the sparseness of the data for the time slots. In other words, 198 users, which we have selected after data pre-processing, watch TV at most in daytime and early morning.

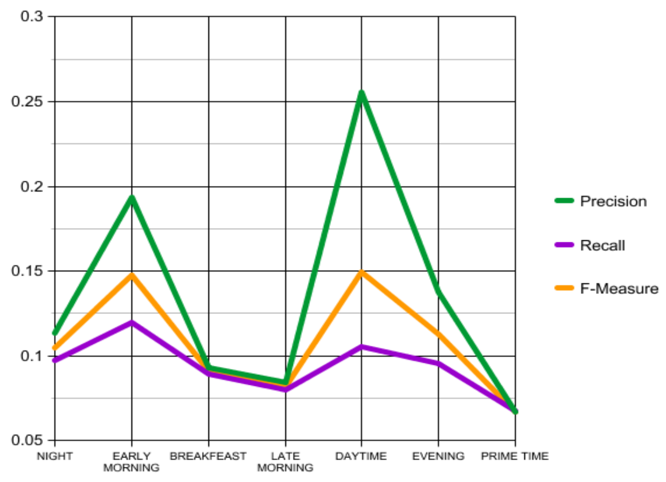


Figure 4-5: Performance of Algorithm with Pre-filtering according to Time Slots

With changing α, we have tested the performance of context aware pre-filtering according to time of day. The average values of evaluation metrics for all time slots are calculated. Results, which are presented in Table 4-2, show similar behavior to the pure collaborative algorithm.

Table 4-2: Performance of Algorithm with Time Aware Filtering

Context aware pre-filtering according to time of day			
α value	**10**	**20**	**50**
Precision	0.1348	0.1097	0.0961
Recall	0.0935	0.0991	0.1747
F-Measure	0.1078	0.1017	0.1225

Figure 4-6 visualizes the precision, recall and f-measure values for context-aware collaborative graph based algorithm when time of day of a program is used as context. With increasing α, recall values increases significantly. This is because of the newly recommended items in the increased length of recommended item list.

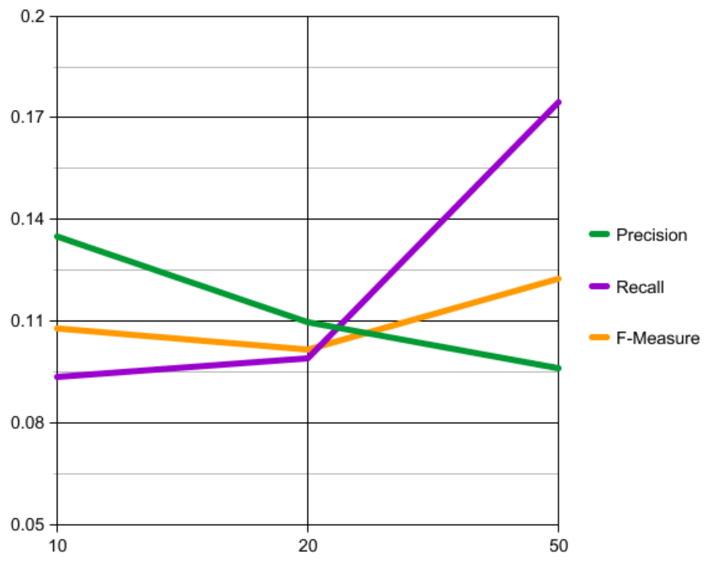


Figure 4-6: Performance of Collaborative Algorithm with Time Aware Pre-filtering w.r.t. α

Figure 4-7demonstratesthe success of our recommender system according to the selected genre when α is set to 10. The collaborative algorithm works best for genres historical, youth, action and detective. The common characteristic of all these genres is that they are all overly specialized genres that contain fewer programs compared to genres like news, series and life. From this point of view, it can be concluded that performance of our algorithm gets better when the candidate set of programs shrinks.

