Semantic Indexing for Music Search with Adaptive Recommendation

DENG Jie

Abstract

Due to the rapidly increasing availability of digital music, advanced semantic music search and music recommendation are becoming important issues. Thus how to index music for semantic search and how to make recommendations in intrinsic of musical art are challenging problems currently. In this context, this paper used a semantic indexing method that is capable of flexibly retrieval music in semantic level. When the semantic and dynamic index has been established, updating, increasing and decreasing index scores are necessary to build a reinforcement index to gain accurately search results. Moreover, the paper also proposed useful and flexible music recommendation approaches, for example song-to-song matching and multi-song matching methods. In order to optimize the order of search result, genetic algorithm also used to re-ranking those near hidden music in the top. The proposed method will also support users with diverse needs when searching for music. The observations indicate that the present approach is able to get better performance.

Keywords*: semantic index, genetic algorithm, ranking, recommendation, music search*

1. Introduction

As the digital music becomes more and more huge and ubiquitous, music search becomes the vital important tool when people use music services in the web. The first generation search engines make great contribution to finding textual information on the web. Yahoo!Music represent the first generation search engine attempts to support audio search, which adopts text-based approach and the service is limited [6, 7]. Thus, non-textual multimedia documents such as music audio bring new challenges [3] to search engines – how to incorporate search by the musical content, which will achieve better music search result. Therefore, music search has significant commercial and research promise in nowadays. Many groups have already made a great effort on this area, for example, Last.fm is a popular music search engine also with a music recommender system. Pandora is an automated music recommendation service and custodian of the Music Genome Project which captures the essence of music at the fundamental level. Thus intelligent music search and recommendations will have becoming more and more popular.

According to the research, there are three key issues in audio-based music search: how to index the content of music objects, how to present the user with intuitive methods of querying music objects, and which music objects to present to the user and in which order. This paper mainly addresses the first and the third key issues and proposes a novel approach for semantic index of music data objects and improved method for music data ranking and recommendation when music browsing.

This paper mainly focuses semantic index of music and music recommendation. Thus, the paper is organized as follows. In section 2, previous literature reviews on music search and recommendation will be described. Section 3 gives an overview of music search. A detailed description on proposed methodology will be introduced in Section 4, which consists of music representation, semantic indexing, index updating, incremental and decreased Index, music matching and browsing. Experiment observations will be given in Section 5. Finally, some conclusions and future work are discussed in Section 6.

2. Related Work

Extraction of music feature vectors is the basis of music search system. Nicola Orio [5] has already given some music characteristics and features in “Music Retrieval: A Tutorial and Review”. According to the Music Genome Project depict, a given song contains approximately 400 attributes. Each attribute corresponds to a characteristic of the music. Most of the today’s approaches content-based music search systems are based on melody, timbre and rhythm as the main features and often only content descriptors. Therefore, depending on these main features, there are three category approaches: index terms, sequence matching, and geometric methods which deal with polyphonic music scores. S. Downie and M. Nelson has presented that melodies were indexed through the use of N-grams. M. Melucci and N. Orio presented an alternative approach, where indexing was carried out by highlighting musically relevant approaches. Most importantly, an architectural paradigm for collaborative semantic indexing of multimedia data objects has been present in [1]. Leung has also researched multimedia data mining and searching through dynamic index evolution in [2]. C. R. Buchanan presented some approaches to

semantic-based audio recognition and retrieval in [9]. Douglas Turnbull [4] gives semantic annotation and retrieval of music.

Some of the famous commercial music services (e.g. Pandora.com [15], Last.fm [16]) own music search and recommendation functions, which also rank the results based on relevance, quantified by music similarity. The founder of Pandora.com has presented the most used approach which calculates the distance between the source song and each of the database song. Each distance is regarded as a function of the differences between the musical features of the source song and database song. In addition, some hybrid similarity measures were also presented in recent work (e.g., [8, 13]), which combine music features with social tags. There are also some combined approaches which rank the search results by both their importance and relevance to the query. GenJam [14] proposed an interactive computer system improvising over a set of jazz songs using genetic algorithms. Douglas Turnbull, Luke Barrington, etc in [10] presented a computer audition system that can both annotate novel audio tracks with semantically meaningful words and use a semantic query to retrieve relevant tracks form database of unlabeled audio content.

