TAG-BASED MUSIC RECOMMENDATION SYSTEMS USING SEMANTIC RELATIONS AND MULTI-DOMAIN INFORMATION

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

BY

İPEK TATLI

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

SEPTEMBER 2011

Approval of the thesis:

TAG-BASED MUSIC RECOMMENDATION SYSTEMS USING SEMANTIC RELATIONS AND MULTI-DOMAIN INFORMATION

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

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

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

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

Examining Committee Members:

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

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU ________________

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

Assoc. Prof. Dr. Pınar Şenkul
Computer Engineering Dept., METU ________________

M.Sc. Güven Fidan
AGMlab Information Technologies ________________

Date: ________________

ABSTRACT

TAG-BASED MUSIC RECOMMENDATION SYSTEMS USING SEMANTIC RELATIONS AND MULTI-DOMAIN INFORMATION

Tatlı, İpek
M.Sc., Department of Computer Engineering
Supervisor : Dr. Ayşenur Birtürk

September 2011, 79 pages

With the evolution of Web 2.0, most social-networking sites let their members participate in content generation. Users can label items with tags in these websites. A tag can be anything but it is actually a short description of the item. Because tags represent the reason why a user likes an item, but not how much user likes it; they are better identifiers of user profiles than ratings, which are usually numerical values assigned to items by users. Thus, the tag-based contextual representations of music tracks are concentrated in this study.

Items are generally represented by vector space models in the content based recommendation systems. In tag-based recommendation systems, users and items are defined in terms of weighted vectors of social tags. When there is a large amount of tags, calculation of the items to be recommended becomes hard, because working with huge vectors is a time-consuming job. The main objective of this thesis is to represent individual tracks (songs) in lower dimensional spaces. An approach is described for creating music recommendations based on user-supplied tags that are augmented with a hierarchical structure extracted for top level genres from Dbpedia. In this structure, each genre is represented by its stylistic origins, typical instruments, derivative forms, sub genres and fusion genres. In addition to very large vector space models, insufficient number of user tags is another problem in the recommendation field. The proposed method is evaluated with different user profiling methods in case of any insufficiency in the number of user tags. User profiles are extended with multi-domain information. By using multi-domain information, the goal of making more successful and realistic predictions is achieved.

Keywords: Recommendation systems, User Profiling, Social Tagging, Semantic Relations, Dimensionality Reduction

ÖZ

SEMANTİK İLİŞKİ VE ÇOKLU ALAN BİLGİSİ KULLANAN ETİKET TABANLI MÜZİK TAVSİYE SİSTEMLERİ

Tatlı, İpek
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Dr. Ayşenur Birtürk

Eylül 2011, 79 sayfa

Web 2.0 'ın evrimleşmesiyle sosyal ağ siteleri üyelerine içerik oluşturmada olanak sağlamıştır. Web sitelerinin çoğunda kullanıcılar içerikleri etiketleyebilmektedir. Etiket herhangi bir kelime olabilir, ama aslında içerik hakkında kısa bir açıklamadır. Etiketler bir kullanıcının bir içeriği ne kadar sevdiğini değil, neden sevdiğini gösterir. Etiketler kullanıcılar için kullanıcılar tarafından atanan sayısal değer olan oylamadan daha iyi bir tanımlayıcıdır. Bu nedenle bu tez çalışmasında müzik parçalarının etiket tabanlı bağlamsal temsilleri üzerinde yoğunlaşılmıştır.

İçerik tabanlı tavsiye sistemlerinde, öğeler genellikle vektör alan modellerinde temsil edilmektedir. Etiket tabanlı tavsiye sistemlerinde, kullanıcılar ve öğeler, ağırlıklı sosyal etiket vektörleri ile tanımlanır. Büyük bir miktarda etiket olduğunda, tavsiye edilecek öğeleri hesaplamak zor olur. Çünkü büyük vektörler ile çalışmak zaman alıcı bir iştir. Bu tezin ana amacı, müzik parçalarını düşük boyutlu alanlarda temsil etmektir. Kullanıcı tarafından sağlanan etiketlere göre müzik önerileri oluşturmak için Dbpedia'dan üst düzey müzik türleri için çıkarılan bir hiyerarşik yapı açıklanmaktadır. Bu yapıda her tarz; üslup kökenleri, tipik enstrumanları, türev formları, alt türleri ve füzyon türleri tarafından temsil edilmektedir. Vektör alan modellerinin aşırı büyüklüğüne ek olarak, kullanıcı etiketi sayısının az olması da tavsiye alanında önemli bir problemdir. Olası kullanıcı etiketi azlığı problemi ihtimali nedeniyle, önerilen matris indirgeme yöntemi farklı kullanıcı profili yöntemleri ile değerlendirilmektedir. Kullanıcı profilleri çoklu alan bilgisi kullanılarak genişletilir. Çoklu alan bilgisi kullanarak, daha başarılı ve gerçekçi tahminler yapılmaktadır.

Anahtar Kelimeler: Tavsiye Sistemleri, Kullanıcı Profilleme, Sosyal Etiketler, Semantik İlişkiler, Matris Boyut İndirgeme

To my family

ACKNOWLEDGEMENTS

Firstly, I want to thank to my thesis supervisor Dr. Ayşenur Birtürk for her advices, motivation and support during my thesis study.

I am grateful to my family -Yeter and Ahmet Tatlı- for their love and support. Without them, I could not have completed this study. I thank to my dear friends Nihal Karagöz, Çağrı İlçe, Çiğdem Okuyucu and Serra Tekiroğlu for their help and support whenever I needed. I would also like to thank to Güven Fidan for giving me an opportunity to learn semantic issues in AGMlab.

Moreover, I appreciate the contributions from my director Erdem Alptekin, my team leader Gökhan Akca, my Latex teacher Yunus Emre Esin and other colleagues within SIEMENS A.S.

Lastly, thanks to anonymous reviewers of eChallenges 2010, ACM RecSys 2011, IEEE ICDM 2011 Conferences; WOMRAD 2011-2nd Workshop on Music Recommendation and Discovery and MMIS 2011-The Fifth International Workshop on Mining Multiple Information Sources.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CB	Content-based
CF	Collaborative Filtering
IR	Information Retrieval
RS	Recommendation System
TB	Tag based
TBRS	Tag-based Recommendation System
VSM	Vector Space Model

CHAPTER 1

INTRODUCTION

Social networking and communication has become trendy in the past few years. People can easily join social networking sites like Facebook [1], Twitter[2] and MySpace[3]. Communication and maintaining their relationships with friends are the most important reasons why users use such sites [4]. Adding friends, joining groups, attending events, sharing photos and presenting personal interests are some of the activities in such social networking sites. There have been some studies about how much information users share with others in these networks. Studies show that Facebook users share a considerable amount of information about themselves [5]. In addition to the aforementioned activities, members indicate their favorite movies, television series, games, books and music bands on their Facebook profiles which are examples of explicit data collection regarding listing of preferred items [6].

Internet radios like Last.Fm[8], Pandora[9] and GrooveShark[10] are other examples of social networking. People use these radio stations interactively in terms of specifying whether they like/dislike the tracks; tagging bands, albums and tracks; and making comments about the music. Most of the sites let their members participate in the content generation. For example, users can label artists, albums and tracks with tags in Last.Fm. A tag can be anything; but it is actually a short description about the item [11]. Because tags represent the reason why a listener likes a song, but not how much he/she likes it [12]; they are better identifiers of the user profiles than the ratings which are usually numeric values that users give items. Thus, we have concentrated on the tag-based(TB) contextual representations of music tracks.

In some systems, users select tags from a list of words; but they can also enter what they want which means that most of the time there is no limitation in tagging process. In music domain, tags can be about genre, locale, mood, instrumentation, style, misc and personal opinions as stated in [6]. It can be said that tags are mostly human-based annotations so that they suffer from the problems of natural language processing. They may be misspelled and misspelling causes increasing number of tags for the same tag. Using a spell checking and a stemming algorithm may help solving this problem. Moreover, tags may be noisy and may not provide sufficient information about the item. In addition to these problems, tags may be used for vandalism and hacking [6]. As another problem, some bands/artists may not be tagged enough and there may not be enough tags at track level which cause a sparsity problem. Because of the reasons like synonymy, polysemy, subjectivity and noise; the amount of tags to be used in recommendation systems is always huge.

In document classification, bag-of-words is a common approach in terms of representing each text as a vector of weighted terms as stated in [13]. Same as document classification, in recommendation systems, both users and items may also be represented as vectors of weighted tags. When the amount of the tags is huge, calculation of the items to be recommended is becoming hard because working with huge vectors is a time-consuming job. Moreover, quality of the recommendations is inefficient when compared with the time consumed. Thus, dimensionality reduction is important for faster and efficient recommendations.

In the recent studies, only one information domain (movie or book or music) has been used. However, it is now easier to use multi-domain information about users because nowadays most people use more than one social networking site.

1.1 Aim of The Study

In this thesis, it is aimed to provide faster and efficient recommendations in content-based music recommendation systems. Real Last.fm data has been used. Last.fm does not provide rating mechanism to its users. However, it provides a social tagging mechanism so that users can enter tags for tracks, albums and artists. We have concentrated on TB representations of tracks and users. Both tracks and users are represented by weighted vectors of social tags. Tags are huge in size and working with such huge vectors results in slow recommendation process. Moreover, huge vectors are mostly sparse and noisy which causes inefficient recommendations. Thus, dimensionality reduction methods are applied on huge vector spaces. LSA is the most common and reliable method. However, it suffers from the lack of ”morphology” issues. We have implemented a well-organized semantic-relations method in order to reduce the dimensionality in the music RSs. Each track/user vector size has been reduced to the number of music genres in the music domain. In our hierarchical structure each genre is represented with its sub genres, fusion genres, instruments, stylistic origins and derivative forms[61]. Our approach has performed better when compared with LSA (with a computable ranking). If the ontology is ready, LSA is complex in terms of time and space when compared with our dimensionality reduction method. Moreover, our method considers ”morphology” issues whereas LSA does not.

In Last.fm, users may listen to music however they may not enter tags for tracks. Because of the insufficient number of user tags, most of the users get poor recommendations. In order to provide better recommendations to those who does not enter sufficient number of tags, users have been profiled by different methods using personal tags, friends’ tags, tags of all tracks they listened to, tags of artists of all tracks they listened to. In addition to these methods, user profiles have been extended with the Facebook profiles.

Briefly, it is aimed to compare the content-based recommendation systems using different dimensionality methods (semantic relations and LSA), different user profiling methods and different similarity functions. Moreover positive effect of multi-domain information is expected. The contributions of this study are that: (1) we provide a ”semantic relations” method for dimensionality reduction in very huge vector spaces; (2)we perform the comparison of our method against the classical Singular Value Decomposition (SVD) method which is the base of Latent Semantic Analysis (LSA), our method outperforms the traditional one; (3) different user profiling methods are used for the users having insufficient number of tags; (4) user profiles are extended with Facebook profiles.

1.2 Outline of The Thesis

The rest of this thesis is organized as follows:

Chapter 2 - Background Information and Related Work summarizes the literature on recommendation systems, TB music recommendation systems and dimensionality reduction methods.

Chapter 3 - Our Approach describes the proposed dimensionality reduction algorithm and the user profiling methods. It also presents the user profile extension with multi-domain information.

Chapter 4 - Evaluation states the evaluation methodology and metrics used in the system evaluation. Comparison with the state-of-the-art method (LSA) is presented.

Chapter 5 - Conclusion provides a general overview of this thesis and draws the conclusions of the study. It also suggests some possible improvement issues which can increase the success of the studied recommendation approach.

CHAPTER 2

BACKGROUND INFORMATION AND RELATED WORK

This chapter provides a background in the area of recommendation systems (RSs) and music recommendation. Firstly, the recommendation problem and RSs are described. Then the techniques used in RSs are explained. Consequently, music recommendation, TB music recommendation and dimensionality reduction methods in large vector spaces are summarized.

2.1 Recommendation Systems

Nowadays, the Internet is the main source of information which has about 15-30 million web pages. However, all the information is heterogeneous and unstructured. Search for information is becoming an important task. Google and Yahoo are the most popular search engines. About 150 million searches are done through Google per day. Thus, academic and industrial communities develop methods to index and search this large amount of information. Once a web page is found on Google or Yahoo, users must search on it in order to verify if the information needed is there. Search engines perform mainly 2 processes: indexing and searching. In indexing process; it can be thought that there are cyber robots that visit every web page, process their contents and index them by words [14]. Indexing can be done manually or in an automated way. In automated process of indexing; methods like feature extraction, eliminating stop words, stemming, counting and mapping to concepts can be used[15].

Information filtering is the removing process of all information which is of no interest whereas Information Retrieval (IR) is the process of finding relevant data within a large amount of unstructured data. Web search engines, library catalogs,store catalogs, e-mail notification engines and cookbook indexes are examples of IR systems. ”Information Storage and Retrieval" (ISAR) is another name of IR. IR is mostly about texts. However, IR of speech, cross-language, question-answering, image retrieval and music retrieval are also becoming popular[16].

Recommendation process can be treated as an IR problem. Both RS and IR systems deliver information to users using a large amount of unstructured data which is not from a controlled database. RSs also compares a history of queries, extracts relevant information and organizes it for searching.

Today web-crawling is popular in order to perform the IR tasks in RSs. With the help of crawling, RSs gather pages from the Internet in order to index them. As an example, everybody has a Facebook profile. In Facebook profiles, there is a large amount of information; some of the information is useful (like birth-date, favourite movies, etc..) and some are unnecessary like advertisements.Gathering useful information is IR, removing unuseful information is IF.

Acquiring meaningful information from Facebook is easy. Sometimes, this task may be harder. Data collection in RSs can be done in 2 ways. Asking a user to rate an item, asking a user to rank a collection of items from favourite to least favourite, presenting two items to a user and asking him/her to choose the best one, asking a user to create a list of items that he/she likes are some examples of implicit data collection. On the other hand, analysing item/user viewing times, keeping a record of the items that a user purchases on-line, for music RSs obtaining a list of tracks that a user has listened to and for video RSs obtaining a list of videos that a user has watched on can be given as examples of explicit data collection [6].

People make decisions every day like 'Which movie should I watch?', 'Which city should I visit' or 'Which music should I listen to?'. Today, there are many choices and people have a little time to explore them all. Recommendation booms because of this problem and helps people make decisions in these big spaces. Simple principle of the RSs is to compare the user interests which are acquired from their profiles. In other words, RSs are a specific type of Information Filtering. Extracting user interests from user profiles is the IR process which will be explained in the following section. The comparison of the acquired user profiles is the recommendation process and can be done in 3 different ways which will also be explained in 2.3.

Table 2.1: Content analysis in content-based approach

Film Name	**Director**	**Cast**	**Genre**	**Year**
Ed Wood	Tim Burton	Johnny Depp, S. Jessica Parker	Drama, Biography	1994
The Crow	Alex Proyas	Brandon Lee, Ernie Hudson	Action, Crime, Fantasy	1994
Arizona Dream	Emir Kustarica	Johnny Depp, Jerry Lewis	Fantasy, Drama	1993

2.2 Recommendation Techniques

RSs find new and relevant items for individuals. User or item profiling is done through the IR process. For a RS, the next important thing is to decide what should be recommended. After calculation of similarities between users and items, individual generation of suggestions are performed. Thus, recommendation process can be done in a collaborative, content-based or hybrid way.

2.2.1 Content Based Recommendation Systems

In content-based (CB) recommendation, it is assumed that users like items similar to the one that they liked in the past. CBR deals with the content of items, it analyses the descriptions of items. A sample of item description analysis can be seen in the Table 2.1. In this example, each movie has some attributes like director, cast, genre and year. Data is structured and mostly saved in the database tables. However, most of the times data is unstructured like news texts. The best way to deal with it is to treat each word as they are different attributes.

Items are represented as attribute vectors. Each vector element is the weight of the corresponding attribute. In the calculation of weights, some methods can be used. Here are some of them:

1. Term Count Method: Each matrix entry is set to the number of occurrences of the term in that document.

2. Binary Method: If the term is in the document, that matrix element is set to 1. Otherwise, it is set to 0.

3. TF IDF Method: This is the most common technique. Each matrix entry is set to TF-IDF value of that term. TF is computed as follows:

$$TF(t_i) = f(t_i, d)$$

$$IDF(t_i) = log(\frac{D}{|d : t_i \in d|})$$

$$Weight(t_i) = TF(t_i) \times IDF(t_i)$$

Where,

- t_i is the $i^t h$ term.
- $f(t_i, d)$ is the frequency of term t_i in document d.
- D is the total number of documents in the corpus.

Moreover **log-entropy**, **root type** and **modified tf idf** methods can be used [19].

In the user profiling part of the CBR systems, user profiles have a model of user preferences and a history of user interactions [20]. Decision trees, rule induction and nearest neighbour algorithms are used in order to find the items similar to the user profiles. CBR systems suffer from the over-specialization problem that recommendations are very similar to the items that were liked before. New, interesting and unexpected items could not be recommended[20].

Pandora is a very good example of CBR systems. 50 music experts are working for Pandora such that they listen to a song each 30 minutes period and specify its properties according to a predefined feature set. As it can be seen, this is an expensive approach. However, there are also some feature extraction tools that can describe the audio automatically [6].

