Music Genre Classification Through Lyric and Music Analysis

David J. Garcia
Dept. of Computer Science
University of Texas at Austin

Ivan Oropeza
Dept. of Computer Science
University of Texas at Austin

1. INTRODUCTION

The increasing accessibility to vast amounts of music in recent years has created demand for systems to manage and retrieve music of interest for users. The typical approach to accomplish this task is through music annotations. One of the most informative annotations to this day is music genres. A music genre is described as a categorical label created by humans to describe music [5]. However, music genres are not well defined and the boundaries between them are unclear due to the cultural interactions throughout human history — genres have been created, merged, or even split into sub-genres. Consequently, automatic genre classification is a complex task [4]. In this investigation we explore the use of lyric data in order to further improve genre classification. We train a classifier with audio features available in the Million Song Dataset (DMS) and language features extracted from lyrics and we evaluate the performance of our classifier as we increase the training data of our language feature extractor.

2. PROBLEM DEFINITION AND ALGORITHM

2.1 Task Definition

Genre classification is a special case of the more general problem of music annotation. In music annotation, a song is associated with descriptive labels such as mood, artist, and instruments in order to efficiently retrieve and manage songs from large music libraries [2]. Hence, the problem can be simplified and worked out as a classification problem. The input for this problem is the song file that needs to be annotated. In our investigation we augment the input to include the lyric file as well. This additional input can be fetched from an API or the web itself using the song's title and artist's name. The output is a collection of descriptive labels. However for our experiments, it will only be a single-class classification label, the genre. Specifically, the output will be a string from one of the genres we study (Electronic, Pop, Rock, and Punk). The difficulty of the problem arises from the fact that music genres are poorly defined and their boundaries remain unclear. It is historically common for songs to have properties of multiple genres. For example, the Rococo period is the era when songs had properties of both the Baroque and Classical periods. As a consequence, new genres have been created in an attempt to preserve the single class nature of the problem. Moreover, genres have also seen specialization and as a result new sub-genres have been created. For example, Jazz has various sub-genres like Fusion, Piano, and Quartet and each has their unique characteristics while still maintaining Jazz's characteristics [5]. Consequently, genre classification, even for the average person, is hard. Past studies have shown that humans have an accuracy between 53% and 70%. Therefore, automatic genre classification is a difficult task.

2.2 Algorithm Definition

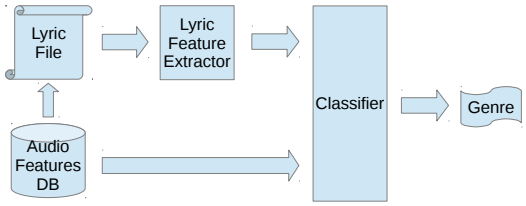

Figure 1: Overall experimental framework. The MSD files contains audio features which are aggregated/summarized in the classifier script. Each song's respective lyrics-file passes through the lyric-extractor where it is scored by four Bigram models; the scores are concatenated to the summarized audio features. These combined features are quantized (using K-Means) into four dimensional vectors. The quantized vectors are applied directly to the classifier.

In our project we use Scikit. Scikit is a python module for data analysis that provides a consistent framework for various classification algorithms [3]. We train two "one vs one" SVM classifiers with a polynomial and radial kernel respectively. We also train a "one vs rest" SVM classifier, K-Nearest Neighbors with one and three neighbors, and a decision tree. We use the default configurations for all classifiers.

In early experiments, the 52-dimensional vector (48 audio-features, and 4 language features) became unwieldy. To

compensate, we decided to quantize the data by first applying K-Means to generate four clusters. The fact that we were trying to classify songs into four genres led us to choose $K = 4$. After clustering, each 52-dimensional vector was replaced by a 4-dimensional vector— each cell of which represented the distance between the original 52-dimensional vector and the 52-dimensional cluster point the cell represented.

We decided to utilize a bidirectional Bigram for our language model. As we mentioned, we train a Bigram for each genre. Perplexity scores are generated by each model for each song. The scores are then normalized and concatenated to the audio features. The normalization is discussed in detail in the experiments section.

3. EXPERIMENTAL EVALUATION

3.1 Methodology

In our investigation we consider four genres (Rock, Pop, Electronic, and Punk). Scaringella et al. report that Punk is an easily classifiable genre — systems reach 100% accuracy. On the other hand, Rock and Electronic are confused about 20% of the time[4]. We also consider Pop due to the variety of topics and themes in this genre. For each genre we sampled 500 songs. For each song, we retrieved the audio features from MSD and manually retrieved the lyrics from the web in order to construct lyric features. We also labor to sample in such a way as to reflect artists' representation in the sample; that is, if an artist is highly prolific, the probability that one of his/her songs will be included in our sample is relatively high. We also try to refrain from including non-English songs. The lyrics came from various websites so there is no consistent format for the lyrics.

