Genre Classification of Symbolic Music with SMBGT

Alexios Kotsifakos [1], Evangelos E. Kotsifakos [2], Panagiotis Papapetrou [3], and Vassilis Athitsos [4]

[1,4]*Computer Science and Engineering Department, University of Texas at Arlington, USA*
[1]alexios.kotsifakos@mavs.uta.edu, [4]athitsos@uta.edu

[2]*Department of Informatics, University of Piraeus, Greece*
ek@unipi.gr

[3]*Department of Computer Science and Information Systems, Bickbeck, University of London, United Kingdom*
Department of Information and Computer Science, Aalto University, Finland
panos@dcs.bbk.ac.uk

ABSTRACT

Automatic music genre classification is a task that has attracted the interest of the music community for more than two decades. Music can be of high importance within the area of assistive technologies as it can be seen as an assistive technology with high therapeutic and educational functionality for children and adults with disabilities. Several similarity methods and machine learning techniques have been applied in the literature to deal with music genre classification, and as a result data mining and Music Information Retrieval (MIR) are strongly interconnected. In this paper, we deal with music genre classification for symbolic music, and specifically MIDI, by combining the recently proposed novel similarity measure for sequences, SMBGT, with the k-Nearest Neighbor (k-NN) classifier. For all MIDI songs we first extract all of their channels and then transform each channel into a sequence of 2D points, providing information for pitch and duration of their music notes. The similarity between two songs is found by computing the SMBGT for all pairs of the songs' channels and getting the maximum pairwise channel score as their similarity. Each song is treated as a query to which k-NN is applied, and the returned genre of the classifier is the one with the majority of votes in the k neighbors. Classification accuracy results indicate that there is room for improvement, especially due to the ambiguous definitions of music genres that make it hard to clearly discriminate them. Using this framework can also help us analyze and understand potential disadvantages of SMBGT, and thus identify how it can be improved when used for classification of real-time sequences.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining; H.2.4 [**Systems**]: Multimedia Databases

General Terms

Algorithms

PETRA '13, May 29 - 31 2013, Island of Rhodes, Greece.

http://dx.doi.org/10.1145/2504335.2504382

Keywords

Classification, sequence, genre, MIDI, SMBGT

1. INTRODUCTION

Music can be seen as an effective means of stimulating and focusing attention. Particularly for specific groups of people who are weak in responding to other physical interventions, music can be highly significant. Hence, music can be considered as an assistive technology with high therapeutic and educational functionality for children and adults with disabilities. Skill areas such as mental retardation, autism, and learning disabilities can be affected by music therapy. Music can be used to structure an entire therapeutic intervention in order to maintain attention. It can be used as means of alerting people for important information or interactions, or it can function as a calming down mechanism when anxiety intervenes with cognitive focus. In addition, music is of high importance in learning, as it can provide significant assistance in memorization.

A problem of particular interest to the community of music informatics as well as to the communities of data mining and assistive technologies is that of classifying a query song of unknown category to one of several existing and known categories. This problem can be abstracted in different ways depending on the application domain. In this paper, we focus on the classification of music songs in genres.

Music genre classification is highly related to the area of assistive environments, since it can facilitate many of the music functionalities described above. For example, music systems with therapeutic and educational functionalities need to be adapted based on the music preferences of the end-users, such as children, patients, or disabled. Hence, being able to identify the genres of songs that fit to each category of end-users is imperative in such settings as it can assist effectively in medical treatments. As another example, consider the task of learning and memorization. It would be very useful to identify for each individual the music genres that are more suitable when performing a learning task or are less disruptive than other genres.

Songs are essentially music audio signals that can be either represented in symbolic format based on musical scores, or in audio format based on analogue signals, which can be sampled and encoded in different ways. A widely used format of the first type of representation is the Musical Instrument Digital Interface (MIDI), while the Humdrum has also been used. Although the MIDI has some disadvantages, such as that the sound quality depends on synthesizer and it cannot store voice, it has some important advantages. It takes very little space making it easy to store and communicate,

and it provides a format that can be easily handled allowing easier comparison between different instruments and music pieces. Thus, it has been widely accepted, and in this paper we focus on this type of symbolic format.