2.2.2 Collaborative Filtering Systems

In Collaborative Filtering(CF), it is assumed that users with similar tastes rate items similarly. It does not depend on the contents or features of items. CF needs ratings for its recommenda-

Table 2.2: Rating scales in collaborative filtering

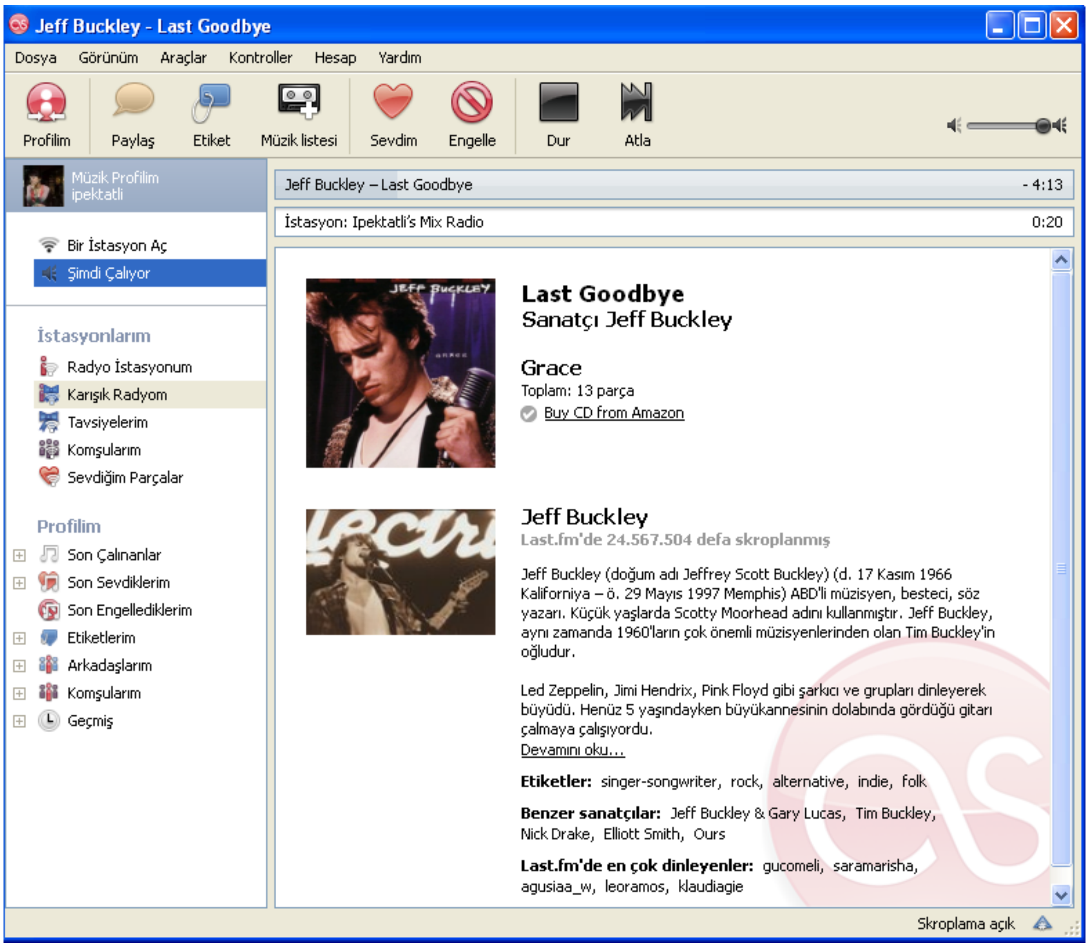
Rating	**Scale**
unary	”good” or ”not know”
binary	”good” or ”bad”
integer	integers:1-5 or 1-10

tion process. The most common ratings are shown in the Table 2.2[17].

For better recommendations, systems should know more about users. In fact, the more a system knows about a user, better recommendations it generates. Rating scales plays an important role in better recommendations. If rating scale of a system is 1-100, finding similarity of users becomes harder because users may have given 88 and 89 to an item although they are similar in a 1-5 scale system. Moreover, explicit ratings require a dedicated time for users in order to get efficient recommendations. Rating process may be boring for users. Thus, implicit data collection may be more realistic. Last.fm’s AudioScrobbler, which can be seen in Figure 2.1, is a good example of implicit data collection. With the help of this software, Last.fm keeps a history of tracks that users are listening to.Moreover, the functionalities like ”love this track”, ”skip this track” and ”ban this track” helps in the user profiling process. Last.Fm is the best example for the RSs that uses CF methods. It uses social tagging strategy in order to enrich its recommendations.

If we compare CF with CB approach, CF uses ratings of users for finding user similarities whereas CB approach uses contents and features of items in order to find item similarities. CF assumes that similar users like similar things. On the other hand CB approach assumes that if a person likes an item in the past, he/she likes items similar to the one that he/she liked in the past. CF is better than CB in domains where there is not much content associated with items. Moreover, CF suggests serendipitous items which may not be like the items that user have liked before. In some domains with lots of items, users may have rated only a few of them which causes sparsity. In addition to sparsity, some items which have not been rated are not recommended.

Breese et al. classified CF algorithms as memory-based and model-based algorithms in [18]. Pure memory-based algorithms keeps users, items and rating in the memory which increases run-time duration. Today, most of the systems uses pre-calculated model of the system



Last Goodbye
Sanatçı Jeff Buckley

Grace
Toplam: 13 parça

Jeff Buckley
Last.fm'de 24.567.504 defa skroplanmış

Jeff Buckley (doğum adı Jeffrey Scott Buckley) (d. 17 Kasım 1966 Kaliforniya – ö. 29 Mayıs 1997 Memphis) ABD'li müzisyen, besteci, söz yazarı. Küçük yaşlarda Scotty Moorhead adını kullanmıştır. Jeff Buckley, aynı zamanda 1960'ların çok önemli müzisyenlerinden olan Tim Buckley'in oğludur.

Led Zeppelin, Jimi Hendrix, Pink Floyd gibi şarkıcı ve grupları dinleyerek büyüdü. Henüz 5 yaşındayken büyükannesinin dolabında gördüğü gitarı çalmaya çalışıyordu.

Etiketler: singer-songwriter, rock, alternative, indie, folk

Benzer sanatçılar: Jeff Buckley & Gary Lucas, Tim Buckley, Nick Drake, Elliott Smith, Ours

Last.fm'de en çok dinleyenler: gucomeli, saramarisha, agusiaa_w, leoramos, klaudiagie

Figure 2.1: AudioScrobbler of Last.fm

and uses it in run-time. In [18], CF algorithms are classified into probabilistic and non-probabilistic ones. The most common nearest neighbour algorithms, graph-based algorithms, neural algorithms and rule-mining algorithms are non-probabilistic.

2.2.3 Hybrid Systems

There are some RSs that combine these two methods. They exploit the advantages of the two method while avoiding their disadvantages. However, other problems rise such as the increasing number of users and documents to be saved [21].

Fab is a good example of hybrid systems [24]. In the Fab system, users are represented by the content analysis of the items that they rate. Then they receive items both which belong to the most similar users' profiles and when they score highly against their profiles. With the help of content-based collaborative filtering, advantages of CB and CF approaches are used. Moreover, such hybrid systems solve the shortcomings of both. Over-specialization is the most important problem of CB systems that such systems recommend only the most similar items to the previously rated/seen/ items. Any unusual (but similar) items that a user may like are not recommended. CF solves this problem by recommending new and unusual items. On the other hand, CF has a problem that newly added items are not rated by anyone so they are not recommended to anyone. Hybrid systems do not have such problems. Users are represented as vectors of the contents. In this vector space, features are the elements of the vectors. For the movie domain; release date, type (movie or tv series), director, writer, cast, language, country, company, genre and keywords may be the features of the movies.

In [22], recommendation techniques are classified into 5 types: CF, CB systems, Demographic, Utility-based and knowledge-based. CF and CB systems are explained in the previous sections. **Demographic** method uses categorization of users based on their personal attributes like age, place, and gender etc... An early example of this type of recommendation can be seen in the one which is designed by Rich et al [30]. It is a book RS based on the personal information of its users. Demographic approach uses people to people similarities like CF. It has an advantage that it does not need ratings; it deals with the demographic data of the people. On the other hand; **utility-based approach** does not build a long term generalization. System produces a utility-function. Although producing a utility-function is a problem, utility-based approach is a good approach. For example, price is a trade-off preference after a while and utility-based systems updates the preferences very often and produces user-specific functions about user's short-term needs. System relies on the information provided by users through questionnaires and interviews. This approach may use constraint-satisfaction techniques. Thirdly, **knowledge-based** RSs keep user profiles in knowledge-based structures. Reasoning may be done between item profiles and user profiles. For this kind of inference there are some frameworks that can be used such as Clips [31] and Jess[32].

Each technique has its own advantages and disadvantages. A pure CB approach has some shortcomings. Generally a very limited analysis of certain kinds of content is provided. The second problem is over-specialization. User is limited to see the items similar to ones that are already voted. Finally, the last problem is to expect user feedback because it is a boring task for users whereas it is the only factor that effects the future performance of the system. On the other hand, CF has other disadvantages. If the number of users is very small relative to the volume of information in the system, then the coverage of the ratings becomes very sparse. As an another problem; if tasting of a user is very unusual than the others, then any nearest neighbours will not be calculated and this leads to a poor recommendation.

The crucial problem is how best to learn about a new user because RSs knows nothing about the new user. This is called ”the new user problem” and the system should acquire some information about the new user. The most direct way to do is to ask for ratings for some items. But the questions should be selected carefully. For example, a food RS should not ask to the new user if he/she likes vanilla ice cream because most people likes it and this is not a useful information for predicting the user’s taste. Same as vanilla ice cream, a travel RS should not ask to a new user if she travelled and liked Burkina Faso; because he/she might not have even heard about its name and this also does not help system predict the new user’s style. Moreover, this rating part is boring for new users, so this should be done in a funny way while learning enough about him/her. Thus, hybrid RSs combine two or more of these techniques in order to prevent the aforementioned disadvantages. There are some hybridization methods which are now explained.

Weighted method uses the scores of several recommendation techniques that are combined together to produce a single recommendation.

Switching method switches between recommendation techniques depending on the situation.

Mixed method generates recommendations from several different RSs at the same time.

Feature combination method generates features from different recommendation data sources which are used together in a single recommendation algorithm.

Cascade method is the one in which one recommender refines the recommendations given by another.

In **feature augmentation method**; output from one technique is used as an input feature to another.

In **Meta-level method**; the model learned by one recommender is used as input to another.

2.2.4 Semantic Recommendation Systems

Apart from these types of recommender systems, semantic RSs are very popular nowadays. Vocabulary or ontology based systems, trust network based systems and context-adaptable systems are said to be the main types of semantic RSs [22]. Most accurate and most interesting recommendations are done by using artwork features and their semantic relations in Cultural Heritage Information Personalization(CHIP) project [23]. Although it seems that using semantic relations affect the recommendation process positively, determining which relations are interesting for users is important.

Wang and Kong have designed a system which uses categorical information of items [25]. Similarity of user pairs are calculated with the help of these measures: the similarity of user evaluation histories; the similarity of these user's demographic data; and the users similarity in interest or preference based on the semantic similarities of the items retrieved and/or evaluated. Farsani et al. suggest a system which classifies the products and customers with OWL for product-client similarity[26]. Another system in the field of e-commerce is designed by Ziegler et al.[27]. The system is uses the collaborative-recommender paradigm through content using a product taxonomy from which the user profiles are defined. Jung et al. propose a RS based on personal information in terms of Semantic Web[28]. The system stores user profiles and web services in RDF files and extracts the most relevant RDF objects in accordance with the user profile. Another example is the system of Cantador et al[29]. Their system uses a multi-layer semantic framework. It generates user profiles with the help of ontological concepts and classifies people in terms of their preferences. Using these classes, system generates different semantic levels. It finds implicit social networks which may help in recommendation process.

Here are the advantages of semantic relations usage in RSs[22]:

1- Inter-operatibility of system reources

2- Homogeneity of item/ user representations, improvement of these representations

3- Dynamic contextualization

4- Better performance in CF systems and social networks

5- Handling the cold-start problem by completing missing parts of the profiles

6- Improvement of the system with pre-defined rules

7- Ability to extend profiles semantically

8- Generation of descriptions enriched by web services

2.3 Music Recommendation

People listen to music, but listening to the same tracks and singers may be boring most of the time. Poeple may get suggestions from their friends in order to find new and relevant tracks; they may find out new music in some radio stations, on social networks (like Facebook, Youtube, MySpace), the movies thay have watched and in shopping stores. However, finding new music is mostly on-line. Thus, internet radios are becoming popular that people may find relevant music with the help of recommendations of the internet radios. Online internet radios can be seen in the Fiure 2.2.

Figure 2.2: Online Internet radios

2.3.1 Music Recommendation Systems Overview

In the past, people listen to music on their walkmans, radios or CD-players.However, today they prefer on-line music sources; because with the growth of Internet, everything including music is available to everyone. In today's world, thousands of albums are arriving to music market every week and the music domain becomes bigger. Music recommendation gains more and more importance because finding new and similar music is hard in this big space.

FM radios, TV programs, friends, shopping stores, clubs, bars, advertisements, movies, internet radios, websites and fan pages are sources of new music. According to a survey mentioned in study [6], 86 percent of the people uses on-line sites in order to get suggestions for new and relevant music.

Music recommendation can be thought as a prediction problem. Main task of a recommendation engine is to predict items (songs, albums, artists) that a user may be interested in. There are some use-cases mentioned in [6].

1- Find good: RSs aiming to ”find good” provides a ranked list of items some of which may be novel.

2- Find all good items: RSs aiming to find all good items must have a low false positive rate.

3- Recommend sequence: RSs aiming to recommend sequence provides an ordered sequence as a whole (ex: play-lists).

4- Just browsing: Some RSs are for just browsing and searching items.

5- Improve profile: Improving profiles is important for RSs having a strong community component.

6- Express self: This is important in social networking sites such that people communicate and interact with each other according to common interests.

7- Influence others: This is important in social networking sites such that some people may want to influence others about his/her interests.

Users can be described in a demographic way; with their age, gender, language, education, family status and income. In addition to demographic profiling, users can be described according to their geographic location. Moreover, users can be described in a psychological way; with their general interests, hobbies, music preferences etc.. This can be done explicitly through lists of preferred/hated artists/song or users' ratings/reviews/opinions. Tracking listening habits and keeping a history of pages/blogs visited are some ways of implicit psychological user description. Explicit one is not always reliable because:

1- People do not bother.

2- Explicit ratings/opinions/reviews provide only a partial information about the user. Or people may even lie.

3- If people tell the truth about themselves, they often fail to update their information over time.

There are some representations for user descriptions. As the complexity grows, expressiveness of the representations also grow. Thus GUMO is the most complex bur expressive one.

1- UMIRL (User Modelling for Information Retrieval Language): UMIRL can be thought as an XML-file. Users are described in terms of their demographic and geological information. Their music background and music preferences are kept. With these information, rules like "while having a special dinner with girlfriend, a romantic piece has a slow tempo, lyrics are related with love, and has a soft intensity can be " are created in XML.

2- MPEG-7: MPEG-7 is a standard for multimedia content description. It has an XML schema. It keeps the user preferences for content-filtering and browsing.

3- FOAF (Friend of a Friend): It is a huge ontology of users and their social networks.

4- GUMO (General User Model Ontology): It is the top level user ontology. In addition to basic user dimensions,personality (talkative, quiet, shy, kind, helpful...) and characteristic (extrovert, introvert, optimistic, pessimistic...) information are also kept.

In music RSs, tracks can be profiled in terms of their audio contents (like rhythm, timbre, tonality, instrumentation). In addition to audio descriptions, tracks can be profiled in terms of their text descriptions like their metadata, reviews mined from various blogs, lyrics and tags [6]. Metadata information is mostly applied by experts. Artist name, album name, genre, duration and year are some attributes in the metadata information. Attributes are global descriptions of items and are same to all users whereas tags are local descriptors and might change from user to user [33]. In our study, we focus on the text descriptions, namely tags in track profiling.

2.3.2 Tag-based Music Recommendation

Tags are associated with Web 2.0. Since they are in natural language, they suffer from the problems of natural language processing. Thus,they may be misspelled. Misspelling causes increasing number of tags for the same tag. Using a spell checking and a stemming algorithm may help solving this problem. Stemming is a technique to convert similar words to a common base form. This base form does not have to have a meaning from a linguistic point of view (such as reducing synonyms to a single word, or finding the root of the word). Various stemming algorithms exist for English language. With the help of stemming, similar words (which has the same roots) can be found and number of attributes in the item set can be decreased. The other problems in social tags are:

Polysemy: Some tags may have more than 1 meaning. Ex: progressive, love

Synonymy: People may enter different words for the same tag. Ex: hip-hop, hiphop, rap

Personal tags: Some tags are about users' own experiences and opinions. Ex: seen live, I own it

Noise: Some tags are meaningless. Ex: x, whoa

Sparsity of data: New bands/albums/tracks may not have enough tags.

Hacking and vandalism: Some people may enter irrelevant tags for tracks/albums/artists. Ex: death metal for Paris Hilton

Users are now content-generator rather than content-reader because they help the generation of contents on the web. In the music domain, tags can be entered for songs, albums and artists. Most of the existing previous works are about recommendation of tags. Because social tagging is getting more and more important, tag-based recommendation systems (TBRSs) systems are becoming a new research topic for academicians. Most of the current music RSs use CF technique with 2- dimensional data:

1- user - tag

2- item - tag

3- user - item

A 3-dimensional data (user-item-tag) should be thought in TBRSs. Data is mostly very sparse in these systems because each user generally tags a small amount of items. In [34], a 3-order-tensor framework is represented.Each 'user-item-tag' data is a tensor in this study. With this framework, data sparsity problem is handled because a Higher Order Singular Value Decomposition (HOSVD) method is applied on this 3-dimensional matrix. Their method outperforms the traditional algorithms.

RSs either predict ratings for unseen items or predict items that can be liked. Most of the social we-sites like Last.fm does not have a rating mechanism. Instead of explicit ratings, today's RSs use implicit ratings (user's listening habit, user's purchase history etc.). Thus rating scale in implicit rating mechanisms is 0-1. Tags can be used in rating-based CF systems with the help of implicit rating mechanism [33]. If the tag is used by the user, its rating is 1; otherwise its rating is 0. In most of the previous studies, 2-dimensional spaces in music space are taken into consideration (item-user or user-tag or item-tag). User-tag and item-tag relations can be used in order to extend the rating data [33].

2.3.2.1 CF Approaches in Tag-based Music Recommendation

In [33], user-item matrix is extended by including user tags as items. They also extend user-tag matrix by including item tags as users. Details can be seen in Figure 2.3. After extension with tags, they use a fusion method on traditional CF algorithm which captures users, items and tags. With this approach, 3-dimensional relation <user,item, tag >is held unlike global attributes which only have a 2 dimensional relation <item, attribute >. This is known as tag-aware method.

On the other hand, the three 2-dimensional relations among users, tags and items are used in a new similarity measure generation which outperforms the traditional item-based and user-based CF methods [35]. In this approach, neighbourhood generation is done through the similarity of users' tags, similarity of users' items and the similarity of user's tag-item

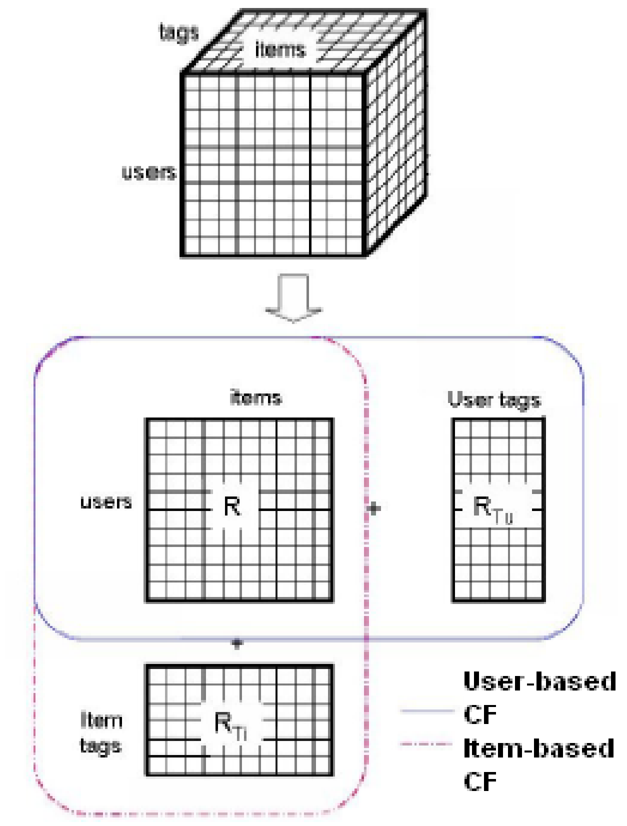


Figure 2.3: Extension of user-tag-item matrix

relationships.

