GENRE CLASSIFICATION VIA AN LZ78-BASED STRING KERNEL

Ming Li
School of Computing Sciences
University of East Anglia
NORWICH NR47TJ, UK
mli@cmp.uea.ac.uk

Ronan Sleep
School of Computing Sciences
University of East Anglia
NORWICH NR47TJ, UK
mrs@cmp.uea.ac.uk

ABSTRACT

We develop the notion of normalized information distance (NID) [7] into a kernel distance suitable for use with a Support Vector Machine classifier, and demonstrate its use for an audio genre classification task. Our classification scheme involves a relatively small number of low-level audio features, is efficient to compute, yet generates an accuracy which compares well with recent works.

Keywords: genre, classification, LZ78 String Kernel, SVM.

1 INTRODUCTION

As collections of music, both personal and public, become ever larger, the desire for increasingly intelligent music retrieval systems grows. A key problem for such systems is the audio genre classification problem. This has become an increasingly active area of research in recent years. The state of the art is that automatic genre classification systems are beginning to approach, and sometimes exceed, human performance.

Generally, such a system can be divided into two parts: feature extraction and classification. In the first part, audio sampled are converted into a sequence of acoustic features. In the second part, these features are input to a classifier which yields an estimate of the sample genre.

Features used typically fall into three categories: timbral texture, rhythm and pitch [3]. Intuitively, the presence of features from all three categories would seem desirable, but good performances have been reported using only some. For example, [1] reports good results using only pitch contour. Advanced signal processing ideas can be used to extract higher level features: for example in [9], a wavelet-based feature called DWCHs is used to obtain good results.

A wide variety of classifiers are available for the second stage, e.g. Gaussian Mixture Modelling (GMM), K Nearest Neighbour (KNN), Support Vector Machine (SVM) as described in [3,9,10]. Artificial Neural Networks (ANNs) and learning tree vector quantizer have also been used [11,12].

One difficulty is that most classifier technology is oriented towards fixed length vector data. Thus somewhere between the raw data and the classifier, the variable length signal is converted to a fixed length vector.

A simple way of converting variable to fixed length is to use some sort of averaging over the whole music piece. This might work well where timbral features are particularly important in discriminating between genres, especially when used in conjunction with a good classifier e.g. as in [2]. However, it seems a pity to lose sequence information by such averaging, and this has motivated the use of sequence oriented approaches such as Hidden Markov Model (HMM) and Statistical (*n*-gram) Language Models. These have been explored respectively in [19] for genre classification and in [15] for singing language identification.

HMM and *n*-gram models seem attractive but for various technical reasons they tend to be used only with small numbers of states. Our main contribution in this paper is to introduce and explore a novel approach to passing information about variable length data sequences to a standard vector classifier in a way which takes account of the patterns which actually appear in the training data, as opposed to those restricted to some arbitrary maximum length.

For this work we concentrate on using SVM for the classification stage. This has become a popular tool in the last few years, and it achieves good performance in a wide range of pattern recognition tasks [1,4].

Specifically, we present a new class of string kernel, which is based on Lempel-Ziv-type coding algorithm [16], and use it to drive an SVM classifier for music genre classification. In our scheme, the audio sample represented by a sequence of spectrum-based feature vectors are approximated and converted by a Vector Quantizer (VQ) into a sequence of 1-dimensional vector. Then, we use our string kernel to calculate the similarity of two such vector sequences by analyzing the temporal characteristics shared between them. Our scheme for calculating the similarity between variable length feature sequences is explained below. Our results indicate that we obtain an improved performance compared with fixed *n*-gram techniques.

The rest of this paper is organized as follows: in the next section, we give an overview of our classification scheme; section 3 explains briefly the notion of normalized information distance (NID) – this is a similarity function proposed in [7] and based on the concept of Kolmogorov Complexity (KC); in section 4, we explain how we extend NID into a valid kernel distance and used it for sequence classification. Essentially, our method maps a sequence to a high-dimensional feature space which is indexed by all the phrases identified by a simple adaptive coding parse of the sequence. Experiment results and discussion are shown in section 5 and 6. Finally, we conclude in section 7.