Similarity search is the basis for a variety of applications such as playlist generation, similarity-based browsing interfaces, and recommendation systems. It has to be mentioned though that end users are more likely to browse and search by genre than by recommendation, artist similarity or music similarity [18]. As a result, genre classification has been studied in the music literature and research focuses on exploiting machine learning and data mining methods for meaningful results.

As mentioned by Scaringella et al. "Music genres are categories that have arisen through a complex interplay of cultures, artists, and market forces to characterize similarities between musicians or compositions and organize music collections" [34]. Musical genres can be characterized by using several types of rules [9], while a good work by Aucouturier and Pachet on how to represent musical genre can be found at [2].

Constructing automatic music classifiers is of great interest for a variety of reasons [24]. Starting with genre classifiers, which are the focus of this work, they can be used to understand the important characteristics of music that help distinguishing particular genres and how these characteristics vary. What is more, classifiers can be used to categorize recordings whose authorship is unknown to composers. In addition, results produced by automatic classifiers can be combined with sociological and psychological research and give insight on the way that humans understand musical similarity and construct clusters of music pieces.

Although much research has focused on how to represent musical genres and on genre classification, still there are many problematic aspects that should be addressed, as mentioned below [25]. First of all, there is not reliable ground truth, which is not only fundamental to effectively train genre classifiers, but also to evaluate genre classification systems. Secondly, genre classification is most of the times done by artist or album and not by individual recording that is certainly inappropriate for the classification task. Moreover, most of the times no consensus can be achieved by human annotators both on the genre of songs, due to the non-existing strict boundaries between genres [21], and on the different categories that can be used to classify songs in. Consequently, it is tough to build up a taxonomy of genres that is widely acceptable [29]. In addition, most genres have fuzzy definitions that may change from source to source and the criteria to define some genres vary. Also, some genres may overlap and individual songs may be classified to more than one genre making the classification task even harder, if only one genre per recording is allowed. Apart from these difficulties, not much psychological research has been performed on how humans perform genre classification [21, 32], while a significant amount of time is required for human annotators to classify songs. It is also important to have large training sets to create meaningful models and large test sets to have a better guarantee on the performance of the classification systems. Finally, it has been shown in the literature that increasing the number of genres and the number of songs reduces the performance of classification. All the aforementioned problems that arise in automatic music genre classification make current software tools not achieving satisfyingly high accuracy, and as a result research on this field should be continued.

In order to perform genre classification systems first extract important features representing salient information from the recording, and then classify the music piece to a music genre based on the output of one or more combined classifiers that take as input these features. These features are low-level and/or high-level. Features of the first category are based on signal processing quantities and include temporal, energy, perceptual, and spectral shape features. Furthermore, melodic and harmonic content are better described by low-level attributes than notes or chords. High-level features, which are also referred to as semantic, include tempo, rhythm, key, instrumentation, vocal style, and meter, among others. For music classification humans use high-level information, which is better provided by recordings in symbolic format, such as MIDI. It should be noted that selecting appropriate features requires knowledge of diverse domains, such as signal processing, musicology, music theory, psychology, and in the last few years text and web-mining.

Next, we provide some basic terms used in music that will help the reader throughout the paper. Every piece of music (as shown in Figure 1) is a sequence of notes characterized by a *key*, which defines the pattern of allowed intervals that the sequence of notes should conform with, and a *tempo* that regulates the speed of the music piece. Moreover, each *note* is described by its *pitch* and *duration*. A *pitch interval* is the distance between two pitches, and *transposition* is a term that refers to shifting a melody of a piece written in a specific key to another key. Finally, there is a discrimination between *monophonic* and *polyphonic* music, where the first does not allow two or more notes to sound at the same time. Figures 1 and 2 show a monophonic and its corresponding polyphonic part, respectively, of a well-known instrumental Classical piece of Wolfgang Amadeus Mozart, the "Rondo Alla Turca", which is also used in our experiments. Figure 3 presents the initial part of a polyphonic traditional orthodox church melody (with Greek lyrics) named "Here is darkness and morning" arranged for choir by S.N. Chatzistamatiou.

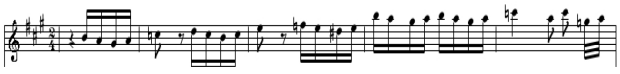

Figure 1: Initial part of the "Rondo Alla Turca" monophonic music score.

