

Improving Music Genre Classification Using Collaborative Tagging Data

Ling Chen
L3S Research Center
Leibniz University Hannover
lchen@l3s.de

Phillip Wright
College of Computing
Georgia Institute of
Technology
phillip.wright@cc.gatech.edu

Wolfgang Nejdl
L3S Research Center
Leibniz University Hannover
nejdl@l3s.de

ABSTRACT

As a fundamental and critical component of music information retrieval (MIR) systems, music genre classification has attracted considerable research attention. Automatically classifying music by genre is, however, a challenging problem due to the fact that music is an evolving art. While most of the existing work categorizes music using features extracted from music audio signals, in this paper, we propose to exploit the semantic information embedded in *tags* supplied by users of social networking websites. Particularly, we consider the tag information by creating a graph of tracks so that tracks are neighbors if they are similar in terms of their associated tags. Two classification methods based on the track graph are developed. The first one employs a classification scheme which simultaneously considers the audio content and neighborhood of tracks. In contrast, the second one is a two-level classifier which initializes genre label for unknown tracks using their audio content, and then iteratively updates the genres considering the influence from their neighbors. A set of optimizing strategies are designed for the purpose of further enhancing the quality of the two-level classifier. Extensive experiments are conducted on real-world data collected from Last.fm. Promising experimental results demonstrate the benefit of using tags for accurate music genre classification.

Categories and Subject Descriptors

H.3 [Information Systems]: Information Storage and Retrieval; J.5 [Computer Applications]: Arts and Humanities—*music*

General Terms

Algorithms, Experimentation, Performance

Keywords

Music genre classification, exploiting tag semantics, relax-

ation labelling

1. INTRODUCTION

Due to the rapid advancement of digital technology in the last two decades, there has been an increasingly large amount of music files available on the Web. It is very likely that in the near future all recorded music in human history will be available online [26]. The enormous—and continuously growing—volume of online music data necessitates the development of efficient and effective music information retrieval (MIR) systems. Although most of the traditional music databases index songs by title or artist name, it was observed that people often need to search music by music content instead [8]. Music genre (e.g., Jazz, Rock, Country etc.) is one of the top-level descriptions of music content. Consequently, accurate music genre classification is a critical step toward the success of modern music information retrieval systems.

Most of the existing MIR systems (e.g., AllMusic Guide¹ and MSN Music Search Engine) rely on human experts as well as amateurs to categorize music by genres. However, this kind of approach is not only expensive but also time-consuming. As noted by Weare [9], Microsoft required the assistance of 30 musicologists over a period of one year in order to manually label a “few hundred thousand songs”. Not surprisingly, huge research interest has been triggered to perform the task *automatically*. Aucouturier and Pachet [2] summarized the representative efforts in this regard (a survey on automatic genre classification of music content can be found in [23]), where most of the proposed approaches trained their classifiers with low-level features extracted from music audio signals.

Automatically classifying music by genre is a challenging problem considering that music is an evolving art, where performers and composers have been influenced by music in other genres [15]. Hence, automatic classification approaches depending on low-level features alone may not be able to obtain satisfactory results. As observed and suggested in [20], incorporating other features, such as high-level features based on music abstractions, could result in performance improvements.

With the recent rise of Web 2.0 technologies, Web users from different backgrounds began to annotate/tag resources on the Web at an incredible speed. Many social media applications like Flickr, Del.ici.ous, and Last.fm provide features which allow users to assign tags to information on the web,

¹<http://www.allmusic.com>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSDM'09, February 09-12, 2009, Barcelona, Spain.
Copyright 2009 ACM 978-1-60558-390-7 ...\$5.00.

i.e. images, blog entries, or video clips. This emerging meta-data provides useful information which has been explored in applications such as navigation [11], enterprise search [10] and Web search [3]. We believe that the valuable information embedded in user-supplied tags can be beneficial in many other applications. Particularly, in this paper, we investigate *performing automatic music genre classification by taking advantage of available tag data*. To the best of our knowledge, this is the first attempt in the literature which exploits the usage of tags in music genre classification.

Using tags to facilitate music genre classification is a non-trivial problem due to the fact that tags are usually *freely* chosen by users. The varied tags supplied by different users provide not only related information but also noise which is either useless or ambiguous. For example, the track “I hope you dance” by Lee Ann Womack was labelled by Last.fm users with tags including *country* as well as *songs to remember*. While the former indicates the genre of the track, the latter does not seem to be helpful. Furthermore, it is often observed that a track is assigned tags which imply different genres, such as *blues* and *rock*².

In our work, we take into account the tag information by constructing a graph of tracks so that tracks are neighbors if they are semantically similar in terms of their associated tags. Two classification methods, integrating audio content information as well as tag information, are developed based on the track graph. The first method, called *Single-Layer Classifier*, simultaneously considers the audio content of tracks and the tag-based neighbors. The second one, called *Double-Layer Classifier*, firstly estimates the genres for unknown (unclassified) tracks using the audio content information, and then improves the estimation by updating the genre of each unknown track based on not only its known neighbors but also its unknown neighbors. A set of optimizing strategies are also designed to further improve the quality of the *Double-Layer Classifier*. Performance evaluations are conducted on a set of tracks and tags collected from Last.fm³, an internet radio and music community website. The promising results demonstrate the usefulness of tags in music genre classification.

The rest of the paper is organized as follows. Section 2 presents background information by reviewing related work in the literature. In Section 3, we formally define the classification problem and describe the two classification schemes. Section 4 evaluates the performance of the two classifiers. In Section 5, we discuss the optimizing strategies we adopt to further improve the performance of the *Double-Layer Classifier*. Finally, we draw some conclusions and discuss our future work in Section 6.

2. RELATED WORK

We address related work from two related research areas, music genre classification, collaborative tagging systems, and collective classification.