2 OVERVIEW OF CLASSIFICATION SCHEME

Audio signal are temporal in nature, and our feature extraction scheme is designed to respect this. As shown in Figure 1, it begins with a sequence of audio samples representing a particular piece of music, and then applies the following steps: 1) convert the sequence of audio samples into a sequence of cepstral features inspired by speech recognition technology; 2) apply a vector quantizer to discretize the feature space into small blocks and label them by index; this converts the sequences of cepstral feature vectors into a (variable length) sequence of VQ codebook indices; in step 3), pairs of these event sequences are used to compute a similarity measure (in a manner developed in section 4) to construct a kernel matrix for an SVM classifier.

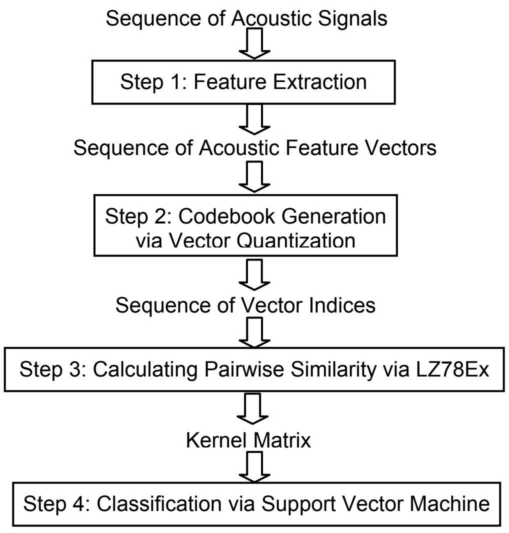
Sequence of Acoustic Signals
Step 1: Feature Extraction
Sequence of Acoustic Feature Vectors
Step 2: Codebook Generation via Vector Quantization
Sequence of Vector Indices
Step 3: Calculating Pairwise Similarity via LZ78Ex
Kernel Matrix
Step 4: Classification via Support Vector Machine

Figure 1 Block diagram of classification scheme

Our scheme has aspects in common with Tsai and Wang's work [15], but we use only one codebook to be generated across all the classes instead of one for each class. This means that the pre-processing stages (steps 1 and 2) are independent of the actual classes of interest, and so the pre-processing result (i.e. codebook) learnt from one task might be regarded as background knowledge and used for a variety of classification tasks. A more fundamental difference is that the fixed-bi-gram language model used in [15] is replaced by our string kernel method that can build a set of characteristic patterns of varying length from the data.

2.1 Audio Parameterization

Since our work focuses on a novel string kernel and its performance in audio genre classification, we decided to adopt a simple but very popular low level signal feature set that is perceptually motivated and has been successfully used in speech recognition: MFCC[1] coefficients plus an energy term. Moreover, since the performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters, we explored their use. In our experiments, the final feature set includes delta values and acceleration values that are calculated using the following regression formula:

$$energy = \log \sum_{n=1}^{N} s_n^2 \qquad (1)$$

$$time\_derivative = \frac{\sum_{i=1}^{M} i(v_{t+i} - v_{t-i})}{2\sum_{i=1}^{M} i^2} \qquad (2)$$

s_n is the signal energy for audio samples, v_i is a delta coefficient at time t computed in terms of the corresponding static coefficients v_{i-t} to v_{i+t}.

2.2 Vector Quantization (VQ)

The VQ step maps the potentially huge space of MFCC-level signal sample to much smaller element space of codebook indexes.

Formally, a vector quantization of n dimensions can be described as a mapping of vector V from an n-dimensional space S to a subset S' of it, where S' consists of K codewords in S. The VQ scheme block diagram is shown in Figure 2. In the encoding procedure, the input vector searches the codebook and finds the closest codeword for that input and the corresponding vector reference index is output. In our work, a diagonal-covariance Mahalonobis distance metric is employed and a linear codebook is constructed for which minimizes the quantization error.

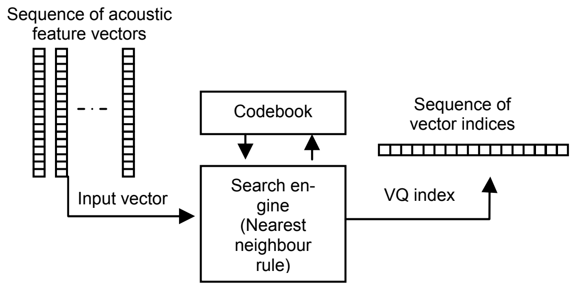
Sequence of acoustic feature vectors
Codebook
Sequence of vector indices
Input vector
Search engine (Nearest neighbour rule)
VQ index

Figure 2 Block diagram of vector quantization

2.3 Key Concepts in SVM

SVM is a powerful supervised learning algorithm, which maps data into a high dimensional feature space where data is more likely to be linearly separable. A linear decision boundary is obtained by maximizing the geomet-

[1] Mel Frequency Cepstral Coeficients

ric margin in order to make it as far away from the data as possible whilst separating the two classes correctly. The decision boundary learnt by SVM is entirely based on the information provided by a kernel matrix (called Gram Matrix) with entries of similarity score measured by pairwise inner product.

