Automatic Lyrics-Based Music Genre Classification

Ruoxi Zhang
Department of Music
Stanford University
Palo Alto, CA 94304
ruoxi17@stanford.edu

Kezhen Zhao
Department of CEE
Stanford University
Palo Alto, CA 94306
zkz@stanford.edu

Peiling Lu
Department of Music
Stanford University
Palo Alto, CA 94304
peilingl@stanford.edu

Abstract

Our project is to build neural network models to classify the genre of music pieces by their lyrics. Basically, we are going to use different natural language process (NLP) methods to generate context analysis with respect to lyrics, so as to tag them with proper genre labels. First, we tried to implement traditional machine learning algorithms like SVM and Random Forest as baseline with small amount of data. Then with increasing data amount, we proposed several recurrent neural networks: LSTM, GRU and attention neural networks, comparing their performances for classifying 15 genres of music based on lyrics.

1 Introduction

With technologies in Natural Language Processing advanced fast, people are not satisfied with just approaching tasks in plain text, but explore more applications in entertainment field to better meet the need for spiritual level. Music recommendation system is an emerging research to help people find out the music they have preference for. It gives users personal advice based on their tastes. This recommendation system can figure out users' personal taste by exploiting the songs they already listened to. In early periods, many music database system are indexed by song title and artist name. This simple index method would definitely bring about incorrect results. More efficient methods make use of audio features like timbre and pitch contour. But it cannot provide distinct boundaries between different genres, especially when there is a trending in combining several different genres into one composition. Although some great achievements have been made when using audio features, we still need to add other music features to enhance performance. Lyrics, as another important component of music, has bee taken into consider. However, most research focused more on using traditional machine learning algorithms such as SVM and clustering since they don't have enough data. However, with increasing data amount and algorithm advanced, deep learning can be applied when automatically classify lyrics data. Thus, in this paper, we implemented four neural network models from simple sequence model to models with attention layers and compared the performance each made.

In section 3, we described our approached in detail, including data preprocessing, baseline model, and the essence of neural network models. In section 4, we provided datasets we use, model configurations, evaluation method and results. Finally, we made a conclusion about our discovery, and pointed out some future work we suggested to further examined.

2 Background

Natural Language Processing now is an essential part in artificial intelligence. It is aimed at helping computer "understand" natural languages so that we can do some useful tasks like language translation, keyword searching, etc. Researches of Natural Language Processing has been did for a long time since 1940. At that time, machine translation was the first application field related to natural language processing. However, since technology was developing and the essence of language remained to investigate more, researchers simply concluded that it is the vocabulary and order of words that determine the meaning of languages. But with researches got deeper, syntactic structure has been brought to consider. During that time, other applications like speech recognition has appeared.

Nowadays, natural language processing has developed to a systematic discipline. Many technologies has emerged to help deal with hard tasks. For language processing, we have developed technologies like WordVec, Part of Speech and Bag of Words. WordVec is a word embedding technology to map words or phrases from the vocabulary to vectors of real numbers, which preserves both syntactic and semantic information. Part of Speech clusters words with similar grammar structure. And Bag of Words model is to predict word from sum of surrounding word vectors.

Also, with technology advanced and the amount of data increasing, deep learning has come to the stage. Recurrent neural network is a kind of neural network mainly to deal with sequence prediction or classification problem. It can use its internal state to process sequence input, which is also the format of languages. Long short-term memory (LSTM) are units of RNN. That is, RNN constructed by units of LSTM is often called LSTM networks. Unlike traditional neural networks, it can help memorize through time period like human reading.

Many researches has been did to help advance technologies. In ”Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank” [1], Richard et al. proposed a sentiment treebank for predicting the sentiment of movie reviews using natural language processing technology. In ”Reading Wikipedia to Answer Open-Domain Questions” [2], Danqi et al. combines a search component based on bigram hashing and TF-IDF matching with a multi-layer recurrent neural network model trained to detect answers in Wikipedia paragraphs.

Music genre classification is a significant part for music recommendation system. Music features for classifying genres are mainly lying in lyrics and audio signal. Many researches focused more on audio features, since different genres of music appears to have distinct representations in audio part. For example, George and Perry [3] proposed an automatic music genre classification system by using characteristic like instrumentation, rhythmic structure, and harmonic content of the music. The accuracy achieved 61% for 10 genres, which is comparable with manually labeling. In ”Music Genre Classification with the Million Song Dataset” , Liang, Haijie and Brendan [4] combines both audio and lyrics features to classify music genres by developing a cross-modal retrieval framework. Specifically, they implement Hidden Markov Model for audio features and super linear model SVM for lyrics part. Finally, they get 40% accuracy by only using lyrics data. And they just work best for mental and hip-hop.