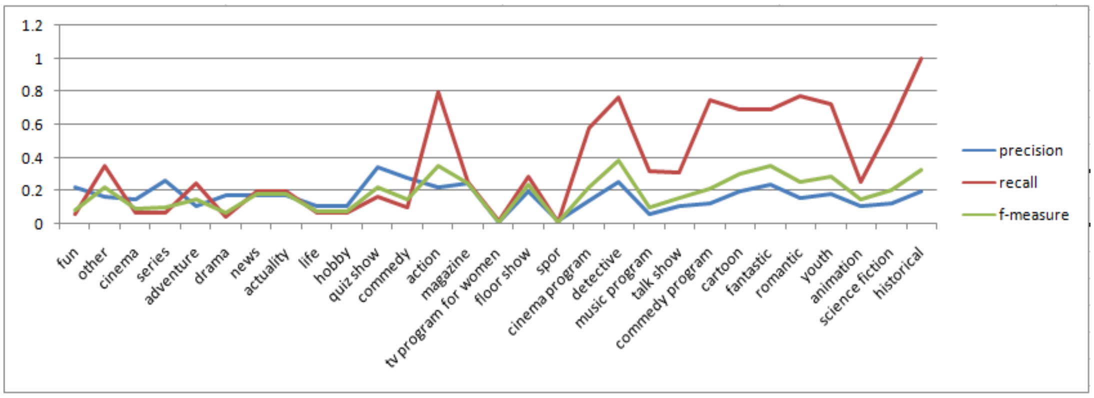


Figure 4-7: Performance of Algorithm with Pre-filtering according to Genres

In Table 4-3, performance of our algorithm on the graph which is filtered according to genre is presented with respect to α by using the average values of metrics in the tests for all genres.

Table 4-3: Performance of Algorithm with Genre Aware Filtering

Context aware pre-filtering according to genre			
α value	**10**	**20**	**50**
Precision	0.1766	0.1432	0.0964
Recall	0.3704	0.5490	0.7033
F-Measure	0.1993	0.1891	0.1481

As it can be inferred from the Table 4-3, recall value of our algorithm increases importantly with increasing α. The reason for this is that the number of relevant items is small for specific genres and, most of the relevant items are recommended when we increase the length of the list of recommended items. In Figure 4-8, the visual representation of this situation is presented.

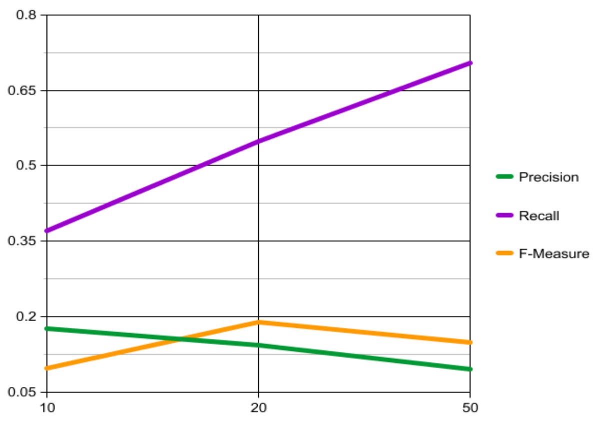


Figure 4-8: Performance of Collaborative Algorithm with Genre Aware Pre-filtering w.r.t. α

We have also tested the effect of contextual pre-filtering by using both contexts together. By setting the time of day and genre at the same time, the contextual graph is restricted to even a smaller size. This situation brings opportunity to our algorithm for generating better recommendations on considerably small transition probability matrix which is less sparse than the graphs which are filtered by using one contextual attribute. In Table 4-4, the average performance of our graph based collaborative algorithm on the graph which is filtered according to both time of day and genre attributes is represented.

