Using Semantic Relations in Context-based Music Recommendations

İpek Tatlı, Ayşenur Birtürk
Department of Computer Engineering, METU
Inonu Bulvari, 06531
Ankara, Turkey
{ipek.tatli, birturk}@ceng.metu.edu.tr

ABSTRACT

In this paper, we describe an approach 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. We use this well-organized structure in dimensionality reduction in user and item profiling. We compare two recommenders; one using our method and the other using Latent Semantic Analysis (LSA) in dimensionality reduction. The recommender using our approach outperforms the other. In addition to different dimensionality reduction methods, we evaluate the recommenders with different user profiling methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models, information filtering, selection process*

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Recommendation systems, user profiling, social tagging, semantic relations, dimensionality reduction

1. INTRODUCTION

These days, most social-networking sites let their members participate in 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 of the item. Because tags represent the reason why a listener likes an item, but not how much he/she likes it they are better identifiers of user profiles than ratings, which are usually numerical values assigned to items by users. Thus, we concentrate on the tag-based contextual representations of music tracks.

Items are mostly represented in vector spaces in the 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 is to represent individual tracks (songs) in lower dimensional spaces. In order to reduce the dimensionality, we focus on the genre information of the tags. Each genre has a relationship with some instrumentation, with some subgenre information and with style information each of which may be entered as tags in the music domain. In our work, for each genre Dbpedia[1] (a structured form of Wikipedia[2]) is crawled to set the relationships between genre and its stylistic origins, typical instruments, derivative forms, sub genres and fusion genres. The contributions of our approach are that: (1) we provide a "semantic relations" method for dimensionality reduction in very huge vector spaces and (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.

2. RELATED WORK

In music recommendation systems, 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, lyrics, tags and reviews mined from various blogs [1]. Metadata information is mostly supplied by experts. The artist's name, the album's name, genre, duration and year are some attributes in the metadata. Attributes are global descriptions of items and do not change according to users whereas tags are local descriptors and might change from user to user [2]. In our study, we focus on text descriptions, namely tags in track profiling.

Recommender systems either predict ratings for unseen items or predict items that can be liked. Most of the social web-sites like Last.fm do not have a rating mechanism. Instead of explicit ratings, today's recommender systems use implicit ratings (users' listening habits and purchase histories etc.). Thus the rating scale in implicit rating mechanisms is 0-1. Tags can be used in rating-based collaborative

[1]http://dbpedia.org
[2]http://en.wikipedia.org

WOMRAD 2011 2nd Workshop on Music Recommendation and Discovery, colocated with ACM RecSys 2011 (Chicago, US)

filtering systems with the help of an implicit rating mechanism [2]. If the tag is used by the user, its rating is 1; otherwise its rating is 0. In most 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 to extend the rating data [2]. A new approach which uses all dimensionalities of the music space is proposed in [3]. Each 'useritem- tag' data is a tensor in this study and the researchers propose a Higher Order Singular Value Decomposition (HOSVD) technique for 3-dimensional social tagging data. The HOSVD method outperforms the classical methods. In contrast, the three 2-dimensional relations among users, tags and items have been used in a new similarity measure generation which outperforms the traditional item-based and user-based collaborative filtering methods [4]. In this approach, neighborhood generation is effected through the similarity of users' tags, similarity of users' items and the similarity of user's tag-item relationships. In addition to user similarities item similarities have been calculated with common tags, common users and common tag-item relationships. Moreover, tags can be clustered and these clusters improve the personalized recommendations [5].

Up to our knowledge, [6] uses a similar approach to ours in extracting top 50 music facets from Wikipedia but the main objective of [6] is to provide an automatic method for uncovering the music facets and to classify tags according to these facets. On the other hand, we create a hierarchical genre structure and evaluate the usefulness of our approach in music recommendation.