3. Overview of Music Search

Intelligent music search contains basic search and recommendation. The following figure 1 shows the whole process of music search and recommendation.

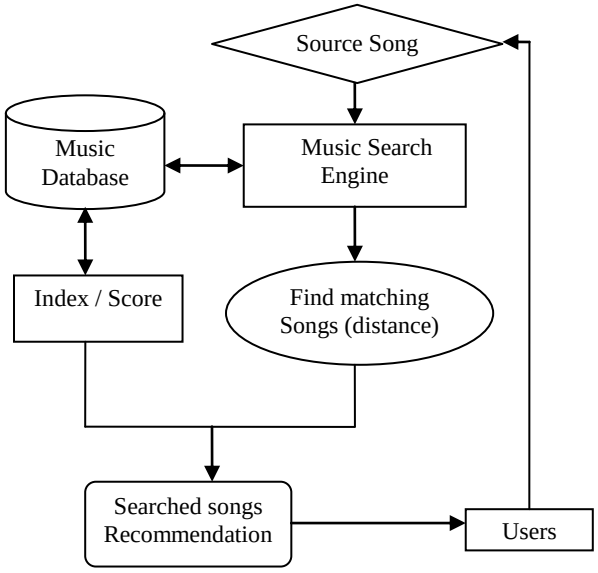

Figure 1, Overview of Music search

First of all, a music database has been built, which consists of two components: actually music songs and semantic index contents. The songs part contains the basic attributes of the music, and the indexing part contains four tables (music songs, songs with attributes, index term and index table). In addition, the recommendation will make prediction about what kind of music you are going to like next based on the search conditions and users feedback. More details will be described in the following sections.

4. Proposed Methodology

With the limitations of the current technologies, it's very difficult to extract the semantic content of multimedia data directly, especially for the music which is an art form. Thus indexing of these music data may become more costly and time-consuming. Therefore, this paper shall employ a novel music index approach to better support music search and retrieval.

4.1. Music Representation

The first task of music search is music representation. Without regard to the format of digital audio music, just think about the attributes and characteristics of music songs. Therefore, a given music song M is represented by an n-dimensional vector, which contains many different attributes (approximately 300 - 400). $M = <$ music_id, e_1, e_2, e_3, …, $e_n >$. Each attribute stand for a vector element, which is a characteristic of music songs. Suppose an n-dimensional music database vector which corresponds to the musical characteristics of a source song is determined. According to the Music Genome Project, a lot of music attributes are sorted by categories, which used for classifying music songs, for example Hip-Hop/R&B, Rock/Pop, Jazz/Blues, Country/Folk, etc.

By reference, Rock and Pop songs have 150 attributes; Rap songs has 350 attributes; And Jazz songs have approximately 400 attributes. Thus, these sufficient numbers of attributes have enough used to represent a given music. Each attribute is assigned a magnitude number which is between one and five, in have-integer increments every time. Thus, these attributes have been digitized. Therefore, the simple distance between any two songs in an n-dimensional space is able to be calculated by the Euclidean distance. As the different attribute may have different weight, thus the music song can be added a weighting vector $W = < w_1, w_2, w_3, ..., w_n >$. The sample formulation is in the following.

Music Representation:

Music Song $S = < s_1, s_2, s_3, \ldots, s_n >$;
Song Attribute Weight $W = < w_1, w_2, w_3, \ldots, w_n >$;
Where $1 < = s_n <= 5$, and $0 <= w_n <= 1$.

4.2. Indexing

Indexing is the process of collecting, parsing, and storing data to facilitate fast and accurate information retrieval. The index describes partial or whole content of the documents in one way or another. Index term may be related keywords, phrase which are meaningful, and it can be a well-defined hierarchy structures. Thus, without regard to the metadata of the music, for example song name, artist, album, the paper focus on the semantic content of the music data, for example style, mood, etc., which is more challenging than indexing metadata. The following part will describe the semantic content of music indexing approach in detail.