Table 4-4: Performance of Algorithm with Genre and Time of Day Aware Filtering

Context aware pre-filtering according to genre and time of day			
α value	**10**	**20**	**50**
Precision	0.2230	0.1628	0.0974
Recall	0.6606	0.7882	0.9570
F-Measure	0.2907	0.2471	0.1688

In our implementation, context aware filtering by using genre and time of day improve the success of our recommender algorithm. Genre is particularly successful compared to time of day, and our algorithm performs even better when two contextual attributes are used in conjunction. In Figure 4-9, the performance of our recommender algorithm with context aware pre-filtering is presented when the contextual attributes are used both separately and together with α=10.

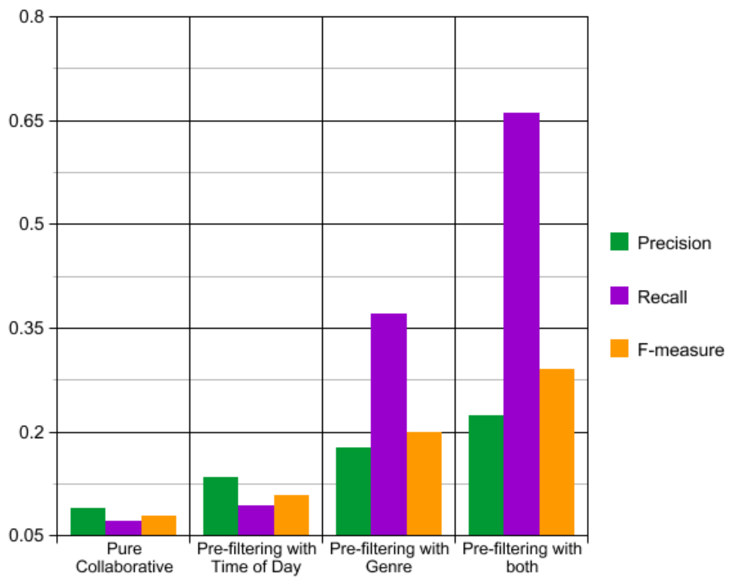


Figure 4-9: Comparison of the Pre-filtering Results w.r.t. Selected Context

4.4.3. Collaborative Algorithm with Context Aware Post-Filtering

In the scope of our experiments, we perform context aware post-filtering on the list of recommended items that is produced by our collaborative algorithm in order to re-rank it according to some contextual attributes. For our experiments we have used the contextual attributes actor and director for post-filtering purposes. We have tested the effect of these contextual variables together since they usually co-exist in a program. The results of experiments are presented in Table 4-5 for different values of α.

Table 4-5: Performance of Algorithm with Context Aware Post-filtering

Context aware post-filtering			
α value	**10**	**20**	**50**
Precision	0.0949	0.0759	0.0688
Recall	0.0716	0.1394	0.1578
F-Measure	0.0812	0.0982	0.1034

Table 4-6 presents the improvement on the performance by adding context aware post-filtering to the pure collaborative algorithm.

Table 4-6: Effect of Post-filtering to Pure Collaborative Algorithm

	Without post-filtering	**With post-filtering**
Precision	0.0889	0.0949
Recall	0.0701	0.0716
F-Measure	0.0783	0.0812

Context aware post-filtering has provided restricted gain of success to the collaborative algorithm. The reason for this situation is that context aware post-filtering does not provide a solution to the data sparseness problem. Instead, it tries to re-rank the list of the recommended item list according to actors and directors that the user has rated via the programs he/she watched. Furthermore, actors and directors are present for only some genres. This situation causes a limited performance improvement when post-filtering is applied.

4.4.4. Collaborative Algorithm with Context Aware Pre-filtering and Post-filtering

In the scope of our experiments, we have applied context aware post-filtering not only to the recommended items generated by pure collaborative algorithm but also to the recommended items which are generated as the output of collaborative algorithm applied on contextually filtered graphs. The aim is to measure the success of the collaborative algorithm when all contextual information is used for recommendation in these experiments.

In the experiments, context aware post-filtering is performed on the recommended items generated by the collaborative algorithm that uses time aware pre-filtering. Figure 4-10 demonstrates the performance of graph based collaborative algorithm enriched with both post-filtering and pre-filtering with respect to time of day in all time slots when α is set to 10.