$UTsim(ui, uj)$: The similarity of users' tags, which is measured by the percentage of common tags used by the two users:

$$UTsim(u_i, u_j) = \frac{|T_{u_i} \cap T_{u_j}|}{max(|T_{u_k}|)}$$

$UPsim(u_i, u_j)$: the similarity of user's items, which is measured by the percentage of common items tagged by the two users:

$$UPsim(u_i, u_j) = \frac{|P_{u_i} \cap P_{u_j}|}{max(|P_{u_k}|)}$$

$UTPsim(u_i, u_j)$: the similarity of the users' tag-item relationship, which is measured by the percentage of common relations shared by the two users:

$$UTPsim(u_i, u_j) = \frac{|TP_i \cap TP_j|}{max(|TP_k|)}$$

Thus, the overall similarity measure of two users is defined as below:

$$Simu(u_i, u_j) = w_{UT} \cdot UTsim(u_i, u_j) + w_{UP} \cdot UPsim(u_i, u_j) + w_{UTP} \cdot UTPsim(u_i, u_j)$$

Where,

- $w_{UT} + w_{UP} + w_{UTP} = 1$.
- w_{UT}, w_{UP} and w_{UTP} are the weights to the three similarity measures,respectively.

In addition to user similarities; item similarities are calculated with common tags, common users and common tag-item relationships.

$PTsim(p_i, p_j)$: The similarity of two items based on the percentage of being put in the same tag.

$$PTsim(p_i, p_j) = \frac{|T_{p_i} \cap T_{p_j}|}{max(|T_{p_k}|)}$$

$PUsim(p_i, p_j)$: the similarity of two items based on the percentage of being tagged by the same user.

$$PUsim(p_i, p_j) = \frac{|U_{p_i} \cap U_{p_j}|}{max(|U_{p_k}|)}$$

$PUTsim(p_i, p_j)$: the similarity of the two items based on the percentage of common tag-item relationship.

$$PUTsim(p_i, p_j) = \frac{|UP_i \cap UP_j|}{max(|UP_k|)}$$

$$Simp(p_i, p_j) = w_{PU} \cdot PUsim(p_i, p_j) + w_{PT} \cdot sim(p_i, p_j) + w_{PUT} \cdot PUTsim(p_i, p_j)$$

Where,

- w_{PU}, w_{PT} w_{PUT} are the weights.
- Their sum is also 1.

This 2 methods outperforms the tag-aware method. Traditional CF, tag-aware method and this TB method is compared in Figure 2.4. First one is user-based, the second one is item-based.

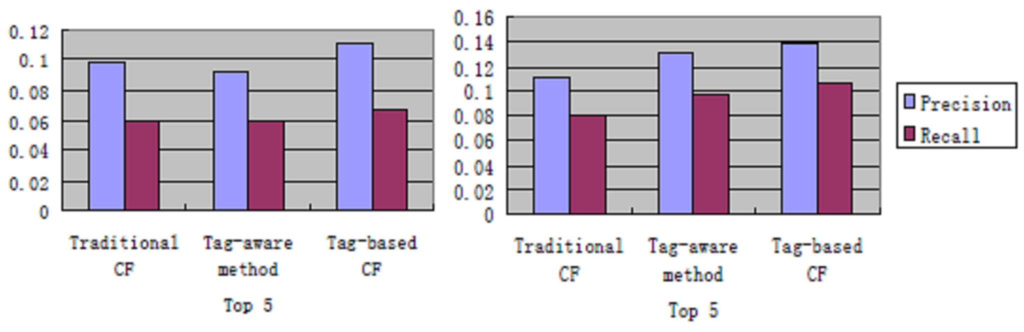


Figure 2.4: Comparison of Traditional CF, tag-aware method and tag-based methods

In [12], search-based algorithms are applied on TB profiles. A query is created. Tags in the current user's profile are added to the query. Then the tracks which are tagged with the tags in the query are searched. The cosine similarity between the resulting tracks and the query are calculated. The most similar tracks are recommended. This search-based method outperforms the classical TB CF methods.

In most of the previous studies, user profiling is done in 4 ways:

1- Collecting all the data and descriptions in the user profiles.

2- Using only the tags specified by user.

3- Using the tags used by friends.

4- Using the tags associated with the user's social media reflecting his/her interests and activities.

In [43], users are profiled by his/her own tags as well as the tags specified by his/her social contacts. Moreover, semantic relations between tags are found via a tag to tag(T2T) matrix. Semantic relations between tags are calculated with the following formula:

$$T2T[u_{t_i}][c_{t_j}] = \frac{P(u_{t_i}) \cap P(c_{t_j})}{P(u_{t_i})}$$

Thus the value of $T2T[u_{t_i}][c_{t_j}]$ is the proportion of the people with both tags i and j within the people with tag i. Users are represented with vectors of their personal and social tags. For each tag, its semantically related tags are selected in the T2T matrix and these tags are also added to the user profile. Their relationship degree affects the weight of the tag in the user vector. This method does not provide a good precision (about 2,7 percent with the last.fm dataset). However, it is good in accuracy.

Tags are user-annotated free texts. Users are free to enter tags whichever item he/she wants which causes an uncontrolled of tag redundancy. Using all these tags in vector spaces is mostly time-consuming. Moreover the vector space is sparse. Tags can be clustered according to their similarities and these clusters improve the personalized recommendations [36]. Clustering is one method to reduce the dimensionality, the other method is the usage of semantic relations between tags. This is done with some pre-defined ontologies in CHIP project [23]. This solves the cold-start problem in CB RSs. When there are a few items, recommender uses semantic relations to find similar items. Moreover, it helps the over-specialization problem. It sometimes finds interesting and surprising items by combining feature relations.

In addition to tags, web-mined texts are also used instead of tags. In MusicSun, a web-based similarity between artists is used in [41]. A query like ”<artist name >+ review” is entered on Google, top 50 pages are retrieved, each page is parsed into words and lastly each artist is represented as a word vector. Similar artists are found with vector distance techniques. [40] compares top ten tags of Last.fm and top ten web-mined texts for the music group Portishead and observes that web-mined texts are very unlikely in the music domain. However, this is a good solution for artists/bands which are not tagged in Last.fm-like specialized sites. [42] propagates the artist tags with the help of Wikipedia abstracts. It addresses the solution to cold-start problems of artists.

2.3.2.2 CB Approaches in Tag-based Music Recommendation

TB profiles are not used only in CF approaches, but also in CB approaches. [39] evaluates the performance of CB profiles in different domains (Last.fm and Delicious[46]). They propose 5 different CB recommendation algorithms:

1- TF-based similarity:

User's usage of tags appearing int he item profile is utilised. Scales are normalized.

$$g(u_m, i_n) = tf_u(u_m, i_n) = \frac{\sum_l tf_{u_m}(t_l)}{max(tf_u(t))}$$

$$g(u_m, i_n) = tf_i(u_m, i_n) = \frac{\sum_l tf_{i_n}(t_l)}{max(tf_i(t))}$$

2- TF-cosine based similarity:

Both user and item profiles are represented by a list of weighted items. Weights are calculated using Tf method. Cosine similarity measure is used to compute the similarity between user and item.

$$g(u_m, i_n) = cos_t f(u_m, i_n) = \frac{\sum_l tf_{u_m}(t_l) \cdot tf_{i_n}(t_l)}{\sqrt{\sum_l (tf_{u_m}(t_l))^2} \cdot \sqrt{\sum_l (tf_{i_n}(t_l))^2}}$$

3- TF-IDF cosine based similarity:

Both user an item profiles are represented by a list of weighted items. Weights are calculated using Tf-idf method. Cosine similarity measure is used to compute the similarity between user and item.

$$g(u_m, i_n) = cos_{tf-idf}(u_m, i_n) = \frac{\sum_l tf_{u_m}(t_l) \cdot iuf(t_l) \cdot tf_{i_n}(t_l) \cdot iif(t_l)}{\sqrt{\sum_l (tf_{u_m}(t_l) \cdot iuf(t_l))^2} \cdot \sqrt{\sum_l (tf_{i_n}(t_l) \cdot iif(t_l))^2}}$$

4- Okapi BM25 based similarity:

The similarity between user and item is calculated using Okapi-BM25 method.

$$g(u_m, i_n) = bm25_u(u_m, i_n) = \sum_l bm25_{u_m}(t_l)$$

$$g(u_m, i_n) = bm25_i(u_m, i_n) = \sum_l bm25_{i_n}(t_l)$$

5- Okapi BM25 cosine-based similarity:

Both user an item profiles are represented by a list of weighted items. Weights are calculated using Okapi-BM25 method. Cosine similarity measure is used to compute the similarity between user and item.

$$g(u_m, i_n) = cos_{bm25}(u_m, i_n) = \frac{\sum_l bm25_{u_m}(t_l) \cdot bm25_{i_n}(t_l)}{\sqrt{\sum_l (bm25_{u_m}(t_l))^2} \cdot \sqrt{\sum_l (bm25_{i_n}(t_l))^2}}$$

Okapi-BM25 is a probabilistic ranking method used in IR in search engines. The okapi-bm25 weighting function is given below [48].

$$Okapi - bm25(q_i) = \frac{(k_1 + 1) \cdot f(q_i, D) \cdot IDF(q_i)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})}$$

Where,

- q_i is the i^{th} term of query q.
- $f(q_i, D)$ is the frequency of term q_i in document D.
- $|D|$ is the length of document D.
- $avgdl$ is the average document length (in words) in the corpus.
- k_1 and b are the parameters that are usually chosen as $k_1 = 2.0$ and $b = 0.75$.
- $IDF(q_i)$is the Inverse Document Frequency of q_i.

$$IDF(q_i) = log(\frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5})$$

Where,

- $DF(q_i)$ is the number of documents containing q_i.
- N is the total number of documents.

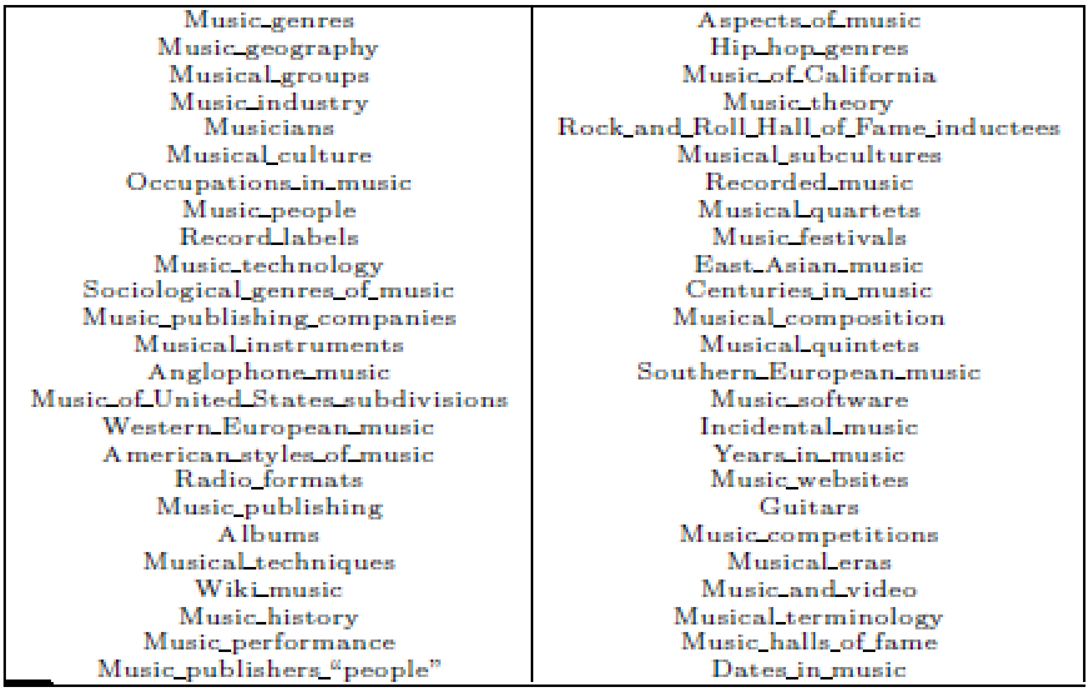
cos_{bm25} method outperforms cos_{tf-idf} method considering Last.fm data-set if CB approach is used[47].

In [49], heterogeneity in music recommendations is explained. Tags, track listenings and social contacts are said to be the source of information in music recommendations. Tags are proved to be the most valuable source of information. Tags provides effective coverage. On the other hand, friends provide diversity in recommendation and track listenings provide novelty.

Up to our knowledge, a study very similar to our study focuses on the top 50 music facets extracted from Wikipedia and assigns Last.fm tags to these facets [42]. The main object of study [42] is to provide an automatic method for uncovering the music facets and to classify tags according to these facets. The extracted music facets can be seen in the Figure 2.5. And the assigned tags to the facets can be shown in the Figure 2.6. The researchers of the study did not evaluate the usefulness of their approach in music recommendation. The just categorized the tags using Wikipedia.

2.4 Dimensionality Reduction in Recommendation Systems

As explained before, TBRSs are similar to the document-query structures in IR. Documents and queries are represented as a weighted list of terms in IR systems. Similarly, tracks and users are represented as a weighted list of tags in RSs. Thus, Vector Space Model is the standard representation for both document-term and track-tag representations. Text and image domains may have a large amount of attributes (terms or tags) which causes noise effects,



Music_genres	Aspects_of_music
Music_geography	Hip_hop_genres
Musical_groups	Music_of_California
Music_industry	Music_theory
Musicians	Rock_and_Roll_Hall_of_Fame_inductees
Musical_culture	Musical_subcultures
Occupations_in_music	Recorded_music
Music_people	Musical_quartets
Record_labels	Music_festivals
Music_technology	East_Asian_music
Sociological_genres_of_music	Centuries_in_music
Music_publishing_companies	Musical_composition
Musical_instruments	Musical_quintets
Anglophone_music	Southern_European_music
Music_of_United_States_subdivisions	Music_software
Western_European_music	Incidental_music
American_styles_of_music	Years_in_music
Radio_formats	Music_websites
Music_publishing	Guitars
Albums	Music_competitions
Musical_techniques	Musical_eras
Wiki_music	Music_and_video
Music_history	Musical_terminology
Music_performance	Music_halls_of_fame
Music_publishers_"people"	Dates_in_music

Figure 2.5: Top 50 Wikipedia music facets

ambiguities and redundancies. In order to remove the noise and reduce the complexity, some unsupervised dimensionality reduction methods are enhanced [52].

When the dimensionality of data increases;

1- performance of the query decreases,

2- time and space complexity increases,

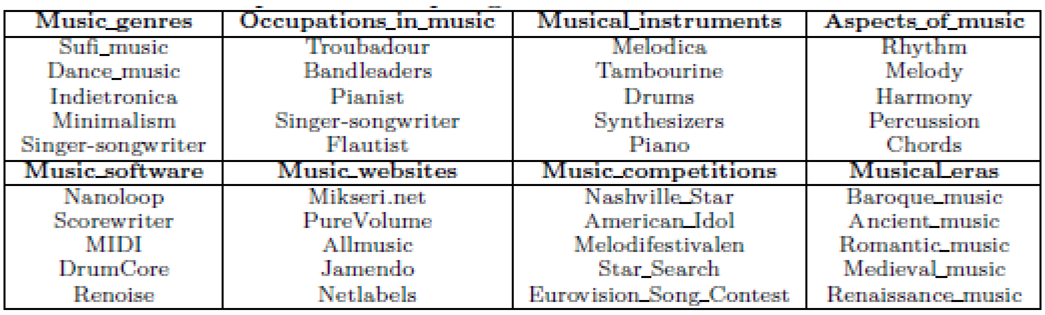
3- efficiency for data indexing decreases,

4- computation becomes slower,

5- noise increases, and

6- semantic gap problem arises.

Aim in the dimensionality reduction is to compute faster with a reasonable accuracy. LSI (Latent Semantic Indexing) is the oldest technique in the solution to this problem. However, alternative matrix decomposition methods are enhanced because of the time and space complexity of LSI.



Music_genres	Occupations_in_music	Musical_instruments	Aspects_of_music
Sufi_music	Troubadour	Melodica	Rhythm
Dance_music	Bandleaders	Tambourine	Melody
Indietronica	Pianist	Drums	Harmony
Minimalism	Singer-songwriter	Synthesizers	Percussion
Singer-songwriter	Flautist	Piano	Chords
Music_software	**Music_websites**	**Music_competitions**	**Musical_eras**
Nanoloop	Mikseri.net	Nashville_Star	Baroque_music
Scorewriter	PureVolume	American_Idol	Ancient_music
MIDI	Allmusic	Melodifestivalen	Romantic_music
DrumCore	Jamendo	Star_Search	Medieval_music
Renoise	Netlabels	Eurovision_Song_Contest	Renaissance_music

Figure 2.6: Top tags for some music facets

In [52], researchers evaluate 4 different dimensionality reduction methods in terms of complexity, approximation error and quality of the retrieval. The compared methods are explained below.

2.4.1 Latent Semantic Analysis

Latent Semantic Analysis is a common approach in document categorization and text summarization. In our case, we treat tracks as a list of tags just like that documents are a list of words. LSA is an intelligent technique that it finds the semantic relations between documents by using the information about the usage of words in the context. It finds the similar documents by knowing the common words between the documents. Basically, LSA uses a term-document matrix which has the counts of terms in each document as the matrix element. Most of the time, the matrix is sparse and in the columns there are documents and in the rows there are terms. The methods which can be used in the construction of the matrix may vary. Term Frequency, binary, TF-IDF, log-entropy, root type and modified tf-idf methods can be used in the matrix construction[19]. After the construction of term-document matrix, LSA finds a low-rank approximation in order to achieve its basic goals. Firstly, it reduces the dimensionality for easier computation. Secondly, it eliminates the anecdotal instances of terms in order to get a denoisified matrix. Lastly, synonymy is taken into account that similar words to the words of a document are added to the representation of the document. Besides; there are some limitations of LSA. It uses only its inputs (terms and documents); not the world knowledge. It does not use morphology and word order, either. Singular Value Decomposition (SVD) is the base of LSA. It models the relationship between terms and documents. In

SVD, the term-document matrix is thought to be created from 3 matrices. Mathematically,

$$A = U \cdot S \cdot V^T$$

Where,

- A is the $m * n$ term-document matrix.
- U is a $m * m$ unitary matrix.
- S is a $m * n$ diagonal matrix.
- V is a $n * n$ unitary matrix.