The state-of-the-art in music genre classification primarily focuses on classifying music based on low-level features extracted from audio signals. Most of the existing work proceeds in two steps: *feature extraction* and *multi-class classification*. As summarized in [23], three sets of fea-

tures for representing timbral texture, rhythmic content, and melodic and harmonic content are widely used in the literature. Features characterizing timbre mainly analyze the spectral distribution of the signal. G. Tzanetakis et al. [26] described some of the most common timbral features: Mel-Frequency Cepstral Coefficients (MFCC), Spectral Centroid, Spectral Rolloff, Spectral Flux, Zero Crossings and Low Energy. Rhythmic content refers to the temporal regularity of a music work. The features for representing rhythmic content are based on detecting the most salient periodicities of the signal. An in-depth study on low-level rhythmic descriptors extracted from different periodicity representations is given in [13]. Harmony represents the actual or implied pitch simultaneity in music, while melody is defined as a succession of pitched events perceived as a single entity. Although melodic and harmonic analysis have been used by musicologists to study music structures for a long time, there have only been limited attempts [26] at building a genre classifier based on pitch features. After a set of relevant and significant features are extracted, different classification schemes can be applied. For example, the K-Nearest Neighbor (KNN) is probably the simplest classifier which assigns a song with the genre most commonly represented in its K neighbors [21]. Other classification techniques used in music genre classification include Support Vector Machines (SVM) [18], Gaussian Mixture Models (GMM) [6], and Linear Discriminant Analysis (LDA) [27]. Our method can be applied on top of any of these automatic music classification algorithms based on low-level features (e.g., these methods can serve as the Base Classifier in our approach). A recent direction in music genre classification is to combine features from multiple aspects. For example, some promising results have been obtained in [19] by combining low-level features with high level ones. Whitman and Smaragdis [28] combined cultural features, such as terms frequently occurring together with artists in Web pages, with low-level features and achieved significantly improved performance over the use of low-level features alone.

Tags have recently become popular as a means of annotating objects on the Web. Golder and Huberman [12] provided a detailed account of different types of tagging systems. Halpin et al. [14] studied the dynamics of tagging data and pointed out that the frequency distribution of the tag set for popular sites from Del.icio.us follows a power law. Besides analyzing the trends and properties of various tagging systems, some efforts have explored the usage of tags in Web applications. For example, P. Schmitz [24] discussed extracting an ontology from Flickr tags. X. Wu [29] used a tripartite conceptual model to derive emergent semantics in annotated social bookmark services. The learned semantics were applied to discover and search for shared Web bookmarks. In [10], the authors discussed using tags to lighten the limitation of the amount and quality of anchor text to improve enterprise search. The usage of tags in Web search has also been investigated in [3]. There are also a few works applying tags in data mining tasks. Berendt and Hanser [4] compared the performance of blog post classification using features derived from tags, titles and body. They found that tags together with body yielded better classification accuracy than any other feature alone. Brooks and Montanez [5] performed unsupervised learning on blogs using tags, which showed tags are useful in clustering blog posts into broad categories.

²Classifying music tracks sitting on the boundaries between genres is beyond the scope of this paper.

³<http://www.last.fm>

Another related area is collective classification. Collective classification refers to the simultaneous prediction of vertex labels of a graph. The intuitive is that the label of a vertex can be influenced by its own attributes and the labels and attributes of other nodes in the graph. Many collective classification methods [16, 17, 25, 7] have been proposed in the literature which model such dependencies differently. However, most the existing works focus on classifying documents. To the best of our knowledge, this paper is the first effort which investigates the effectiveness of collective classification on collaborative tagging data. Considering that tag data is noisy, several feature processing strategies are proposed to improve the classification performance.

3. GENRE CLASSIFICATION

In this section, we first define the music genre classification problem. After introducing the graph structure which models music tracks and the semantic similarity measure between tracks, two classification schemes, considering tag information as well as audio content information, are illustrated respectively.

3.1 Problem Description

It is extremely difficult to give a precise definition of a music genre [15]. Similarly, there is no consensus on the construction of music genre taxonomies [2]. All the existing work on automatic genre classification, which either classifies music to a list of genres or to a hierarchy of genres, makes the same assumption that a genre list/taxonomy is given and should be imposed on the music database. In this paper, we make the same assumption and consider the problem of automatically classifying music to a list of genres – though our approach can be extended easily to classify music into a hierarchy of music genres.

In the context considered in this paper, the music genre classification problem can be specified as follows. Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a set of music tracks, where each track x_i is associated with a set of low-level audio signal features $\Gamma(x_i)$ and a set of tags $T(x_i)$. Let $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$ be a list of music genres such that $\forall x_i \in \mathcal{X}$, the label of x_i is $C(x_i) \in \mathcal{C}$. The goal is to infer a function $f_{\mathcal{X} \rightarrow \mathcal{C}}$ which sufficiently approximates the mapping of the set of music tracks to the list of music genres.

3.2 Graph Structure

A straightforward solution which classifies music using tag data in addition to audio content data is an *ensemble method*, which constructs two base classifiers using the two sources of information respectively and performs classification by taking a vote on the predictions made by each classifier. The base classifier which focuses on tag information can directly use tags as features. However, due to the data sparsity problem, we consider tag information indirectly in our work. Particularly, we construct a graph of tracks so that tracks which are semantically similar, in terms of tags that are assigned to them, are adjacent nodes in the graph.

Given the complete set of tags associated with all tracks, $\mathcal{T} = \{t_j | t_j \in T(x_i) \wedge x_i \in \mathcal{X}\}$, we first compute a *TFIDF*-like score for each tag t_j associated with a track x_i as below:

$$TFIDF(t_j, x_i) = Freq(t_j, x_i) \times \log \frac{|\mathcal{X}|}{\sum_{x_i \in \mathcal{X}} Occurs(t_j, x_i)}$$

where $Freq(t_j, x_i)$ is the frequency that t_j was assigned

to x_i , $|\mathcal{X}|$ is the number of all tracks in the dataset, and $Occurs(t_j, x_i)$ is a boolean function returning 1 if t_j is used to annotate track x_i and 0 otherwise.

After that, each track x_i is associated with a vector, $\vec{T}(x_i)$, of length $|\mathcal{T}|$. Each element of $\vec{T}(x_i)$ is the TFIDF value of a tag associated with the current track. We define the similarity between two tracks x_i and x_j , denoted as $Sim(x_i, x_j)$, as the cosine similarity of the two corresponding vectors $\vec{T}(x_i)$ and $\vec{T}(x_j)$:

$$Sim(x_i, x_j) = \frac{\vec{T}(x_i) \cdot \vec{T}(x_j)}{\|\vec{T}(x_i)\| \|\vec{T}(x_j)\|}$$

A graph $G = (N, E)$ can then be created, with N being the set of nodes, each representing a track, and E the set of edges, containing only edges $(x_i, x_j) \in E$ with $Sim(x_i, x_j)$ above a certain threshold ϵ . The terms *node* and *track* are used interchangeably hereafter.