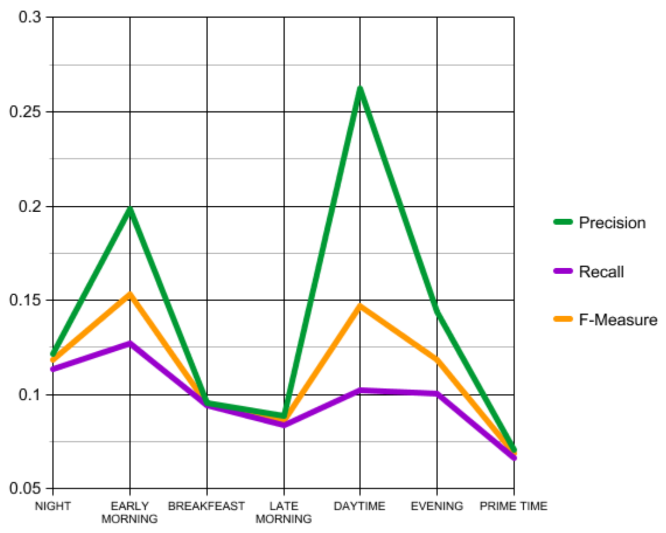


Figure 4-10: Performance of Algorithm with Post-filtering and Pre-filtering according to Time Slots

Similar to the pure collaborative algorithm, context aware post filtering brings limited performance improvement to the time aware pre-filtering. Results are presented in Table 4-7 when α is set to 10.

Table 4-7: Effect of Post-filtering to Algorithm with Context Aware Pre-filtering according to Time of Day

	Without post-filtering	**With post-filtering**
Precision	0.1348	0.1397
Recall	0.0935	0.0982
F-Measure	0.1078	0.1124

In our experiments, we have applied context aware post-filtering to the recommendation list generated by the collaborative algorithm as well. In Figure 4-11, the performance of graph based collaborative algorithm which is enhanced by both post-filtering and pre-filtering with genre is presented with all genres.

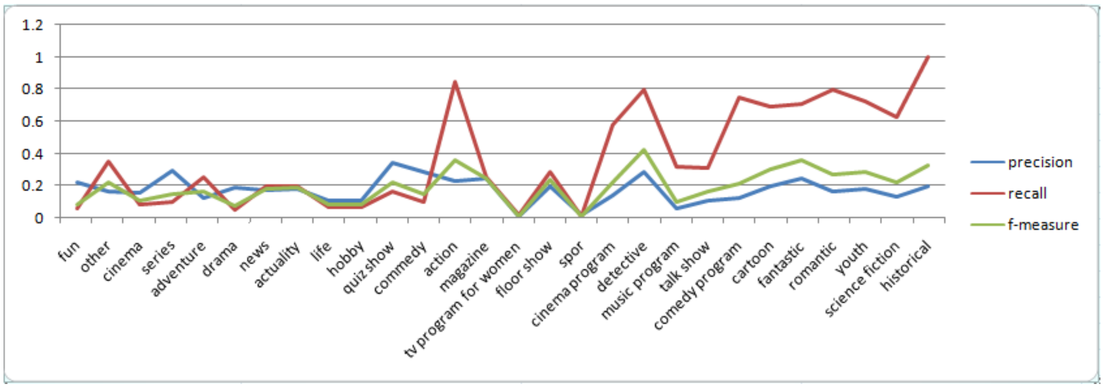


Figure 4-11: Performance of Algorithm with Post-filtering and Pre-filtering according to Genres

Although context aware post-filtering improved the performance of our algorithm for some genres such as action and cinema program, its overall affect to genre aware pre-filtering is limited like in time aware pre-filtering. Results are presented in Table 4-8 when α is set to 10.