3 Approach

3.1 Data Preprocessing

Raw data we collected is just raw text files with lyrics. It contains some special characters like Latin characters, punctuations and numbers, which can cause incorrect results. There are several data preprocessing steps we used that are common in natural language preprocessing [5].

- Convert text to lower case: convert all words into lower case in case that two same words are regarded as different.
- Remove numbers: numbers cannot provide language benefits in analysis.
- Remove punctuations: Although punctuations can give some grammar structure information, it gets removed since it does not add value.
- Strip white spaces: Remove extra white spaces.

3.2 Baseline

We use the bags-of-words model and traditional machine learning approaches for our baseline model. We choose Support Vector Machine (SVM) as our classifier. Since the songs are represented as high-dimensional sparse vectors, linear kernel is suitable for our model. We choose the scikit-learn library [6] as our tool to implement our model. On top of raw term (word) frequency, we run the same process on the tf-idf processed data, since certain terms have little or no discriminating ability in determining genres [7].

However, SVM becomes slow when the data amount increase. And SVM is more often used to two-class problem. To deal with these problem, we then try to implement Random Forest model, because it is capable of making use of a large corpus. Moreover, it is intrinsically suited for multiclass problems, which fits well in our data.

3.3 Neural Network Model

3.3.1 Word Embedding

The embedding method we used in our project was Glove [8], which is an unsupervised learning algorithm for obtaining vector representations for words. Input data of GloVe was a concatenation of all lyrics. We firstly pre-processed lyrics into tokens(with all special characters, punctuation and extra whitespace removed), and concatenated tokens with single whitespace as our Corpus. The embedding size was set at 50, and the output of Glove was words with vectors, which were in the format:

$$word \quad vector \rightarrow I, -0.270010 \quad 0.009907 \quad -1.071562 \ldots,$$

with each word have a vector length of 50.

The data fed in our model was generated corresponding to the output of Glove. For each song, we had a list with separated words in order, and each word was represented by its index.

3.3.2 Simple Sequence Model

Since there are limitations for the dataset of sequential lyrics, we firstly build two basic sequential model using different types of complex activation units, i.e., Gated Recurrent Units (GRUs) and Long-Short-Term-Memories (LSTM), as reference.

The GRU (Cho et al., 2014 [9]) represents its hidden state h_t as:

$$\begin{aligned} h_t &= (1 - z_t) \circ \tilde{h}_t + z_t \circ \tilde{h}_{t-1} && \text{(Hidden state)} \\ \tilde{h}_t &= \tanh(r_t \circ U h_{t-1} + W x_t) && \text{(New memory)} \\ z_t &= \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}) && \text{(Update gate)} \\ r_t &= \sigma(W^{(r)} x_t + U^{(r)} h_{t-1}) && \text{(Reset gate)} \end{aligned}$$

where $\tilde{h}_t$ is the new memory, a consolidation of current input and previous hidden state; z_t is the update gate, conditioning on the past hidden state; r_t is the reset gate, determining how the previous state is important to the new memory; and $W, U, W^{(z)}, W^{(r)}, U^{(z)}$, and $U^{(r)}$ are the trainable parameters.

The LSTM unit represents its hidden state h_t as:

$$\begin{aligned} h_t &= o_t \circ \tanh(c_t) && \\ \tilde{c}_t &= \tanh(W^{(c)} x_t + U^{(c)} h_{t-1}) && \text{(New memory cell)} \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t && \text{(Final memory cell)} \\ i_t &= \sigma(W^{(i)} x_t + U^{(i)} h_{t-1}) && \text{(Input gate)} \\ f_t &= \sigma(W^{(f)} x_t + U^{(f)} h_{t-1}) && \text{(Forget gate)} \\ o_t &= \sigma(W^{(o)} x_t + U^{(o)} h_{t-1}) && \text{(Output gate)} \end{aligned}$$

where the new memory cell is the consolidation of current input and past state; the input gate determines whether the new input x_t is worth preserving; the forget gate conditions on previous memory cell; the memory cell combines new memory and past memory; the output gate determines what information is necessary for the hidden state.

3.3.3 Attention Networks

When determining the genre of a song by its lyrics, not all words are equally important when contributing to the classification. An attention mechanism is proposed by Bahdanau et al. [10], allowing the model to learn which words are more important than others in machine translation. We apply this idea in genre classification using lyrics to learn which words are more important to determine the music genres.

The architecture of our neural network consists of sequence encoder, attention layer, and a classification softmax layer as illustrated in Fig. 1.