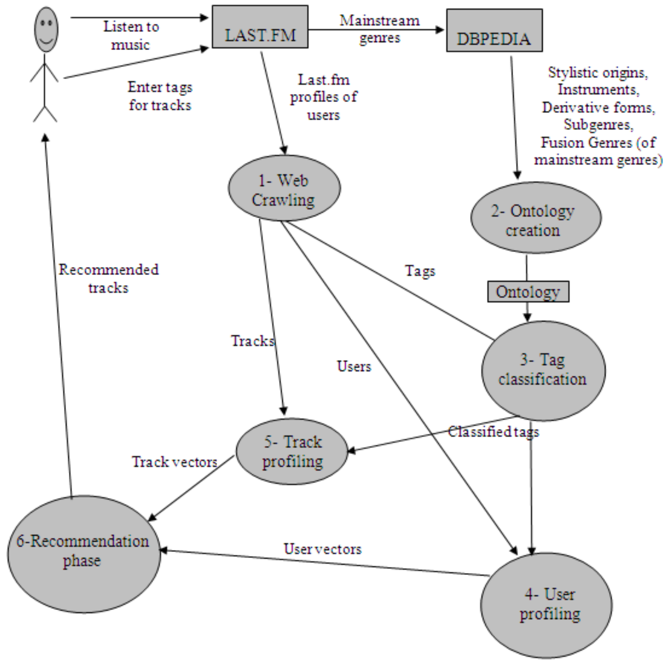

Figure 1: System architecture of the proposed approach

3. OUR APPROACH

Our system performs 6 main tasks, shown in Figure 1: web crawling, creating an ontology of musical genres, classifying tags according to the ontology, track profiling, user profiling and enacting the recommendation process. The circles denote the phases of the system.

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

Tags may be about genre, instrumentation, location, moods, style, personal opinions and/or artists' names [1]. 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 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 in our system. Instrumentation, subgenre, fusion genre, derivative forms and stylistic information are used in our track and user representation. Firstly, we decided the main genres in the musical domain. In [8], 14 mainstream genres (country, folk, jazz, blues, r&b, heavy metal, alternative/indie, punk, rap, electro, reggae, classical, rock and pop) are used. We enriched these genres 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 our 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&b, rock, soul, world. Having identified our genres, we decided to crawl the Wikipedia page for each main genre but then we switched to Dbpedia since it is more structured for web-crawling. We crawled Dbpedia page for each main genre. Obtained information is illustrated in Table 1 for "rock music".In the **ontology creation phase**, we created a small ontology-a hierarchical structure -with the help of the data crawled from the Dbpedia. Relations in our ontology can be seen in Table 2. In this structure instrumentation, stylistic origins, derivative forms, subgenres and fusion genres are the classes; the crawled data are the instances. For example, "New Age Music" and "Synthpop" are instances of the class "Derivative forms" which can be seen in Table 1.

LSA does not use word order and morphology. In order not to differentiate "electronic" from "electronica", we applied some stemming algorithm. 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. We used the Porter stemmer[3] which is a classical stemming algorithm. By using a stemming algorithm, morphology is taken into consideration in our approach. In the **tag classification phase** of our system, we parsed instances existing in the ontology into single words. We applied the stemming algorithm to each single word. Then we concatenated the stemmed roots with "%" in order to consider "word order" that LSA does not use. Some

[3]http://tartarus.org/ martin/PorterStemmer

Table 1: Wikipedia/Dbpedia page of "rock music"

Rock music
Stylistic origins: Rock and roll, electric blues, folk music, country, blues
Typical instruments: Vocals, electric guitar, bass guitar, drums, synthesizer, keyboards
Derivative forms: New Age Music - Synthpop
Subgenres: Alternative rock - Art rock - 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 - Countryf rock - Flamencorock - Folk rock - Funk rock - Glam Punk - Indo-rock - Industrial rock - Jazz fusion - Pop rock - Punta rock - Raga rock - Rai rock - Rap rock - Rockabilly - Rockoson - Samba-rock - Space rock - Stoner rock - Sufi rock

Table 2: Relations in our ontology

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

examples of the stemming results can be seen in Table 3.

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 these versions of the instances 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 100000 tags out of 160000 tags in the track representation. In the **track profiling phase**, the 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:

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

Table 3: Concatenating the stemmed words of the instances

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

Where, $g(i,j)$ is the i^{th} term (genre) in j^{th} track; and $hasInstrumentation(i,j)$ 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 term count is usually normalized to prevent a bias towards longer documents (which may have a higher term count regardless of the actual importance of that term in the document). The term frequency (TF) value gives local information about a tag. An inverse document frequency (IDF) value is calculated for each different tag in the training set. This is calculated by dividing the total number of tracks by the number of tracks that refer to that feature. The IDF value gives global information of a tag. Thus, tracks in our dataset are represented as a weighted list of genres and the weights of the genres are calculated with TF*IDF.

$$w_i = \frac{n_{i,t}}{n_t} \times log(\frac{|T|}{|T_i|})$$

In the formula above, w_i is the weight of i^{th} genre; $n_{i,t}$ equals the number of times i^{th} genre appears in t^{th} track; n_t is the total number of genres in t^{th} track; $|T|$ is the total size of the tracks, and $|T_i|$ equals the number of tracks in which i^th genre appears. In the **user profiling phase**, 3 different methods are used:

1- using the users' own tags (personal tags) that they entered

2- using the users' friends' tags (friends' tags) that their friends entered

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

In the first method, users are profiled with their own tags. In the second method, users are profiled with their friends' tags. And in the last method, users are profiled with all the tags of the tracks that they listened to. Semantic relations are also used in user profiling method 1 and method 2, just the same as in track profiling. In method 3, a user profile is the sum of the tracks that he/she has listened to. Weights of the genres in user vectors are also calculated with TF*IDF method. The main goal after creating a user profile from the training set is to recommend the items in the test set.