Table 4-8: Effect of Post-filtering to Algorithm with Context Aware Pre-filtering according to Genre

	Without post-filtering	With post-filtering
Precision	0.1762	0.1777
Recall	0.3852	0.3864
F-Measure	0.2421	0.2428

In the scope of our experiments, we have tested our graph based collaborative algorithm by using all contextual information for both pre-filtering and post-filtering operations. Similar to the previous tests that uses post-filtering with two pre-filtering techniques separately; the effect of post-filtering is again limited to the collaborative algorithm when it is used with both pre-filtering attributes together. Results are provided in Table 4-9 when α is set to 10.

Table 4-9 : Effect of Post-filtering to Algorithm with Context Aware Pre-filtering according Genre and Time of Day

	Without post-filtering	With post-filtering
Precision	0.2230	0.2249
Recall	0.6660	0.6712
F-Measure	0.2907	0.2951

4.5. Evaluation of Experiment Results

In the scope of our experiments, we have tested the effect of our parameters β (path length of random walk), γ (number of similar users in k-NN) and α (length of recommendation list), and fixed two of them because of performance issues.

As a result of our experiments, we realized that the success of the recommender algorithm increases with the path length for random walk on the graph. As we state, we had to keep the maximum length of path as 6 for our experiments. Therefore, it is concluded that our algorithm might provide better results if it is performed on more powerful systems.

In the scope of our experiments, we have also tested the number of similar users to be picked for k-nearest-neighbor algorithm which highly depends on the dataset used. In our implementation, our algorithm performed best when the number of similar users is set to 12. We think that this value does not provide insight about our data since it can perform better with different γ values for another data pre-processing technique.

With increasing sizes of recommended item list α, it can be seen that precision values are decreasing while recall values are increasing since the algorithm can find more relevant items with bigger values of α. However, the newly discovered relevant items are not sufficient to keep the precision values at the same values.

In order to measure the effect of contextual attributes, we have tested all contextual variables that are used both for pre-filtering and post-filtering. Evaluation metrics for all possible combination of contextual attributes are presented in Table 4-10when α is set to 10.

It is obvious that context aware pre-filtering produces bigger improvement compared to post-filtering, and genre information performs better compared to time of day for pre-filtering. Finally, the best performance of our algorithm is reached when all of the contextual variables are used.

Our algorithm performs better with context aware pre-filtering because a small set of candidate programs causes our random walk to traverse most of the graph with six steps on it. On the other hand, the effect of post-filtering has remained limited to the success of our recommender system. Since our attributes that are used for post-filtering is only valid for specific genres, positive effect of post-filtering does not change the overall success of our recommender algorithm.

Table 4-10: Performance Comparison w.r.t. All Possible Contextual Variables

	No Context	**Post-filtering only**	**Pre-filtering with Time of Day**	**Pre-filtering with Genre**	**Pre-filtering with Time of Day and Genre**	**Pre-filtering with Time of Day + Post-filtering**	**Pre-filtering with Genre + Post-filtering**	**Pre-filtering with Time of Day and Genre + Post-Filtering**
Precision	0.0889	0.0949	0.1348	0.1766	0.2230	0.1397	0.1777	0.2249
Recall	0.0701	0.0716	0.0935	0.3704	0.6606	0.0982	0.3864	0.6712
F-Measure	0.0783	0.0812	0.1078	0.1993	0.2907	0.1124	0.2428	0.2951

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, a graph based collaborative recommender algorithm is presented. The proposed algorithm is empowered with both context-aware pre-filtering and post-filtering. The approach presented by Bogers and the graph model presented by Phuong are used as a baseline and we modified it according to performance issues and architectural needs. The recommender system uses real channel usage data provided by Arçelik A.Ş. and merges this data with program guide retrieved from Radikal TV guide.

The design and implementation of the system are described in detail. First, the graph based collaborative algorithm is developed which is able to produce top-N recommendation list. Secondly, this algorithm is enriched with context aware pre-filtering which utilizes genre and time of day of program. Finally, the resulting recommended items list is re-ranked by using contextual attributes actors and directors of a program.