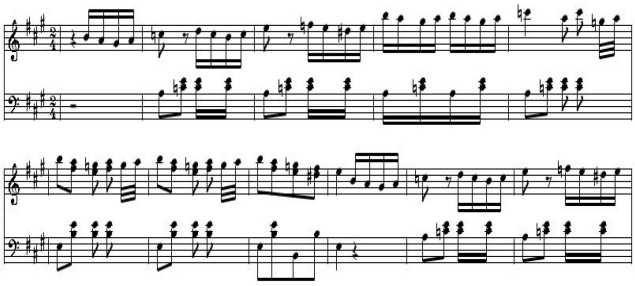

Figure 2: Initial part of the "Rondo Alla Turca" polyphonic music score.

The remainder of this paper is organized as follows: in Section 2 we provide the related work, in Section 3 we sketch the recently proposed SMBGT method for comparing sequences and also describe how to measure the similarity between songs with the use of SMBGT. Our experimental evaluation and results are discussed in Section 4. Finally, in Section 5 we conclude the paper and provide directions for future work.

2. RELATED WORK

Expressing pitch can be done with absolute pitch [26] or pitch interval, which can also be quantized [38]. Duration can be en-

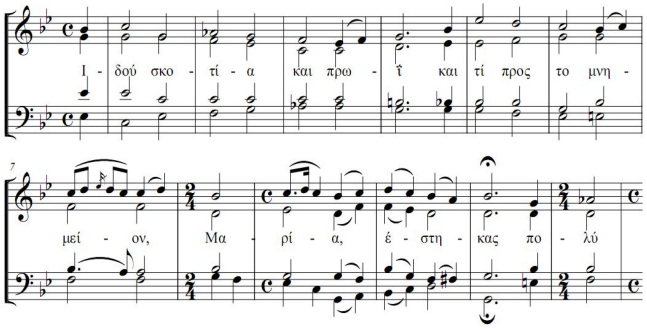

Figure 3: Initial part of the "Here is darkness and morning" polyphonic music score (with Greek lyrics).

coded as IOIR, which is the ratio of the differences in time onsets of two consecutive notes (IOI). IOI and the logarithm of IOIR with variations have also been used [30]. Thus, using pitch interval and IOIR (or their variations) gives transposition and time invariance when comparing melodies, while taking advantage of both pitch and duration makes melody sequences more unique [26, 38].

Next, a brief overview of the methods that have been applied to music retrieval and QBH that perform *whole sequence matching* or *subsequence matching* is given. For a more detailed survey on *Query-By-Humming* (*QBH*) similarity methods please refer to [15].

Regarding whole sequence matching, the method in [1] operates using Dynamic Time Warping (DTW) [17] and is based on absolute pitches. To deal with tempo variations methods have incorporated them in their scheme [28] or scale the target sequences before applying DTW [23]. Transposition invariance can be embedded as a cost function in the scheme of Dynamic Programming (DP) computation [19], but the goal in this approach is to do *whole query matching* while duration is not used. Another DP-based approach that allows for gaps on both sequences during their alignment is the Longest Common SubSequence (LCSS) [4], which has been used for evaluating melodic similarity [38]. Iliopoulos et al. proposed an algorithm that is suitable for whole query matching in melodic recognition, accounts for a constrained number of gaps only in the target sequence, tolerates mismatches by a constant value, and uses absolute pitches [12]. Another approach segments each database song into several disjoint pieces and whole sequence matching is performed between the query and each such piece [40].

Two similar methods for subsequence matching that use only absolute pitches and implicitly deal with tempo scaling have been proposed [11, 13]. Variations of Edit distance used for music retrieval can be found at [19, 31], while its most version is applied to QBH exploiting both pitch and duration information [14, 16, 39]. SPRING is appropriate for finding the subsequences of evolving numerical streams that are closest to a query [33], and has been applied to the problem of QBH [14]. In this paper we use *Subsequence Matching with Bounded Gaps and Tolerances* (*SMBGT*), which is a recently proposed novel method for subsequence matching that has been applied to QBH [14, 16]. However, since it can be easily used as a general method for comparing sequences, it can perform whole sequence matching and be applied to many data mining tasks, such as genre classification, which is the goal of this paper.

HMM-based methods have also been proposed for music retrieval, for QBH [27, 35, 39], and for genre classification [36]. Although the HMMs are suitable for modeling the probabilistic behavior of music pieces, due to the fact that their training is computationally expensive it is very hard to create models discriminating different genres of music, especially for large databases.

