Team: Zachariah Zhang (zz1409), Lingshan Gao (lg2755), Binqian Zeng(bz866)
DS-GA 1003
Advisor: Vitaly Kuznetsov

Term Project Proposal

Abstract:

Music tagging is the problem of assigning a set of tags to a song based on its content. This is an important problem for music services for applications such as search or recommendation. In this paper we explore different models for music tagging using both conventional and deep learning approach. We frame the problem as a multilabel classification problem. We found that deep learning is able to outperform conventional methods as it is able to create a rich representation of complicated language.

1 Introduction:

In this project, we will focus on applying Machine Learning techniques, particularly NLP-related techniques, to label song genres using lyrics. Traditionally, songs are labelled by the production companies when they are first released. Alternatively, people may also assign certain genre tags to a song based on its melody, rhythm and harmony. Besides the sound component of a song, we are curious if the text component, such as lyrics, artist and composer, will tell us anything about its genre. The objective of the project is to create a model that is able to assist song classification and to tag them automatically based on their lyrics.

The product of this project can be applied to music recommendation systems. Usually a song is tagged with multiple genre labels, when a user is listening to a multi-labelled song, we can recommend other songs that share the same labels to the user. Traditionally, music recommendation systems use collaborative filtering. Our model gives the option of recommending songs that are rarely listened by the crowd but potentially match the taste of the user. Another use of our model is that it may help music platforms, such as Google Music or iTunes, to tag the genres more accurately. It is commonly seen that production companies have a tendency to add as many genre tags to a song as they can to attract more audience. Our model automatically and objectively labels songs with the genre that they belong to. It will not only save manpower from manually labelling each song, but also generate objective tagging results without any incentives to abuse genre tags.

Throughout the rest of the paper we will:

- Give a more detailed description of the data and problem
- Show our methodology and results using a shallow model
- Show improvements that can be made using Deep Learning

Description of the Data

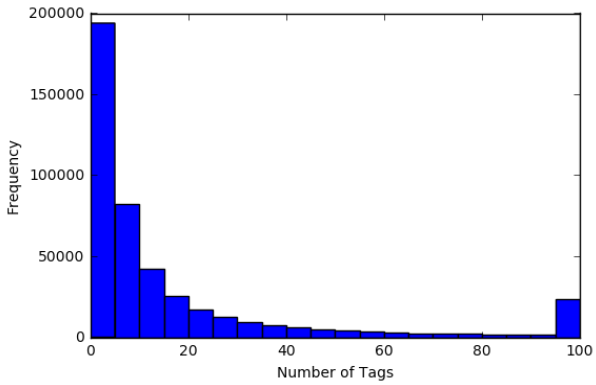
Overview of the data

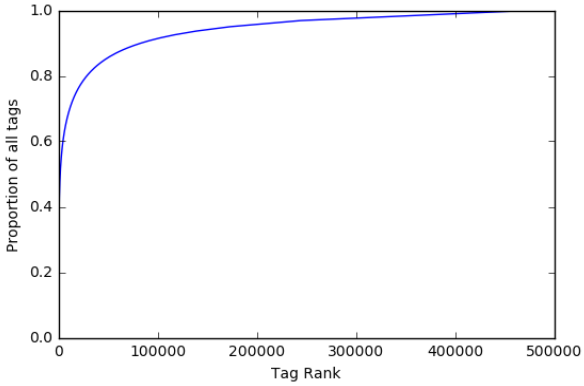
We created a lyrics data set from the Last FM dataset. The dataset includes information such as artists, song names, tags, timestamp and IDs. Target variables are tags. This dataset contains metadata for 943,347 songs, from which 505,216 songs are labeled with one or more genre tags. We used the PyLrics API to collect lyrics for around 190,000 of these songs. We have included an example below.

Song	Jimi Hendrix - Hear My Train A Comin'
Lyrics	Well, I wait around the train station Waitin' for that train Waitin' for the train, yeah Take me home, yeah From this lonesome place Well, now a while lotta people put me down a lotta changes My girl had called me a disgrace...
Tags	Rock, Blues, Psychedelic Rock, '60s', Guitar...

Tags:

The dataset contains 475,080 unique tags. Many of these tags occur with very low frequency. For our experiments we focus on the top 100 tags. These cover 44.1% of all tags in the dataset. Each song has 16.71 tags on average.