3.3 Single-layer Classification

Given the audio content of tracks, $\Gamma(x)$, as well as the graph structure G , which reflects the similarities between tracks in terms of tags, we assign a genre c_i to a track x_i so that the probability $Pr(c_i | \Gamma(x_i), G)$ is maximized. Based on the *Markov Random Field* (MRF) assumption that the class of a node is conditionally independent of the classes of other nodes in a graph, given the classes of its immediate neighbors, we have $Pr(c_i | \Gamma(x_i), G) = Pr(c_i | \Gamma(x_i), N(x_i))$, where $N(x_i)$ is the set of neighbors of x_i . Assuming that the audio content of a track has no direct coupling with its neighbors' genres, and manipulating with Bayes Rule, the probability can be computed as follows:

$$Pr(c_i | \Gamma(x_i), N(x_i)) \propto Pr(c_i) Pr(\Gamma(x_i) | c_i) Pr(N(x_i) | c_i) \quad (1)$$

By further assuming the independence between each pair of nodes in the neighborhood of x_i , the right side of Equation 1 can be written as:

$$Pr(c_i) Pr(\Gamma(x_i) | c_i) \prod_{x_j \in N(x_i)} Pr(C(x_j) | c_i) \quad (2)$$

Thus, the genre of x_i will be

$$c_i = \arg \max_{c_i} Pr(c_i) Pr(\Gamma(x_i) | c_i) \prod_{x_j \in N(x_i)} Pr(C(x_j) | c_i) \quad (3)$$

It can be observed from Equation 3 that this classification scheme simultaneously considers the audio content information and tag information. For easy reference, we call this classifier *Single-Layer Classifier*. Note that, the audio content information is taken into account explicitly, while the tag information is studied implicitly by considering the class labels of neighboring tracks with similar tags.

3.4 Double-layer Classification

In order to enhance the classification quality, learning from unknown data has been used in hypertext categorization [7]. In our work, we adopt this idea to consider neighborhoods including not only tracks with known genres but also unknown tracks whose genre labels need to be predicted. Then, the *Relaxation Labelling* technique [22] is used to iteratively update the genre labels for unknown tracks.

Let Δ^k denote all of the known information (e.g., the audio content of tracks and the known genre labels of tracks),

and $N^u(x_i)$ be the set of unknown neighbors of x_i . We aim to find the class c_i for x_i to maximize the probability $Pr(c_i|\Delta^k)$, which can be written as a summation over unknown neighbors:

$$Pr(c_i|\Delta^k) = \sum_{N^u(x_i)} Pr(c_i|N^u(x_i), \Delta^k) Pr(N^u(x_i)|\Delta^k) \quad (4)$$

Based on the MRF assumption and the independence assumption among neighbors as described in the previous subsection, the two components on the right side of Equation 4 respectively yield:

$$Pr(c_i|N^u(x_i), \Delta^k) = Pr(c_i|N^u(x_i), N^k(x_i)) \quad (5)$$

$$Pr(N^u(x_i)|\Delta^k) = \prod_{x_j \in N^u(x_i)} Pr(C(x_j)|\Delta^k) \quad (6)$$

where $N^k(x_i)$ refers to the set of known neighbors of x_i . Combining Equations 5 and 6 with Equation 4, an iterative process can be obtained:

$$Pr(c_i|\Delta^k)^{(r+1)} = \sum_{N^u(x_i)} [Pr(c_i|N^u(x_i), N^k(x_i)) \prod_{x_j \in N^u(x_i)} Pr(C(x_j)|\Delta^k)^{(r)}] \quad (7)$$

Equation 7 illustrates the framework of the double-layer classifier, which is presented in Figure 1. Basically, there are two classifiers involved: a *Base Classifier*, and a *Graph Classifier* built on top of the Base Classifier. Given the set of known tracks (training data) and the set of unknown tracks (test data), the Base Classifier estimates the genres of unknown tracks using the known information. That is, the component $Pr(C(x_j)|\Delta^k)$ in Equation 7 will be computed. In our approach, we calculate this probability by employing a Naive Bayes Classifier using the audio content information, though other classification schemes (e.g. the *Single-Layer Classifier* described in the previous subsection) and information sources (e.g. both audio content and tag data) can be used as well.

Next, a *Graph Classifier* is used to iteratively update the genres of unknown tracks according to Equation 7. The degree to which the genres estimated in the previous iteration are taken into account in this iteration is controlled by $Pr(c_i|N^u(x_i), N^k(x_i))$, which, based on Bayes Rule, can be manipulated as below:

$$Pr(c_i|N^u(x_i), N^k(x_i)) = Pr(c_i) \prod_{x_j \in N^u(x_i) \cup N^k(x_i)} Pr(C(x_j)|c_i) \quad (8)$$

The relaxation procedure described in Equation 7 is guaranteed to converge to a locally consistent assignment of genre labels. In addition, since the computation of Equation 7 is intractable, we adopt the *hard labelling* technique in [1]. The basic idea is to approximate Equation 8 by considering only the most probable neighborhood labelling. That is, for each unknown neighbor x_j , we calculate using its most probable label $C_{max}(x_j)$ as of the previous iteration.

4. EXPERIMENTS

In this section, we first describe the data set used in our experiments. Then, we respectively evaluate the performance of each individual classifier.

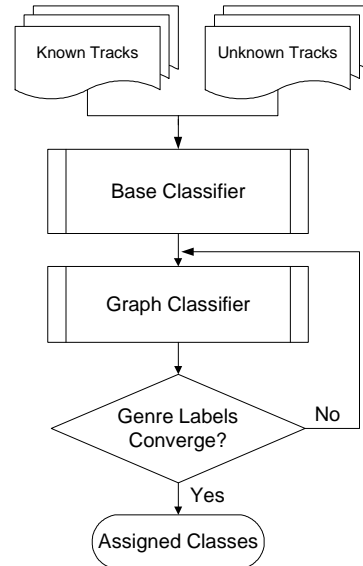


Figure 1: The Framework of the Double-Layer Classifier.

Table 1: Track genre distribution

1	Jazz	451	4	R&B	287
2	Rock	287	5	Electronica	386
3	Country	602	6	Rap	249

4.1 Data Set

Three sets of data were collected for our experiments: audio tracks, tag data and genre data. Over 10,000 MP3 files were crawled from the Last.fm site by accessing “radio stations” based on genre and keywords. We then used the jAudioTagger library⁴ to extract the artist and title information from the ID3 tags contained in each MP3 file. With the artist and title information, we collected the “ground truth” genre data for each track from the All Music Guide, a popular edited Web resource for music information. Tracks belonging to genre classes with too few class representatives were removed. The remaining 2,262 tracks represent six genres, *Jazz*, *Rock*, *Country*, *R&B*, *Electronica* and *Rap*, with the class distribution shown in Table 1.