At the heart of SVM is a constraint quadratic optimization solver which works with the dual problem using only the inner product of input data pairs (this is the similarity measure). This inner product may be specified in terms of a kernel function which does the mapping to the high dimensions, but by working with the dual problem only inner products are needed and these may be evaluated in R^2 or even specified directly as a similarity measure between pairs.

When the data is not linearly separable in the high dimensions, a modified SVM procedure can be used, which finds a trade-off between maximizing geometric margin and minimizing the cost of misclassification. This is achieved by introducing "slack" variables, which allow the margin constraint to be violated

3 NORMALIZED INFORMATION DISTANCE

A Normalized Information Distance (NID) as proposed in [7] is a metric measuring pairwise similarity between sequences. Informally, it is the ratio of the information not shared by the two sequences to the total information content of the pair of sequences. This is illustrated in Figure 1 where circle A is K(x) that represents the Kolmogorov complexity of object x, circle B is K(y) and the total area of two circles (A+B+C) is K(xy). Two identical sequences will have NID=0, whilst two sequences with no common information content will have NID=1.

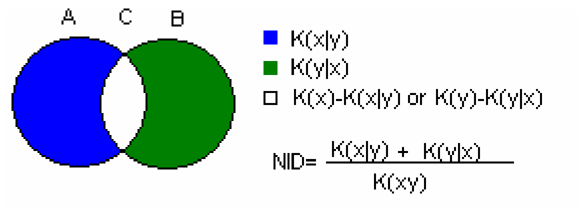

Figure 1. Conceptual illustration of normalized information distance.

3.1 Conditional Kolmogorov Complexity

Given an object encoded as a binary string x, its Kolmogorov complexity $K(x)$ or algorithm entropy is the minimum number of bits into which the string can be compressed without losing information [13]. Intuitively, Kolmogorov complexity indicates the descriptive complexity contained in an object and it is the length of the shortest program for some universal machine which, when run without any input, outputs that string. This is an idealized notion, because it is not computable. However, any compression algorithm gives an upper bound and this can be taken as an estimate of the Kolmogorov complexity.

A random string has relatively high complexity since no structural pattern can be recognized to help reduce the size of program. Strings like structured texts and melodies should have lower complexity due to repeated terms and musical structure.

As for conditional Kolmogorov complexity, $K(x|y)$, it is defined as the shortest program that can regenerate the sequence x from the sequence y. $K(x)$ is the special case of $K(x|\lambda)$ where λ is the empty sequence. In [7], $K(x|y)$ is estimated as the difference of the unconditional complexity estimates $K_c(xy)$ and $K_c(y)$:

$$K_c(x|y) = K_c(xy) - K_c(y)$$

Here xy stands for the concatenation of sequences x and y. A minor issue arises when using algorithms such as compressors to estimate KC: in general the order of concatenation affects the size of the compressed concatenation, so that the relation $K_c(xy) = K_c(yx)$ may not hold for our estimates. This issue can be handled by using the average of the two ordering.

3.2 Normalized Information Distance

The information distance [14] between two sequences x and y can be defined as the length of a shortest binary program that computes x given y, and also computes y given x. However, such a distance does not take the length of the sequence into account. This motivates the desire for a relative (or normalized) measure that takes account of sequence length. If two pairs of sequences have the same information distance but with different lengths, the longer pair should be given a smaller distance measure than the shorter pair, reflecting the fact that more information is shared between longer sequences.