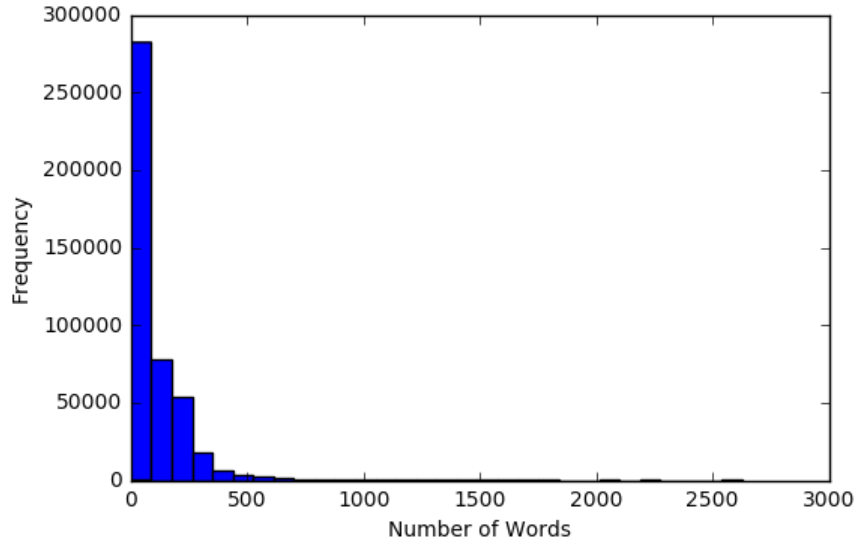
Here are some of the most frequent labels and their associated frequencies.

8 Most Frequent Tags

Tag	Frequency
rock	89052
pop	61020
alternative	49398
indie	42174
electronic	38535
female vocalists	37548
favorites	35450
love	31029

Lyrics:

Each song contains 80.47 words on average. Another consideration for this problem is how many words we will use as features. By ignoring low frequency words, we can greatly reduce our features space. Using the 20,000 most frequent words, we are able to represent 98% of all the words in the dataset. We also remove stop words as they have very little relevance for classification.



Multi-Label Classification

Tagging can be viewed as a multi-label classification problem, as each song can have multiple tags. These problems are typically handled in one of the two ways.

1. **Problem Transformation**- These methods try and change our data into fit within standard classification methods. For example, we can use the binary relevance method, where we train a binary classification for each label.
2. **Algorithm Adaptation-** This is the approach of adapting algorithms for multi-label classification. An alternative adaptation is having the model predict a binary vector across labels and modify the cost function for BCE.

In our experiment we use models of both varieties can compare accuracy.

Evaluation Metrics

We experimented with several different evaluation metrics

Hamming Loss: This is the proportion of misclassified labels for each prediction. We then average across all predictions.

F-score: F score is a metric that factors precision and recall into a single number.

AUC: Area under the receiver operator curve.

Hamming loss gives a very interpretable measure of how well our model is doing but it can be misleading due to the high number and sparsity of labels (always predicting no label will have a low hamming loss). Furthermore, we found AUC to be misleading because of the sparsity of labels as well as the imbalance of frequencies of classes. We decided to use F-score to get a scale free measure of our prediction. F score balances precision and recall into a single score.

Features

We tested several different strategies for representing text.

Bag of Words

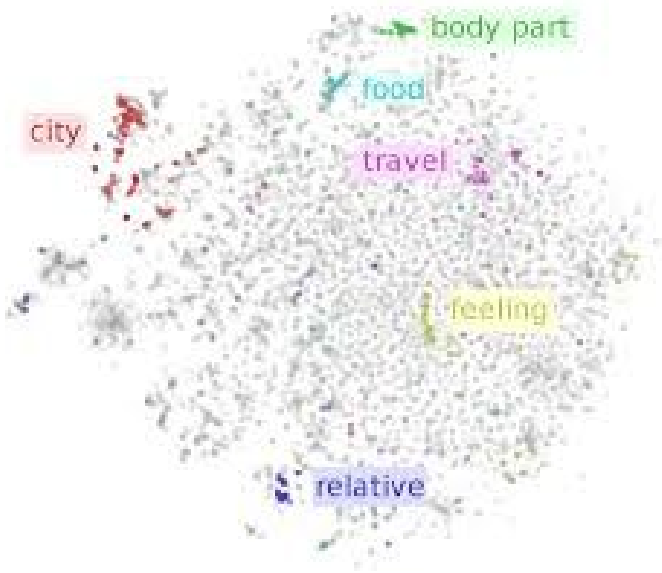
A bag of words representation of a document has a feature for each unique word in the document. The value of this feature is the number of times the word occurs. We experimented using TF-IDF features as well. TF-IDF normalizes the counts by the number of times the word is seen in the whole corpus. The intuition being infrequent words contain more information than common words. These representations can be extended with n-gram phrases.