Artist and genre classification where vectors of low-level audio features (MFCCs) are mapped to points in a Euclidean anchor space is proposed in [3]. These points are vectors of posterior probabilities of membership in the anchor classes/dimensions, which are represented by neural networks. Since points can be seen as samples from a distribution characterizing a song, it is modeled by GMMs and an approximation to KL-divergence is used to compare songs. However, due to lack of ground truth the classification accuracy is not very satisfying. McKay et al. showed the effectiveness of using large feature sets that are combined with feature weighting systems, high-level features, and also the importance of instrumentation-based features for genre classification when performed with k-NN, neural networks and combination of several classifiers [24]. Cataltepe et al. exploit the MIDIs and their audio versions. The MIDI is turned into a string from a finite alphabet and the distance between pieces is computed by the Normalized Compression Distance. MIDIs are also converted to audio from which timbre, rhythmic, and pitch features are extracted. Finally, diverse and independent classifiers based on Linear Discriminant Classifier and k-NN are combined with weights to perform classification [6]. SVMs have also been used for genre and music classification [20, 22]. Moreover, McKay et al. suggested that background research from musicology and psychology has to be exploited for genre classification, as should happen with cultural features and metadata mined from the web [25]. Other very interesting approaches for genre classification can be found at [5, 7, 8, 37]. Finally, surveys on audio-based music classification and annotation, and automatic genre classification have been proposed [10, 34].

3. MEASURING SONG SIMILARITY

Next, we give some notations and how we define a variable error-tolerant match, briefly describe the SMBGT method used for measuring the similarity between sequences, and then describe how to compare songs represented in the MIDI symbolic format.

Let us denote $X = \{x_1, \ldots, x_n\}$ a sequence that represents a music piece, with $|X|$ being its length. Each $x_j = \langle x_j^p, x_j^r \rangle \in X$ represents a pair of real values, with x_j^p and x_j^r corresponding to pitch and duration information, respectively. We call a set of N sequences $DB = \{X_1, \ldots, X_N\}$ a music database, the size of which is $|DB|$, and we use the letter S to refer to a song. Finally, we denote a *subsequence* of X as $X[ts : te] = \{x_{ts}, \ldots, x_{te}\}$, which is a set of elements from X that appear in the same order as in X and do not have to be continuous.

As proposed in [14], given two sequences X and Y, we say that their elements $x_i \in X$ and $y_j \in Y$ *match with variable ϵ-tolerance*, and denote it as $x_i \approx_\epsilon^f y_j$, if, we use absolute or relative tolerance, and for a set of constraints $\epsilon^f = \{\epsilon_p^f, \epsilon_r^f\}$:

$$\epsilon_p^f(i) = f_p(x_i^p) \text{ and } \epsilon_r^f(i,j) = f_r(x_i^r, y_j^r), \tag{1}$$

where f_p is a function of x_i^p and f_r a set of constraints on x_i^r, y_j^r. One possible definition for ϵ_p^f is the following:

$$\epsilon_p^f(i) = \lceil x_i^p * t \rceil, \text{ with } t = 0.2,\ 0.25,\ 0.5. \tag{2}$$

Regarding $\epsilon_r^f(i,j)$, if we consider IOIR for representing duration information, which is the case in this paper, it can be defined as follows:

$$\epsilon_r^f(i,j) = \{y_j^r \leq 2 * x_i^r,\ y_j^r - x_i^r \geq -0.5\}. \tag{3}$$

3.1 SMBGT

The similarity measure that we use to compare sequences is named SMBGT [14], which outperforms several existing methods in terms of accuracy when used for the Query-By-Humming application domain.

Given a query Q and a target sequence X ($|Q| << |X|$), SMBGT finds the subsequence of X that best matches Q. This method allows variable tolerance levels in the matching (where tolerances are defined as functions of the query and target elements), since there may be slight or more serious variations in both key and tempo when matching melodies. In addition, during the alignment it allows skipping a constrained number of consecutive elements on both X and Q. These gaps are positive integers, α and β, respectively and provide a setting where the expansion of the matched subsequences is controlled. Moreover, SMBGT bounds the matching range (that is the maximum matching subsequence length in X) by r, which is tuned according to the application domain. Also, it allows for optionally bounding the minimum number of matching elements by δ, so as to ensure that the matching subsequences will include at least a specified number of matching elements. An example of SMBGT is shown in Figure 4.