We partition each genre's 500 samples into two groups: 1) The language model training set (50 to 250 data points), 2) The classifier dataset (200 data points).

The classifier dataset (200 points) is selected uniformly at random and remains unchanged throughout all experiments. A seed is used to ensure that subsequent experiments produce identical results. The language model dataset (50-250 data points)for each genre is selected uniformly at random; as before, a seed is used. Our experiments are meant to show how classifier performance improves/degrades as more data points are used to train the language model. To this end, we incrementally increase the number of language-model data points by 50.

The audio features come from the MSD dataset but they are preprocessed since they are temporal. The mean and standard deviation are calculated for all intervals and both measures for all the 24 timber and pitch distinct features are used as the audio features. Consequently, a total of 48 audio features are generated for each song.

Then, we create lyric features. We calculate the perplexity of each song lyric with respect to each of the four genres. With these four scores, we then construct a four dimensional vector for each respective song. In our initial experiments, this strategy proved to be degenerate; that is, it actually made classification much worse than the baseline. We were able to normalize the perplexity scores by calculating Kullback-Leibler divergence for each pair of Bigram distributions. For example, if we wish to calculate the normalized perplexity of *rock*, we first calculate the following:

$$KL(rock||all) = \sum_{genre \neq rock} KL(rock||genre) \quad (1)$$

We then have to calculate the other side:

$$KL(all||rock) = \sum_{genre \neq rock} KL(genre||rock) \quad (2)$$

Finally we use these two sums to normalize the rock perplexity score:

$$NP_{rock} = \frac{P_{rock}}{KL(rock||all) + KL(all||rock)} \quad (3)$$

This is done for each of the perplexity scores. We construct a feature vector by concatenating the audio features with these "normalized" perplexity features.

It's worth noting that KL-divergence is calculated using the classifier dataset (200 points) with respect to each pair of distributions for each point; that is, each lyric file produces the KL values. Since KL is defined over a population, the lyric-files were broken up into their constituent 2-grams. The KL value was then calculated over this finer-grained representation of the file. A production system could do the same, but it might be better to estimate the divergence with some large representative dataset. Exploring the best way to calculate these values is a question worth exploring as follow-on work.

Next, we reduce the dimensionality of the feature vector using the previously described method. Since, previous experiment results showed little changes to the classifier performance. After reducing the dimensionality of our data points, we perform 10-fold cross validation on the classifier dataset. For each fold, 20 data points are used for testing and 180 are used for training the classifiers. Prior to training the classifier, both the training and testing data sets are normalized. All the features are divided by the standard deviation of the training data set. Finally, we train the classifier using the normalized training data and evaluate the classifier performance using the testing dataset.

3.2 Results

In our work, we attempt to use language features to improve upon the performance of a baseline classifier (i.e. the classifier trained only on audio-features):

The confusion matricies are read from top-bottom. For example, in Figure 2 electronic was classified as electronic 101 times; it was misclassified as pop 26 times, as punk 39 times and as rock 31 times. Although we trained several classifiers, we report the performance of the radial kernel SVM

pred	*elec*	*pop*	*punk*	*rock*
elec	101	76	27	51
pop	26	41	15	20
punk	39	31	117	53
rock	31	49	38	73

Figure 2: Audio-only baseline confusion matrix. Columns represent the actual genre and rows the predicted genre respectively.

machine since it is the classifier with the best performance overall.

After training 4 Bigram language models with 250 data points (one for each genre), and constructing feature vectors as described above, our classifier produces the confusion matrix in Figure 3:

pred	*elec*	*pop*	*punk*	*rock*
elec	98	71	24	51
pop	26	42	12	22
punk	40	29	120	49
rock	33	55	41	75

Figure 3: Audio-Bigram 250-lyric confusion matrix. Columns represent the actual genre and rows the predicted genre respectively.

Each of the Bigrams were trained on approximately 250 lyric files from each genre. We present our best result which (as expected) resulted after training the Bigrams with the most amount of data. At the very least, we feel that these results suggest that the addition of language features holds some promise. Although there is small drop in electronic performance, we feel that the construction of electronic music lends itself to more language ambiguity. We will discuss this in more detail in the following sections.

3.3 Language Only Classification

In addition to our primary experiment (combining language features to audio features), we also trained the classifier with language-only features. For the sake of comparison, we will contrast the 50-lyric experiment with the 250-lyric experiment. The 50-lyric language-only confusion matrix is as follows

pred	*elec*	*pop*	*punk*	*rock*
elec	50	31	26	32
pop	23	31	30	26
punk	21	23	20	15
rock	103	112	121	124

Figure 4: Bigram-only 50-lyric confusion matrix. Columns represent the actual genre and rows the predicted genre respectively.