The diagonal entries (S_i) are known as the singular values of A. The m columns of U and the n columns of V are called the left singular vectors and right singular vectors of A, respectively. The approximated matrix A_k is the rank-k-approximation of the original matrix and is defined as

$$A_k = U_k \cdot S_k \cdot {V^T}_k$$

2.4.2 Non-Negative Matrix Factorization

Non-negative Matrix Factorization (NMF) is another method which is successful in reducing the dimensionality of matrices with non-negative components[52]. In the NMF, each document is represented as a combination of base topics and each axis in the space stores the base topic. For a matrix A with size $t \times d$, NMF factorizes A as

$$A = W \cdot H$$

Where, W is $t \times k$ and H is $k \times d$ and $k \leq d$. The columns of W contain the basis vectors and the columns of H contain the weights. NMF does not need to be orthogonal. Moreover, each document takes only non-negative values.

2.4.3 Independent Component Analysis

Independent Component Analysis (ICA) transforms the original matrix into its independent components [52]. ICA assumes that the original matrix is linear or non-linear mixtures of some latent variable with some coefficients. The latent variables are called the independent components of the matrix. The aim is to find the most independent components. For a matrix A of size $t \times d$ ICA decomposes A as

$$A = C \cdot F$$

Where C is the mixing matrix of size $t \times k$ and F is the matrix of independent components with size $k \times d$.

2.4.4 Fuzzy K-Means Clustering

The technique which uses clustering in matrix decompositions is called Concept Indexing [52]. Concept index represents the linear combinations of centroids of clusters. CI requires less memory than LSA. An improved version of the CI uses Fuzzy K-means algorithm. In FKM clustering, objects can belong to multiple clusters, and clusters are fuzzy sets.

In [52], 4 different datasets are used in order to compare the 4 methods. LSA performs the best in retrieving. Quality of its retrieval is the best. NMF and ICA methods gives poor results when compared with FKM and LSA. LSA is more complex than FKM in terms of time and space, but best approximation is achieved by LSA.

2.5 Motivation

Recommendation is a hot topic in both data mining and artificial intelligence areas. Most of the previous studies had focused on movie domain. Music is a a less studied domain for IR and recommendation. With the emergence of Internet radios like Last.fm, GrooveShark and Pandora, music recommendation gained more importance. There are some differences between movie recommendation and music recommendation:

1- People watch a movie in 1-2 hours whereas they listen to a music track in 5-6 minutes. Thus, people need more music recommendation than movie recommendation.

2- The amount of the items(tracks) in the music domain is much more than the amount of items(movies) in movie domain.

3- It is easy to keep history of listened to tracks with some software like Last.fm's Audioscrobbler; however there is not such a software for keeping a log of watched movies.

4- Movie recommendations mostly rely on ratings whereas music recommendations rely on listening habits.

5- False positives in music recommendations is not as annoying as false positives in movie recommendations. People does not mind when an irrelevant track is recommended, because it takes only 5 minutes. On the other hand, getting irrelevant movie recommendations annoys people, because it takes 2 hours.

6- People listen to music while travelling, coding, studying, reading, even sleeping. People go to concerts, join to festivals and watch live performances. Music is everywhere including movies. Movies are watched only once whereas soundtrack albums are listened more than once.

The history of musical interests of the members are kept well in Last.fm. Thus, the profiles on Last.fm constitute a good resource for user profiling in music recommendation. Last.fm does not provide its users a rating mechanism. However, it provides social tagging mechanism. Users can enter tags for artists, tracks and albums. Thus, rather than rating-based recommendation, TB recommendation is focused in this thesis study. The vector space model used in such systems is large in size. Moreover, it is sparse and noisy. Thus a dimensionality reduction method should be used in order to produce more efficient and faster recommendations.

In TB user profiling, tags which users entered for tracks are used. If a user is an active listener but does not label any tracks with tags; that user will not be able to get recommendations due to the lack of tags. Thus, this kind of users should be profiled by different methods other than using only user's own tags.

Facebook is another popular social networking site that people of any age have already registered. Everybody uses Facebook actively. A small research has been done on Facebook profiles of friends and it has been seen that most of the users had already specified their favourite movies, music bands, games, sports, books and TV programs on their Facebook profiles. Thus extending user profiles with their Facebook interests seems to improve the performance of the recommender.

CHAPTER 3

PROPOSED APPROACH

This chapter presents a novel dimensionality reduction method for large vector spaces. First, an overview of the system is presented. Then the data sources are explained. Data representation is described. Finally system architecture and each component is discussed in detail.

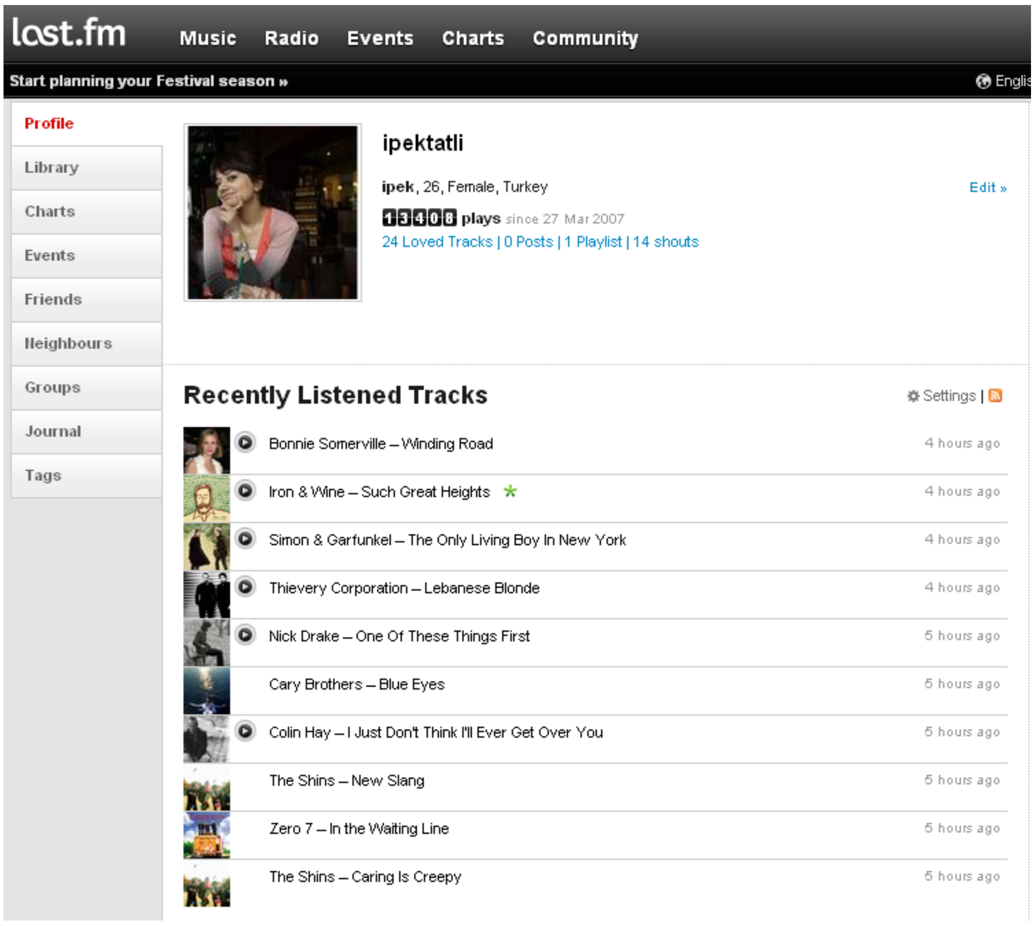
3.1 Overview

The task of our system is to recommend tracks to users considering their personal interests acquired from their Last.fm and Facebook profiles. Our system creates a music ontology with the help of Dbpedia. Then it uses this ontology in order to reduce the dimensionality of the VSM (track-tag, user-tag). It creates TB user and track profiles using 5 different user profiling methods. It generates rules from Facebook profiles (rules like ”if a person like the movie ’Godfather’, he/she may also like the band ’Pink Floyd’”). Then it extends user profiles by inferring the rules generated from Facebook. Finally it generates personal recommendations.

3.2 Data Sources

3.2.1 Last.fm, Music Recommendation Engine

Last.fm is founded in U.K, in 2002 [44]. Today it has about 40 million of users from about 190 countries. Last.fm provides an Audioscrobbler to keep the history of songs users listened to. Moreover, it provides applications for iphone, ipad and some other devices.



ipektatli

ipek, 26, Female, Turkey

13408 plays since 27 Mar 2007

24 Loved Tracks | 0 Posts | 1 Playlist | 14 shouts

Recently Listened Tracks

Bonnie Somerville – Winding Road	4 hours ago
Iron & Wine – Such Great Heights	4 hours ago
Simon & Garfunkel – The Only Living Boy In New York	4 hours ago
Thievery Corporation – Lebanese Blonde	4 hours ago
Nick Drake – One Of These Things First	5 hours ago
Cary Brothers – Blue Eyes	5 hours ago
Colin Hay – I Just Don't Think I'll Ever Get Over You	5 hours ago
The Shins – New Slang	5 hours ago
Zero 7 – In the Waiting Line	5 hours ago
The Shins – Caring Is Creepy	5 hours ago

Figure 3.1: User profile on Last.fm

As it can be seen in Figure 3.1; there are mainly profile, library, charts, events, friends, neighbours, groups, journal and tags tabs on a Last.fm user page.

On **profile tab**; recently listened tracks, recent activities, user's archive, popular tracks, popular artists are seen.

On **library tab**; users can see their most listened artists, loved tracks and created playlists.

On **Charts tab**; statistics about listened tracks, albums and artists are shown.

On **Events tab**; users can see their joined events and friends' events.

On **Friends tab**; friends and friend requests are seen.

On **Neighbours tab**; weekly calculated neighbours are shown. Users can listen to their neighbour's radio stations.

On **Groups tab**; joined and created groups are seen.

On **Journal tab**; users can write diaries and use their Last.fm profiles as blogs.

On **Tags tab**; user can see his/her tags.

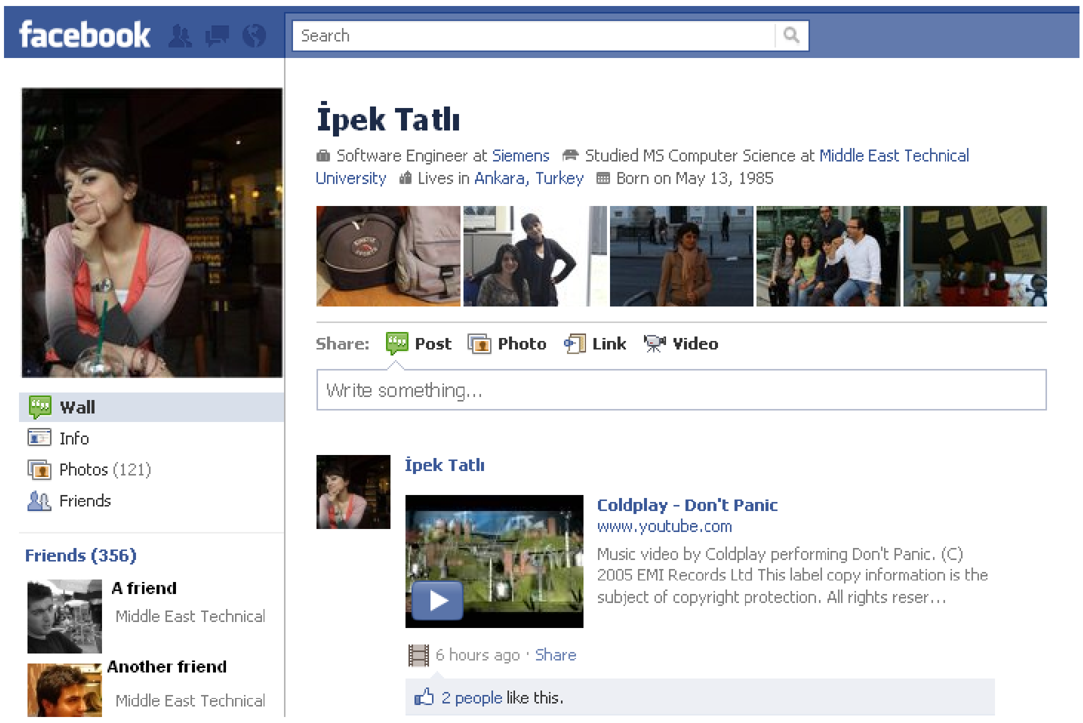
Last.fm is a social networking site as well as being the most popular Internet radio. Users can add friends. They can leave notes (shout mechanism on Last.fm) on their friend's profiles. They can send private messages to each other. They can join to groups or events. Moreover, they are involved in the content-generation like adding photographs to fan pages or entering tags for artists/albums/songs. This process is called social tagging.

When a user is registered to Last.fm, a musical profile is created either by his/her tracks on the computer with Audioscrobbler or the tracks that are listened on the Internet radio. Audioscrobbler keeps a log of listened tracks. This process is called scrobbling. Scrobbling helps Last.fm to calculate similar users and users can see their neighbours on their own profiles. Neighbours are being calculated every week on Sundays. In addition to neighbour recommendation, Last.fm recommends artists that individuals may like. Users can see their recommendations on their own profiles.

Last.fm does not explain its recommendation method. However it is obvious that Last.fm uses CF methods. It uses user-based CF for neighbour recommendation and item-based CF for artist recommendation. It enriches the CF methods with social tagging mechanism. In other words meta data, audio information and tags are used in Last.fm's CF algorithm. Moreover, user profiles are enriched with the tags they entered and the comments they made on the friends', artists', albums' and tracks' profiles. Evaluating is done with "love/ban/skip" mechanism.

3.2.2 Facebook, Social Network

Facebook is one of the most popular social networking sites. It was founded in 2004 by Mark Zuckerberg from Harvard University. It was thought to be used by the students of only Harvard University. Then it was expanded to all over the world. Today, Facebook has more than 750 million users [45].


facebook

Search

İpek Tatlı

Software Engineer at Siemens · Studied MS Computer Science at Middle East Technical University · Lives in Ankara, Turkey · Born on May 13, 1985

Share: Post Photo Link Video

Write something...

Wall
Info
Photos (121)
Friends

Friends (356)

A friend
Middle East Technical

Another friend
Middle East Technical

İpek Tatlı

Coldplay - Don't Panic
www.youtube.com
Music video by Coldplay performing Don't Panic. (C) 2005 EMI Records Ltd This label copy information is the subject of copyright protection. All rights reser...

6 hours ago · Share

2 people like this.

Figure 3.2: User profile on Facebook

Users add friends, send private messages, join to groups, add photos, share videos and write comments on Facebook. The inspiring part of Facebook for my study is that users specify their interests (favourite movies, favourite music bands, favourite books, favourite TV programs) on their profiles. It is an advantage for a recommender to know what people like in different domains.

3.2.3 Wikipedia, Dbpedia

Wikipedia was founded in 2001 by Jimmy Wales and Larry Sanger [63]. Wikipedia was just the next generationof Nupedia which was a free online English encyclopedia project. Articles in Nupedia had been written by experts and they had been reviewed under a formal process. First contributors of Wikipedia were from Nupedia, Slashdot postings, and web search engine indexing. Wikipedia had been having about 20000 articles in 18 different languages at the end of 2001. By 2002, it had been having 26 language editions, 46 by the end of 2003, and 161 by the final days of 2004. English Wikipedia had been having about two million articles in 2007 which makes it the largest encyclopedia.

Wikipedia is a free, web-based, multilingual encyclopaedia. For the data mining tasks, it is not appropriate. Instead of its unstructured content, Dbpedia can be used. Dbpedia was started by Free University of Berlin, the University of Leipzig and OpenLink Software [64]. The first datasets were published in 2007. There are more than 3,5 million entries in Dbpedia and half of them belong to a consistent ontology with links to related data sets.

3.3 Design Issues

3.3.1 Domain Description

The main aim of this study is to reduce the dimensionality of the vector space in our system. We use the semantic relations between the tags in the music domain. In order to create a well-organized semantic relations structure, we use Dbpedia information of our mainstream music genres.

We use the music domain in order to apply the proposed approach in a CB system. Tracks are appropriate for the TBRSs because their top tags can be gathered through Last.fm API.

Moreover; in order to find the relation between movies and artists, we use users' interests (favourite movies, TV shows and music bands) acquired from their Facebook profiles. Thus, we extend the user profiles with the artists that users may like according to their favourite movies.

3.3.2 Data Representation

Data used in our RS is crawled from Last.fm, Dbpedia and Facebook.

1- Track, artist, tag, singer and mainstream genre information and the mapping between each are collected from Last.fm.

2- For each mainstream genre specified in Last.fm; ontology classes (instrumentation, fusion genres, subgenres, derivative forms and stylistic origins) and the mapping between mainstream genre and the classes are gathered from Dbpedia.

3- Movie, TV-program and music band information for each user are collected from Facebook.

More details about the system database tables are shown in Table 3.1 and Table 3.2.

3.4 System Architecture and Components

System functionality is achieved by six main components. Processes performed in our system can be seen in Figure 3.4. Circles are the phases performed.