The authors in [7] use conditional Kolmogorov complexity as the basis of a normalized information distance $D(x, y)$ for measuring the similarity relations between sequences. Two versions of the similarity metric are proposed in [7].

$$d1(x,y) = \frac{K(x|y) + K(y|x)}{K(xy)} \quad (3)$$

$$d2(x,y) = \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))} \quad (4)$$

The second definition can be shown to satisfy the conditions enumerated above without qualification, and in that sense is more satisfactory than the first. But conditional Kolmogorov complexity is not computable, so we have to rely on estimates of KC. This makes it less clear that strict application of the mathematical elegance criterion is the most important guideline for practical classification work. Indeed, in our earlier work in [1], we found the practical performance of equation 3 is slightly better than that of equation 4.

4 THE LZ78-BASED STRING KERNEL

In [1], we used an LZ78 parser to provide the estimates of the Normalized Information Distance (NID) between pitch contours extracted from two MIDI files. The results were encouraging. We then began looking how to analyze sequences of audio instead of symbolic data: here the 'symbols' which should be chosen as a basis for a NID method are not so clear – this is the purpose of our VQ preprocessing stages.

We also wanted to explore the recent success of string kernel methods in the context of music. It occurred to us that we could use the dictionaries generated by an LZ78 parser to identify commonly occurring words in a sequence, and then use a string-kernel like approach to drive an SVM classifier. Hence we introduce a new class of string kernel, called LZ78Ex kernel.

Given two sequences x and y, our kernel distance is estimated by the degree of sharing between the phrases of two dictionaries D_{lz78ex}(x) and D_{lz78ex}(y) produced by a modified LZ78 parse of the sequences. The modified LZ78 generates frequency counts, and these are used in computing the kernel distance. Note that LZ78 is run exactly once for each piece of music in the data set: the pairwise kernel metric takes as input the two dictionaries.

4.1 A Modified LZ78 Parsing Scheme

At the heart of our kernel is a simple adaptive parsing algorithm based on the LZ78 compression scheme, which is originally proposed in [16]. It works by identifying patterns, called phrases, of the data and stores them in a dictionary (i.e. encoding table) that defines commonly occurring substrings, and associates them with dictionary indexes that are used to represent the phrases in the compressed output. The outline code shown below captures the essence of our modified LZ78 encoder omitting irrelevant details:

```
Clear dictionary;
w = λ[1];
while (more input)
  C = next symbol;
  Pattern=wC
  if ( Pattern in dictionary)
    w = Pattern;
  else
    add Pattern to dictionary
    w = λ;
  endif
  UpdateFrequency(pattern)
endwhile

Return dictionary;
```

Normally, the LZ78 parsing algorithm starts with a dictionary initialized with codes for each possible value in the sequence and emits code consisting of a compact reference to the each new entry. As we are only interested in the entries which actually occur in the data sequences, in our implementation, we initialize the dictionary to be empty and update pattern frequency at each step. Note that (1) unlike the *n*-gram approach, LZ78 parsing produce a set of features with varying length (2) it identifies increasingly long initial portions of repeated phrases gradually: that is, it will need to see a phrase of length *L* on *L* occasions before it remembers the whole phrase in its dictionary.

[1] 'λ' represent the empty string.

4.2 LZ78Ex Kernel

Recall that the pre-processing steps of our scheme convert a sequence of acoustic feature vectors into a sequence *S* of symbols from a finite alphabet (the set of VQ indexes). When this sequence is parsed using our modified LZ78 parser, we obtain a dictionary *D(S)* of commonly occurring 'words' in the VQ index alphabet ∑. The set of all such words ∑* provides an infinite set of features. We take as the basis for our similarity measure between two sequences *S* and *S'* the inner product of the vectors representing *D(S)* and *D(S')*. In fact our representation of a sequence *S* in this space is a weighted vector Φ_{lz78ex} with *i*-th element indicating the number of times *i*-th pattern occurring in *S*. The LZ78Ex kernel distance between two sequences *S* and *T* is then defined as the inner product of their feature vectors in which the elements are weighted by their occurrence frequencies:

$$K_{lz78ex}(s,t) = < \Phi_{lz78ex}(s), \Phi_{lz78ex}(t) > \quad (5)$$

Finally we normalise our kernel to take account of the actual size of the two feature vectors as follows:

$$K^{norm}_{lz78ex}(s,t) = \frac{K_{lz78ex}(s,t)}{\sqrt{K_{lz78ex}(s,s)K_{lz78ex}(t,t)}} \quad (6)$$

Note that while the feature space has infinite dimensionality, computing the inner product is a linear in the size of the features actually present in the data. The LZ78 parse itself is highly efficient, especially when using appropriate trie structures and/or hashing. It follows that the overall cost of computing LZ78Ex kernel is bounded above by *O*(*n*). Note also that the LZ78Ex kernel satisfies Mercer's conditions (symmetry and positive semi-definiteness). This follows from its definition as the inner product of two feature vectors.