However, we found that due to the highly repetitive nature of songs (chorus / hook repeated multiple times), it is best to ignore the count information and us binary features for the presence of a word or n-gram.

Word Embedding

Bag of words has several shortcomings. Firstly, they ignore ordering in the words. This can be very important for capturing the complexity of natural language. Another weakness is they are unable to model the semantic relationship between words. For example, we would like ‘burrito’ and ‘taco’ to be treated similarly by the model.

To address these problems we use a different features representation called word embeddings. Word embeddings map each word to a continuous vector such that words that appear in the same context are close to each other. We initialize our model it a 200 dimensional GloVe word embedding that has been trained on Google News. We the represent a document as a sequence of vectors.



Model

Baseline

We first establish a baseline using conventional multi-label classification techniques. We use the binary relevance method of multilabel classification. For each label we fit a one vs rest classifier. Each model uses 20,000 features corresponding to the most frequent n-grams in the lyrics. We limit n-gram length to 3. We trained naive bayes, logistic regression, and SVM models. Models were trained on 80% of the data and tested on the remaining 20%. We performed a sweep of hyperparameters to select the best result for each model. We also experimented with TF-IDF features but found it to be less robust.

Bag of Words Features

Model	Naive Bayes	Logistic Regression	SVM
F-score	0.27	**0.303**	0.218

TF-IDF Features

Model	Naive Bayes	Logistic Regression	SVM
F-score	0.263	0.250	0.264

MultiLabel SVM

We also experimented with reformulating the objective for multi-label prediction. Consider a song x_i with labelset Y_i .We define a multi-label hinge loss by:

$$L(x_i, Y_i) = max(0, argmax_{y' \in Y_i} 1 + < w, \Psi(x, y') > - argmin_{y' \in Y_i} < w, \Psi(x, y') >)$$

Where $\Psi(x, y)$ is a class sensitive feature map. The loss can be interpreted more intuitively as trying to maximize the margin between the least likely true label and most likely false label. In our experiment $\Psi(x, y)$ maps to word-label co occurrence features of the form “word a and label b occur”. We use gradient descent to minimize L with respect to weights, w . We found that this performs similarly to the binary relevance approach with a testing F-score of **.207** .

Deep Learning

In order to address the shortcomings of the binary relevance model, we implemented several different deep learning techniques. In the following sections we describe the two architectures that we tested. We will then present of experimental results.

Deep learning models present several advantages over our binary relevance model.

Word Embeddings- Word embeddings are vector representations of words. By modeling words as a vector we are able to capture the semantic similarity of different words. For example would want the words “king” and “queen” to be represented as being similar words. We initialize the model with 200 dimensional GloVe vectors pretrained on the Google news dataset. We Allow the model to fine tune these embeddings for song tagging during training.

Word Order- Our baseline model disregards word ordering beyond trigrams. Using a deep model we can effectively model longer term dependences without the issue of sparsity in high order n-grams.

Scalability- The binary relevance model requires us to fit a new model for each label. Wh

Convolutional Neural Network

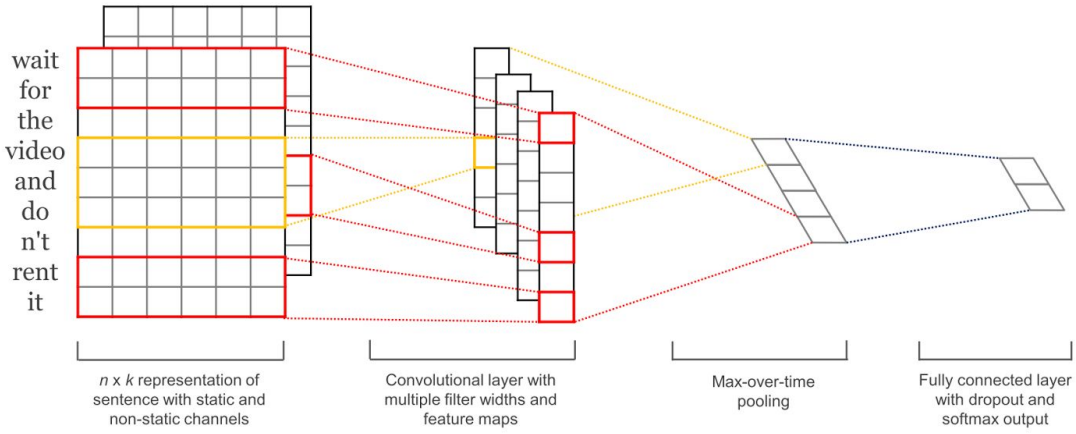
Convolutional networks have been extremely successful in computer vision and base on this they have been adapted for text classification in Kim(2014). Following this model, we represent each song as a matrix $X \in R^{T \times N}$, where T is the length in words of the song and N is the