4.2.1. Semantic Indexing Approach

The proposed music indexing approach is different from traditional text indexing approaches. Let's consider a collection of music songs { S_j }, where their semantic features and characteristics cannot be directly and automatically extracted, it has been determined manually, as what Music Genome Project has done. Then every music song links with an index set { I_j }, and this set contains a number of elements e_{j1}, e_{j2}, e_{j3}, …, e_{jm}. And each element is made up of a triple which has three components: song_id, index term, and index score. The index score stands for the signification of the triple of the index set to that music song. If the index score is higher, the index term to that music song is more important. According to the music representation and index representation, the following relations show their whole representation and relationship.

Music M = < music_id, music_name, description >
Song S = < song_id, $attribute_1$, $attribute_2$, ..., $attribute_n$ >
Index Term I = < index_id, index_term_name >
Index Table T = < index_id, music_id, score >

Thus the index table has a many-to-many relationship between the music song and the index term. In order to effectively and efficiently index all the music songs in the database, hierarchy the built index is required. Let's layer the index according to the different intervals of the score, thus there are N levels L_1, L_2, …, L_n with a set of parameters P_1, P_2, …, P_n. Therefore, for the given score value x, if $P_i \leq x \leq P_{i+1}$, the given index term with score value x will be assigned to level L_i. According to this rule, all the index term will be placed in the suitable layer.

4.2.2. Index Score Updating

Though the music index has already built in the above, the operations (add, delete, update) of the music index are also required. Because the index score is directly affected by the user search behavior and feedback, modified reinforcement learning algorithm is able to update music index score. Reinforcement learning is to maximize some notion of long-term reward. Here is to get accurate music search result. According to the SQL query, the query score for a music song can easily get from the index table.

Suppose when user input search terms Q(T_1, T_2, T_3), music search engine will display the suggested N music songs (m_1, m_2, …, m_n) result in descending order by relevance (corresponding score s_1, s_2, …, s_n). Then the users will choose the result and give some feedback. When the user chooses the m_i in the result list, the s_i will be strengthened by α_1. In addition, if the user provides the promising comments, the s_i will be strengthened by α_2. Conversely, if the user does not choose any song from the search result list, the related index score on T_1, T_2, and T_3 for the related songs will be decreased by β_1, with the different probability $(1 - \varepsilon)$. Furthermore, if the negative feedback will be given to the engine, the penalty will be performed on the related songs in the database by different level γ. The following if statements show the above structure and calculations.

Algorithm 1: Updating Index Score

```
1.  If ( choose the mi in the result list) {
2.      Mi.score = Mi.score + α1
3.      If ( positive feedback ) {
4.          Mi.score = Mi.score + α1 + (1 – γ) * α2
5.      }
6.      Else {
7.          Mi.score = Mi.score + α1 - (1 – γ)
8.      }
9.  Else
10.     Mi.score = Mi.score – (1 – ε) * β1
```

4.2.3. Incremental and Decreased Index

Apart from updating the music index, in order to maintain the index become more comprehensive and complete, adding and deleting some index terms are necessary. As we all know, if we give more search terms (index term) in the query, the more accurate search results will be gain. Let's consider this situation: when a user input a symphony "Eine kleine Nachtmusik", which is a very famous song composed by Mozart. However, the index table only maintains the several index terms, which don't contain the composer of the song. Thus when the users search this song by inputting Mozart, they may be not able to find this song in the return list. Therefore, adding the new index term "Mozart" to the index table is necessary.

Because the index is hierarchy built, properly assigning the score to the new index and placing it to the suitable hierarchy is very important, which affects the future search results. Suppose a new index term is added, the score of the related music song to the new index term is initially related to the highest score (S_{max}) of that music song in the index table. Let's give a level factor μ ($0 \leq \mu \leq 1$) to the added index term. Thus, the initially index score to the specific song can be gained by $(1 - \mu) * S_{max}$. Thus, the new index has been properly added, with the future continuously updating this index score, the search result by this index term will become more accurate.