Then, the Last.fm API was used to download the tags for each artist-track pair. A total of 17,383 tags were collected. Each track has at most 99 tags and at least 1 tag. On average, each track is associated with 29.9 tags. Note that, we did not prune tracks with very few tags, because we are interested in examining the robustness of the classification approaches when classifying such tracks in addition to more frequently tagged tracks.

All of the audio tracks were converted into 22K, 16bit, mono WAV files, which were further processed using the MARSYAS system⁵ — an open source software framework for audio processing — to extract low-level features such as MFCC, spectral centroid, spectral rolloff, and spectral flux.

⁴<http://www.jthink.net/jaudiotagger>

⁵<http://marsyas.sness.net>

Table 2: Baseline accuracy

Training Data	Classes				
	1-2	1-3	1-4	1-5	1-6
10%	66.53	58.93	49.43	45.09	42.33
20%	67.24	60.45	51.78	46.77	43.97
30%	68.07	61.80	53.16	48.68	44.84
40%	67.60	61.87	53.24	48.53	44.93
50%	69.32	62.52	53.50	49.41	45.35

(a) Using Naive Bayes on Low-level Features

Training Data	Classes				
	1-2	1-3	1-4	1-5	1-6
10%	72.93	63.34	52.84	46.42	44.61
20%	76.31	66.88	56.01	48.92	46.78
30%	77.43	68.78	57.29	49.12	47.68
40%	77.95	68.63	58.11	50.71	48.67
50%	78.89	68.85	58.78	51.10	49.43

(b) Using SVM on Low-level Features

Table 3: Single-layer classification accuracy

Training Data	Classes				
	1-2	1-3	1-4	1-5	1-6
10%	78.36	70.06	60.10	57.44	54.25
20%	83.64	71.61	63.28	60.36	50.09
30%	83.72	73.36	64.68	63.61	61.91
40%	86.25	74.03	66.98	66.27	64.67
50%	87.41	75.60	68.77	68.37	66.01

4.2 Performance Evaluation

Baseline. In order to evaluate the idea of classifying music using tags, we first conduct experiments for baseline approaches which perform music genre classification using audio content data only. Two classifiers, Naive Bayes (NB) and Support Vector Machine (SVM), are trained on the set of low-level features extracted from tracks. We carried out the experiments by varying the number of classes and the percentage of training data. The resulting accuracy is presented in Tables 2 (a) and (b) respectively. The class numbers correspond to those of Table 1 (which is same for Tables 3 and 4). Each result is averaged over 30 runs of the method (as are all of the following results presented in this paper). It can be observed that SVM surpasses NB slightly. While, the performance of both approaches is limited when all of the six classes are taken into account.

Single-Layer Classification. We conduct similar experiments with the *Single-Layer Classifier* (i.e. Equation 3), which simultaneously considers the audio content information as well as the tag information. When constructing the track graph, the similarity threshold ϵ is set to 0.2. As shown in Table 3, the accuracy of the single-layer classification surpasses all corresponding values in Table 2 which demonstrates the usefulness of tags in music genre classification.

Double-Layer Classification. We then investigate the performance of the *Double-Layer Classifier*. Similarly, the similarity threshold ϵ is set to 0.2 in building the graph structure.

Table 4: Double-layer classification performance

Training Data	Classes				
	1-2	1-3	1-4	1-5	1-6
5%	60.07	59.75	63.14	62.97	60.53
10%	94.59	70.82	73.71	72.38	71.74
15%	94.79	84.32	87.73	84.41	83.91
20%	95.29	82.80	87.90	86.41	86.57
50%	95.26	91.66	88.85	87.45	88.32

(a) Double-Layer Classification Accuracy

Training Data	Classes				
	1-2	1-3	1-4	1-5	1-6
5%	70.08	66.12	66.23	65.23	60.67
10%	93.93	72.00	75.85	71.74	72.22
15%	94.46	78.78	85.79	83.07	82.90
20%	95.15	80.40	85.45	83.85	83.13
50%	94.95	87.89	86.73	85.61	85.41

(b) Double-Layer Classification Recall

Table 5: Experiments with variation on similarity threshold

Training Data	Similarity Threshold				
	0.1	0.2	0.3	0.4	0.5
5%	54.40	60.53	61.25	55.57	54.23
10%	70.09	71.74	71.78	68.25	67.09
15%	82.89	83.91	83.40	81.40	81.39
20%	86.36	86.57	86.07	85.98	84.49
50%	87.06	88.32	88.11	88.05	88.12

The accuracy values are shown in Table 4 (a), where we examine on an extended range of the size of the training data. Compared with the baseline approaches, the results show again that tags are beneficial in improving the performance of music genre classification. Compared with the *Single-Layer Classifier*, the performance of the *Double-Layer Classifier* with 10% training data is even better than the performance of the *Single-Layer Classifier* with 50% training data. The probable reason is that the *Double-Layer Classifier* utilizes not only known tracks but also unknown tracks in inferring the label of an unknown track. Hence, if unknown neighbors provide reliable information in pulling the test tracks to the right class, the performance can be good as well even with fewer number of known neighbors. Thus, the good performance of the *Double-Layer Classifier* verifies that tag data is reliable information source in discovering neighboring tracks of same genres. The recall values of the *Double-Layer Classifier* are shown in Table 4 (b).

We further evaluate the performance of the *Double-Layer Classifier* by varying the similarity threshold ϵ from 0.1 to 0.5. Table 5 presents the results on all 6 classes, from which we notice that the accuracy is dependent on the similarity threshold, especially when the size of the training data is small. When the similarity threshold is low (e.g., 0.1), each unknown node has a relatively larger neighborhood – so the chances are greater that its neighbors belong to genres different from the true genre of the test node. Hence, the accuracy is poor when the similarity threshold is set too low.

Table 6: Per class performance of double-layer classifier

5% Training Data	Class					
	Jazz	Rock	Country	R&B	Electronica	Rap
Precision	81.58	43.77	75.53	39.17	70.74	52.38
Recall	75.57	22.81	88.18	57.54	72.44	48.47

On the other hand, when the similarity threshold is high, the neighborhood size shrinks so that it does not provide sufficient influence any more. That’s why the accuracy falls down when the similarity threshold is increased.