Rock clearly dominates this matrix; that is, for a given song, the probability that it will be classified as rock is relatively high. We were not able to completely investigate why this behavior is seen. However, this "bias" towards rock attenuates by training the Bigrams with more data. The 250-lyric Bigram confusion matrix is shown in Figure 5.

pred	*elec*	*pop*	*punk*	*rock*
elec	38	33	29	19
pop	45	41	35	39
punk	45	33	49	41
rock	69	90	84	98

Figure 5: Bigram-only 250-lyric confusion matrix. Columns represent the actual genre and rows the predicted genre respectively.

The 250-lyric model clearly balances the classifier, but it still does not perform very well. Combining the language model with the audio-features is clearly beneficial. We will discuss possible problems with the language model in the next sections.

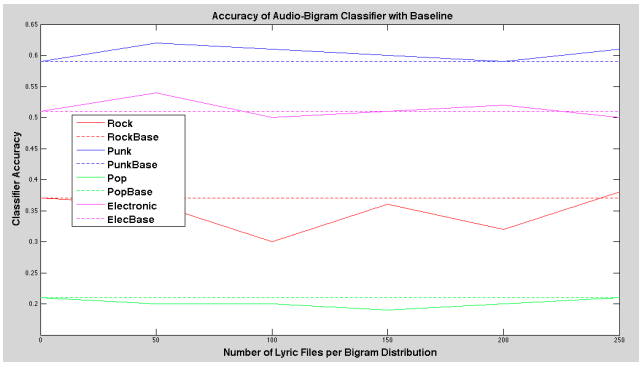

Figure 6: Classifier accuracy when trained with audio and lyric features. The dashed line represents the baseline (performance with audio-only features).

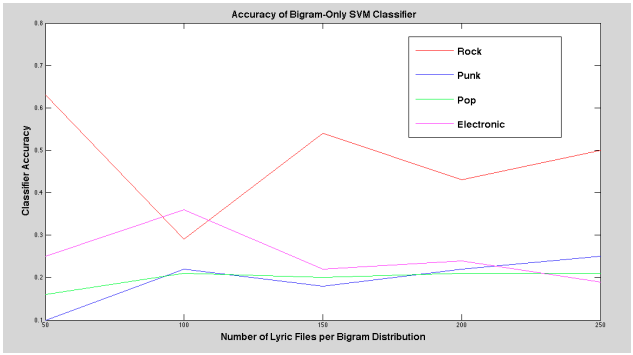

Figure 7: Classifier accuracy when trained with only lyric features.

Figure 7 shows the accuracy scores of our best classifier using only lyric features. As we can see, the discriminatory power of our classifier increases for most genres as training data increases. Hence, our proposed lyric features are in fact capturing some aspects of genres. The decrease in electronic music can be explained by the fact that electronic music revolves around remixes of songs from other genres. Hence

as we increase the training size, we are in reality flattening the distribution for this genre. Rock on the other hand is a very broad genre that encompasses very distinctive subgenres but all with the same core characteristics. If we approximate the distribution of rock songs in the MSD using a small subset of songs (193267 songs) from it, rock and its derivatives cover 13.3% of the space compared to 2.7%, 2.8% and 9.4% for punk, electronic, and pop respectively. Consequently, its distribution may be constantly changing since it is one of the popular genres in the dataset and consequently has higher variability due to its larger population.

For a more comprehensive look at the classifier performance, consider Figure 6. It's worth noting that each successive experiment (50, 100,..etc) uses the lyric data from the previous experiment; that is, the set of 100 lyric files (used to train a 100-lyric Bigram) contain the same files that were used to train the 50-lyric Bigram (plus 50 more). We are simply adding to the language model. As such, we would expect the classifier performance to converge. Note the combined classifier sort of follows the Bigram-only classifier in Figure 7. With more data, we would expect classification accuracy to improve to some limit. There is a hint of an upward trend in the Bigram-only classifier.

3.4 Discussion

By far, the most difficult task was the initial data collection (from the MDS) — specifically the collection of lyrics. Due to music licensing hurdles, and the specific songs we needed lyrics for, we had to manually procure the lyrics. Thus, we were only able to collect approximately 500 lyrics per genre. We feel that a larger dataset would probably reveal the emergent trends.

Additionally, most songs from MDS belonged to multiple genres. Each MDO-song-file contains a special "genre" vector. Each vector with length greater than 1 is necessarily ambiguous. In our collection, we included a song in a genre dataset if the respective genre appeared in the genre vector. Thus it's quite possible, even likely, that a song was included in more than one of our genres. As an extreme example, music by Celine Dion was included in the rock genre. This fact can make classification very problematic for genres that often co-occur with other genres. In future work, it would behoove the researcher to pay close attention to the data collection task and try to quantify the level of ambiguity at an early stage.