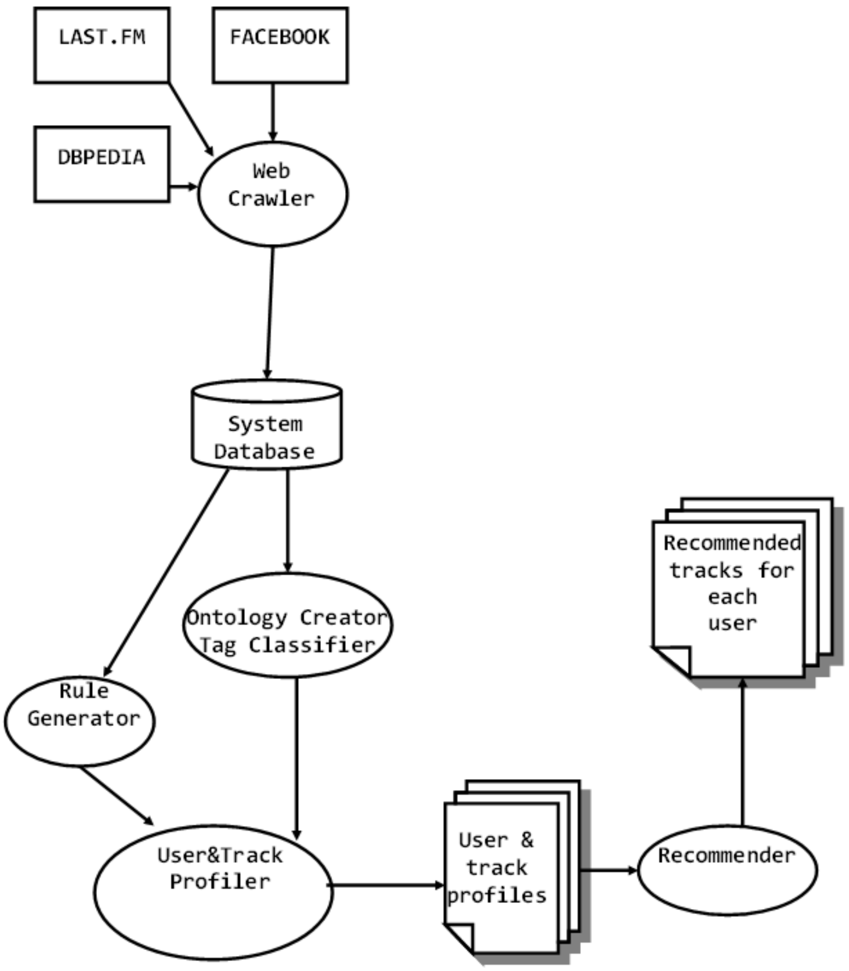


Figure 3.3: System architecture

1. Web Crawler: Crawls Last.fm and Facebook profiles and inserts the extracted data to database tables.

Table 3.1: System Database

Table	**Columns**	**Description**
users	user_id, user_lastfm_name	Keeps user data
tracks	track_id, track_name	Keeps track data
singers	singer_id, singer_name	Keeps singer data
tags	tag_id, tag_name, tag_count	Keeps tag data
track_singer	track_id, singer_id	Keeps singers of tracks
track_tag	track_id, tag_id	Keeps tags of tracks
singer_tag	singer_id, tag_id	Keeps tags of singers
user_track	user_id, track_id	Keeps tracks of users
user_track_tags	user_id, track_id, tag_id, tag_count	Keeps users own tags
friend_track_tags	user_id, track_id, tag_id, tag_count	Keeps users friends tags
mainstream_genres	genre_id, genre_name	Keeps mainstream genres
instrumentation_class	instrument_id, instrument_name	Keeps instrumentation data
subgenre_class	subgenre_id, subgenre_name	Keeps subgenre data
fusiongenre_class	fusiongenre_id, fusiongenre_name	Keeps fusion genre data
derivativeForm_class	derForm_id, derForm_name	Keeps derivative form data
stylisticOrigins_class	stylisticOrigins_id, stylOrig_name	Keeps stylistic origin data
genre_instrument	genre_id, instrument_id	Keeps instruments of genre
genre_subgenre	genre_id, subGenre_id	Keeps subgenres of genre
genre_fusionGenre	genre_id, fusionGenre_id	Keeps fusion genres of genre
genre_derForm	genre_id, derForm_id	derivative forms of genre
genre_stylisticOrg	genre_id, stylisticOrg_id	stylistic origins of genre
tag_genre	tag_id, genre_id	mapping between tag and genres
tag_instrument	tag_id, instrument_id	mapping between tag and instruments
tag_subgenre	tag_id, subGenre_id	mapping between tag and subgenres
tag_fusionGenre	tag_id, fusionGenre_id	mapping between tag and fusion genres
tag_derForm	tag_id, derForm_id	mapping between tag and derivative forms
tag_stylisticOrg	tag_id, stylisticOrg_id	mapping between tag and stylistic origins

Table 3.2: System Database - continued

Table	**Columns**	**Description**
FB_movies	movie_id, movie_name	Keeps movie and TV-program data
FB_artists	artist_id, artist_name	Keeps artist data
FB_user_movie	user_id, movie_id	Keeps favourite movies and TV-programs of users
FB_user_artist	user_id, artist_id	Keeps favourite artists of users
FB_rules	movie_id, artist_id	Keeps the mapping between movies and artists

2. Ontology Creator: Crawls the Dbpedia pages of music genres and creates a hierarchical structure.

3. Tag classifier: Classifies tags using the hierarchical structure.

4. Track Profiler: Creates track-tag vectors.

5. User Profiler: Creates user-tag vectors.

6. Recommendation Generator: Calculates the relevant tracks for users.

3.4.1 Web Crawler

Users listen to music and enter tags for tracks in their Last.fm profiles. In the web crawling phase of the system, our data set is generated. Details of the data set are given in Chapter 4.

Real Last.fm data is used in this thesis study. In order not to use similar users and in order to achieve diversity, 50 users are selected from an application named ”join Last.fm”[58]. In this group, members of the group share their Last.fm nicknames. Moreover, 19 friends who use both Last.fm and Facebook are also added to the data set. Last.fm profiles of the users in the data set are crawled with the help of Last.fm API [50]. This number of users is acceptable for content-based approaches.

Firstly for each user, 300 top tracks are gathered. Top tracks are stored in *tracks* table in the database.

$$APImethod : user.gettoptracks$$

Then their 'loved' tracks are extracted. Loved tracks are also stored in *tracks* table in the database.

$$APImethod : user.getlovedtracks$$

For each track, the artist names, tags and tag counts are crawled. Artists are stored in *singers* table in the database, the mapping between artist and track is kept in *track_singer* table. Tags and tag counts are kept in *tags* table. Mapping between track and tag is stored in *track_tag* table. Mapping between singer and tag is stored in *singer_tag* matrix.

$$method = track.gettoptags$$

Finally for each user their tags entered for tracks and their friends' tags entered for tracks are gathered. Users' own tags are kept in $user - track - tag$ table, and users' friends' tags are kept in *friend_track_tag* table.

$$APImethod : user.getfriends$$

$$APImethod : user.gettoptags$$

$$APImethod : user.getpersonaltags(fortracks)$$

In addition to Last.fm profiles, Facebook profiles of the users are also crawled using Facebook Graph API. Their favourite movies, tv-series and music artists are gathered. Movies and tv-series are kept in *FB_movies* table, and music bands are saved in *FB_artists* table. The mapping between 400 users and movies/tv-series are stored in *FB_user_movie*. The mapping between users and artists are kept in *FB_user_artist* table. Generated rules are stored in *FB_rules* table.

3.4.2 Ontology Creator

Tags may be about genre, location, moods, personal opinions, instrumentation, styles and artists' names [6]. The distribution of tags in Last.fm can be seen in Table 3.3. For example, two users of Last.fm tagged some songs as follows: the first one loved listening to "The Wall" from "Pink Floyd" and tagged the track with the words "energetic" and "seen live". The

Table 3.3: Distribution of Tag in Last.fm

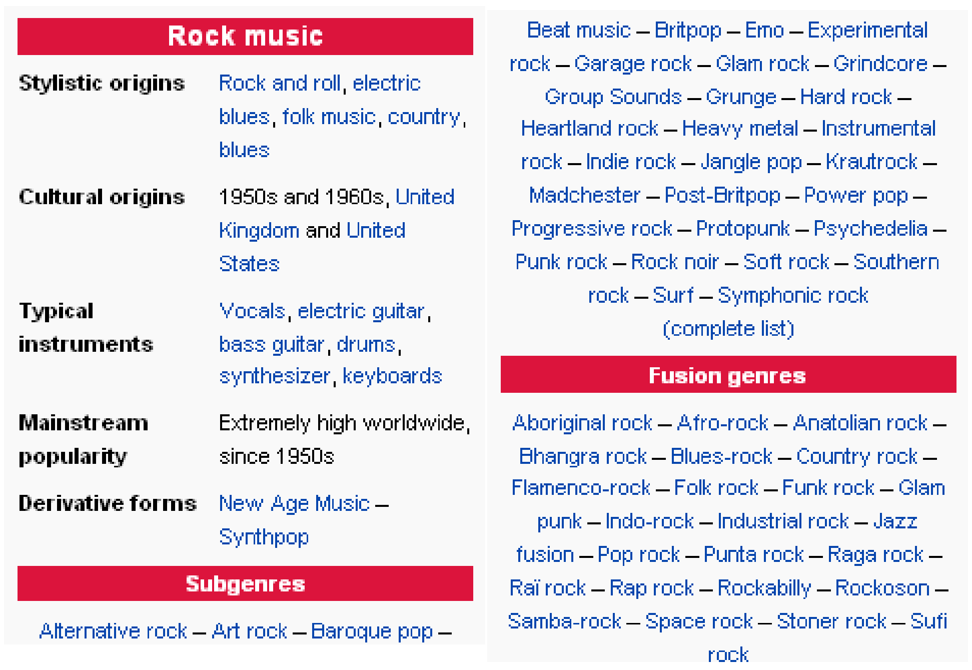
Type	**Frquency**	**Examples**
Genre	68%	Heavy metal, punk
Locale	12%	French, Seattle
Mood	5%	Chill, party
Opinion	4%	Love, favorite
Instrumentation	4%	Piano, female vocal
Style	3%	Political, humor
Misc	3%	Coldplay, composers
Personal	1%	Seen live, I own it

second one loved ”Only Girl” from ”Rihanna” and tagged ”Only Girl” also with the words ”energetic” and ”seen live”. Thus both ”Only Girl” and ”The Wall” have the same tags. According to the recommendation’s similarity function, they appear as very similar tracks, although in most other ways (genre, instrumentation for instance) they are not. Because of such reasons, subjective tags like personal opinions and moods are ignored in the track and user representation. Genre, instrumentation, sub-genre, fusion genre, derivative form and style are used in the track and user representation. In [40], 14 mainstream genres (country, folk, jazz, blues, r and b, heavy metal, alternative/indie, punk, rap, electro, reggae, classical, rock and pop) are used. These genres are enriched with Last.fm’s mainstream genres, which can be reached on the left frame of the page http://www.last.fm/music. The main genres used in the system are as follows:

acoustic, ambient, blues, classical, country, electronic, emo, folk, hardcore, hip hop, indie, jazz, Latin, metal, pop, pop punk, punk, reggae, r and b, rock, soul, world.

The mainstream genres are kept in ”mainstream_genres” table in the database. Having identified the mainstream genres, for each main genre Wikipedia is crawled but then Dbpedia is crawled since it is more structured for web-crawling. The obtained information for ”rock music” is illustrated in Figure 3.5.

In ontology-creation phase, we created an ontology-like structure with the help of the data crawled from the Dbpedia. Classes of the ontology can be seen in Table 3.4. Instance samples are illustrated in Table 3.5. Lastly, relations in the ontology can be seen in Table 3.6. The



Rock music

Stylistic origins	Rock and roll, electric blues, folk music, country, blues
Cultural origins	1950s and 1960s, United Kingdom and United States
Typical instruments	Vocals, electric guitar, bass guitar, drums, synthesizer, keyboards
Mainstream popularity	Extremely high worldwide, since 1950s
Derivative forms	New Age Music – Synthpop

Subgenres

Alternative rock – Art rock – Baroque pop – Beat music – Britpop – Emo – Experimental rock – Garage rock – Glam rock – Grindcore – Group Sounds – Grunge – Hard rock – Heartland rock – Heavy metal – Instrumental rock – Indie rock – Jangle pop – Krautrock – Madchester – Post-Britpop – Power pop – Progressive rock – Protopunk – Psychedelia – Punk rock – Rock noir – Soft rock – Southern rock – Surf – Symphonic rock (complete list)

Fusion genres

Aboriginal rock – Afro-rock – Anatolian rock – Bhangra rock – Blues-rock – Country rock – Flamenco-rock – Folk rock – Funk rock – Glam punk – Indo-rock – Industrial rock – Jazz fusion – Pop rock – Punta rock – Raga rock – Raï rock – Rap rock – Rockabilly – Rockoson – Samba-rock – Space rock – Stoner rock – Sufi rock

Figure 3.4: Rock music on Wikipedia

complete ontology for the ”rock genre” can be found in **Appendix B**.

Instrument classes are kept in *instrument_class* table, subgenre classes are kept in *sbgnre_clss* table, fusion genre classes are kept in *fusionGenre_class* table, derivative form classes are kept in *derivativeFrom_class* table and stylistic origins are kept in *stylisticOrigin_class* table in the database. The corresponding mappings are kept in *genre_instrument*, *genre_subgenre*, *genre_fusionGenre*, *genre_DerForm* and *genre_StylisticOrg* tables.

Table 3.4: Classes in the ontology

Class name
Genre
StylisticOrigin
Instrument
DerivativeForm
SubGenre
FusionGenre

Table 3.5: Instance samples in the ontology

Classes	**Instances**
StylisticOrigin	Rock and roll, electric blues, folk music, country, blues, rhythm and blues, soul music
Instrument	Vocals, electric guitar, bass guitar, acoustic guitar, drums, synthesizer, keyboards
DerivativeForm	New Age Music, Synthpop
SubGenre	Alternative rock ,Art rock ,Baroque pop ,Beat music ,Britpop ,Emo ,Experimental rock ,Garage rock ,Glam rock ,Grindcore ,Group Sounds ,Grunge ,Hard rock ,Heartland rock ,Heavy metal ,Instrumental rock ,Indie rock ,Jangle pop ,Krautrock ,Madchester ,Post-Britpop ,Power pop ,Progressive rock ,Protopunk ,Psychedelia ,Punk rock ,Rock noir ,Soft rock ,Southern rock ,Surf ,Symphonic rock
FusionGenre	Aboriginal rock ,Afro-rock ,Anatolian rock ,Bhangra rock ,Blues-rock ,Country rock ,Flamenco-rock ,Folk rock ,Funk rock ,Glam punk ,Indo-rock ,Industrial rock ,Jazz fusion ,Pop rock ,Punta rock ,Raga rock ,RaÃ¯ rock ,Rap rock ,Rockabilly ,Rockoson ,Samba-rock ,Space rock ,Stoner rock ,Sufi rock

Table 3.6: Relations in the ontology

hasStylisticOrigins	Genre and Stylistic Origins
hasInstruments	Genre and TypicalInstrumentation
hasDerivativeForms	Genre and Derivative Forms
hasSubGenres	Genre and Sub Genres
hasFusionGenres	Genre and Fusion Genres

3.4.3 Tag Classifier

The reason why we classify the tags is to reach their mappings faster. In order not to differentiate ”electronic” from ”electronica”, a stemming algorithm is applied to tags. Stemming is a technique to convert similar words to a common base form. This base form does not have to have a meaning from a linguistic point of view (such as reducing synonyms to a single word, or finding the root of the word). Various stemming algorithms exist for the English language. The Porter Stemmer[51] which is a classical stemming algorithm is used in this study. By using a stemming algorithm, morphology, which LSA does not use, is taken into consideration in our approach. Below is the pseudo code of the tag classifying algorithm:

```
for $i$ = 1 to number_of_instances_in_ontology do
  $instance$ ← getinstance_at_position(i)
  instance = trim(instance)
  word_array = split(” ”,instance)
  $new\_instance$ = ”%”
  for $j$ = 1 to size_of_word_array do
    $word$ ← getword_at_position(j)
    word = stem(word)
    new_instance = concatenate (new_instance, word, ”%”)
  end for
  for $k$ = 1 to size_of_tags_in_dataset do
    if is_like(new_instance) then
      insert_into_tag_instance_table($tag\_id_k$, $instance\_id_i$ )
    end if
  end for
end for
```

In the algorithm above, each instance in the ontology is parsed into single words. The stemming algorithm is applied to each single word. The stemmed roots are concatenated with ”%” in order to consider ”morphology” that LSA does not use. LSA applied on document-term matrices does not consider ”morphology”. Some examples of the stemming results can be seen in Table 3.7. All the tags in our dataset are saved in the *tags* table. The reason why we use ”%” in the new version of instances is that we use the new versions of the instances

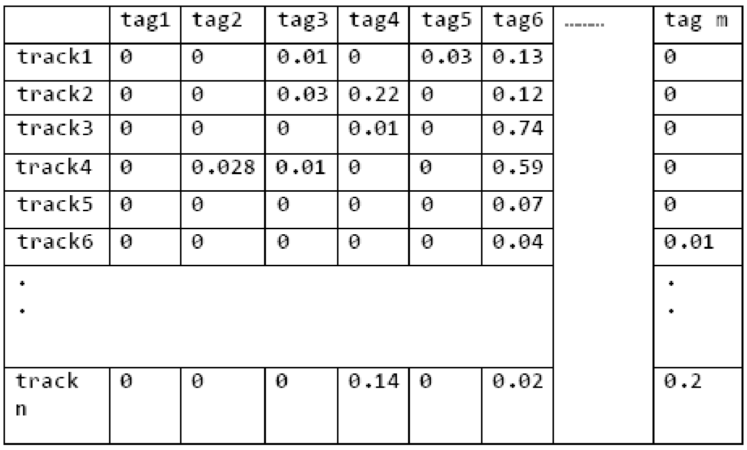
Table 3.7: Concatenating the stemmed words of the instances

Tag before stemming	Tag after stemming
electric blues	%eletr%blu%
Aboriginal rock	%Aborigin%rock%

in our SQL statements. We use the newer instances in SQL statements like "select * from tags where tag_name like '%Aborigin%rock%' ". With this usage, we are using about 80000 tags out of 160000 tags in track representation. The corresponding mappings between tags and instances are kept in *tag_instrument*, *tag_subgenre*, *tag_fusionGenre*, *tag_DerForm* and *tag_StylisticOrg* tables.

Each term in the document-term matrix(used by LSA) is a single word whereas each term in the track-tag matrix may be a group of words (like "electro blues") mapped to a genre.

3.4.4 Track Profiler



	tag1	tag2	tag3	tag4	tag5	tag6	....	tag m
track1	0	0	0.01	0	0.03	0.13		0
track2	0	0	0.03	0.22	0	0.12		0
track3	0	0	0	0.01	0	0.74		0
track4	0	0.028	0.01	0	0	0.59		0
track5	0	0	0	0	0	0.07		0
track6	0	0	0	0	0	0.04		0.01
. .								. .
track n	0	0	0	0.14	0	0.02		0.2

Figure 3.5: Track-tag matrix

In track profiling phase, size of a track vector is the size of mainstream genres (22 in our case). Last.fm provides integer percentages (between 0 and 100) relative to the most used tags per track. We updated these percentages by adding 1 to each percentage value in order not to discard any having 0 percentage. Each entry in the vector is calculated as follows:

$$Term - Count(g(i,j)) = \sum_k hasInstrumentation(i,j) + \sum_k hasStylisticOrigins(i,j) + \sum_k hasDerivativeForms(i,j) + \sum_k hasSubGenres(i,j) + \sum_k hasFusionGenres(i,j)$$

Where,

- $g(i,j)$ is the i^{th} term (genre) in j^{th} track.
- $\sum_k hasInstrumentation(i,j)$ is the total percentage (between 1 and 101) of the tags of the j^{th} track which are found to be similar to the new instance versions of the instrumentation class of i^{th} genre (with the help of the aforementioned SQL statements).