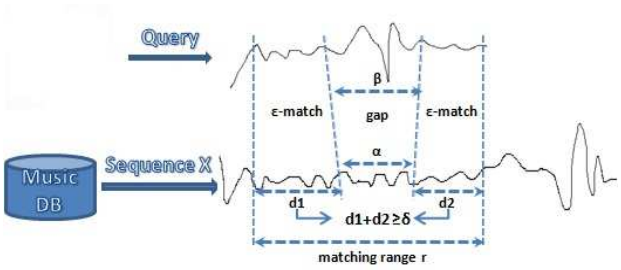

Figure 4: SMBGT: error-tolerant matching is denoted as ϵ-match.

Although SMBGT was proposed motivated by the QBH application, due to the fact that it takes into account all of the above aspects when matching a query and a target sequence, it is a general similarity measure for sequences guaranteeing a robust and meaningful matching even in potentially noisy application domains, such as music, and can be used for both whole sequence and subsequence matching.

3.2 Similarity between songs

To perform the genre classification task based on MIDI recordings we should be able to define the similarity or distance between songs. Based on SMBGT, which is a similarity measure, the scheme that we use in this paper for song similarity is the following.

Assume that we would like to compare two songs, namely S_i and S_j, which are in MIDI format and consist of $|c_i|$ and $|c_j|$ channels (that are at most 16 per song). First, for each song and channel, we extract the highest pitch at every time click, a method called *all-channels extraction* [38]. It has to be noted that in the extraction process we exclude channel 10, since it is used for drums and cannot offer any melodic information. Then, we have to convert the tuples ⟨pitch, click⟩ of each channel to ⟨pitch interval, IOIR⟩, resulting in $|c_i|'$ and $|c_j|'$ converted channels; as mentioned in Section 2 this representation provides transposition and time invariance. After these steps, we compare all channels $|c_i|'$ of S_i with the $|c_j|'$ channels of S_j using SMBGT, and get $|c_i|' * |c_j|'$ scores. Finally, we retrieve the maximum score s_{ij} out of all the pairwise similarity scores, and normalize it by dividing with the maximum length of the two channels of S_i and S_j that gave s_{ij}. The resulting score is assigned as the similarity between S_i and S_j. The similarity score of a song S_i to itself is considered to be 0. This is done basically to exclude the query song itself from being a neighbor in the k-NN classification task; the less the similarity score is between two songs the less similar they are.

4. EXPERIMENTS

In this section we provide the setup for our experiments and the experimental results.

4.1 Experimental Setup

4.1.1 Data

For the purposes of our experiments we collected 100 freely available on the web MIDI songs that cover four genres of music: *Classical*, *Blues*, *Rock*, and *Pop*. In Table 1 there is more information for the dataset, i.e., the total number of channels it covers or else $|DB|$, the number of 2-dimensional points corresponding to these channels, and also the number of songs per genre of music.

Table 1: Characteristics of the dataset used in the experiments. There are 100 songs, and the total number of channels comprising these songs in their MIDI representation, along with the number of 2-dimensional points corresponding to these channels are shown. In the final column, the number of songs belonging to each of the four selected genres is presented.

# of songs	# of channels	# of points	Genres
100	821	310.798	Blues (20), Rock (31) Classical (27), Pop (22)

We have to note that during the process of selecting MIDI songs we observed that pop and disco songs have many common characteristics and acoustic similarities as genres. As a result, we picked pop songs, which we considered to be sufficient for our experiments. Moreover, each song was labeled with one genre. This is because assigning a song to more than one genres would make the classification task biased and providing higher accuracies, since the number of songs and genres is not huge.

Following the selection of the 100 MIDI songs, for each of them we followed the procedure mentioned in Section 3.2 to get for each song the 2-dimensional representation of its channels. In other words, for each song we first applied the all-channels extraction process, and then converted the tuples ⟨pitch, click⟩ of each channel to ⟨pitch interval, IOIR⟩. This pre-processing procedure was done offline and only once, guaranteeing that there is no chance of missing a melody existing in any channel of a song that may be similar to melodies of other songs' channels.

Experiments were run on an AMD Opteron 8220 SE processor at 2.8GHz, and implemented in Matlab.