5. OPTIMIZING STRATEGIES

Although the *Double-Layer Classifier* performs well when there are enough known tracks, its performance is limited when a smaller set of training data is used. We consider it expensive to acquire the training data, as this usually requires human assessment. Therefore, we aim to improve the performance when a smaller percentage of known data is used.

We examine why misclassification is caused when the percentage of known data is low. Table 6 shows the per class precision and recall given 5% training data. It can be observed that the misclassification usually occurs among tracks of the three genres: “Rock”, “R&B”, and “Rap”. By taking a closer look at the intermediate results, we find there exist many cross-class edges between tracks of the three genres. This kind of edges affect not only the neighborhoods of unknown tracks but also those of known tracks, which are used to estimate the conditional probabilities between genres (i.e. $Pr(C_j|c_i)$). Recall that the *Double-Layer Classifier* uses both the conditional probabilities estimated from known tracks and the current neighborhood of unknown tracks to update genres. The existence of cross-class edges hampers the classification performance.

We believe such cross-class edges are caused by the noise problem of tag data. For example, besides the obvious noisy tags such as “favorite” and “mood”, we also observe tags like “pop” and “rock” are frequently used to annotate tracks from both “Rock” and “R&B”. Thus, in order to decrease the number of cross-class edges, we consider several strategies to process tag features.

5.1 Feature Processing

Particularly, three feature processing strategies are studied in our work: *Tag Discrimination*, *Tag Augmentation*, and *Content Combination*, which are illustrated as follows respectively.

Tag Discrimination. The first strategy we test is to discriminate between tags occurring in single class and tags occurring in multiple classes. To this end, We re-adjust the weights of tags based on their class distributions. As discussed in Section 3, initially, tags are weighted using a *TFIDF*-like score. The *TFIDF* method assigns more weight to a tag which occurs frequently in a particular track while occurs rarely in other tracks. Thus, it emphasizes tags occurring in a particular track. In order to emphasize tags occurring in a particular genre class, we weight tags based on their class distributions. The basic idea is that the weight of a tag should be low, if the tag is supplied with tracks of

Table 7: Performance of double-layer classifier with tag discrimination

Measure	Method	Training Data			
		5%	10%	15%	20%
Precision	Double-Layer	60.53	71.74	83.91	86.57
	Tag Discrimination	49.62	78.79	77.36	72.79
Recall	Double-Layer	60.67	72.22	82.90	83.13
	Tag Discrimination	49.97	73.89	70.36	69.49

Table 8: Per class performance of double-layer classifier with tag discrimination

10% Training Data		Class					
		Jazz	Rock	Country	R&B	Electronica	Rap
Precision	Double-Layer	80.40	57.37	81.29	60.67	84.42	66.29
	Tag Discrimination	87.41	77.92	90.01	53.34	72.91	91.16
Recall	Double-Layer	84.15	19.13	97.29	63.89	90.84	78.01
	Tag Discrimination	84.29	34.87	96.78	72.47	86.81	68.10

different genres. Otherwise, it should be assigned a higher weight. Hence, first we use an entropy based measure to justify the class distribution of a tag.

Let $\mathcal{X}^k = \{x_1, x_2, \dots, x_n\}$ be the set of known tracks. The notations \mathcal{T} , \mathcal{C} , $T(x_i)$, and $C(x_i)$, as defined in Section 3, respectively represent the complete set of tags, genre classes, the tags associated with a track x_i , and the genre label of a track x_i . We compute the frequency that a tag t_j occurs in a class c_k as $S(t_j, c_k) = |\{x_i | t_j \in T(x_i), C(x_i) = c_k, x_i \in \mathcal{X}^k\}|$. Then, the *class entropy* of a tag t_j is:

$$E_C(t_j) = - \sum_{c_k \in \mathcal{C}} \frac{S(t_j, c_k)}{S_C} \log_2 \frac{S(t_j, c_k)}{S_C}$$

where $S_C = \sum_{c_k \in \mathcal{C}} S(t_j, c_k)$ is the overall occurrence of tag t_j in all classes. The lower value of the class entropy of a tag, the more certainly the tag distributes in classes, and the higher weight we should assign to the tag.

Recall that each track is associated with a vector $\vec{T}(x_i)$, where each element is the *TFIDF* value of a tag t_j associated with x_i . To include the class distribution of a tag in the existing *TFIDF* method, the new weight of a tag t_j is the product of the tag’s *TFIDF* value and the inverse of its class entropy, $TFIDF(t_j, x_i) * \frac{1}{E_C(t_j)}$.

As a result of weighting tags based on their class entropy, the similarity values between tracks decrease. The performance with the similarity threshold $\epsilon = 0.05$ is shown in Table 7. For easy comparison, we include the performance of the original *Double-Layer Classifier* in the figure as well. We notice that this strategy improves the performance of the original *Double-Layer Classifier* only when the percentage of known data is 10%. When the percentage of known data is low, discriminating tags based on class distribution even damages the performance of the original *Double-Layer Classifier*. The possible reason is that, without enough training data, the class entropy of tags may not be computed reliably. The reason that the performance of this strategy decreases given more training data is because of the fixed similarity

Table 9: Performance of double-layer classifier with tag augmentation

Measure	Method	Training Data			
		5%	10%	15%	20%
Precision	Double-Layer	60.53	71.74	83.91	86.57
	Tag Discrimination	49.62	78.79	77.36	72.79
	Tag Augmentation	62.94	81.78	86.35	86.77
Recall	Double-Layer	60.67	72.22	82.90	83.13
	Tag Discrimination	49.97	73.89	70.36	69.49
	Tag Augmentation	67.30	81.43	83.97	85.86

threshold we use. As we observe from the intermediate results, when more training data is given, the similarity values of tracks are getting smaller. Hence, a lower similarity threshold should be used. Otherwise, the neighborhood of tracks shrinks so that it does not provide sufficient information for accurate classification.

The per class performance of the *Double-Layer Classifier* with the *Tag Discrimination* strategy is shown in Table 8. For genres of ‘‘Jazz’’ and ‘‘Rock’’, both the precision and recall are improved. For genres of ‘‘Country’’ and ‘‘Rap’’, the precision is improved significantly with a tiny sacrifice in the recall. For the genre of ‘‘R&B’’, the recall value is improved evidently while the precision decreases slightly.