The rest of the formula is constituted in the same logical way. The calculated term counts are used in the term-weighting. There are several ways of term weighting in IR systems. In VSMs, terms are weighted in contrast to Boolean Model spaces. It provides partial matching so that it provides better matching. In this model, term weights are calculated by several methods like TF*IDF and Okapi-bm25 [53]. In this thesis, TF*IDF and Okapi-bm25 are used in the track and user profiling.

Tag counts are usually normalized to prevent a bias towards longer documents. Term Frequency value gives local information of a tag. An inverse document frequency value is calculated for each different tag in training set. This is calculated by diving total number of tracks by the number of tracks that reference to that feature. IDF value gives global information of a tag. The intuition behind this weighting schema is to assign higher weights to tags that occur frequently in a document (high TF), and rarely in the data set (high IDF). This prevents the effect of assigning high weights to tags that are most frequent in the data set.

Okapi bm-25 is yet another term-weighting method used in IR. It has been shown to perform well in search tasks over web data so that BM stands for "best match". BM25 is the combination of BM11 and BM15 which are one of the oldest weighting schemas[54]. BM25 is proved to be successful in content-based recommendations[49]. It uses TF and IDF in the core. In addition to TF and IDF, there are some constants and track length normalisation. When a track profile is dense, and it may be selected as similar to many of the users by default. Thus, length of the tracks are normalized in this function. Briefly, okapi-bm25 is a combined version of TF-IDF method.

Tracks in our dataset are represented as a weighted list of genres. Semantically related tags are counted as the same genre in this representation. The weights of the genres are calculated with TF*IDF and Okapi-bm25. Formulations are given below:

$$TF(q_i) = f(q_i, t)$$

$$IDF(q_i) = log(\frac{T}{|t : q_i \in t|})$$

$$Weight(q_i) = TF(q_i) \times IDF(q_i)$$

Where,

- q_i is the $i^t h$ genre in the vector.
- $f(q_i, t)$ is the frequency of genre q_i in track t.
- T is the total number of tracks.

The okapi-bm25 weighting function is given below[55]:

$$Okapi - bm25(q_i) = \frac{(k_1 + 1) \cdot f(q_i, t) \cdot IDF(q_i)}{f(q_i, t) + k_1 \cdot (1 - b + b \cdot \frac{|t|}{avgtl})}$$

Where,

- q_i is the i^{th} genre in the vector.
- $f(q_i, t)$ is the frequency of genre q_i in track t.
- $|t|$ is the length of track t.
- $avgtl$ is the average track length (in words) in the corpus.

- k_1 and b are the parameters that are usually chosen as $k_1 = 2.0$ and $b = 0.75$.
- $IDF(q_i)$is the Inverse Document Frequency of q_i.

$$IDF(q_i) = log(\frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5})$$

Where,

- $DF(q_i)$ is the number of tracks containing q_i.
- N is the total number of tracks.

3.4.5 User Profiler

In the user-profiling phase, classical recommenders use tags that users assign to tracks. Most of the time, users do not enter tags for the tracks. Thus a very sparse user-tag matrix is created which results in poor recommendations. In order to handle this problem, we create user-tag matrix in 5 different ways. User-tag matrices are created by

1- using the users' own tags (personal tags) that they entered for tracks (Figure 3.6),

2- using the users' friends' tags (friends' tags) that friends entered for users' tracks (Figure 3.7),

3- using all the tags of the tracks (social tags) that they listened to (Figure 3.8),

4- using all the tags of the artists of tracks that they listened to (Figure 3.9),

5- using all the tags of the artists of tracks that they listened to and all the tags of the artists that are inferred from the rules generated from their Facebook profiles (Figure 3.10).

In the first method, users are profiled with their own tags which they entered for tracks. In the second method, users are profiled with their friends' tags which their friends entered for tracks the user listened to. Semantic relations are used in user profiling method-1 and method-2, just the same as in track profiling.

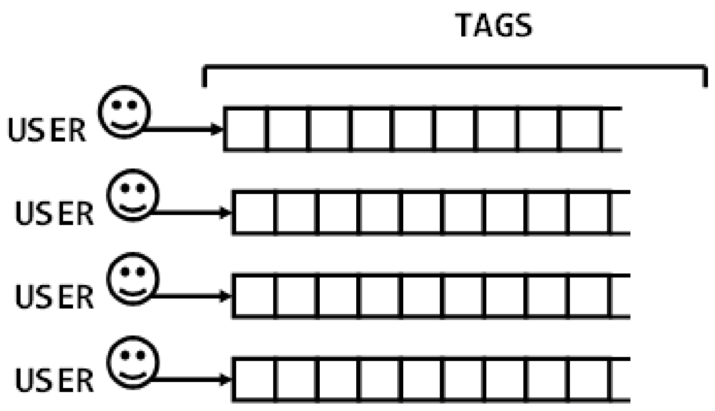


Figure 3.6: User profiling with user's own tags that user entered for tracks

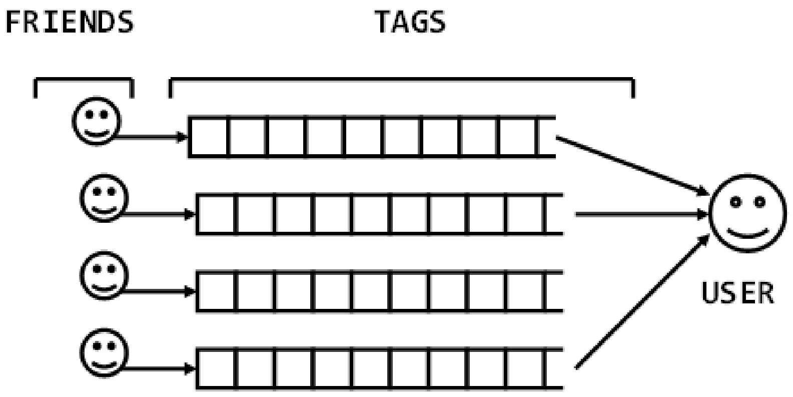


Figure 3.7: User profiling with user's friends' tags that they entered for tracks user listened to

The main problem in the user-profiling is that most of the users do not enter tags for the artists/albums or tracks they listened to. Because of the lack of the personal tags, method-3, method-4 and method-5 are used in user profiling. In the third method, users are profiled with all the tags of the tracks that they listened to. As explained before, tracks are profiled by the use of semantic relations. Thus the user vectors are calculated as follows:

$$U_k = \sum_j Track\_profiles(k, j)$$

Where $\sum_j Track\_profiles(k, j)$ is the sum of the track profiles that the user k listened to.

In the forth method, users are profiled with all the tags of the artists of the tracks that they

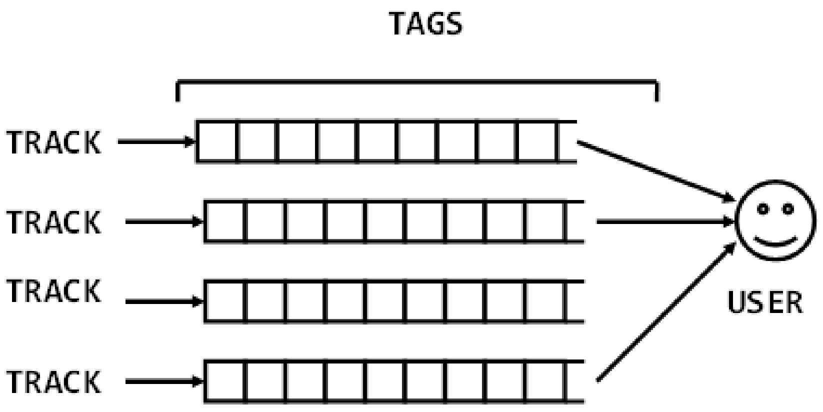


Figure 3.8: User profiling with the tags of all the tracks user listened to

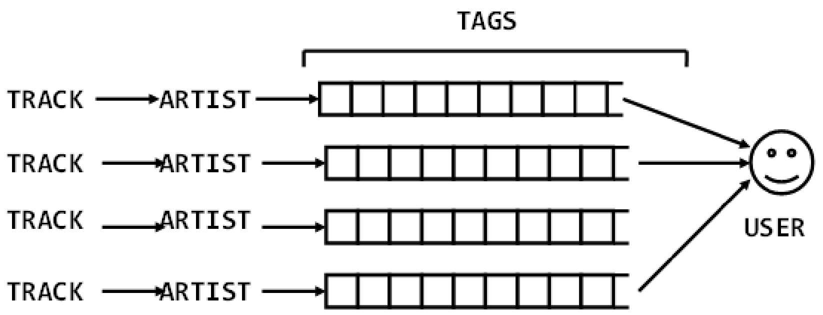


Figure 3.9: User profiling with the tags of artists of all the tracks user listened to

listened to. For this representation, artists are profiled just the same as tracks. So the users are profiled as follows:

$$U_k = \sum_j Artist\_profiles(k, j)$$

Where $\sum_j Artist\_profiles(k, j)$ is the sum of the artist profiles that the user k listened to.

In order to take the advantage of multi-domain information, we have generated rules like ”if a person like the movie ’Godfather’, he/she may also like the band ’Pink Floyd’” using about 400 Facebook profiles. Facebook provides a Graph API which helps developers to get any information about the people in the developers’ friend lists. We gathered the favourite movies, TV-programs and music bands of the people in our friend lists and the people in the dataset used in the scope of this study.

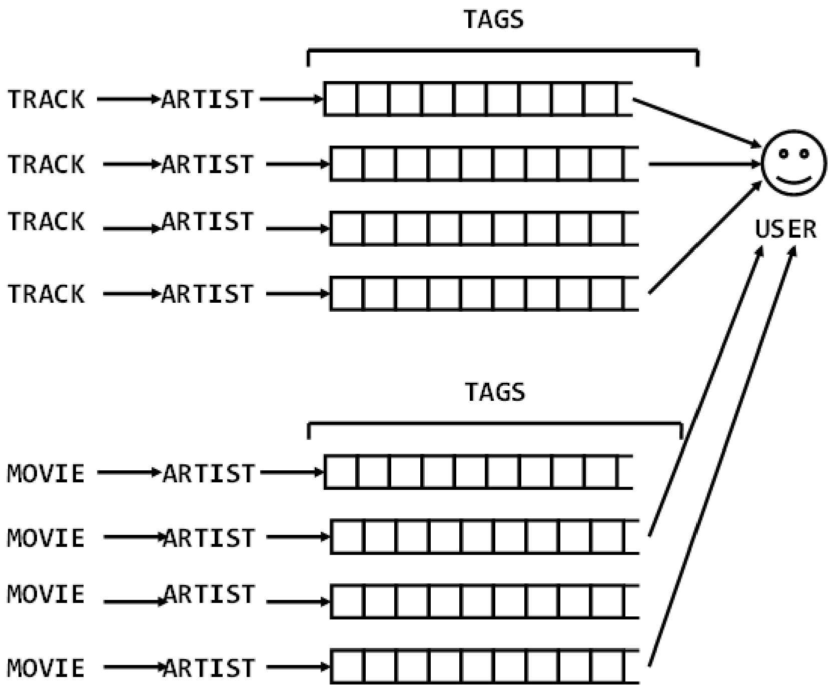


Figure 3.10: User profiling with the tags of extended set of artists

Some of the rules can be seen below:

- The users who loves ”Fight Club” most probably also like ”Radiohead”.
- The users who loves ”Fight Club” most probably also like ”Muse”.
- The users who loves ”Eternal Sunshine of the Spotless Mind Red” most probably also like ”Hot Chili Peppers”.
- The users who loves ”Back to the Future Trilogy” most probably also like ”Queen”.
- The users who loves ”Slumdog Millionaire” most probably also like ”Farid Farjad”.

In the fifth method, artist set of each user is extended with the artists which are inferred from the rules generated from the Facebook profiles. In addition to the artists that the user listened to, the possible artists inferred from the rules are also used in the creation of the user profiles. Users are profiled as follows:

$$U_k = \sum_j Extended\_Artist\_profiles(k, j)$$

Where $\sum_{j} Extended\_Artist\_profiles(k, j)$ is the sum of the artist profiles that the user k likes and may like according to the generated rules.

In the user vectors, the weights of the tags are also calculated with TF*IDF and Okapi-bm25. TF is calculated for each user and the IDF, which has been calculated in the track profiling, is used as the IDF value. The main goal after creating a user profile from the training set, is to recommend the items in the test set is the main goal.

3.4.6 Recommendation Generator

In the recommendation phase, the common cosine similarity with TF-IDF and cosine similarity with Okapi bm25 are used because of their success in CB recommenders. The most similar items to the user profile are found. The cosine similarity formula is given below:

$$Cos\_sim = \frac{user\_vector \cdot track\_vector}{|user\_vector||track\_vector|}$$

Jaccard index, correlation, Manhattan distance and hamming distance are some of the other methods available for finding the similarities, but we find out that cosine similarity gave the best results in our study, in producing items that were more similar to those of the user profile.

3.5 Implementation Details

Music ontology has been created at the beginning of the study because Java would have been used. Then we switched to PHP because of its simplicity and faster implementation. Thus, LAMP (Linux, Apache, Mysql, Php) is used in the web-crawling. The combined components of LAMP are as follows:

1- Linux is an open-source operating system,

2- Apache is a web server,

3- MySQL is a database,

4- PHP is a scripting language.

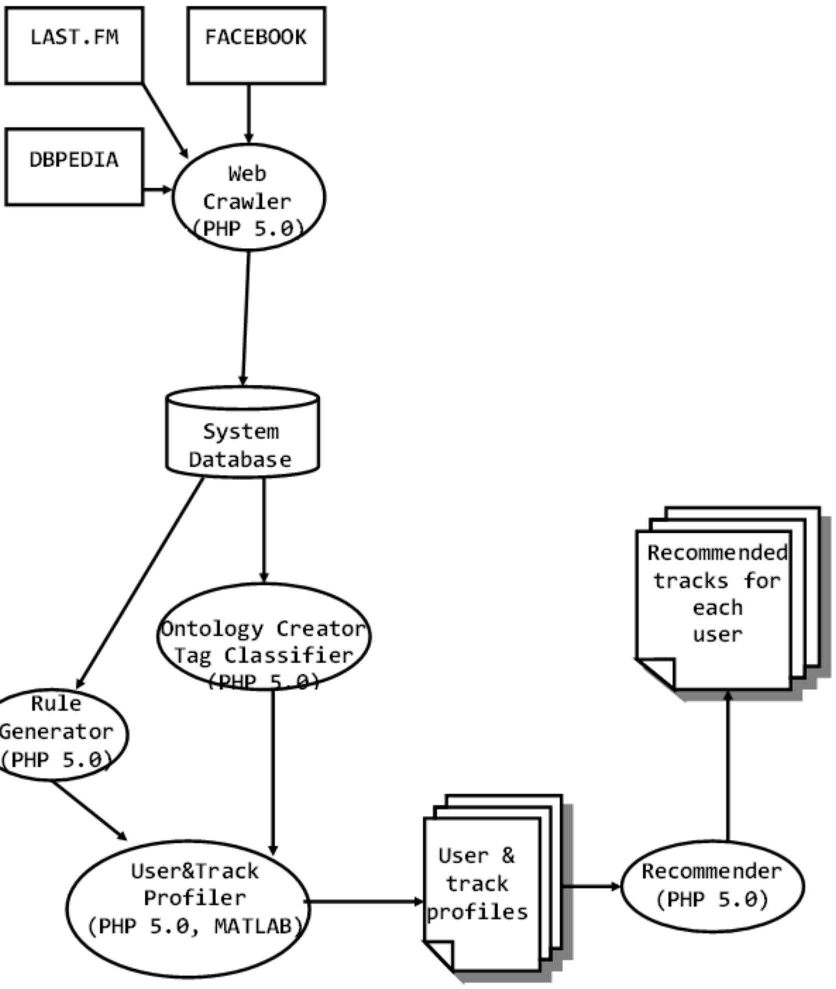


Figure 3.11: Implementation Architecture

Choosing LAMP seems to be the best way for the system for complete control. Moreover, it is easy to install. Its performance, security and reliability are considered to be the best architecture. Moreover, PHP has a simple web integration and Apache is known for its security features.

Because tracks vectors are huge in size, MATLAB is used for the track and user profiling phases. MATLAB performs well for big matrices. For the recommendation system using LSA in dimensionality reduction; HPC (high performance computing) machines in Department of Computer Engineering (METU) are used because of the parallel MATLAB utility[59]. Parallel MATLAB eases our work in TF*IDF and Okapi-bm25 weighting , however SVD function (base of LSA) in MATLAB cannot be fastened. The most time-consuming part of the implementation is the SVD with an optimal ranking in MATLAB. Instead of MATLAB's *svds()* function, we have used a "Sparse Matrix SVD" function [62] which is faster and consumes less memory than the original $svds()$ function.

CHAPTER 4

EVALUATION

This chapter provides the evaluation of the proposed method. Firstly, the details of the data set are given. The evaluation methodology and evaluation metrics are explained. Then the evaluation results are discussed.

4.1 Data Set

Real Last.fm data is used in this thesis study. As explained before in the ”Web Crawler” section of Chapter 3, in order not to use similar users and in order to achieve diversity, 50 users are selected from an application named ”join Last.fm”. In this group, members of the group share their Last.fm nicknames. Moreover, 19 friends who use both Last.fm and Facebook are also added to the data set. Last.fm profiles of the users in the data set are crawled with the help of Last.fm API [50]. Since our approach is not collaborative but content-based, this number of users is acceptable. User selection process is shown in Figure 4.1.

Table 4.1 demonstrates the data crawled from Last.fm profiles of the aforementioned 69 users. For each user, 300 top tracks, ’loved’ tracks and friends are gathered. Moreover, tags and tag counts that the users and their friends assigned to tracks are extracted. For each track; artist, tag and tag counts are crawled. For each artist; tag and tag counts are extracted.

In addition to Last.fm profiles, Facebook profiles of the users are also crawled. Their favourite movies, tv-series and music bands are gathered. This information is used in rule generation process which was explained in Chapter 3. Details are given in Table 4.2.