dimension of the word embedding. We either truncate or pad songs so they all have the same length. We then compute a hidden representation for each layer (L) using the following formula.

$$c^{(l+1)}_{\ i} = f(W^{(l)}x^{(l)}_{\ i:i+h-1} + b^{(l)})$$

Where f is a nonlinear activation function. In our experiments we use rectified linear units. We alternate these layers with pooling layers. Our final hidden activation is then fed to a logistic layer to predict the labels.

$$y = \sigma(W^{(L)}c^{(L)} + b^{(L)})$$



Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a natural choice for modeling temporal data. RNN's maintain an internal hidden state that is updated based on the input at each timestep. We implemented a variation of recurrent network that uses Gated Recurrent Units. GRU's have been shown to be more stable and better at modeling long range dependencies than the standard RNN. GRU's use an update gate and a forget gate to control whether or not a unit is updated. Given an input x_t and hidden state h_{t-1} we compute the reset gate, r, and update gate, z.

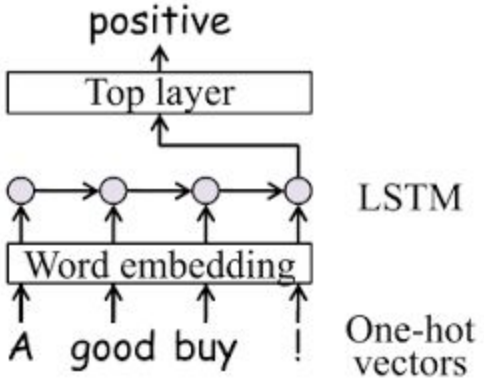
$$z = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

We then use the reset gate to control how much of the previous state we should carry over to the new candidate state, which we define as

$$h^* = tanh(W x_t + r \cdot U h_{t-1} + b)$$
$$h_t = z_t \cdot h_{t-1} + (1 - z) \cdot h^*$$

Similarly to the CNN, we use the final hidden state to perform classification.

$$y = \sigma(W^{(L)} h_T + b^{(L)})$$



Results

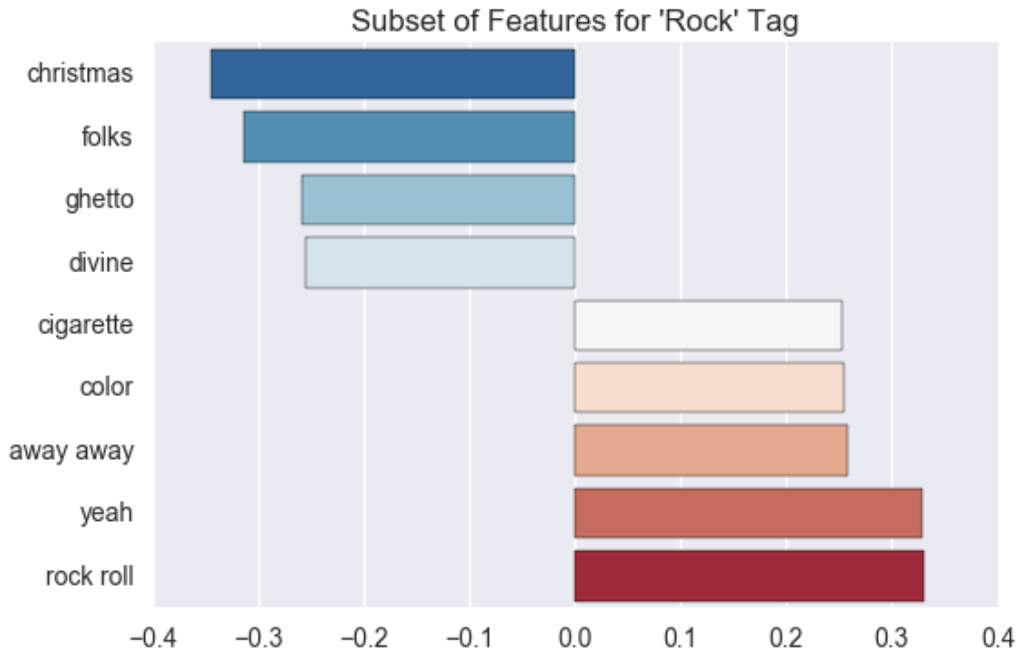
We trained both models using a vocabulary size of 20,000 to predict the top 100 tags. Both models use a 200 dimension GloVe word embedding that has been pretrained on Google news. We use the CNN architecture from Kim(2014). This contains 3 convolutional layers with kernel sizes of 3,4, and 5 and 100 feature maps each. We also use a GRU with 512 hidden units. We use Adam optimizer with learning rate of .001 to fit models. We have for the GRU to be a more effective model for this task.