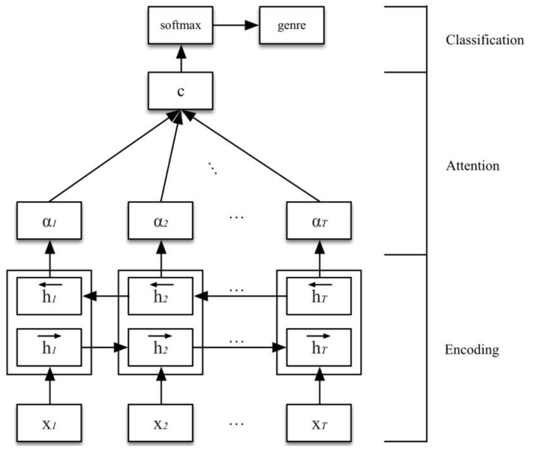


Figure 1: Attention Network Architecture

Sequence Encoder

We build different models by using GRU-based and LSTM-based encoders. Given a song consists of a sequence of T words, we represents each word as $x_i, i \in [1, T]$ in the form of a embedded word vector.

For the GRU-based encoder, we applies a bidirectional GRU to represent the hidden state h_i:

$$\overrightarrow{h}_i = \overrightarrow{GRU}(x_i) \qquad \text{(Forward hidden state)}$$
$$\overleftarrow{h}_i = \overleftarrow{GRU}(x_i) \qquad \text{(Backward hidden state)}$$
$$h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i] \qquad \text{(Combined hidden state)}$$

where $\overrightarrow{GRU}$ reads a song in the normal order, $\overleftarrow{GRU}$ in the reverse order, and $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$ summarizes the information of the sentence around word x_i.

Similarly, the bidirectional LSTM encodes the sequence as:

$$\overrightarrow{h}_i = \overrightarrow{LSTM}(x_i) \qquad \text{(Forward hidden state)}$$
$$\overleftarrow{h}_i = \overleftarrow{LSTM}(x_i) \qquad \text{(Backward hidden state)}$$
$$h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i] \qquad \text{(Combined hidden state)}$$

Word Level Attention

By adding the attention mechanism, the model is able to learn which words are more important to represent the meaning of a song. For each hidden state h_i, the attention mechanism can be represented as:

$$u_i = \tanh(W_a h_i + b_a) \qquad \text{(Representation of hidden state)}$$

$$\alpha_i = \frac{\exp(u_i^T u_a)}{\sum_{k=1}^{n} \exp(u_k^T u_a)} \qquad \text{(Attention vector)}$$

$$c_i = \sum_{j=1}^{n} \alpha_{i,j} h_j \qquad \text{(Context vector)}$$

where the term u_i is a hidden representation of hidden state h_i; $u_i^T u_a$ gives a score indicating the importance of the word, and α_i is the normalized importance weight; the context vector c_i is the weighted average of the hidden layer, capturing the relevant high-level contextual information of the song; and W_a, b_a, and u_a are trainable parameters.

Classification

With the output of the attention layer, we add a softmax layer to obtain the probability distribution of the genre:

$$p = softmax(W_p c + b_p)$$

where W_p and b_p are trainable parameters. We choose the genre with the highest probability as the final label. Since it is classification problem, we choose cross-entropy as loss function.

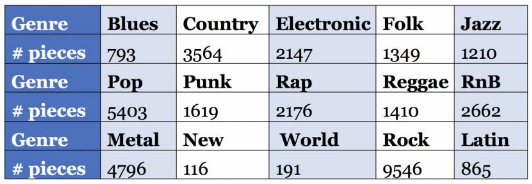
4 Experiments

4.1 Datasets

For baseline model, the dataset we used was The musiXmatch Dataset(MXM [11]), which is the official lyrics collection of the Million Song Dataset (MSD [12]). The MSD is a collection of audio features and meta-data for a million contemporary popular music tracks, which include titles, artists, tags, genre labels, lyrics and other features. Among which we mainly use the a subset of lyric dataset given by musiXmatch as our project dataset, which contains music track ID (identity for each music piece), lyrics, and the tagged genre labels.

The lyrics of MXM are given in the form of bags-of-words of 5000 most frequent words. For our milestone, we take a subset of 10661 songs in 15 genres from the dataset for SVM model and 73659 songs in 15 genres for Random Forest Model.