4.1.2 Evaluation

The classifier that is used for genre classification is the k-NN. First, we created the similarity matrix of all songs to all others (of size 100x100), based on the procedure mentioned in Section 3.2. Then, each of the 100 MIDI songs was considered to be a query to which we applied k-NN classification, for k in [1,10]. A query is classified correctly if the majority of its k Nearest Neighbors belong to the same genre as that of the query, and the measure that we use for evaluating our framework is the *classification accuracy*, which is defined as the percentage of the query songs that were

classified correctly. Clearly, we are interested in a large number of queries for which there is clear majority for one genre in their k neighbors and are also classified correctly.

In case that for a query song there are more than one genres with the highest number of "votes" in the returned k-NN results, then there is no clear majority for a genre (independently of whether that genre is the correct one for the query or not). To resolve ties of such cases we applied the following scheme. In the ordered similarity scores of the k-NN that are returned, the output genre of the classifier is the first genre for which the maximum number of votes is reached (by taking one by one the k neighbors and increasing the counter of the genre they belong to). This is an intuitively good approach for the case of ties, instead of, for example, randomly picking one of the genres that has the maximum number of votes or, even worse, not classifying the query to any genre (and just return all tied genres as possible answers). We believe that further looking at the structure of the songs may provide more accurate results.

With regard to the parameters of SMBGT, the number of elements allowed to be skipped in a query Q was set to $\beta = 6$, and for the target sequence X to which Q is compared was set to $\alpha = 5$. For ϵ_p^f, absolute variable tolerance was studied with $t = 0.2$ in the scheme proposed in Section 3. In addition, the maximum matching range r was set to 1.2 and the minimum number of matching elements δ to 0.1. The reason for instantiating the parameters of SMBGT with these values is because they have been proved to provide the best accuracy for the QBH application [14]. Although QBH is not the target application in this paper we believe that these values should be tested for the genre classification as well. As part of our future work, cross-validation will also be done to obtain the best combination of parameter values.

4.2 Experimental Results

In Table 2 we present the classification accuracies that we obtained for several values of k in the k-NN classification. The "Non-Tied" column shows the number of queries (out of the 100) for which the classifier could clearly decide and return one genre, i.e., there was only one clear winner genre with the maximum number of votes. Column "Non-Tied Classified" gives the number of queries that were correctly classified out of those for which there was a clear winner genre by the classifier (as indicated by column "Non-Tied"). "Accuracy (Non-Tied)" provides the classification accuracy for the queries that were correctly classified when there was a clear winner genre. In other words, for a specific k it is the division of the number in column "Non-Tied Classified" with that of column "Non-Tied". Column "Tied" is basically the remaining number of query songs up to 100 when we consider column "Non-Tied". "Tied Classified" shows the number of queries that were correctly classified when there were more than one genres with the same maximum number of votes in the returned results (ties) and we treated them following the scheme described in Section 4.1.2. In "Accuracy (Tied)" we present the classification accuracy obtained when dividing column "Tied Classified" with "Tied". Finally, the total classification accuracy is shown in "Accuracy Total", which refers to the total number of queries that were classified correctly (whether they had a clear winner genre in the returned results or after resolving ties).

From Table 2 we can observe that the best accuracy for non-tied queries is achieved for $k = 10$ and $k = 7$, and is 40%. We note here that, although 40% accuracy is also achieved for $k = 2$ the total number of non-tied queries is only 30, and thus it is not considered to be a reliable value for accuracy. Regarding the accuracies for "tied" queries the best ones are obtained for $k = 4$ with 43.48% and $k = 9$ with 41.67%. Finally, the best total accuracy is 40% for $k = 10$. According to these accuracies, we observe that the best value for k among all tested values is 10, especially if we observe the total classification accuracy, which is essentially the measure returned to the user by an automatic genre classification system.

There are several reasons for not getting total classification accuracies greater than 40%, while also requiring to look at the %10 of the database songs' scores for each query ($k = 10$) to attain this accuracy. First of all, there is a significant percentage of queries for which we have to resolve ties. This indicates that a more thorough study of the structure of songs should be done so as to apply a more clever mechanism to deal with ties, or even use another similarity measure for such cases that will be based on low-level features of the songs. What is more, we perform genre classification with k-NN, which is a simple classifier that does not require any kind of training. However, since in the music domain describing a genre requires knowledge of not only intrinsic properties of songs but also cultural features (that greatly influence the characterization of songs into genres), more complex and trained classifiers on larger databases of songs should provide more promising classification accuracies.