Table 4.1: Details of the Data Set (gathered from Last.fm)

# of users	69
# of tracks	13312
# of tags	169174
# of artists	4253
Average # of tracks per user	527
Average # of tags per track	45
Average # of tags per artist	70
# of users having own tags	4
Average # of tags per user	18
Average # of friend tags per user	37

Table 4.2: Details of the Data Set (gathered from Facebook)

# of users	400
# of artists (by 171 users)	986
# of movies (by 131 users)	519
# of tv-series (by 146 users)	269
Average # of movies per user	11
Average # of tv-series per user	10
Average # of bands per track	9

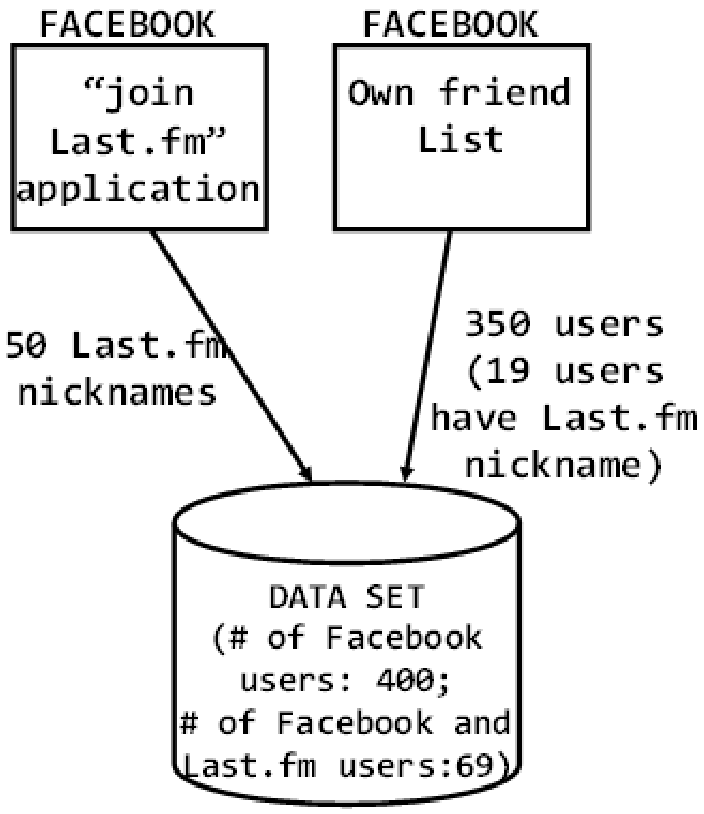

FACEBOOK
"join Last.fm" application
FACEBOOK
Own friend List
50 Last.fm nicknames
350 users (19 users have Last.fm nickname)
DATA SET (# of Facebook users: 400; # of Facebook and Last.fm users:69)

Figure 4.1: Users in the data set

4.2 Methodology

RSs can be evaluated in many different ways. The most common one is to get the user feedbacks. Because getting user feedbacks is an expensive process, multi-fold validations are used in order to simulate the recommendation environment [56]. For each user, we performed a 4-fold-cross validation in which the training data size was 75% of the listened tracks and the test data was 25% of the listened tracks. User profiles were created using the training set and the task of our RS was to predict the correct items in the test set. This process was repeated 4 times with every test and training sets; and at the end, results are averaged. Figure 4.2 depicts the methodology we followed:

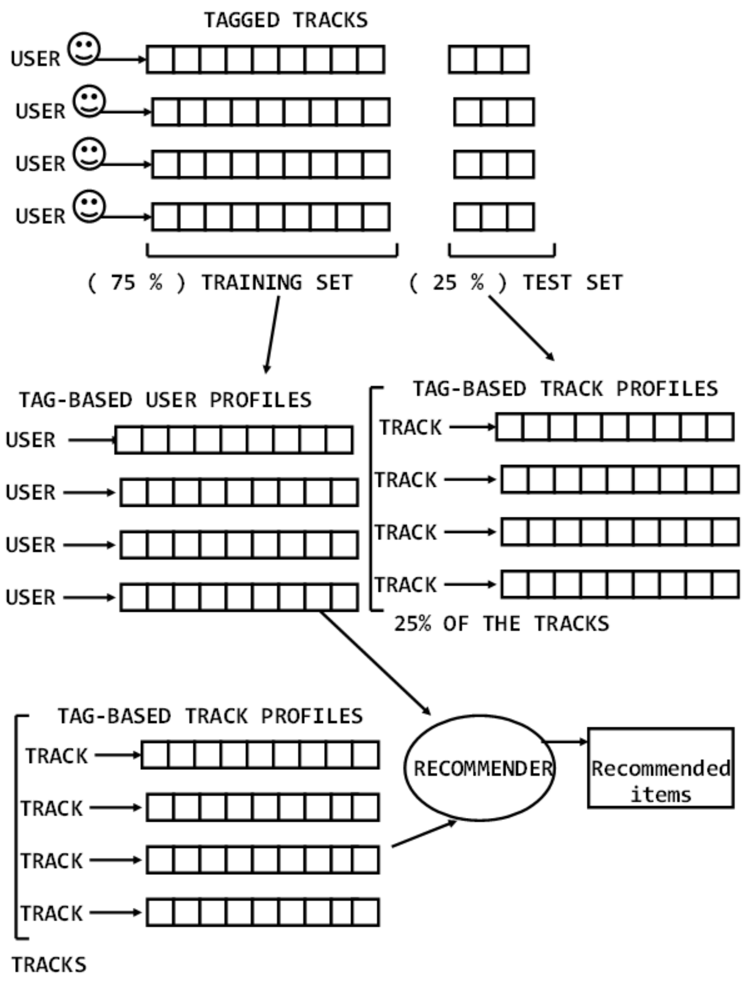


Figure 4.2: Description of the followed experimental methodology

4.3 Evaluation Metrics

Most popular metrics used in RSs are explained below [57]:

- **Predictive accuracy metrics** are used to find the closeness between predicted ratings and actual user ratings. MAE (mean absolute error) which is an average of absolute errors is an example for the predictive accuracy metrics.
- **Classification accuracy metrics:** are used to find the ratio of relevant items to all items/all recommended items (namely recall and precision).
- **Ranks accuracy metrics:** measure the ability of a RS to produce the correct order of items.

- **Prediction-rating correlation:** is the correlation between the variance of the recommender's result and the actual user's result.

In this study we used the most common classification accuracy metrics precision and recall.

Precision is the ratio of relevant tracks recommended correctly to the number of tracks recommended. Specifically, we are interested in the top N ranked results (P@N).

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}}$$

Where,

- N_{TP} is the total number of true positives namely, total number of items recommended correctly.
- N_{FP} is the total number of false positives namely, total number of items recommended incorrectly.
- $N_{TP} + N_{FP}$ is the sum number of true positives and false positives namely, total number of all items recommended.

Recall is the ratio of relevant tracks recommended correctly to the number of tracks user listened to.

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

Where,

- N_{TP} is the total number of true positives namely, total number of items recommended correctly.
- N_{FN} is the total number of false negatives namely, total number of items not recommended incorrectly.
- $N_{TP} + N_{FN}$ is the sum number of true positives and false negatives namely, total number of all items available.

Table 4.3: Categorization of items

	Actual positive	**Actual negative**
Predicted as positive	TP	FP
Predicted as negative	FN	TN

Recall is said to be impractical [57]. For an actual recall calculation, the system should know all items are whether liked or disliked by all users. This is impractical for very large music recommendation systems.

Break even point is also used in the evaluation of the system in long term. Break even point is the point at which the system's recall rate matches its precision rate.

4.4 Experimentation Results

In Last.fm, although users listen to music, they rarely enter tags for the tracks that they like. Thus, user profiles in Last.fm are smaller than in other social tagging sites, so that the performance of the pure CB recommendation is not satisfying [39].

In Table 4.5; two recommenders using LSA with an optimized parameter -k- and our method in dimensionality reduction are compared in terms of recommending the corresponding tracks in the test set. LSA is applied to the track-tag matrix whose size is 13312*169174 (13312 tracks, 169174 tags). On the other hand, the recommender using semantic relations method decreases the matrix size to 13312 * 22 (13312 tracks, 22 genres). In this recommender, each genre is semantically related to instruments, stylistic origins, subgenres, fusion genres and derivative forms. Thus, semantically related tags are counted as the same genre in this representation.

The aforementioned 2 recommenders are evaluated with using different user profiling methods explained in Chapter 3. Track-tag matrix used in LSA is very huge in size so that we have used HPC (High Performance Computing)[59] facility of METU Computer Engineering Department. Even though we ran the MATLAB codes on this machines using parallel programming, LSA for big rankings takes more than 5 hours which is not optimal for real-time RSs. Elapsed time for track profiling and recommendation process using LSA (with ranking 22,100,1000 and 5000) is shown in Table 4.4.

Table 4.4: Average results and time duration obtained by the CB approaches using LSA with different rankings

User profiling method	**ranking k**	**duration**	**P@5**	**P@10**	**P@20**	**B.E.P.**
All tracks	22	11 min.s	0.036	0.043	0.042	0.037
All tracks	100	48 min.s	0.058	0.053	0.050	0.040
All tracks	1000	307 min.s	0.077	0.076	0.070	0.052
All tracks	5000	721 min.s	0.079	0.077	0.071	0.055

As seen in the Table 4.5, it is obvious that the recommender using semantic relations outperforms the recommender using LSA in dimensionality reduction. In this comparison; ranking for LSA is chosen to be the optimal one in terms of both optimal time and optimal space. The comparison between LSA and semantic-relations are given in Table 4.6.

The precision and break-even points for the recommendation systems seems to be low. In the movie domain, recommenders generally has more than 70 % precision. On the other hand, in the music domain precision values are lower because of the huge number of the tracks. Recommending exactly the same item with the items in the test set results in lower precision values. To get higher precision values; similar tracks to the tracks in the test set may be assumed as ”true positives”. As an assumption; all the tracks of an artist can be assumed as similar tracks.

As it is obvious in Table 4.5, top-5 recommended items are the most successful ones. Thus precision values for top-5 items are greater than precision values of top-10 items. Same as, precision values of top-20 items are smaller than precision values of top-10 items.

Cos_{BM25} outperformed Cos_{tf-idf} while using LSA in dimensionality reduction. On the other hand, Cos_{tf-idf} is more successful than Cos_{BM25} while using semantic-relations in dimensionality reduction. In general, Cos_{BM25} is better than Cos_{tf-idf} because B.E.P. with Cos_{BM25} is always higher than B.E.P. with Cos_{tf-idf}. Details are given in Table 4.7.

Personal tags are the best identifiers for users; however insufficient number of personal tags result in poor recommendations. Friend tags have been thought to perform well at the beginning of the study. However, it is seen that friends have not entered tags for user’s tracks. The recommender using all social tags in the user profiling seems to provide the best results because it handles the semantic gap problem in social tagging. When all the tags of the artists of the tracks that user listened to are used, quality of the recommendation increases. It can be said that tags of artists are better identifiers than tags of tracks for the user profiles. On the other hand, using multi-domain information in user profiling provides better recommendations. Comparison of the user profiling methods are given in Table 4.8.

Table 4.5: Average results obtained by the CB approaches using LSA and semantic relations

Dimensionality reduction method	**User profiling method**	**Similarity method**	**P@5**	**P@10**	**P@20**	**Break Even Point**
Semantic Relations	User Tags	cos_{tf-idf}	0.000	0.100	0.175	0.064
Semantic Relations	Friend Tags	cos_{tf-idf}	0.000	0.000	0.000	0.000
Semantic Relations	All tracks	cos_{tf-idf}	0.178	0.168	0.134	0.050
Semantic Relations	All artists	cos_{tf-idf}	0.200	0.184	0.163	0.053
Semantic Relations	All extended artists	cos_{tf-idf}	0.231	0.196	0.168	0.052
Semantic Relations	User Tags	cos_{bm25}	0.000	0.000	0.000	0.000
Semantic Relations	Friend Tags	cos_{bm25}	0.000	0.000	0.000	0.000
Semantic Relations	All tracks	cos_{bm25}	0.126	0.110	0.121	0.056
Semantic Relations	All artists	cos_{bm25}	0.115	0.110	0.102	0.052
Semantic Relations	All extended artists	cos_{bm25}	0.136	0.121	0.113	0.053
LSA (with optimal k)	User Tags	cos_{tf-idf}	0.000	0.065	0.081	0.057
LSA (with optimal k)	Friend Tags	cos_{tf-idf}	0.000	0.000	0.016	0.011
LSA (with optimal k)	All tracks	cos_{tf-idf}	0.079	0.077	0.071	0.055
LSA (with optimal k)	All artists	cos_{tf-idf}	0.081	0.076	0.074	0.055
LSA (with optimal k)	All extended artists	cos_{tf-idf}	0.084	0.086	0.079	0.054
LSA (with optimal k)	User Tags	cos_{bm25}	0.000	0.107	0.085	0.059
LSA (with optimal k)	Friend Tags	cos_{bm25}	0.000	0.000	0.015	0.011
LSA (with optimal k)	All tracks	cos_{bm25}	0.083	0.081	0.075	0.064
LSA (with optimal k)	All artists	cos_{bm25}	0.082	0.079	0.077	0.064
LSA (with optimal k)	All extended artists	cos_{bm25}	0.086	0.085	0.081	0.065

Table 4.6: Comparison of Semantic Relations and LSA methods in dimensionality reduction

	Semantic Relations Method	**LSA**
(+)	• Once created,faster • Uses lower memory-spaces • Performs better (compared to SVD with a computable ranking) • Handles ”morphology” problem (uses stemming algorithm)	• Reliable • Common method, used for years • Easy to implement
(-)	• Creating ontology takes a bit long time (crawling Dbpedia)	• High time and space complexity • ”Morphology” problems

Table 4.7: Comparison of the methods used in similarity

$Cosine_{BM25}$ **Method**	$Cosine_{TF-IDF}$ **Method**
• more successful while using LSA in dimensionality reduction • provides better break-even points	• more successful while using semantic-relations in dimensionality reduction

Table 4.8: Methods used in ”user profiling”

User Profiling Method	**Properties**
Using user’s own tags user entered for tracks	• Best identifiers for users • Generally, user profiles suffer from the lack of tags (insufficient number of tags)
Using friends’ tags they entered for user’s tracks	• Maybe better recommendations than users’s own tags if there is a higher number of tags per track • Crawling friends’ tags for the user’s tracks takes too long
Using tags of tracks user listened to	• No extra implementation, uses the track vectors • Better than using friends’ tags for the tracks that user listened to; however worse than using tags of artists of the tracks user listened to
Using tags of artists of tracks user listened to	• Better than using tags of tracks that user listened to; however worse than using tags of extended artist set • Extra implementation for singer profiles
Extending the set of user’s artists with Facebook profiles	• Better recommendations than using only the artists of tracks user listened to • Uses multi-domain information • Extra effort for the rule creation

CHAPTER 5

CONCLUSION AND FUTURE WORK

User annotated texts, tags in our case, are huge in size, but the representation matrix is very sparse. Using such giant matrices in calculations is a time- and resource- consuming job. For the document categorization and text summarization, LSA has been used for years because it is easy to use and reliable. As an alternative, with the help of Dbpedia, we created an ontology-like semantic relations structure for the music domain. Within this thesis study, an ontology-based dimensionality reduction method for large vector spaces has been presented. We have compared two recommenders; one using our method and the other using Latent Semantic Analysis (LSA) in dimensionality reduction. The recommenders have been evaluated with different user profiling methods and similarity functions. Our method has the advantage of using ”morphology” with respect to LSA.

In addition to our novel dimensionality reduction method, we have proposed different methods for user profiling. Most of the time, users do not enter tags for the tracks they listen to. Thus users' own tags are insufficient for the construction of user profiles. Users have been profiled in 5 different methods using personal tags that they entered to tracks, using friends' tags that friends entered to tracks, using tags of all tracks they listened to and using tags of artists of all tracks they listened to. By using the information crawled from Facebook profiles, we have generated rules like ”if a person likes movie-M, than he/she may also like artist-A”. We have extended the user profiles with the artists they may like according to the rules. The rules can be extended with the demographic data of the users such as similar schools, locations and ages.

Creation of data set and ontology took a long time. As a future work, ready data sets may be used. Moreover, created ontology can be extended with cultural origins and regional scenes specified in Dbpedia. Now about 80000 out of 170000 tags are assigned to the ontology. By the extension with cultural origins and regional scenes, less number of tags can be discarded. This work can be extended assigning different weights for different relations. For instance, *hasInstrumentation* and *hasSubgenres* may have different weights in the track profiling. Moreover, the generated rules may also have different weights such that if 5 users loving movie-M also loves artist-A and 12 users loving movie-M also loves artist-B then tags of artist-B and artist-A may have different weights in the user profiling. In addition to these improvements, the system can be evaluated more number of users and tracks.

REFERENCES

[1] Facebook, http://www.facebook.com, Last accessed on 8th September 2011.

[2] Twitter, http://twitter.com, Last accessed on 8th September 2011.

[3] MySpace, http://www.myspace.com, Last accessed on 8th September 2011.

[4] Dwyer C.; Hiltz S. R., Passerini K., 2007. *Trust and Privacy Concern Within Social Networking Sites: A Comparison of Facebook and MySpace*. In Proceedings of the Thirteenth Americas Conference on Information Systems (Keystone, Colorado, 2007).

[5] Acquisti A., Gross R., 2006. *Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook*. In Proceedings of 6th Workshop on Privacy Enhancing Technologies (Cambridge, United Kingdom, 2006).

[6] Celma, O., Lamere, P., (2007). *Music Recommendation Tutorial*. In Proceedings of 8th International Conference on Music Information Retrieval (Vienna, Austria, 2007).

[7] The Internet Movie Database, http://www.imdb.com, Last accessed on 8th September 2011.

[8] Last.fm, http://www.last.fm, Last accessed on 8th September 2011.

[9] Pandora, http://www.pandora.com, Last accessed on 8th September 2011.

[10] Grooveshark, http://grooveshark.com, Last accessed on 8th September 2011.

[11] Schedl M., Knees P., 2009. *Context-based Music Similarity Estimation*. In Proceedings of the 3rd International Workshop on Learning the Semantics of Audio Signals (Graz, Austria, 2009).

[12] Firan C., Nejdl W., 2007. *The Benefit of Using Tag based Profiles*. In Proceedings of LAWEB 2007 (Santiago, Chile, 2007).

[13] Wolf F., Poggio T. and Sinha P., 2006. *Human Document Classification Using Bags of Words*. Technical Report.

[14] J. Silva, 2008. *Information Filtering and Information Retrieval with the Web Filtering Toolbar*. Electronic Notes in Theoretical Computer Science, vol. 235, pages 125-136, 2008.

[15] Soergel D., 1985. *Information Retrieval The Scope of IR*. On http://www.dsoergel.com/NewPublications/HCIEncyclopediaIRShortEForDS.pdf, Last accessed on 8th September 2011.

[16] Costa A, Roda F., 2011. *Recommender systems by means of information retrieval*. In Proceedings of WIMS, pages 57:1-57:5. ACM, 2011.

[17] Schafer B., Frankowski D., Herlocker J., Sen S., 2007. *Collaborative Filtering Recommender Systems*. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York (2007)

[18] Breese J. S., Heckerman D., Kadie C., 1998. *Empirical analysis of predictive algorithms for collaborative filtering*. In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence,pages 43-52, July 1998.