Experiments are performed with our recommender system for two purposes. First we tested the effects of parameters such as the length of the recommended items list, the path length of random walk, the number of users to be used for collaborative filtering on the performance of our graph based algorithm. We then evaluated the additional success achieved by context aware pre-filtering and post-filtering.

After we fixed the optimal values for the path length for random walk and the number of similar users, we have performed our tests with only one variable parameter which is the length of the recommended items list. Results show that the most effective context to the success of the recommender system is genre.

Furthermore, it is concluded that the effect of context aware post-filtering is limited because contextual attributes used for context aware post-filtering is valid for only specific programs. The ultimate conclusion to draw from our experiments is that our algorithm produces better results as long as it is enriched with more contextual attributes.

In our implementation, we use pre-defined similarity metric called co-occurrence between terms in our dataset. The next step to improve the performance of our recommender system might be using pre-defined similarities among other node types as well. For example, pre-defined program similarity and user similarity by using demographic information or social media information might bring performance improvements.

In our experiments, we had to restrict the length of random walk due to the performance issues. Results show that the performance of our system increases by increasing the length of path for random walk. Therefore, extending the length of path with more computational power might yield better results.

The evaluation of our recommender system is performed by using offline experiments. However, this work is intended to be embedded in Arçelik TV’s. Before putting our recommender algorithm into use for Arçelik, some online experiments might be carried out for testing the effectiveness of our system by the time of user interaction.

REFERENCES

[1] “Uydu Yayın Lisansı Olan Kuruluşlar Listesi (RD ve TV olarak).” [Online]. Available: http://yayinci.rtuk.org.tr/web/web_giris.php. [Accessed: 19-Aug-2014].

[2] F. S. da Silva, L. G. P. Alves, and G. Bressan, “PersonalTVware: An infrastructure to support the context-aware recommendation for personalized digital TV,” *Int. J. Comput. Theory Eng.*, vol. 4, no. 2, pp. 131–135, 2012.

[3] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, “Context-Aware Recommender Systems,” in *Recommender Systems Handbook*, 2011, pp. 217–253.

[4] I. Konstas, V. Stathopoulos, and J. M. Jose, “On social networks and collaborative recommendation,” in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 195–202.

[5] H. Cheng, P.-N. Tan, J. Sticklen, and W. F. Punch, “Recommendation via query centered random walk on k-partite graph,” in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, 2007, pp. 457–462.

[6] T. Bogers, “Movie recommendation using random walks over the contextual graph,” in *Proc. of the 2nd Intl. Workshop on Context-Aware Recommender Systems*, 2010.

[7] E. Samdan, A. Tasci, and N. K. Cicekli, “A Graph Based Collaborative and Context Aware Recommendation System for TV Programs,” in *1st Workshop on Recommender Systems for Television and Online Video*, (*in press* 2014).

[8] N. D. Phuong and T. M. Phuong, “A graph-based method for combining collaborative and content-based filtering,” *PRICAI 2008 Trends Artif. Intell.*, pp. 859–869, 2008.

[9] P. Resnick and H. R. Varian, “Recommender systems .(Special Section : Recommender Systems)(Cover Story) Recommender systems .(Special Section : Recommender Systems)(Cover Story),” vol. 56, no. March, pp. 1–3, 1997.

[10] B. Mobasher, “Data mining for web personalization,” *Adapt. web*, pp. 90–135, 2007.

[11] F. Ricci, L. Rokach, and B. Shapira, “Introduction to Recommender Systems,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA: Springer US, 2011, pp. 1–35.

[12] M. Balabanović and Y. Shoham, “Fab: content-based, collaborative recommendation,” *Commun. ACM*, vol. 40, no. 3, 1997.

[13] C. Desrosiers and G. Karypis, “A comprehensive survey of neighborhood-based recommendation methods,” *Recomm. Syst. Handb.*, 2011.

[14] R. Van Meteren and M. Van Someren, “Using content-based filtering for recommendation,” *... Mach. Learn. ...*, 2000.