Another important observation that we came up with is that for the majority of k values that we experimented with the best "Accuracy (Non-Tied)" per genre was achieved in the following order: Blues, Rock, Pop, and Classical. This is indeed the case in music. Blues songs have a very well-defined compositional structure, followed by Rock songs, which can also be characterized to have a particular style, but less well-defined than Blues. Additionally, Pop songs are much harder to be treated as having one style, since they cover a wide variety of "popular" music. Finally, Classical music is the hardest genre to classify songs to, because the structure and especially the melody of each piece depends basically on the personal taste and the music era of the composer. According to these music observations that are confirmed by our experiments, SMBGT is proved to be very promising with regard to discriminating genres based on their structure.

5. CONCLUSIONS AND FUTURE WORK

In this paper we applied a recently proposed novel similarity measure for comparing sequences named SMBGT to perform music genre classification with the k-NN classifier. Music genre classification can be highly applicable to assistive environments since music can be seen as a means of stimulating and focusing attention for people with disabilities. Since both the definition of genres and the discrimination among them is in general very vague and problematic, the classification accuracies that we got for a set of 100 queries were not very high for different values of k. Thus, to improve the classification accuracy there are several aspects that we would like to experiment with as future work.

First, instead of using a simple classifier that does not require training, diverse and independent trained classifiers (and combination of them) can be used. Secondly, features can be extracted from short segments of music pieces to create multidimensional sequences, and then evaluate the performance of classifier(s) using SMBGT. Additionally, different parameters for SMBGT will be tested, for example by using cross-validation. What is more, as part of future work, polyphonic music will be analyzed in order to see if the compositional structure of songs is meaningful for the classification task. Finally, to build a more realistic automatic music genre classification system bigger datasets should be collected covering more genres (and even splitting them to subgenres).

Table 2: Classification accuracies with k-NN for the dataset of Table 1. For each value of k in [1,10] we present the following statistics. The number of songs that were classified having a clear winner genre (column "Non-Tied"), and having more than one tied genres to classify them to (column "Tied"). In addition, for each of the aforementioned columns, the number of songs that were correctly classified is shown in columns "Non-Tied Classified" and "Tied Classified", respectively, along with the corresponding classification accuracies (columns "Accuracy (Non-Tied)" and "Accuracy (Tied)"). The final classification accuracy for each k is also presented in column "Accuracy (Total)".

k	Non-Tied	Non-Tied Classified	Accuracy (Non-Tied) (%)	Tied	Tied Classified	Accuracy (Tied) (%)	Accuracy (Total) (%)
1	100	31	31	0	0	NaN	31
2	30	12	40	70	19	27.14	31
3	72	26	36.11	28	10	35.71	36
4	77	27	35.06	23	10	43.48	37
5	76	29	38.16	24	8	33.33	37
6	66	25	37.88	34	14	41.18	39
7	80	32	40	20	7	35	39
8	83	31	37.35	17	7	41.18	38
9	88	34	38.64	12	5	41.67	39
10	75	30	40	25	10	40	40

Acknowledgments

This work was partially supported by grants from the National Science Foundation, IIS-0812601, IIS-1055062, CNS-1059235, IIS-1329119, and CNS-1035913, and in part by the Academy of Finland ALGODAN Centre of Excellence.

6. REFERENCES

[1] N. Adams, M. Bartsch, J. Shifrin, and G. Wakefield. Time series alignment for music information retrieval. In *Proceedings of ISMIR*, pages 303–311, 2004.

[2] J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93, 2003.

[3] A. Berenzweig, D. Ellis, and S. Lawrence. Anchor space for classification and similarity measurement of music. In *Proceedings of ICME*, volume 1, pages 1–29, 2003.

[4] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proceedings of SPIRE*, pages 39–48, 2000.

[5] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and a da b oost for music classification. *Machine Learning*, 65(2):473–484, 2006.

[6] Z. Cataltepe, Y. Yaslan, and A. Sonmez. Music genre classification using midi and audio features. *EURASIP Journal on Advances in Signal Processing*, 2007(1):275–279, 2007.

[7] R. Cilibrasi, P. Vitányi, and R. Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

[8] R. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition. In *Proceedings of ICMC*, pages 344–347, 1997.