Tag Augmentation. Instead of directly decreasing the number of cross-class edges, the second strategy aims to increase the number of in-class edges. Particularly, when the percentage of training data is low, the data sparsity is severe so that the in-class edges are not sufficient (given our particular dataset, this happens more seriously for tracks of ‘‘Rock’’). We consider to handle this problem by augmenting the tag vector of each track with tags from its neighbors.

We now associate two tag vectors with each track, \vec{T} and \vec{T}' , where the former contains the original tags associated with the track and the latter consists of augmented tags. Given the training data, we firstly discover neighborhoods from it with respect to some similarity threshold ϵ . Then, for each known track, the original tags of its neighbors are added to its augmented tag vector, if they do not occur in its original tag vector. The weight of each augmented tag is averaged over the number of neighboring tracks originally having this tag. After augmenting tags from neighbors, the similarity between tracks based on their augmented tag vector can be computed using the cosine similarity measure as well. Let $S(x_i, x_j)$ and $S'(x_i, x_j)$ be the similarities between tracks x_i and x_j based on their original tag vector and the augmented tag vector respectively. The overall similarity between the two tracks can be computed as $\alpha * S(x_i, x_j) + (1 - \alpha) * S'(x_i, x_j)$, where $\alpha \in [0, 1]$ controls the relative importance of each singular similarity.

Note that, tag augmentation is computationally expensive, since the neighborhood of a tag need to be explored twice, before and after adding tags. Consequently, we perform this strategy for training data only. By augmenting tags for known tracks, we aim to estimate better conditional probabilities between genres. Our experimental results suggest the effectiveness of this strategy when only training data is processed.

Table 10: Performance of double-layer classifier with tag entropy and augmentation

Measure	Method	Training Data			
		5%	10%	15%	20%
Precision	Tag Augmentation	62.94	81.78	86.35	86.77
	Discrimination & Augmentation.	70.13	85.18	85.03	85.39
Recall	Tag Augmentation	67.30	81.43	83.97	85.86
	Discrimination & Augmentation.	62.14	82.36	84.27	84.28

Table 9 presents the performance of the *Double-Layer Classifier* integrated with the *Tag Augmentation* strategy, where the coefficient α is set to 0.6 and the similarity threshold ϵ is 0.2, together with the performance of the original *Double-Layer Classifier* and the one with *Tag Discrimination*. The performance of the classifier with *Tag Augmentation* outperforms the other two clearly. It is notable that the *Tag Augmentation* strategy shows some effect when the percentage of training data is as low as 5%, although the improvement is not quite significant.

We further integrate the *Tag Discrimination* and the *Tag Augmentation* strategies. The performance, with $\alpha = 0.6$ and $\epsilon = 0.05$, is shown in Table 10. Given 10% training data, the simultaneous employment of the two strategies performs better than employing the *Tag Augmentation* strategy alone. When more training data is given, the advantage of the integration of the two strategies disappears. The possible reason is similar to as discussed on the *Tag Discrimination* strategy.

Content Combination. Although the *Tag Augmentation* strategy obviously improves the performance, it is expensive to compute neighborhood twice from the training data. Thus, instead of augmenting tag features from neighbors, we consider to augment features with other sources of information. Particularly, we take into account the audio content of tracks, and measure the similarity between known tracks based on not only their associated tags but also their associated low-level features. Similar to tag-based similarity measure between tracks, the cosine similarity is employed to measure the proximity between tracks based on their low-level features. Given the tag-based similarity, $S_t(x_i, x_j)$, and content-based similarity, $S_c(x_i, x_j)$, the overall similarity between tracks x_i and x_j is $\beta * S_t(x_i, x_j) + (1 - \beta) * S_c(x_i, x_j)$, where $\beta \in [0, 1]$ controls the relative importance of similarity based on different information sources.

We apply this strategy on both known and unknown tracks. Consequently, not only the conditional probabilities estimated from known tracks, but also the neighborhoods of unknown tracks are adjusted. Table 11 shows the performance of this strategy with $\beta = 0.6$ and $\epsilon = 0.5$. For easy comparison, we include the results in Table 9 as well. Compared with the original *Double-Layer Classifier*, this strategy improves the performance given the percentage of training data ranging from 5% to 20%. It also outperforms the other two strategies which focus on processing tag features themselves. The results suggest the effectiveness of integrating multiple information sources. Moreover, compared with the *Tag Augmentation*, this strategy is more preferable as it is computationally efficient.

This strategy cannot be easily integrated with the *Tag*

Table 11: Performance of double-layer classifier with content combination

Measure	Method	Training Data			
		5%	10%	15%	20%
Precision	Double-Layer	60.53	71.74	83.91	86.57
	Tag Discrimination	49.62	78.79	77.36	72.79
	Tag Augmentation	62.94	81.78	86.35	86.77
	Content Combination	69.68	83.02	85.32	87.18
Recall	Double-Layer	60.67	72.22	82.90	83.13
	Tag Discrimination	49.97	73.89	70.36	69.49
	Tag Augmentation	67.30	81.43	83.97	85.86
	Content Combination	70.42	80.74	84.83	85.99

Table 12: Per class performance of double-layer classifier with content combination

10% Training Data		Class					
		Jazz	Rock	Country	R&B	Electronica	Rap
Precision	Double-Layer	80.40	57.37	81.29	60.67	84.42	66.29
	Tag Discrimination	87.41	77.92	90.01	53.34	72.91	91.16
	Content Combination	92.04	70.47	86.98	81.35	88.59	78.68
Recall	Double-Layer	84.15	19.13	97.29	63.89	90.84	78.01
	Tag Discrimination	84.29	34.87	96.78	72.47	86.81	68.10
	Content Combination	93.74	51.80	87.69	62.73	91.39	97.08

Discrimination strategy, considering the latter leads to very small similarity values between tracks. Our preliminary experimental results show that the integration of *Content Combination* and *Tag Augmentation* is capable of generating performance which is a bit more better than *Content Combination* only. However, due to the inefficiency of the *Tag Augmentation* strategy, we advise against this integration.

Table 12 presents the per class precision and recall of *Double-Layer Classifier* using the *Content Combination* strategy, where the results in Table 8 are included for comparison. This strategy improves the accuracy of almost every class, although the precision and recall of “Rock” and “R&B” are still not as good as those of the other genres.