In theory, the more indexes, the better search results. However, consider the time cost, keeping a proper number of indexes to a specific music is promising. Sometimes some existed index term may have a very low relevance with some music. Thus, in order to decrease the complexity of the computation and speed up search process, deleting those index terms and some records in the index table are required, which will keep the database complete and flexible. Suppose an existed record in index table is deleted, the delete criteria is related to the highest score (S_{max}) of that music song in the index table. Let's give a threshold factor Θ ($0 < \Theta \leq 0.05$), if the index score of the specific music song is smaller than $\Theta * S_{max}$, those records satisfying the above condition will be delete. Therefore, according to adding and deleting index terms or records, the music database and search engine will maintain an efficient and effective performance.

Algorithm 2: Increasing and Deleting Index Term

```
1. If (add new index term)
2.     mj.score = (1 - μ) * Smax
3. If (delete index term or index table record)
4.     While (mj.score < Θ * Smax)
5.         Delete (Ij)
```

4.3. Matching

Though the above indexing of music is able to successfully retrieve music, it focuses on the semantic music content. In order to consider more music attributes or characteristics, the paper has used a hybrid matching approach to find the similar songs. According to the music representation, considering given a song $S = < s_1, s_2, s_3, \ldots, s_n>$ and a song $T = <t_1, t_2, t_3, \ldots, t_n>$, and each attributes of these vectors have been assigned a number between zero and five, in have-integer increments for every value. Thus, the simple distance between these two songs in n-dimensional space can be calculated by the following formulation:

$$\text{Distance}(S, T) = \text{sqrt}[\ \sum(s_i - t_i)\sim 2\ ], \text{ for } i = 1 \text{ to } n \quad (1)$$

When consider the weight $W = < w_1, w_2, w_3, \ldots, w_n >$ of these attributes of music songs, where $1 <= s_n <= 5$, and $0 <= w_n <= 1$, the revised distance can be calculated as follows:

$$\text{Distance}(S, T) = \text{sqrt}[\ w_i * \sum(s_i - t_i)\sim 2\ ], \text{ for } i = 1 \text{ to } n \quad (2)$$

In order to provide the user a flexibly control to the matching behavior, customizing some search conditions of the music attributes is required, which may be used to re-weight the song of the above matching approach and refine some searches for matching songs to include or exclude some selected attributes. Suppose when a user choose to modify the weighting vector, for example, increasing some weights of the attributes which are specific to the selected conditions, to complete particular matching result. Thus, the new resulting songs will be resembled closely to the source song in the new selected conditions.

4.4. Recommendation

Recommendation attempts to recommend information items (music, etc.) that are likely to be interest to the user. So they give us what they think we want, based on what we and other people like us have wanted in the past experience. Most current existent recommendation engines work backward instead, using information that comes not from the art but from their customers or audience, and they work on the principle that the behavior of a lot of people can be used to make educated guesses about the behavior of a single individual. Here is the idea: if most people who liked "The Second Waltz" also like "Medley Strauss And Co", then if we know that a particular individual liked "The Second Waltz", we can make an educated guess that that individual will also like "Medley Strauss And Co". This technique called collaborative filtering. Take music recommendation for example, the key point to grasp about collaborative filtering is that it knows absolutely nothing about the music, which has no preconceptions, and it works entirely on the basis of the audience's reaction. However, this mechanism may result some problem: when most people claim to have enjoyed "The Second Waltz" and also liked "Eine Kleine Nachtmusik K.525", the recommendation engine would be forced to infer that those two songs share some common quality that the users liked. Collaborative filtering works only as well as the data it has available, and humans produce noisy, low-quality data. Therefore, collaborative filtering has to improve to get accurate predictions. So the recommendation engine should use the information not only from the audience but also from the Intrinsic of art.

According to above song to song matching approach, the computation of the distance of the song in the same

category can be calculated. Let's give a threshold σ, all the distance smaller than σ will be recommended to the users. By improving above matching approach, multi-song matching approach can be used to strength recommendation. It builds functionality that will return the best matches to a group of source songs. This functionality provides a list of songs which are similar to the collection of an artist or album. Thus it will generate recommendations for the users, purely on their taste, without any source songs. Let's group the songs into a single virtual song, and the virtual center is defined to be a song vector whose attributes are the arithmetic average of the songs in the collection. Then associating this center a deviation vector represents the distribution of the songs within the collection. The following figure 2 shows the process of multi-song matching. So an individual attribute which has narrow distribution of values around the average value will have a strong affinity for the center value. Therefore, the small deviation will be assigned higher weight.