For Neural Network models, we extracted lyrics with the Developer API given by MusiXmatch Developer[13] to crawl lyrics dataset. We used the identification $trackID$ from MXM as our linked key to get access to lyrics. Each data record included a unique $trackID$, a unique $lyricsTrackID$, which was the 'fingerprint' parameter we used to extract lyrics, a label('genre') and a string value included 30% length of lyrics(due to the copyright, we could extract at most 30% sequential lyrics of a song). In our model, we used altogether 37847 songs in 15 genres as our dataset. The number of songs in each genre is shown in Figure 2.



Genre	Blues	Country	Electronic	Folk	Jazz
# pieces	793	3564	2147	1349	1210
Genre	Pop	Punk	Rap	Reggae	RnB
# pieces	5403	1619	2176	1410	2662
Genre	Metal	New	World	Rock	Latin
# pieces	4796	116	191	9546	865

Figure 2: Dataset Genre Distribution

4.2 Model Configuration

Table 4.2 shows the basic model configuration for the two simple layer sequential model.

Parameter	**Value**
Learning rate	0.05
Epochs	30
Batch size	64
Hidden layer size	72

Table 1: Model Configuration for Simple Sequential Model

Parameter	**Value**
Learning rate	0.9
Epochs	20
Batch size	64
Hidden layer size	72
Attention size	32

Table 2: Model Configuration for Attention Networks

	Positive	**Negative**
Positive	True Positive	False Positive
Negative	False Negative	True Negative

Table 3: Confusion matrix for binary classification problem

When training the model, we separate the training dataset in the ratio of 5:1 for the training data and validation data. We set an early stopping layer monitoring the validation accuracy with patience of 10 to avoid the problem of overfitting. Additionally, we add a dropout layer with rate of 0.2 to avoid overfitting.

Since the data is imbalanced, we set the class weight inversely proportional to the number of data points in a class, which adds penalties when incorrectly classifying the minor classes as major class, to avoid classifying all songs as the major class.

Table 2 shows the configuration of attention networks. Similarly, class weight and dropout are applied to attention network.

Training the simple sequential model takes about 35 minutes, while the attention network takes 40 to 60 minutes.

We use sparse categorical cross entropy as our loss function, since it's a multi-genre classification problem and our output labels are integers instead of one-hot vectors. we calculate a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

Where M is the number of classes, y is the binary indicator (0 or 1) if class label c is the correct classification for observation o, and p is the predicted probability observation o is of class c.

4.3 Evaluation Metric

We used accuracy and confusion matrix as our quantitative evaluation methods. Accuracy is the percentage of correct predictions. But it is not always a perfect way to evaluate methods, since it does not consider the number of classes. With imbalanced data, predicting that every instance belongs to the majority class would give us high accuracy.

$$\text{accuracy} := \frac{\text{correct classifications}}{\text{number of classifications}}$$

However, confusion matrix is a clean and unambiguous way to present the prediction results of a classifier. It can show that what kinds of error your classifier is making. For a binary classification problem, we can get confusion matrix as table 3:

For multi-class classification problem, we can just extend the rows and columns in the table, each item represent the corresponding class, and the content in each cell indicates the number of samples falls into that situation. All correct predictions are located in the diagonal of the table, so it is easy to visually inspect the table for prediction errors, as they will be represented by values outside the diagonal.

	GRU	LSTM	GRU ATT	LSTM ATT
Acc without class weight	0.4374	0.4381	0.4207	0.4155
Acc with class weight	0.3037	0.3164	0.2809	0.3038

Table 4: Confusion matrix for binary classification problem

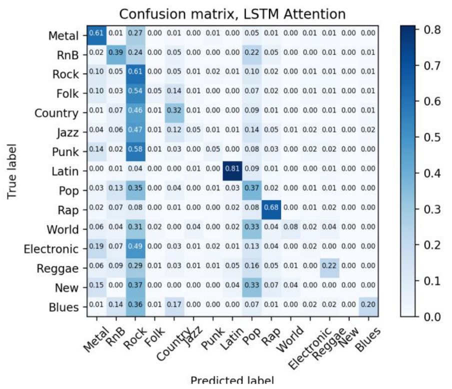


Figure 3: Confusion Matrix of LSTM ATT, without Class Weight

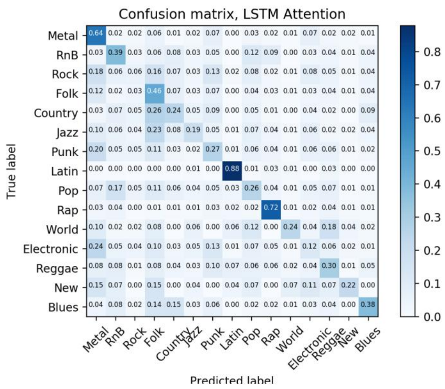


Figure 4: Confusion Matrix of LSTM ATT, with Class Weight

For qualitative evaluation, we manually test some lyrics examples by checking if the lyrics' genres we labeled ourselves are consistent with the predicted labels.