5 EXPERIMENTAL SETUP

5.1 Dataset and Tools

The dataset[2] used in our experiments appears to be that used in [3,9], though we are not certain of this. The set consists of 10 genres: classical(Cl), country(Co), disco(Di), hip-hop(Hi), jazz(Ja), rock(Ro), blues(Bl), reggae(Re), pop(Po) and heavy-metal(Me). Each genre consists of about 100, 30 second samples. For all LZ78-related experiments, the feature vectors were extracted and quantized using HTK.3.2.1[3]. All the classification

[2] http://opihi.cs.uvic.ca/sound/genres

[3] A toolkit for building hidden markov model available at http://htk.eng.cam.ac.uk.

experiments were conducted using LIBSVM package [17], which provides an interface to utilize user-defined kernel matrix. Performance evaluation was accomplished using a stratified 10 cross validation (CV), which avoids overlapping test sets and approximate the original proportions of labels within each subset. Extensive experiments have shown that this is the best choice to get an accurate estimate. There is also some theoretical justification for this approach.

5.2 Sub-Genre Effect: Biased and Unbiased 10-Fold Cross Validation

One of the many drawbacks of not having properly constructed corpuses for running experiments on music genre classification is that it is necessary to either build ones own corpus, or borrow from others. An unconscious effect of this is that present data sets are stratified into sub-genres – for example the songs from a special singer or a sub-source with other special characteristics (e.g. vocal classical as opposed to concerti). This can lead to subtle effects on performance evaluation, and it is not clear that all work to date has guarded against this effect.

To see how important the effect is, we use two different strategies to conduct the 10-fold cross validation which will be called *un-biased* and *biased* respectively:

1) *Un-biased Strategy:* this approach refers to standard stratified 10-fold CV that firstly shuffles the data randomly and then cross validates them. We call this un-biased since the examples from all the sub-genre can be well presented in both training and test dataset.

2) *Biased Strategy*: in this approach, each genre is manually divided into 10 groups (sub-genres) according to the number used to label the audio files (e.g. metal.00000 to metal.00009 belong to the first group in genre metal' etc). We found that two closely numbered files in each genre tend to sounds similar than the files numbered far away from each other. Thus, each group is presumably to represent one special sub-genre. In *i*-th cross validation, the test data are constructed by combining the *i*-th group from each genre and the rest groups are used for training. We call this 'biased' since the stratification effect may lead to biased presence of some sub-genres in the training/test sets.

6 RESULT AND ANALYSIS

Preliminary experiments described in 6.1 and 6.2 were run to determine reasonable choices of MFCC granularity, codebook size for the vector quantization processing, and the choice of audio frame length. Sections 6.3 and 6.4 compare our method with other approaches. Unless otherwise stated, all the results are obtained with biased data using pairwise SVM[1].

6.1 Performance Comparison of 5 and 12 Coefficients

We looked at the performance of 12 coefficients as against 5 coefficients. The latter are reported in [3] as providing the best genre classification results when using a Gaussian mixture model (GMM). In contrast[2], as can be seen in Table 1, the choice of 5 gives worse performance than 12 when we use our string kernel based method. This suggests that our technique is exploiting the additional information provided. We obtained a small improvement in performance by adding one additional energy term. The effect of this addition is shown in table 1 below.

Table 1. Results based on 100ms hamming-windowed frame without overlap. Codebook size is 512.

Features	Accuracy(STD)	
	Without the energy term	With the energy term
5 MFCC	62.86 (6.14)	63.84 (7.22)
12 MFCC	67.15 (6.99)	67.65 (7.15)

6.2 Effects of Codebook Size and Audio Frame Length

The choice of codebook size relates in part to the structure and amount of the audio training data. Although a value of 256 is commonly used in speech recognition, there is no commonly agreed value in audio genre classification. We found that the codebook with 512 VQ symbol appears to give a stable good performance. Also, from Table 2, the average classification accuracy based on 25ms frame length is up to 75%, which is significantly higher than 68% on 100ms frame length.

Table 2. Accuracy based on different codebook size and frame length.