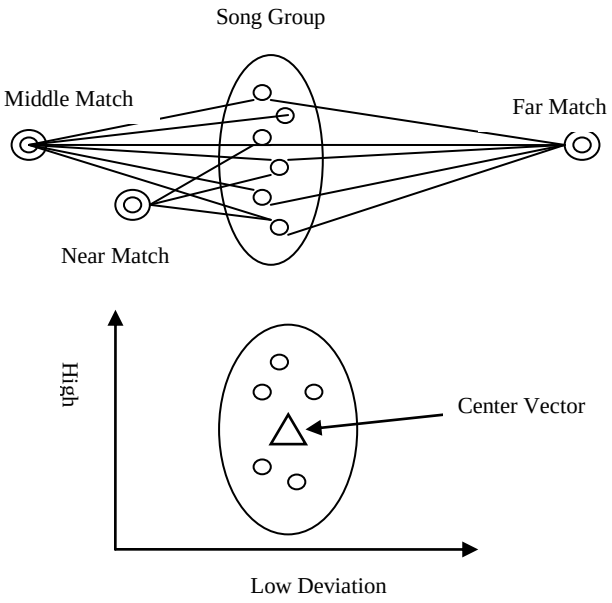

Figure 2, Multi-song Matching

Suppose the center vector of the selected song group $C = < c_1, c_2, \ldots, c_n >$, the standard deviation vector of the song group $D = < d_1, d_2, \ldots, d_n >$, thus the distance between the target vector to the center vector is in the following formulation:

$$\text{Distance}(S, T) = \text{sqrt}\left[\sum(1 / d_i) \sim 2 \ * (c_i - t_i) \sim 2 \right] \qquad (3)$$

Where i from 1 to n; thus the smallest distance are the best matches, thus those songs with smallest distance will be recommended to the users.

4.5. Browsing

In order to make users conveniently find what they want in the search result list, which music songs to present to the user and in which order is related to browsing. Ranking music data by relevance and importance has been proposed by Maria M.Ruxanda, etc. in [12]. According to the previous music indexing, the simple approach is to rank the result by the index score in descending order. Thus the high scored music songs in the index table are always ranked in the top of the result list. Because the users choosing the top music list has a very large probability, the scores of these music songs have a great chance to be increased. Thus those new added songs which have great relevance to the search term as the above issue will be ranked lowly, which have affected the users' behavior. In addition, initially the search engine may be not work perfectly, thus the top search result may not what the users really want.

As the above issues, optimizing the search result will solve these problems. Genetic algorithm generates solutions to optimization problems using techniques inspired by natural evolution. Thus, by genetic algorithm, those related songs which ranked lowly will have a chance to place in a top result list, so the users will find those near hidden , actually important songs. Consider the above situation, let's give a probability to the each song Mi with a corresponding index score Si in the result list. Suppose there are N songs in the result list, according to the probability theory, initially the score of each returned song can be calculated by the following formulation:

$$P_i = S_i / \sum S_j \text{ , where i, j from 1 to N} \qquad (4)$$

Thus the songs with higher scores would have a higher probability. So those songs with higher probability would a higher rank in the search result list. After initializing, then it would search for those index terms, and then select acceptable songs from the search result list and mark down some index terms from it. This procedure would utilize a fitness function. Fitness evaluation is based on external feedback from the users. So do this until the results are approximately what you are really looking for, then stop do it if you are searching over and over for many times but you are not getting good results.

5. Observations

In order to best evaluate the suggested indexing and recommending approaches on music retrieving, some observations have to make. Initially, as the index term is limited, the search result not always works perfectly. With gradually adding new index term to the music database, the search result will be more accurate. Then though the

users continue to increase new indices, the accurate of search results tend towards an equilibrium level. When users set different search conditions, the more search terms input, the more accurate result gain. If the dimensional of music representation is very large, the songs will be effectively distinguished, while the computation may be very complex. On the contrary, it is not able to accurately recommend songs the user likes.