5.2 Updating Optimization

Besides processing feature data, several strategies have been designed in an attempt to optimize the genre updating procedure. Particularly, we consider two sets of weighting schemes to adjust the influence of neighbors in predicting genres of unknown tracks. The first set of schemes assign static weights to neighbors, while the second set computes dynamic weights. However, according to our experimental results, most of the weighting schemes do not show strong and positive effect. We thus briefly introduce these schemes and analyze the reasons that they fail to work.

The following three static weighting schemes are considered. *Similarity-based Weights*. Intuitively, neighbors which are more similar to an unknown track should have more impact on its classification than neighbors which are less

Table 13: Performance of double-layer classifier with static weighting schemes

Measure	Method	Training Data			
		5%	10%	15%	20%
Precision	Content Combination	69.68	83.02	85.32	87.18
	Similarity-based	69.36	76.77	83.39	86.08
	Knowledge-based	76.18	80.18	84.96	84.45
	Degree-based	58.77	77.25	84.26	86.60
Recall	Content Combination	70.42	80.74	84.83	85.99
	Similarity-based	68.47	75.41	82.64	84.82
	Knowledge-based	79.01	80.21	83.18	83.26
	Degree-based	64.29	74.74	82.79	84.17

similar. Thus, we assign a weight to a neighbor using the similarity between the target and the neighbor. That is, Equation 8 is computed as:

$$Pr(c_i) \prod_{x_j \in N^u(x_i) \cup N^k(x_i)} Sim(x_j, x_i) Pr(C(x_j)|c_i) \quad (9)$$

Knowledge-based Weights. Considering that the information provided by known and unknown neighbors might not be equally reliable, we introduce a coefficient $\gamma \in [0, 1]$ to adjust the relative impact of the two sets of neighbors (γ is set to 0.6 in our experiments to emphasize the influence of known neighbors). Then, $Pr(c_i|N^u(x_i), N^k(x_i))$ is updated as

$$Pr(c_i) \prod_{x_j \in N^k(x_i)} \gamma Sim(x_j, x_i) Pr(C(x_j)|c_i) \prod_{x_j \in N^u(x_i)} (1 - \gamma) Sim(x_j, x_i) Pr(C(x_j)|c_i) \quad (10)$$

Degree-based Weights. The last scheme takes into account the topology of the track graph. The neighborhood of each node can vary in size. Intuitively, a large neighborhood may have more variety in genre classes. A node with a large neighborhood is, therefore, probably less capable of confidently influencing its unknown neighbors’ genres. Hence, we use information from such a neighbor to assign a weight which is inversely proportional to its node degree. Applied to $Pr(c_i|N^u(x_i), N^k(x_i))$, we get:

$$Pr(c_i) \prod_{x_j \in N^u(x_i) \cup N^k(x_i)} \delta^{D(x_j)} Sim(x_j, x_i) Pr(C(x_j)|c_i) \quad (11)$$

where $D(x_j)$ is the node degree of x_j and $\delta \in [0, 1]$ is an experimentally decided parameter which results in an inverse ratio between assigned weight and node degree (δ is set to 0.2 in the experiments shown in Table 13).

The performance of the *Double-Layer Classifier* integrated with the static weighting schemes, based on the *Content Combination* feature processing strategy, is shown in Table 13. Unfortunately, apart from the *Knowledge-based Weight* scheme for 5% training data given, the static weighting schemes do not have positive impact on the performance.

Further investigation of the data shows that the similarity between some “Rock” and “R&B” tracks is very high, even

Table 14: Performance of double-layer classifier with dynamic weighting schemes

Measure	Method	Training Data			
		5%	10%	15%	20%
Precision	Content Combination	69.68	83.02	85.32	87.18
	Neighborhood Entropy 1	66.42	84.83	87.01	86.18
	Neighborhood Entropy 2	70.53	79.94	86.05	87.23
Recall	Content Combination	70.42	80.74	84.83	85.99
	Neighborhood Entropy 1	67.50	82.80	86.04	83.49
	Neighborhood Entropy 2	66.93	78.37	85.27	85.62

when low-level features are taken into account. Therefore, the *similarity-based Weight* is rendered mostly useless and is not able to disambiguate tracks.

The *Degree-based Weight* scheme is designed based on the assumption, that a node with a larger neighborhood is less capable of confidently influencing its unknown neighbor. This assumption, however, does not hold for our particular data set. A node with a larger neighborhood does have more variety in genre classes. Nevertheless, in most cases, the majority of its neighbors indicating the true class of the node. Consequently, this node is still capable of correctly influencing its unknown neighbors. Another interesting observation is that, the performance cannot be improved either when we assign more weights to neighbors with higher node degree. This further suggests that, for our particular data set, node degree is not a distinctive feature reflecting the capability of neighbors in confidently influencing unknown tracks.

When the percentage of known data is low, a neighborhood of an unknown track is probably overwhelmed by unknown neighbors. In such cases, it is reasonable to assume that at least known neighbors are more reliable than unknown neighbors. This seems to be the reason why the *Knowledge-based Weight* scheme improves the performance for 5% training data.

The dynamic weighting schemes assign weights to nodes based on the class distributions of their neighborhoods. For each unknown node during each round of genre updating, we compute its *neighborhood entropy*. Given a node x_i and its neighborhood $N(x_i) = \{x_1, x_2, \dots, x_n\}$, where $\forall j \in [1, n]$, $C(x_j) \in \mathcal{C} = \{c_1, c_2, \dots, c_k\}$, the *neighborhood entropy* of x_i , denoted as $E_N(x_i)$, is computed as:

$$E_N(x_i) = - \sum_{j=1}^k \frac{H(c_j)}{H_C} \log_2 \frac{H(c_j)}{H_C}$$

where $H(c_j) = \sum_{x_j \in N(x_i) \& C(x_j)=c_j} Sim(x_j, x_i)$ is the sum of similarities between x_i and all its neighbors which are assigned to class c_j . $H_C = \sum_{c_j \in \mathcal{C}} H(c_j)$ is the normalization constant.