Codebook Size	256	512	1024
100ms audio frame	64.44 (5.18)	67.65 (7.15)	68.57 (6.48)
25ms audio frame	71.04 (6.94)	73.69 (5.99)	74.45 (6.01)

Parameters such as codebook size and frame length can be tuned to affect performance significantly, as shown for example by Aucouturier and Pachet [8]. These authors report detailed experiments exploring a

[1] SVM is a binary classifier in nature. Pairwise and one-against-the-rest are two prediction strategies to extend SVM for multi-class problem. More details are shown in [18]

[2] Perhaps the discriminating power of the extra 7 coefficients is blunted somewhat by the constrained nature of a GMM.

large parameter space, and suggest that there is a 'glass ceiling' of around 65% on performance for the techniques which they investigated.

6.3 Classification Results

Table 3 shows the details of the confusion matrix generated by our method with a 25ms Hamming-windowed frame without frame overlap. The columns are placed in an order such that the most confused genres are put close to each other. The matrix column corresponds to the predicted genre and the row shows the real genre. For example, the value 8 in row 2 column 1 indicates that around 8% of hip-hop music (column 1) was misclassified as reggae.

Table 3. Confusion matrix for the 10-genre task.

	Hi	Re	Di	Po	Me	Ro	Co	Bl	Ja	Cl
Hi	**77**	14	5	3	0	2	0	0	0	0
Re	8	**68**	6	1	0	5	1	2	0	0
Di	4	5	**73**	3	1	5	4	3	0	0
Po	4	4	7	**85**	0	6	8	0	0	1
Me	2	1	0	0	**85**	5	0	3	0	0
Ro	3	3	9	4	12	**54**	16	10	4	1
Co	2	2	0	4	0	12	**63**	3	4	1
Bl	1	3	0	0	1	8	4	**76**	4	0
Ja	0	0	0	0	1	2	4	4	**80**	4
Cl	0	0	1	0	0	2	0	0	9	**94**

The genres and their two letter codes are: classical(Cl), country(Co), disco(Di), hip-hop(Hi), jazz(Ja), rock(Ro), blues(Bl), reggae(Re), pop(Po) and heavy-metal(Me).

The general level of performance is high, the main exception being for Rock music which is fairly evenly confused with a number of other popular music genres: this may indicate that rock is better treated as a super genre in a hierarchical classifier.

6.4 Comparison with Fixed-*n*-gram Methods

The N-Gram model is a simple and effective approach which has been successfully applied in series of tasks for modelling temporal constraints in the sequences. One of its variants called k-spectrum kernel is implemented here for comparison purpose. The k-spectrum of a sequence input is the set of all the k-grams (contiguous) contained in it. Given two sequences, the k-spectrum kernel is defined as the inner product of their k-spectrum feature vectors [5]. The results in Table 4 demonstrate the advantage of our method over fixed-n-gram model. We attribute this to the one of the principal differences between n-gram approach and ours, i.e. the n-gram restriction over the pattern (i.e. feature) length. Our method effectively builds an appropriate variable length set of n-grams from the data, without pre-determining a limit on *n*. The effect of artificially constraining *n* is to cut off the tail from the distribution shown in figure 2 – not surprisingly, this affects performance adversely.

In principle of course, an *n*-gram approach which sets *n* according to the largest word encountered in an LZ78 scan of the training set could be used. However, this approach involves a pre-scan, and encounters problems of increasing computational cost. There are also problem with increasing sparsity in the resulting kernel matrix for n-gram methods as *n* gets larger.

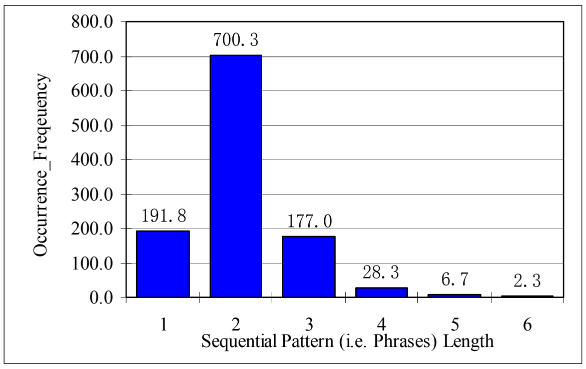

Figure 2. Averaged phrase length distribution within a single audio piece (audio signal is analyzed with 25ms hamming windowed frame without overlap).