4.4 Results

Table 4 shows the accuracy (on test data) of 4 models.

Firstly, we run the models without the consideration of class weight. The accuracy is as high as 0.4. However, we observe the fact that many songs are incorrectly classified as rock song by observing the normalized confusion matrix (Fig. 3), the reason of which is that the data is imbalanced, where the majority of them are of rock genre. We use normalized confusion matrix because data are not uniformly distributed in the data set.

To solve this problem, we add class weight inversely proportional to the number of data points in a class. The confusion matrix shows better diagonal line (Fig. 4) at the cost of a drop of accuracy of around 0.1. The major reason for this drop is because the class weight of rock songs is very small compared to others, making it less costly when mistakenly classifying rock songs as other genres when a large proportion of the dataset is of rock genre.

However, the confusion matrix is more self-explanatory. It can be seen that genre Latin is the most outstandingly classified class because the language difference (Spanish instead of English) in lyrics. Rap is another class that is successfully classified because the special vocabulary (e.g., *yo*) and consecutive or nearby rhyming words (e.g., *sittin, gettin, thinkin*) in rap music.

The incorrect classifications are also illustrative. As shown in the matrix, the group of metal, rock, and punk songs are confused, which is acceptable since these genres have some similarity musically. Similarly, pop, RnB, and Blues songs are confused more than other genres of music.

5 Conclusion

Data preprocessing is a simple but critical part in our task. To make data more clean and reasonable, we still need to remove stop words, remove common word endings like "ed", "es" and "s". Also, comparing with removing stop words, we can also consider remove the infrequent words to avoid the sparsity problem.

The results can vary because of the data distribution, models' structures, evaluation methods, etc. Imbalanced data would persuasive network to classify more data to the majority class. Applying class weights, decrease the number of majority class data, or increase the amount of minority class data can help get a more uniform data distribution. LSTM network works the best in our results since our kinds of genres are not that much. But in "Music Genre Classification by Lyrics using a Hierarchical Attention Network", Alexandros [13] adapted the hierarchical attention network for the same task, with more genres networks with attention layers works the best. Moreover, by evaluating the results using different methods, we find out that accuracy is not enough in some situations, but helps us improve processing work and model structure. With confusion matrix, we can relatively trust our results in belief. In fact, it is reasonable when we manually examine our data.

Future work can be taken in processing the imbalanced data problems, for example, we can collect more data if possible. A larger dataset may have a more uniform data distribution. Also, we can change or add more evaluation metric like F1 score and precision. Moreover, applying different weights to classes can also be an efficient way to distribute importance of classes. Class weights parameterization would be a convenient way to easily adjust the arbitrary values of weights. Besides, we can also use the method SMOTE [15] given by Chawla et al, which is basically an approach to generate synthetic samples is to randomly sample the attributes from instances in the minority class.

References

[1] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

[2] Chen, Danqi, et al. "Reading wikipedia to answer open-domain questions." arXiv preprint arXiv:1704.00051 (2017).

[3] Tzanetakis, George, and Perry Cook. "Musical genre classification of audio signals." IEEE Transactions on speech and audio processing 10.5 (2002): 293-302.

[4] Liang, Dawen, Haijie Gu, and Brendan OConnor. "Music genre classification with the million song dataset." Machine Learning Department, CMU (2011).

[5] https://datascience.stackexchange.com/questions/11402/preprocessing-text-before-use-rnn/11421

[6] Scikit-learn - Machine Learning in Python, http://scikit-learn.org/

[7] Introduction to Information Retrieval - Inverse Document Frequency, https://nlp.stanford.edu/IR-book/html/htmledition/inverse-document-frequency-1.html

[8] GloVe, Jeffrey Pennington, Richard Socher, Christopher D. Manninghttps://nlp.stanford.edu/projects/glove/

[9] Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation"

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[11] MusiXmatch, https://forum.musixmatch.com/

[12] Million Song Dataset, https://labrosa.ee.columbia.edu/millionsong/

[13] MusiXmatch Developer https://developer.musixmatch.com/

[14] Tsaptsinos, Alexandros. "Music Genre Classification by Lyrics using a Hierarchical Attention Network."

[15] Nitesh V. Chawla, Kevin W. Bowyer, Kevin W. Bowyer, Kevin W. Bowyer. SMOTE: Synthetic Minority Over-sampling Technique. https://www.jair.org/media/953/live-953-2037-jair.pdf