Neighborhood Entropy based Weight 1. The higher the neighborhood entropy value, the more even the class distribution of x_i 's neighbors. When its neighbors' classes are distributed evenly, it is not easy to determine x_i 's class. Therefore, it probably should not be drastically influenced in this iteration. We design the first neighborhood entropy based weighting scheme to anchor the node to its current probability distribution in this iteration. That is, the Equa-

tion 7 is updated as

$$Pr(c_i|\Delta^k)^{(r+1)} = Pr(c_i|\Delta^k)^{(r)}$$

$$\sum_{N^u(x_i)} [Pr(c_i|N^u(x_i), N^k(x_i)) \prod_{x_j \in N^u(x_i)} Pr(C(x_j)|\Delta^k)^{(r)}] \quad (12)$$

Neighborhood Entropy based Weight 2. Considering that a node with a higher neighborhood entropy is prone to be classified incorrectly, its influence to its unknown neighbors should be demoted to some degree. Therefore, the second neighborhood entropy based weighting scheme is designed as follows. Let x_i and $N(x_i)$ be an unknown node and its neighborhood. Let ξ be some neighborhood entropy threshold, and $N'(x_i) = \{x_j | x_j \in N^u(x_i) \wedge E_N(x_j) \geq \xi\}$ be the subset of $N(x_i)$ such that each node is an unknown neighbor of x_i and has a high neighborhood entropy. Then, the *Double-Layer Classifier* updates $Pr(c_i|N^u(x_i), N^k(x_i))$ as

$$Pr(c_i) \prod_{x_j \in N'(x_i)} \pi Pr(C(x_j)|c_i) \prod_{x_j \in N(x_i) \setminus N'(x_i)} Pr(C(x_j)|c_i) \quad (13)$$

where $\pi \in [0, 1]$ decides how much the influence of neighbors with high neighborhood entropy should be decreased (π is set to 0.75 in our experiments). Note that, this scheme is similar to the *Degree-based Weight* in the way that both take into account the genre variety of a neighborhood. However, this scheme might be more accurate as it directly measures the genre variety of a neighborhood, while the *Degree-based Weight* scheme infers from the size of a neighborhood. Furthermore, the *Degree-based Weight* assigns fixed weights to nodes, while this scheme dynamically decides whether or not to dampen the influence of nodes in every round.

Table 14 shows the performance of the two neighborhood entropy based weighting schemes. We notice that, although both schemes show certain improvement with certain size of training data, the improvement is not significant. Compared with feature processing strategies, the effectiveness of these weighting schemes focusing on revising the genre updating procedure is limited for our problem.

6. CONCLUSIONS

Automatic music genre classification is a pivotal problem in MIR systems, which has attracted numerous research efforts in the community. Most of the existing approaches focus on improving classification accuracy by finding better low-level features which represent and distinguish music more precisely. In this paper, we propose to explore the usage of social tags for this task. To the best of our knowledge, it is the first attempt to perform music genre classification using such tags. We consider tag information in an implicit way by creating a graph of music tracks based on their semantic similarity in terms of associated tags. Two classification methods are introduced, *Single-Layer Classifier* and *Double-Layer Classifier*, which respectively considers the explicit audio content information and the implicit tag information simultaneously and sequentially. We evaluated the performance of the classifiers on real world data collected from Last.fm. The experimental results demonstrate the usefulness of tags in music genre classification. The *Double-Layer Classifier* performs better than the *Single-Layer Classifier* as unknown data may be utilized as well in predicting the genres of unknown tracks. Further improvement of the *Double-Layer Classifier* has been considered from two

aspects: feature processing and optimized updating. Our experimental results suggest, that strategies focusing on improving tag feature data are more effective than those focusing on revising the genre updating procedure.

In the future, we are going to apply ontological information to tags so that the semantic relationship between tracks can be captured more precisely. Then, we are interested in applying our methods to classifying music into a hierarchy of genres. Furthermore, investigating the usefulness of tags in music *mood* classification is an intriguing problem as well.

7. ACKNOWLEDGMENTS

This work was partially supported by the PHAROS project funded by the European Commission under the 6th Framework Programme (IST Contract No. 045035).

8. REFERENCES

- [1] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. In *SIGIR*, pages 485–492, 2006.
- [2] J.-J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research.*, 32(1):83–93, 2003.
- [3] S. Bao, G.-R. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW*, pages 501–510, 2007.
- [4] B. Berendt and C. Hanser. Tags are not metadata, but just more content - to some people. In *ICWSM*, 2007.
- [5] C. H. Brooks and N. Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW*, pages 625–632, 2006.
- [6] J. J. Burred and A. Lerch. A hierarchical approach to automatic musical genre classification. In *6th Int. Conf. Digital Audio Effects*, 2003.
- [7] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD Conference*, pages 307–318, 1998.
- [8] B. Cui, H. V. Jagadish, B. C. Ooi, and K.-L. Tan. Compacting music signatures for efficient music retrieval. In *EDBT*, pages 229–240, 2008.
- [9] R. Dannenberg, J. Foote, G. Tzanetakis, and C. Weare. Panel: new directions in music information retrieval. In *Proc. Int. Computer Music Conf.*, 2001.
- [10] P. A. Dmitriev, N. Eiron, M. Fontoura, and E. J. Shekita. Using annotations in enterprise search. In *WWW*, pages 811–817, 2006.
- [11] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. In *WWW*, pages 193–202, 2006.
- [12] S. Golder and B. A. Huberman. The structure of collaborative tagging systems. In <http://www.hpl.hp.com/research/idl/papers/tags/tags.pdf>, 2006.
- [13] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *AES 25th Int. Conf.*, 2004.
- [14] H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In *WWW*, pages 211–220, 2007.
- [15] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *SIGIR*, pages 282–289, 2003.
- [16] Q. Lu and L. Getoor. Link-based classification. In *ICML*, pages 496–503, 2003.
- [17] S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. volume 8, pages 935–983, 2007.
- [18] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *ISMIR*, pages 594–599, 2005.
- [19] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *ISMIR*, 2004.
- [20] C. McKay and I. Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR*, pages 101–106, 2006.
- [21] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *ISMIR*, pages 628–633, 2005.
- [22] L. Pelkowitz. A continuous relaxation labelling algorithm for markov random fields. volume 20, pages 709–715, 1990.
- [23] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *IEEE Singal Processing Magazine.*, 23(2):133–141, 2006.
- [24] P. Schmitz. Inducing ontology from flickr tags. In *the workshop on Collaborative Web Tagging at WWW*, 2006.
- [25] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, pages 485–492, 2002.
- [26] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. on Speech and Audio Processing*, 10(5):293–302, July 2002.
- [27] K. West and S. Cox. Finding an optimal segmentation for audio genre classification. In *ISMIR*, pages 680–685, 2005.
- [28] B. Whitman and P. Smaragdis. Combining musical and cultural features for intelligent style detection. In *ISMIR*, 2002.
- [29] X. Wu, L. Zhang, and Y. Yu. Exploring social annotations for the semantic web. In *WWW*, pages 417–426, 2006.