Table 4. LZ78Ex compared with fix-n-gram.

Method	Accuracy (STD)
LZ78Ex Kernel	74.45 (6.01)
2-Gram Kernel	70.77 (5.01)
3-Gram Kernel	62.81 (5.75)
'2+3'-Gram Kernel	69.50 (5.64)

6.5 Comparison with Other Methods

We believe that our dataset is similar, if not identical to that used by Tzanetakis for the work published in [3]. Tzanetakis used a comprehensive genre features and obtained an accuracy of around 61(±4)% using a GMM method. Li[9] uses Tzanetakis's feature set but employs a SVM classifier instead of a GMM: this gives an improvement in performance of nearly 10%. In the same paper, the author explores a new wavelet based feature set called Daubechies Wavelet Coefficient Histogram (DWCHs)[1], and reports further improvement in performance achieving 75% to 78% respectively with pairwise and one-against-the-rest SVM [18]. To compare our performance more objectively with those previously published results, we re-implement the method proposed in [9] and use MARSYAS[2] to extracting the genre features proposed in [3]: MFCCs, FFT, Beat and Pitch.

[1] DWCH consists of the first three moments of the histogram plus the sub-band energy. Since not all the frequency sub-bands are informative, the authors in [9] use only selective sub-bands

[2] A public software framework for computer audition application that can be downloaded from http://www.cs.princeton.edu/~gtzan/marsyas .html.

Table 5. Accuracy over the dataset. SVM1 and SVM2 respectively denote the pairwise SVM and the one-against-the-rest SVM.

Feature Category	Kernel Function	Biased Evaluation		Unbiased Evaluation	
		SVM1	SVM2	SVM1	SVM2
Genre Feature (MFCC + FFT + Beat + Pitch)	RBF	69.96 (5.52)	70.39 (3.73)	72.24 (4.89)	72.60 (4.15)
DWCH (MFCC + FFT + statistical moments from selective sub-bands)	RBF	71.01 (7.15)	70.65 (5.67)	74.90 (3.82)	78.16 (3.62)
MFCC + Energy	**LZ78Ex**	**74.45 (6.01)**	**74.45 (5.37)**	**80.35 (5.64)**	**80.72 (4.08)**

Table 5 display the accuracy of various features and classifiers over the dataset. The bottom row shows that our method outperforms the others consistently in both biased and unbiased evaluation. Comparing the first and second row, DWCH performs better than Tzanetakis's genre feature, we attribute this to wavelet decomposition scheme which matches the models of sound octave division for perceptual scales and provides good time and frequency resolution. Li's works in [9] are repeated in our experiments and the best performance of DWCH (74.9% and 78%) is confirmed: note that they occurred only in unbiased evaluation. As shown in second row of Table 5, when training and testing data are constructed in a biased way, DWCH's performance is underestimated as 71%. Apparently, this may be due to the sub-genre effect mentioned previously in section 5.2.

Our LZ78Ex method is also affected by this problem. However, by taking advantage of temporal information inherit in the sequence, our method still outperform alternative techniques in both situations.

7 CONCLUSION AND FUTURE WORK

Feature extraction is important for music genre classification. To date, much work in this area has used averaged features such as FFT, MFCC, Beat, Pitch or DWCHs. These time averages, especially when combined, can produce good performance. However, as reported in [8] there seems to be a 'glass ceiling' on performance.

We have shown that, by utilizing temporal structure in a sequence of spectrum-based features, rather than averages over sampled windows, we can achieve significantly improved performance. Our new techniques, called the LZ78Ex kernel, is now one of a number of methods which outperforms humans, yielding significantly better than 70% human musical genre classification accuracy reported in [6].

We believe that our new highly efficient LZ78Ex technique for mapping pairs of variable length sequences to a pair of finite dictionaries, and then using the dictionary pair to compute a similarity measure, is our most important contribution. However, some of our performance may be due to our choice of classifier: we used the increasingly popular SVM (Support Vector Machine) technique. This, and other factors such as the different choices of features, and the lack of very high quality labelled reference data corpora, make comparison with other techniques hard. To take just one aspect, we showed in our experiments that the use of biased evaluations may 'under-estimate' the performance by 8% due to the imbalanced representation of sub-genres within the training data. We took the view that the biased approach gives a more reliable estimation since it is hard to seek accurate performance over a full range of instances in the real world classifier application.