[19] Ozsoy M., Cicekli I., Alpaslan F., 2010. *Text Summarization of Turkish Texts using Latent Semantic Analysis*. In Proceedings of of the 23rd International Conference on Computational Linguistics (Beijing, China, 2010).

[20] Pazzani M. J., Billsus D., 2007. *Content-Based Recommendation Systems*. Lecture Notes in Computer Science. (4321), 325-341.

[21] Burke R., 2002. *Hybrid Recommender Systems:Survey and Experiments*. In Proceedings of Journal of User Model. UserAdapt. Interact., 12(4):331370,(2002)

[22] Peis E., Castello J.M.M., DelgadoLopez J.A., 2008. *Semantic Recommender Systems: analysis of the state of the topic*. In Hypertext, (2008).

[23] Wang Y., Stash N., Aroyo L., Hollink L., Schreiber G., 2009. *Semantic Relations in Content-based Recommender Systems*. In Proceedings of the Workshop on Ontology Patterns (WOP) at ISWC (Chantilly, U.S.A., 2009).

[24] Balabanovic M.S.Y., 1997. *Fab: content-based, collaborative recommendation*.In Communications of the ACM, v. 40, pp. 66-72.

[25] Wang R.Q., Kong F.S., 2007. *Semantic-Enhanced Personalized Recommender System*.In Machine Learning and Cybernetics: Proc. of the Int. Conference on Machine Learning and Cybernetics , v. 7 (19-22), pp. 4069-4074.

[26] Farsani H.K., Nematbakhsh M.A., 2006. *A Semantic Recommendation Procedure for Electronic Product Catalog*. In International Journal of Applied Mathematics and Computer Sciences, v. 3, pp. 86-91.

[27] Ziegler N., 2004. *Semantic Web Recommender Systems*. In Proceedings of the joint ICDE/EDBT Workshop.

[28] Jung K., 2005. *RDF Triple Processing Methodology for the Recommendation System Using Personal Information*. In Proceedings Of the Int. Conference on Next Generation Web Services Practices, pp. 241-247.

[29] Cantador I., Castells P., 2006. *Multi-layered Semantic Social Network Modelling by Ontology-Based User Profiles Clustering: Application to Collaborative Filtering*. In Managing Knowledge in a World of Networks, pp. 334-349.

[30] Rich E., 1979. *User Modelling via Stereotypes*. In Cognitive Science 3, 329-354.

[31] CLIPS: A Tool for Building Expert Systems, http://clipsrules.sourceforge.net/, Last accessed on 8th September 2011

[32] JESS, http://www.jessrules.com/jess/download.shtml, Last accessed on 8th September 2011

[33] Tso-Sutter K.L.H., Marinho L.B., Schmidt-Thieme L., 2008. *Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms*. In Proceedings of the 2008 ACM Symposium on Applied Computing (Ceara, Brazil, 2008).

[34] Symeonidis P., Ruxanda M., Nanopoulos A., Manolopoulos Y., 2008. *Ternary semantic analysis of social tags for personalized music recommendation*. In Proceedings of ISMIR (Philadelphia, Pennsylvania, USA, 2008).

[35] Liang H., Xu Y., Li Y., Nayak R., 2008. *Collaborative Filtering Recommender Systems Using Tag Information*. In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (Sydney, Australia, 2008), pp. 59-62.

[36] Shepitsen A., Gemmell J., Mobasher B., Burke R., 2008. *Personalized recommendation in social tagging systems using hierarchical clustering*. In Proceedings of the ACM Conference on Recommender systems (Lausanne, Switzerland, 2008).

[37] Peis E., Castello J.M.M., Delgado Lopez J.A., 2008. *Semantic Recommender Systems: analysis of the state of the topic*. In Hypertext (2008).

[38] Wang Y., Stash N., Aroyo L., Hollink L., Schreiber G., 2009. *Semantic Relations in Content-based Recommender Systems*. In Proceedings of the Workshop on Ontology Patterns (Chantilly, U.S.A., 2009).

[39] Cantador I., Bellogin A., Vallet D., 2010. *Content-based Recommendation Systems in Social Tagging Systems*. In Proceedings of the fourth ACM conference on Recommender systems (Barcelona, Spain, 2010).

[40] Levy M., Sandler M., 2008. *Learning Latent Semantic Models For Music From Social Tags*. In Proceedings of Journal of New Music Research (2008), pages 137-150.

[41] Pampalk E., Goto M., 2007. *Musicsun: A new approach to artist recommendation*. In Proceedings of International Conference on Music Information Retrieval, pages 101-104,(2007).

[42] Sarmento L., Gouyon F., Oliveira E., 2009. *Music Artist Tag propagation with Wikipedia abstracts*. In ECIR-WIRSN,(2009)

[43] Hung C.C., Huang Y.C., Hsu J.Y., Wu D.K.C., 2008,. *Tag-Based User Profiling for Social Media Recommendation*. Workshop onIntelligent Techniques for Web Personalization and Recommender Systems, AAAI 2008.

[44] Last.fm on Wikipedia, http://en.wikipedia.org/wiki/Last.fm, Last accessed on 8th September 2011

[45] Facebook on Wikipedia, http://en.wikipedia.org/wiki/Facebook, Last accessed on 8th September 2011

[46] Delicious, http://www.delicious.com/, Last accessed on 8th September 2011

[47] Sordo M., Gouyon F., Sarmento L.,2010. *A Method for Obtaining Semantic Facets of Music Tags*. Workshop on Music Recommendation and Discovery, ACM Conference on Recommender Systems. (Barcelona, Spain, 2010).

[48] Okapi-bm25 tutorial, http://www.miislita.com/information-retrieval-tutorial/okapi-bm25-tutorial.pdf, Last accessed on 8th September 2011.

[49] Bellogin A., Cantador I., Castells P., 2010. *A Study of Heterogeneity in Recommendations for a Social Music Service*. HetRec Workshop at RecSys 2010 (Barcelona, Spain, 2010).

[50] Last.fm API, http://last.fm/api, Last accessed on 8th September 2011

[51] Porter M., 2001. *Snowball: A language for stemming algorithms*. On http://snowball.tartarus.org/texts/introduction.html., Last accessed on 8th September 2011

[52] Kumar C.A., 2009. *Analysis of Unsupervised Dimensionality Reduction Techniques*. Comput. Sci. Inf. Syst., vol. 6, no. 2, Dec. 2009, pp. 217-227.

[53] Baeza-Yates, R., Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Addison Wesley.

[54] Sparck-Jones, K., Walker, S., Robertson, S. E. 2000. *A Probabilistic Model of Information Retrieval: Development and Comparative Experiments (parts 1 and 2)*. Information Processing and Management, 36(6):779-840

[55] TRANS: Transportation Research Analysis using NLP TechniqueS, https://wiki.cs.umd.edu/cmsc734_09/images/6/62/Trans-finalreport.pdf, Last accessed on 8th September 2011.

[56] Fournier F., 2010. *Recommender Systems Technical Report and Literature Review*. On http://knol.google.com/k/recommender-systems#, Last accessed on 8th September 2011

[57] Herlocker J. L., Konstan J. A., Terveen L. G., Riedl, J. T., 2004. *Evaluating collaborative filtering recommender systems*. In ACM Trans. Inf. Syst. 22 (1): 5-53, doi:10.1145963770.963772.

[58] ”join last.fm” application, http://www.facebook.com/#!/group.php?gid=2246697136, Last accessed on 8th September 2010.

[59] High performance computing, http://hpc.ceng.metu.edu.tr/, Last access on 8th September 2011

[60] Dbpedia, http://dbpedia.org/ontology/MusicGenre, Last access on 8th September 2011

[61] Tatli I., Birturk A., 2011. *Using Semantic Relations in Context-based Music Recommendations*. Workshop on Music Recommendation and Discovery, ACM Conference on Recommender Systems. (Chicago,IL, USA, 2011).

[62] Sparse Matrix SVD, on http://www.mcs.anl.gov/ jiechen/software.html#Sparse

[63] Wikipedia on Wikipedia, http://en.wikipedia.org/wiki/Wikipedia, Last access on 8th September 2011

[64] Dbpedia on Wikipedia, http://en.wikipedia.org/wiki/DBpedia, Last access on 8th September 2011

APPENDIX A

PUBLICATIONS

İpek Tatlı, Ayşenur Birtürk,A Content-based Recommendation System for Music Domain based on Tags, The eChallenges e-2010 Conference, 27-29 October 2010, Warsaw, Poland. (withdrawn paper, since not registered)

İpek Tatlı, Ayşenur Birtürk, Using Semantic Relations in Context-based Music Recommendations, Workshop on Music Recommendation and Discovery 2011 (WOMRAD, in conjunction with ACM RecSys), 23-27 October 2011, Chicago, Illinois, USA.

İpek Tatlı, Ayşenur Birtürk, A Tag-based Hybrid Music Recommendation System Using Semantic Relations and Multi-domain Information, The Fifth International Workshop on Mining Multiple Information Sources (MMIS-11, in conjunction with The IEEE International Conference on Data Mining (ICDM-11)), 10 December 2011, Vancouver, Canada, USA.

APPENDIX B

ROCK MUSIC ONTOLOGY

```
<Ontology xmlns=”http://www.w3.org/2002/07/owl#”
      xml:base=”http://www.semanticweb.org/ontologies/2011/7/
Ontology1314790268281.owl”
      xmlns:rdfs=”http://www.w3.org/2000/01/rdf-schema#”
      xmlns:xsd=”http://www.w3.org/2001/XMLSchema#”
      xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
      xmlns:xml=”http://www.w3.org/XML/1998/namespace”
      ontologyIRI=”http://www.semanticweb.org/ontologies/2011/7/
Ontology1314790268281.owl” >
      <Prefix name=”xsd” IRI=”http://www.w3.org/2001/XMLSchema#”/ >
      <Prefix name=”owl” IRI=”http://www.w3.org/2002/07/owl#”/ >
      <Prefix name=”” IRI=”http://www.w3.org/2002/07/owl#”/ >
      <Prefix name=”rdf” IRI=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”/ >
      <Prefix name=”rdfs” IRI=”http://www.w3.org/2000/01/rdf-schema#”/ >
      <owl:Class rdf:ID=”Genre” >
      </owl:Class >
```

```
<owl:Class rdf:ID=”Instrumentation” >
        <rdfs:subClassOf rdf:resource=”#Genre”/ >
        <owl:disjointWith rdf:resource=”#SubGenre”/ >
        <owl:disjointWith rdf:resource=”#FusionGenre”/ >
        <owl:disjointWith rdf:resource=”#StylisticOrigins”/ >
        <owl:disjointWith rdf:resource=”#DerivativeForms”/ >
</owl:Class >
<owl:Class rdf:ID=”SubGenre” >
        <rdfs:subClassOf rdf:resource=”#Genre”/ >
        <owl:disjointWith rdf:resource=”#Instrumentation”/ >
        <owl:disjointWith rdf:resource=”#FusionGenre”/ >
        <owl:disjointWith rdf:resource=”#StylisticOrigins”/ >
        <owl:disjointWith rdf:resource=”#DerivativeForms”/ >
</owl:Class >
<owl:Class rdf:ID=”FusionGenre” >
        <rdfs:subClassOf rdf:resource=”#Genre”/ >
        <owl:disjointWith rdf:resource=”#Instrumentation”/ >
        <owl:disjointWith rdf:resource=”#SubGenre”/ >
        <owl:disjointWith rdf:resource=”#StylisticOrigins”/ >
        <owl:disjointWith rdf:resource=”#DerivativeForms”/ >
</owl:Class >
<owl:Class rdf:ID=”StylisticOrigins” >
```

```
    <rdfs:subClassOf rdf:resource=”#Genre”/ >
    <owl:disjointWith rdf:resource=”#Instrumentation”/ >
    <owl:disjointWith rdf:resource=”#SubGenre”/ >
    <owl:disjointWith rdf:resource=”#FusionGenre”/ >
    <owl:disjointWith rdf:resource=”#DerivativeForms”/ >
</owl:Class >
<owl:Class rdf:ID=”DerivativeForms” >
    <rdfs:subClassOf rdf:resource=”#Genre”/ >
    <owl:disjointWith rdf:resource=”#Instrumentation”/ >
    <owl:disjointWith rdf:resource=”#SubGenre”/ >
    <owl:disjointWith rdf:resource=”#FusionGenre”/ >
    <owl:disjointWith rdf:resource=”#StylisticOrigins”/ >
</owl:Class >
<owl:ObjectProperty rdf:ID=”hasInstrumentation” >
    <rdfs:domain rdf:resource=”#Genre”/ >
    <rdfs:range rdf:resource=”#Instrumentation”/ >
</owl:ObjectProperty >
<owl:ObjectProperty rdf:ID=”hasSubGenre” >
    <rdfs:domain rdf:resource=”#Genre”/ >
    <rdfs:range rdf:resource=”#SubGenre”/ >
</owl:ObjectProperty >
<owl:ObjectProperty rdf:ID=”hasFusionGenre” >
```

```
        <rdfs:domain rdf:resource=”#Genre”/ >
        <rdfs:range rdf:resource=”#FusionGenre”/ >
</owl:ObjectProperty >
<owl:ObjectProperty rdf:ID=”hasDerivativeForms” >
        <rdfs:domain rdf:resource=”#Genre”/ >
        <rdfs:range rdf:resource=”#DerivativeForms”/ >
</owl:ObjectProperty >
<owl:ObjectProperty rdf:ID=”hasStlisticOrigins” >
        <rdfs:domain rdf:resource=”#Genre”/ >
        <rdfs:range rdf:resource=”#StlisticOrigins”/ >
</owl:ObjectProperty >
<owl:Class rdf:ID=”Rock” >
        <owl:onProperty rdf:resource=”#hasInstrumentation” / >
        <owl:someValuesFrom rdf:parseType=”Collection” >
                <owl:Thing rdf:about=”#Vocals” / >
                <owl:Thing rdf:about=”#Electric_guitar” / >
                <owl:Thing rdf:about=”#Bass_guitar” / >
                <owl:Thing rdf:about=”#Drums” / >
                <owl:Thing rdf:about=”#Synthesizer” / >
                <owl:Thing rdf:about=”#Keyboards” / >
        </owl:someValuesFrom >
        <owl:onProperty rdf:resource=”#hasStylisticOrigins” / >
```

```
<owl:someValuesFrom rdf:parseType="Collection" >
        <owl:Thing rdf:about="#rock_n_roll" / >
        <owl:Thing rdf:about="#Electric_blues" / >
        <owl:Thing rdf:about="#Folk_music" / >
        <owl:Thing rdf:about="#Country" / >
        <owl:Thing rdf:about="#Blues" / >
</owl:someValuesFrom >
<owl:onProperty rdf:resource="#hasDerivativeForms" / >
<owl:someValuesFrom rdf:parseType="Collection" >
        <owl:Thing rdf:about="#new_age_music" / >
        <owl:Thing rdf:about="#synthpop" / >
</owl:someValuesFrom >
<owl:onProperty rdf:resource="#hasSubGenres" / >
<owl:someValuesFrom rdf:parseType="Collection" >
        <owl:Thing rdf:about="#Alternative_rock" / >
        <owl:Thing rdf:about="#Art_rock" / >
        <owl:Thing rdf:about="#Baroque_pop" / >
        <owl:Thing rdf:about="#Beat_music" / >
        <owl:Thing rdf:about="#Britpop" / >
        <owl:Thing rdf:about="#Emo" / >
        <owl:Thing rdf:about="#Experimental_rock" / >
        <owl:Thing rdf:about="#Garage_rock" / >
```

```
<owl:Thing rdf:about=”#Glam_rock” / >

<owl:Thing rdf:about=”#Grindcore” / >

<owl:Thing rdf:about=”#Group_Sounds” / >

<owl:Thing rdf:about=”#Grunge” / >

<owl:Thing rdf:about=”#Hard_rock” / >

<owl:Thing rdf:about=”#Heartland_rock” / >

<owl:Thing rdf:about=”#Heavy_metal” / >

<owl:Thing rdf:about=”#Instrumental_rock” / >

<owl:Thing rdf:about=”#Indie_rock” / >

<owl:Thing rdf:about=”#Jangle_pop” / >

<owl:Thing rdf:about=”#Krautrock” / >

<owl:Thing rdf:about=”#Madchester” / >

<owl:Thing rdf:about=”#Post_Britpop” / >

<owl:Thing rdf:about=”#Power_pop” / >

<owl:Thing rdf:about=”#Progressive_rock” / >

<owl:Thing rdf:about=”#Protopunk” / >

<owl:Thing rdf:about=”#Psychedelia” / >

<owl:Thing rdf:about=”#Punk_rock” / >

<owl:Thing rdf:about=”#Rock_noir” / >

<owl:Thing rdf:about=”#Soft_rock” / >

<owl:Thing rdf:about=”#Southern_rock” / >

<owl:Thing rdf:about=”#Surf” / >
```

```
        <owl:Thing rdf:about="#Symphonic_rock" / >
</owl:someValuesFrom >
<owl:onProperty rdf:resource="#hasFusionGenre" / >
 <owl:someValuesFrom rdf:parseType="Collection" >
        <owl:Thing rdf:about="#Aboriginal_rock" / >
        <owl:Thing rdf:about="#Afro_rock" / >
        <owl:Thing rdf:about="#Anatolian_rock" / >
        <owl:Thing rdf:about="#Bhangra_rock" / >
        <owl:Thing rdf:about="#Blues_rock" / >
        <owl:Thing rdf:about="#Country_rock" / >
        <owl:Thing rdf:about="#Flamenco_rock" / >
        <owl:Thing rdf:about="#Folk_rock" / >
        <owl:Thing rdf:about="#Funk_rock" / >
        <owl:Thing rdf:about="#Glam_punk" / >
        <owl:Thing rdf:about="#Indo-rock" / >
        <owl:Thing rdf:about="#Industrial_rock" / >
        <owl:Thing rdf:about="#Jazz_fusion" / >
        <owl:Thing rdf:about="#Pop_rock" / >
        <owl:Thing rdf:about="#Punta_rock" / >
        <owl:Thing rdf:about="#Raga_rock" / >
        <owl:Thing rdf:about="#Rai_rock" / >
        <owl:Thing rdf:about="#Rap_rock" / >
```

```
                    <owl:Thing rdf:about=”#Rockabilly” / >

                    <owl:Thing rdf:about=”#Rockoson” / >

                    <owl:Thing rdf:about=”#Samba_rock” / >

                    <owl:Thing rdf:about=”#Space_rock” / >

                    <owl:Thing rdf:about=”#Stoner_rock” / >

                    <owl:Thing rdf:about=”#Sufi_rock” / >

                </owl:someValuesFrom >

        </owl:Class >

</Ontology >

<!– Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net – >
```