[15] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” *Recomm. Syst. Handb.*, pp. 1–33, 2011.

[16] M. Pazzani and D. Billsus, “Content-based recommendation systems,” *Adapt. web*, pp. 325–341, 2007.

[17] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of.* Addison-Wesley, 1989.

[18] R. R. Walia, “Collaborative Filtering: A Comparison of Graph-based Semisupervised Learning Methods and Memory-based Methods,” 2008.

[19] P. Resnick, N. Iacovou, and M. Suchak, “GroupLens: an open architecture for collaborative filtering of netnews,” *Proc. ...*, 1994.

[20] U. Shardanand and P. Maes, “Social information filtering: algorithms for automating ‘word of mouth,’” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 210–217.

[21] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, “Recommending and evaluating choices in a virtual community of use,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 194–201.

[22] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, “Collaborative filtering recommender systems,” *Found. Trends Human-Computer Interact.*, vol. 4, no. 2, pp. 81–173, 2011.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[24] G. Linden, B. Smith, and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” *Internet Comput. IEEE*, vol. 7, no. 1, pp. 76–80, 2003.

[25] J. Herlocker, J. A. Konstan, and J. Riedl, “An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms,” *Inf. Retr. Boston.*, vol. 5, no. 4, pp. 287–310, 2002.

[26] D. Lin, “An information-theoretic definition of similarity.,” *ICML*, vol. 98, pp. 296–304, 1998.

[27] R. Bell and Y. Koren, “Scalable collaborative filtering with jointly derived neighborhood interpolation weights,” *Data Mining, 2007. ICDM 2007. Seventh …*, 2007.

[28] N. Lathia, S. Hailes, and L. Capra, “Temporal Collaborative Filtering With Adaptive Neighbourhoods Categories and Subject Descriptors,” in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 796–797.

[29] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl, “Enhancing digital libraries with TechLens+,” in *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, 2004, pp. 228–236.

[30] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Model. User-adapt. Interact.*, vol. 12, no. 4, pp. 331–370, 2002.

[31] J. Liu, P. Dolan, and E. R. Pedersen, “Personalized news recommendation based on click behavior,” in *Proceedings of the 15th international conference on Intelligent user interfaces*, 2010, pp. 31–40.

[32] R. Adomavicius, Gediminas Sankaranarayanan, S. Sen, and A. Tuzhilin, “Incorporating contextual information in recommender systems using a multidimensional approach,” *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.

[33] X. Baltrunas, Linas Amatriain, “Towards time-dependant recommendation based on implicit feedback,” in *Workshop on context-aware recommender systems (CARS’09)*, 2009.

[34] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone, “Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems,” in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 265–268.

[35] F. Cena, L. Console, C. Gena, A. Goy, G. Levi, S. Modeo, and I. Torre, "Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide," *AI Commun.*, vol. 19, no. 4, pp. 369–384, 2006.

[36] S. Sawant, "Collaborative Filtering using Weighted BiPartite Graph Projection A Recommendation System for Yelp," 2013.

[37] Z. Huang, W. Chung, T.-H. Ong, and H. Chen, "A graph-based recommender system for digital library," in *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, 2002, pp. 65–73.

[38] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly, "Video suggestion and discovery for youtube: taking random walks through the view graph," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 895–904.

[39] G. Öztürk and N. K. Cicekli, "A hybrid video recommendation system using a graph-based algorithm," *Mod. Approaches Appl. Intell.*, pp. 406–415, 2011.

[40] L. Ardissono, A. Kobsa, and M. T. Maybury, "Personalized digital television," *Human-computer Interact. Ser.*, vol. 6, 2004.

[41] S. Marchand-Maillet, "Content-based video retrieval: An overview," 2000.

[42] R. Bambini, P. Cremonesi, and R. Turrin, "A recommender system for an iptv service provider: a real large-scale production environment," in *Recommender systems handbook*, 2011, pp. 299–331.