Future work needs to explore the effectiveness of other feature extraction and fast vector quantization techniques, which may improve our audio genre classification system both in accuracy and computational cost. In [19], an Octave-based Spectral Contrast feature is proposed and proved to be more discriminative than MFCC in terms of genre classification. In [12], a supervised tree-based quantization is proposed. It attempts to label feature vectors from different classes with a different label. Thus, the new quantizer could discretize the feature space into many more regions than the conventional minimum distortion vector quantizers.

8 ACKNOWLEDGMENTS

We thank Tao Li for useful discussions and the http://opihi.cs.uvic.ca/sound/genres contributor for providing us the data set. We also thank Kris West and Stephen Cox for interesting and useful discussion, and Gavin Cawley who gave invaluable technical advice on the properties of our kernel.

REFERENCES

[1] Li M. and Sleep M.R. "Melody classification using a similarity metric based on Kolmogorov complexity". Proceeding of Conference on Sound and Music Computing, Paris, 2004

[2] Kristopher West and Stephen Cox. “Features and classifiers for the automatic classification of musical audio signal”, In Proceedings of the 5th International Symposium on Music Information Retrieval, Barcelona, SPAIN 2004.

[3] George Tzanetakis and Perry Cook, “Music genres classification of audio signals”, IEEE Transactions on Speech and Audio Processing, 2002.

[4] C. Xu et al., “Musical Genre Classification using Support Vector Machines,” in Proc. ICASSP ’03, Hong Kong, China, Apr.2003, pp. 429–432.

[5] Leslie, Eskin et al. “The Spectrum Kernel: A String Kernel for SVM Protein Classification”. In Proceedings of the Pacific Symposium on Biocomputing, 2002, pp. 564-575.

[6] D. Perrot and R. Gjerdigen. “Scanning the dial: An exploration of factors in identification of musical style”, in Proc. Society for Music Perception and Cognition, 1999, p.88, (abstract).

[7] Li M., Chen X., Ma B. and Vitanyi P. “The similarity metric” Proc. 14th ACM-SIAM Symposium on Discrete Algorithms 2003.

[8] Aucouturier, J.-J. and Pachet F. “Improving Timbre Similarity: How high is the sky?”. Journal of Negative Results in Speech and Audio Sciences, 1(1), 2004.

[9] Tao Li, Mitsunori Ogihara and Qi Li. “A Comparative Study on Content-Based Music Genre Classification”. In Proceedings of Annual ACM Conference on Research and Development in Information Retrieval (SIGIR 2003), Pages 282-289.

[10] H. Deshpande, R. Singh, and U. Nam. “Classification of music signals in the visual domain”. In Proceedings of the COST-G6 Conference on Digital Audio Efffects, 2001.

[11] H. Soltau, T. Schultz, and M. Westphal. “Recognition of music types”. In Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, 1998.

[12] J. Foote. “Content-based Retrieval of music and audio”. In Multimedia Storage and Archiving Systems II, Proceedings of SPIE, pages 138-147, 1997.

[13] Li M. and Vitanyi P. “An Introduction to Kolmogorov Complexity and Its Applications”, Springer 1997

[14] Bennett C.H., Gacs P., Li M., Vitanyi P.M.B. and Zurek W. “Information Distance”. *IEEE Transactions on Information Theory*, 44:4 (1998), pp 1407-1423

[15] Wei-Ho Tsai and Hsin-Min Wang. “Towards Automatic Identification of Singing Language in Popular Music Recordings”. In Proceedings of the 5th International Symposium on Music Information Retrieval, Barcelona, SPAIN 2004

[16] Jacob Ziv and Abraham Lempel. “Compression of Individual Sequences via Variable-Rate Coding”. *IEEE Transactions on Information Theory*, vol. 1T-24, no.5, Sep. 1978

[17] H.-T. Lin and C.-J. Lin. “A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods”. Technical Report, Department of Computer Science and Information Engineering, National Taiwan University. Available at http://www.csie.ntu.edu.tw/cjlin/papers/ tanh .pdf

[18] C.-W. Hsu and C.-J. Lin. “A comparison of methods for multi-class support vector machines”. IEEE Transactions on Neural Networks, 13(2002), 415-425.

[19] Igor Karpov, “Hidden Markov classification for musical genres”, Tech. Rep., Rice University, 2002.