As improved ranking is a genetic algorithm it also suffers from the shortcomings that genetic algorithm has. First of all adaptive search depends on your first initial guess, which may become so unrelated that your search results are really bad. Secondly, it depends on your judgment of accepting a result. If you expect results too soon and accept no result you may end up unhappy. On the other hand, if you expect all results you may end up doing too many searches with all unrelated results.

6. Conclusions and Future Work

This paper presents a semantic indexing method that is capable of flexibly retrieval music in semantic level. In order to retrieve music accurately, first of all, music representation in a given song is represented by an n-dimensional vector, which contains many different attributes. Then four tables (music, songs, index term, index tables) have been established to semantic index. After that some algorithms and formulations of index score updating, increasing, and decreasing have been detailed explained. In addition, useful and flexible music recommendation approaches, for example song-to-song matching, and multi-song matching methods, has been detailed introduced to make better recommendations to the users. Finally, improved genetic algorithm is used to optimize the ranking of the search results list when users browse. Thus, the above proposed methods can be useful when incorporated into commercial music search engines for improving music service.

In future work, other new schemes that adopted in semantic indexing and retrieval areas will be investigated. In addition, new intelligent music recommendation which based on the intrinsic of musical art will also be studied, whether content-based filtering or collaborative filtering, with the popularity of the social media [11]. As well, the suggested music search engines will be constructed, and some experiments, evaluations and user feedbacks will be conducted, to evaluate user's satisfaction with respect to the proposed music search and recommendation methods.

References

[1] C. H. C. Leung, J. Liu, W. S. Chan, and A. Milani. An Architectural Paradigm for Collaborative Semantic Indexing of Multimedia Data Objects. Proceedings of the 10th international conference on Visual Information Systems: Web-Based Visual Information Search and Management, Vol. 5188, pp.216 – 226, 2008

[2] Leung, C. and Liu, J. Multimedia data mining and searching through dynamic index evolution. In 9th proceedings of advances in Visual Information systems, Shanghai, China, pp.298-309, 2007.

[3] Gros, P., Delakis, M., and Gravier, G. Multimedia Indexing: The Multimedia Challenge. In ERCIM News No. 62, 2005.

[4] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic Annotation and Retrieval of Music and Sound Effects. Audio, Speech, and Language Processing, IEEE Transactions on Issue Date: Feb. 2008, Vol.16, pp. 467 – 476.

[5] Nicola Orio. Music Retrieval: A Tutorial and Review. Foundations and Trends in Information Retrieval, Volume1, Issue 1, Pages 1-96, 2006.

[6] Alexandros Nanopoulos, Dimitrios Rafailidis, Maria M. Ruxanda, Yannis Manolopoulos: Music search engines: Specifications and challenges. Inf. Process. Manage. 45(3): pp.392-396, 2009.

[7] Donald Byrd, Tim Crawford: Problems of music information retrieval in the real world. Inf. Process. Manage. 38(2): pp.249-272, 2002.

[8] R. Stenzel and T. Kamps. Improving content-based similarity measures by training a collaborative model, in Proc. ISMIR, 2005, pp. 2264-271.

[9] C. R. Buchanan Semantic-based audio recognition and retrieval, pp. 2005.

[10] D. Turnbull, L. Barrington, D. Torres and G. Lanckriet "Towards musical query-by-semantic description using the CAL500 data set", Proc. SIGIR"07, pp. 439 2007

[11] Music Information Retrieval Using Social Tags and Audio, Levy, M. Sandler, M. Multimedia, IEEE Transactions on page(s): 383 - 395, Volume: 11 Issue: 3, April 2009

[12] Maria M.Ruxanda, Alexandros Nanopoulos, Christian S. Jensen, Yannis Manolopoulos. Ranking music data by relevance and importance. Multimedia and Expo, 2008 IEEE international conference, pp.549- 552, 2008.

[13] C. R. Buchanan J. Kandola, J. Shawe-Taylor, and N. Cristianini. Learning Semantic Similarity, in Proc. NIPS, 2002, pp.657-664.

[14] Biles, J.A. GenJam: a genetic algorithm for generation of jazz sols. In Proceedings of the International Computer Music Conference. Aarhus, Denmark, 1994.

[15] Pandora.cm. www.pandora.com

[16] Last.fm . www.last.fm