[43] "tf–idf - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Tf–idf. [Accessed: 08-Oct-2014].

[44] M.-W. Kim, E.-J. Kim, W.-M. Song, S.-Y. Song, and A. R. Khil, "Efficient recommendation for smart TV contents," in *Big Data Analytics*, 2012, pp. 158–167.

[45] H.-J. Kwon and K.-S. Hong, "Personalized smart TV program recommender based on collaborative filtering and a novel similarity method," *Consum. Electron. IEEE Trans.*, vol. 57, no. 3, pp. 1416–1423, 2011.

[46] B. Martinez, A. Belen, E. Costa-Montenegro, J. C. Burguillo, M. Rey-L{'o}pez, M.-F. F. A, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," *Inf. Sci. (Ny).*, vol. 180, no. 22, pp. 4290–4311, 2010.

[47] Z. Yu, X. Zhou, Y. Hao, and J. Gu, “TV program recommendation for multiple viewers based on user profile merging,” *User Model. User-adapt. Interact.,* vol. 16, no. 1, pp. 63–82, 2006.

[48] T. Bozios, G. Lekakos, V. Skoularidou, and K. Chorianopoulos, “Advanced techniques for personalized advertising in a digital TV environment: the iMEDIA system,” in *Proceedings of the eBusiness and eWork Conference,* 2001, pp. 1025–1031.

[49] C. Shin and W. Woo, “Socially aware TV program recommender for multiple viewers,” *Consum. Electron. IEEE Trans.,* vol. 55, no. 2, pp. 927–932, 2009.

[50] L. Aroyo, L. Nixon, and L. Miller, “NoTube: the television experience enhanced by online social and semantic data,” in *Consumer Electronics-Berlin (ICCE-Berlin), 2011 IEEE International Conference on,* 2011, pp. 269–273.

[51] G. Shani and A. Gunawardana, “Evaluating recommendation systems,” in *Recommender systems handbook*, 2011, pp. 257–297.

[52] J. Wit, “Evaluating recommender systems: an evaluation framework to predict user satisfaction for recommender systems in an electronic programme guide context,” 2008.

[53] “Digiturk - Yayın Akışı.” [Online]. Available: http://www.digiturk.com.tr/yayin-akisi. [Accessed: 10-Aug-2014].

[54] “Türksat Uydu Haberleşme Kablo TV ve İşletme A.Ş. | TV, İnternet, Telefon.” [Online]. Available: http://www.turksatkablo.com.tr/TV-Rehberi. [Accessed: 10-Aug-2014].

[55] “TV Rehberi- Televizyon Programı ve Yayın Akışı Radikal’de.” [Online]. Available: http://www.radikal.com.tr/tvrehberi/. [Accessed: 10-Aug-2014].

[56] “MySQL - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/MySQL. [Accessed: 10-Aug-2014].

[57] “Active Record Basics — Ruby on Rails Guides.” [Online]. Available: http://guides.rubyonrails.org/active_record_basics.html. [Accessed: 10-Aug-2014].

[58] “Ruby on Rails - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Ruby_on_Rails. [Accessed: 10-Aug-2014].

[59] “jsoup Java HTML Parser, with best of DOM, CSS, and jquery.” [Online]. Available: http://jsoup.org/. [Accessed: 11-Aug-2014].

[60] “Dayparting - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Dayparting. [Accessed: 11-Aug-2014].

[61] A. Af, “Zemberek , an open source NLP framework for Turkic Languages.”

[62] “k-nearest neighbors algorithm - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. [Accessed: 12-Aug-2014].

[63] P. Hall, B. U. Park, and R. J. Samworth, “Choice of neighbor order in nearest-neighbor classification,” *Ann. Stat.*, pp. 2135–2152, 2008.

[64] C. J. Van Rijsbergen, *Information Retrieval, 2nd edition*. 1979.

[65] M. K. Buckland and F. C. Gey, “The relationship between recall and precision,” *JASIS*, vol. 45, no. 1, pp. 12–19, 1994.