Model	CNN	GRU
F-score	.413	**.458**

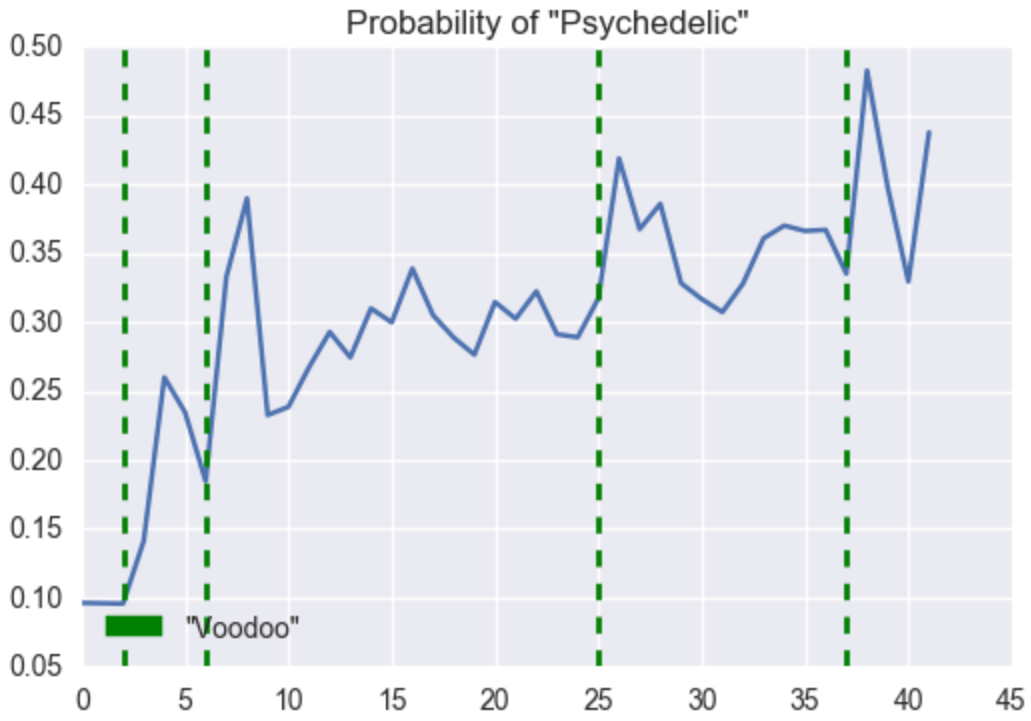
Analysis

We analyzed our models to better understand our model's predictions. We conducted experiment with logistic regression as well as with our GRU.

Logistic regression is a linear model which makes its behavior very easy to interpret. For each label, a parameter having a positive value indicates more confidence in the presence of a label and a negative value indicates more confidence in its absence. All features are binary so we assume the size of the weight roughly corresponds to its importance. Below we have shown weights for a subset of features for the ‘Rock’ tag classifier.



GRU’s don’t have the same natural interpretability as a linear model. However, there are various methods we can use to gain insight into its predictions. One method is to analyze how the probability of different classes chance as new words are observed. Below we show an example of this for the song “Voodoo Child” by Jimi Hendrix. The “Psychedelic” tag is relatively infrequent so the default prediction is very low. We noticed several significant spikes in the probability over the course of the song. We found that each of these spikes follows the word ‘voodoo’. This illustrates one relationship that the model has learned between a word and a label.



Conclusion

In conclusion, we have constructed an automatic music tagging system that is able to successfully predict tags for new songs based on song lyrics. We have compared several different approaches and found deep learning to be the most effective method.