In the **recommendation phase**, we use the common cosine similarity method. The cosine similarity formula is given below:

$$CosSim(track, user) = \frac{track\_vector \times user\_vector}{|track\_vector||user\_vector|}$$

4. DATA SET

We use real Last.fm data in this study. In order to not to use similar users from our own friend lists and in order to achieve diversity, we selected 69 users from an application named "join Last.fm"[4]. In this group, members of the group share their Last.fm nicknames. We crawled their Last.fm profiles with the help of Last.fm API[5]. Since our approach is not collaborative but content-based, this number of users is reasonable. Firstly we gathered their top 300 tracks. Then we extracted their "loved" tracks. For each track, we extracted the singer names and tags. We also gathered the

[4]http://www.facebook.com/group.php?gid=2246697136&v=wall

[5]http://last.fm/api

Table 4: Details of the data set

# of users	69
# of tracks	13312
# of tags	169174
# of singers	4253
Average # of tracks per user	527
Average # of tags per track	45
Average # of tags per user	85
Average # of friend tags per user	451

tag counts per track. Finally for each user we extracted their tags and their friends' tags. Details of our data set can be seen in Table 4.

5. EXPERIMENTS

5.1 Methodology

We performed a 4-fold-cross validation in which the training data size was 75% and the test data was 25%. User profiles were created using the training set and the task of our recommender system was to predict the correct items in the test set.

5.2 Metrics

In this study we used the most common evaluation metric: Precision at the top N ranked results (P@N). Precision is the ratio of relevant tracks selected correctly to the number of tracks selected.

5.3 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 content-based recommendation is not satisfying [7]. In Table 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. As seen in the Table 5, it is obvious that the recommender using semantic relations outperforms the recommender using LSA in dimensionality reduction because it handles the semantic gap problem in social tagging. Moreover, the recommender using users' own tags in user profiling performs better than the recommender using friends' own tags. However, the recommender using all social tags in the user profiling seems to provide the best results because of the increasing number of tags in user vectors.

Table 5: Details of the data set

Dim. reduction method	User profiling method	P@5	P@10	P@20
Semantic relations	Tags of tracks user listened to	0.178	0.168	0.134
Semantic relations	Tags user entered	0.000	0.100	0.175
Semantic relations	Tags friends entered	0.000	0.000	0.000
LSA (with optimal k)	Tags of tracks user listened to	0.079	0.077	0.071
LSA (with optimal k)	Tags user entered	0.000	0.065	0.081
LSA (with optimal k)	Tags friends entered	0.000	0.000	0.016

6. CONCLUSION

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 or 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. In this paper, we evaluated two methods which can be used in dimensionality reduction. In the evaluation Last.fm dataset was used and the recommenders were evaluated with different user profiling methods. Our method has the advantage of using "word order" and "morphology" with respect to LSA. We plan to extend our work, assigning different weights for different relations. For instance, *hasInstrumentation* and *hasSubgenres* may have different weights in the track profiling.

7. REFERENCES

[1] O. Celma and P. Lamere. Music Recommendation Tutorial.*ISMIR.*Vienna, Austria, 2007.

[2] K.H.L. Tso-Sutter, L.B. Marinho and L. Schmidt-Thieme.Tag-aware recommender systems by fusion of collaborative filtering algorithms. *ACM symposium on Applied Computing.* Brazil, 2008.

[3] P. Symeonidis, M. Ruxanda, A. Nanopoulos and Y. Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. *ISMIR.* Philadelphia, USA, 2008.

[4] H. Liang, Y. Xu, Y. Li and R. Nayak. Collaborative Filtering Recommender Systems Using Tag Information. *Web Intelligence and Intelligent Agent Technology.* Sydney, Australia, 2008.

[5] A. Shepitsen, J. Gemmell, B. Mobasher and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. *ACM conference on Recommender systems.* Lausanne, Switzerland, 2008.

[6] M. Sordo, F. Gouyon and L. Sarmento. A Method for Obtaining Semantic Facets of Music Tags. *WOMRAD.* Barcelona, Spain, 2010.

[7] I. Cantador, A. Bellogin and D. Vallet. Content-based Recommendation Systems in Social Tagging Systems. *ACM conference on Recommender systems.* Barcelona, Spain, 2010.

[8] M. Levy and M. Sandler. Learning Latent Semantic Models For Music From Social Tags. *Journal of New Music Research.* pages 137-150.