[9] F. Fabbri. A theory of musical genres: Two applications. *Popular Music Perspectives*, 1:52–81, 1982.

[10] Z. Fu, G. Lu, K. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, 2011.

[11] N. Hu, R. Dannenberg, and A. Lewis. A probabilistic model of melodic similarity. In *Proceedings of ICMC*, pages 509–515, 2002.

[12] C. Iliopoulos and M. Kurokawa. String matching with gaps for musical melodic recognition. In *Proceedings of PSC*, pages 55–64, 2002.

[13] J. Jang and M. Gao. A query-by-singing system based on dynamic programming. In *International Workshop on Intelligent Systems Resolutions*, pages 85–89, 2000.

[14] A. Kotsifakos, P. Papapetrou, J. Hollmén, and D. Gunopulos. A subsequence matching with gaps-range-tolerances framework: a query-by-humming application. *Proceedings of VLDB*, 4(11):761–771, 2011.

[15] A. Kotsifakos, P. Papapetrou, J. Hollmén, D. Gunopulos, and V. Athitsos. A survey of query-by-humming similarity methods. In *Proceedings of PETRA*, 2012.

[16] A. Kotsifakos, P. Papapetrou, J. Hollmén, D. Gunopulos, V. Athitsos, and G. Kollios. Hum-a-song: a subsequence matching with gaps-range-tolerances query-by-humming system. *Proceedings of the VLDB Endowment*, 5(12):1930–1933, 2012.

[17] J. B. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.

[18] J. Lee and J. Downie. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *Proceedings of ICMIR*, pages 441–446, 2004.

[19] K. Lemström and E. Ukkonen. Including interval encoding into edit distance based music comparison and retrieval. In *Proceedings of AISB*, pages 53–60, 2000.

[20] T. Lidy and A. Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of ISMIR*, pages 34–41, 2005.

[21] S. Lippens, J. Martens, and T. De Mulder. A comparison of human and automatic musical genre classification. In *Proceedings of ICASSP*, volume 4, pages 233–236, 2004.

[22] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of ISMIR*, pages 594–599, 2005.

[23] D. Mazzoni and R. Dannenberg. Melody matching directly from audio. In *Proceedings of ISMIR*, pages 17–18, 2001.

[24] C. McKay and I. Fujinaga. Automatic music classification and the importance of instrument identification. In

Proceedings of CIM, 2005.

[25] C. McKay and I. Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved. In *Proceedings of ISMIR*, pages 101–107, 2006.

[26] R. McNab, L. Smith, I. Witten, C. Henderson, and S. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of ICDL*, pages 11–18, 1996.

[27] C. Meek and W. Birmingham. A comprehensive trainable error model for sung music queries. *Journal of Artificial Intelligence Research*, 22(1):57–91, 2004.

[28] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[29] F. Pachet, D. Cazaly, et al. A taxonomy of musical genres. In *Proceedings of Content-Based Multimedia Information Access (RIAO)*, pages 1238–1245, 2000.

[30] B. Pardo and W. Birmingham. Encoding timing information for musical query matching. In *Proceedings of ISMIR*, pages 267–268, 2002.

[31] S. Pauws. Cubyhum: A fully operational query by humming system. In *Proceedings of ISMIR*, pages 187–196, 2002.

[32] D. Perrot and R. Gjerdigen. Scanning the dial: An exploration of factors in the identification of musical style. In *Proceedings of SMPC*, page 88, 1999.

[33] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *Proceedings of ICDE*, pages 1046–1055, 2007.

[34] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *Signal Processing Magazine*, 23(2):133–141, 2006.

[35] J. Shifrin, B. Pardo, C. Meek, and W. Birmingham. HMM-based musical query retrieval. In *Proceedings of JCDL*, pages 295–300, 2002.

[36] H. Soltau, T. Schultz, M. Westphal, and A. Waibel. Recognition of music types. In *Proceedings of ICASSP*, volume 2, pages 1137–1140, 1998.

[37] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[38] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In *Proceedings of ACM Multimedia (Part 1)*, pages 57–66, 1999.

[39] E. Unal, E. Chew, P. Georgiou, and S. Narayanan. Challenging uncertainty in query by humming systems: a fingerprinting approach. *Transactions on Audio Speech and Language Processing*, 16(2):359–371, 2008.

[40] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of SIGMOD*, pages 181–192, 2003.