The issue with electronic music deserves another mention. We speculate that electronic music is much easier to distinguish when heard than when read (the lyrics). As was previously mentioned, electronic frequently "samples" or "remixes" from other songs. The distinguishing feature of electronic music is the manner in which the samples are assembled together. With respect to lyrics, a Bigram model would probably have trouble capturing this aspect of electronic music. For example, suppose that an electronic song combines a sample in a repeating pattern such as: "The system is down; the system is down; the system is down...". A Bigram model would not be able capture the repeating four-word pattern. In general, the variety of "remixing" strategies make the fitting of a language model problematic in the least.

4. RELATED WORK

Tzanetakis and Cook investigate which audio features are better suited for genre classification. They construct a 30-dimensions feature vector using various features for timber, pitch, and rhythm and evaluate, in a 10-genre classification task, three different kinds of systems: K-Nearest Neighbors (KNN), Gaussian classifier (GS), and Gaussian Mixture Model (GMM) [5]. In addition, they evaluate the systems for classification on 4 and 6 sub-genres of Classical and Jazz respectively. The systems are trained with 100 30-second samples from each class and the K value of the system ranges between one and five. The audio samples were gathered from different sources such as radio recordings, compact disks, and MP3 audio files [5]. The results they present show that each feature class captures different characteristics of the music genre but all systems trained only with one feature class score under 50% accuracy. An improvement in the system's accuracy happens when classifying sub-genres. Nevertheless, they show that timber is by far the most important feature class since a system solely trained using timber features doubles the accuracy of systems trained solely on rhythm or pitch [5]. Finally, they compare the system performance to human classification. The system reached an accuracy of 60% versus human accuracy of 53% (trained with 250 ms samples) and 70% (trained with 3 sec samples). Our project builds on top of their findings as we propose to use features extracted from lyrics to improve system's accuracy.

Also similar to our work, with respect to the use of multimodal features, the work done by Bruni et al. [1] trains separate models (i.e. a language model and an image-feature model) and then concatenates the respective features. The model is compared to state-of-the art DM (Distributional Models) on word-similarity tasks as well as categorization tasks. The authors show that the combined model is at least as performant as the basic text-based model.

For word-similarity experiments, the authors show an approximately 5% improvement over the baseline text-DM.

Obviously, our work differs in the type of modality (audio) we are augmenting with language features.

5. FUTURE WORK

As mentioned in the discussion section, we need to refine the initial data collection task. We found a significant amount of ambiguity in the supervised data itself! A simple, yet effective, modification would be to first run each of the MSD files by a human evaluator (using Mechanical Turk or a similar service) and ask them to assign a "weight" for each genre in the genre-vector. This would allow the researcher to make smarter decisions about how classify each song.

Additionally, electronic songs revealed a weakness in using a language model; the construction of much electronic music necessarily "borrows" from songs in other genres; this, by definition, introduces ambiguity that is difficult to compensate for. It's possible that a discriminative model could compensate for this issue. Perhaps a topic model would be able to "capture" the "distribution" of genre sampling in an electronic song. In general, it would be worth while to experiment with different language models.

One of the more interesting aspects of this project was devising a way to "normalize" the language-features with respect to the Bigram distributions. In the interest of full disclosure, we did not have time to fully investigate what "normalization" should mean in this scenario. A more detailed study (or more informed strategy) of this aspect would undoubtedly benefit the multimodal model.

One of the more frustrating aspects of the project pertained to understanding exactly how increased Bigram training affected the performance of the classifier. We were able to experimentally show that sufficient training of the language model seems to add a little to the classifier. It would be worth while to investigate exactly what (if there is any) connection Bigram training has to the performance of the classifier.

6. CONCLUSION

Our results show modest improvements in classification accuracy. However, this is not because lyrics do not capture useful information. Figure 7 shows that lyrics are in fact capturing characteristics of genres. While the accuracy may not be ideal, other studies have shown that similar evaluations only using a specific group of features reach 29% accuracy [2]. Moreover in order to achieve such results, we need to determine how to capture how different genre distributions relate to each other. For example, if a song exists in overlapping tails of two or more genres then its a very hard to classify that point in contrast to a point that just exists in the tail of one. Hence, using only simple features such as probability of membership to each genre is not sufficient.

We have shown that using simple NLP techniques for lyric feature extractions have a noticeable impact on the accuracy in the genre classification task. Hence, it is promising to explore more sophisticated techniques in order to extract better features and thus further improve classifier's accuracy. In essence, this project serves as a stepping stone into bringing together the signal processing and NLP communities in order to jointly solve the complex task of automatic genre classification.

7. REFERENCES

[1] M. B. Elia Bruni, Giang Binh Tran. Distributional semantics from text and images. *GEMS*, pages 22–32.

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

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

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