Mood Classification of Lyrics with Deep Learning

Devashree Mehta
RUID 181003322

***Abstract*—The goal is to train a classifier to separate songs into happy or sad mood. Emotion has always been treated as a gray area for computers and work to delve into discovering exactly how far can AI's understanding mimic that of humans is forever on. We have used Google's pretrained Word2Vec model and Long Short Term Memory Networks to define our own model for the problem stated above. This application has multifold uses especially for online streaming services like Spotify to aid autocreation of playlists of songs sorted by mood.**

I. Introduction

The aim of this project is to classify songs as 'happy' or 'sad' using only lyrics as features. We used pySpark for data preprocessing and PyTorch was used for the final model implementation. The data was obtained from Sebastian Rashcka's blog [1] where he used Naive Bayes to solve a similar problem. We used Google's pretrained Word2Vec model [2] as the initialization vector for embeddings. LSTMs were used to maintain sequence characteristic data within the model, and finally a LogSoftmax layer was applied to a Linear layer to find log likelihoods of the two labels.

II. Data and Preprocessing

The dataset is around 1000 tuples where each tuple comprises of Title, Artist, Lyrics, Genre, Mood, Year. We only focus on Lyrics and Mood. Mood takes two values, 'happy' and 'sad'. The distribution of songs belonging to both classes is more or less equal. Since no other source of data could be obtained with both lyrics and mood labels, a data augmentation technique was attempted where new tuples for class X were generated by randomly sampling from amongst top 50 words in other tuples of class X. However, since the final model chosen for classification was LSTM where sequence of input is also treated as a feature, this technique was abandoned, and we had to thus limit ourselves to the available data. The lyrics for each song were preprocessed by first removing punctuation, then removing stop words and finally tokenizing the input while maintaining the order. This was then converted into a indexed representation where each word was associated with an ID, in order to be fed to an LSTM Network.
This indexed representation was then converted into a matrix of dimensions sequence_lengthx300, using Google's Word2Vec model [2]. Vectors for words that were not present in the model were initialized randomly. Also to be noted, the case of the individual words was not normalized, for example, 'action' and 'ACTION' was treated as separate words, since it was assumed they depict slightly different emotions. Also, Google's trained Word2Vec model treats them differently and returns slightly different vectors for both.

III. Model

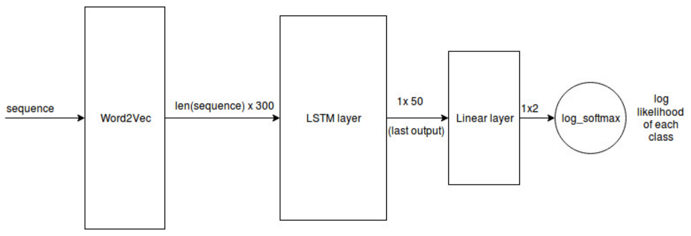

Fig. 1. Model Architecture

The model uses Google's pre-trained Word2Vec embeddings trained on Google News data containing more than 100 billion words [2]. The model outputs a 300 dimension vector and contains vectors for around 3 million words and phrases. The model is trained using skip-grams and the process is described in [3]. Embeddings for the word present in the lyrics corpus were extracted from this model and saved for use in the model. These pre-trained embeddings were used to make up for the lack of sufficient labeled training data. Since the data was not enough for the model to learn contextual information relating to co-occurrences of words, similar meaning words may be treated as widely different by the model. Since the model did not contain all the words that occurred in the vocabulary for the current task, we were forced to initialize those vectors with random values. Given more data, word embeddings for such missing words could be learned by another network specifically used for this task.
Two models were experimented with, one in which these embeddings did not change (or learn) and another in which they did. The model in which we applied gradient to the embeddings was found to perform better than the one in which the embeddings were fixed. We conclude from this that the embeddings were further fine tuned to suit the current application and hence helped improve accuracy. The next layer was the LSTM layer which was used to remember sequential information from the lyrics. Finally, the output from the last cell of the LSTM layer was passed to a Linear layer to reshape into the desired shape, and LogSoftmax was applied to obtain log likelihoods of the two labels. It is defined as,

$$f_i(x) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right)$$

Due to the presence of the LogSoftmax layer, the Negative Log Likelihood loss function was used to compute loss. After experimenting with Adam and SGD optimizers, we settled on SGD optimizer as giving better results.

IV. Results

The available data was shuffled and randomly sampled to make train (80%) and test (20%) sets. The network was trained on a 2GB Nvidia 940M graphics card. Another limitation was the availability of data.

TABLE I
Simulation Parameters and Accuracy

epochs = 100, learning rate = 0.3	78%
epochs = 100, learning rate = 0.1	69%
epochs = 150, learning rate = 0.05	67%

V. Conclusion

While there were limitations on the amount of data and processing power, the network performs reasonably well. In this exercise, we learned how to use Word2Vec models for text classification, and how LSTMs are used on a real world dataset and problem.

VI. Acknowledgement

We thank Professor Gerard Demelo for his continual support and guidance in this project.

References

[1] (2014) MusicMood A Machine Learning Model for Classifying Music by Mood Based on Song Lyrics. [Online]. Available: https://sebastianraschka.com/blog/2014/musicmood.html

[2] (2013) Google Code Archive. [Online]. Available: https://code.google.com/archive/p/word2vec/

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf