LyricLearn

Odikekachukwu Eze-Echesi, Thomas E. Mason

May 2019

Abstract

We tried to predict the year that a song was written based on an analysis of lyrics text data. The first approach we tried incorporated a simple analysis of Term Frequency - Inverse Document Frequency (TF-IDF). We trained this over linear regression with several types of regularization, with some promising results. After this, we tried a more robust approach using deep learning. Here, we tried both character and word-level associations with a Recurrent Neural Network. However, the RNN failed to train effectively due to the vanishing gradient problem, with the loss rate remaining constant even over many epochs. To address this, we tried word-level association with an LSTM network, which appears to show more promise.

1 Introduction

Musical lyrics permeate our lives, influence our thoughts and are usually a good indication of the current state of society within the time period the songs were conceived. We hypothesize that an analysis on the changes in patterns of sentence construction, language organization and historical references in songs within a specific time period, could indicate the era in which the song was produced.

The general classification of music is a fairly well-researched endeavour in the field of natural language processing (NLP). Typically, this sort of analysis is conducted using a combination of audio, lyric and culture data to provide an accurate classification[Ale]. These features are used primarily because they serve as the best indicator of how similar certain songs are to each other and show promise with pattern recognition. Traditional approaches to the classification of text including but not limited to Support Vector Machines (SVM), Naive Bayes (NB) and Logistic Regression (LR) methods have been widely developed and tested but also have their limitations. For example, in NB classification, analyses using huge training sets can create a distortion in the classification scores, which in turn force the scores to be adjusted relative to the data size[Geo].

More recently however, various deep learning methods have shown promise with text classification and analysis including convolutional neural networks (CNN) as well as recurrent neural networks (RNN) which work by passing input through several layers of a network and fine-tuning predictions using back-propagation or dropout algorithms. This error correction process is crucial to our lyric analysis.

In this project, we attempt to provide an estimate of the time period a song was produced based solely on an analysis of it's lyrics. The motivation here stems from a curiosity about the possible changes in patterns within the text and of language choice over time. We approach this problem using multiple methods. In the first approach, we count term frequencies within each lyric data point and identify patterns as a factor of how frequently words appear over the specified time period. In the second approach, we conduct a temporal analysis using an RNN to identify more nuances regarding phrase construction and historical references. We also employ an additional technique using Long Short-Term Memory (LSTM) to account for some of the shortcomings identified in the previously described methods.

2 Approach

2.1 Dataset

We trained our various models over a dataset which was freely available from `kaggle.com`, and which consisted of the lyrics from the Billboard Top 100 songs, from the period 1965-2015. We used only features which could be drawn directly from the lyrics in our training (excluding the name of the artist, the record label, and other metadata) as our analysis is primarily interested in linguistic evolution. We considered this an appropriate dataset for several reasons. Firstly, the data represent a period of 50 years in the United States, during which time both music, popular culture and possibly even language itself have changed dramatically. This ensures that the data is extremely unlikely to be static, and most likely shows significant historical transformations that our models can detect and learn. Secondly, a set of the 100 most popular songs for each year is a very rich snapshot of the music scene in the United States in a given year, and a quick analysis of the dataset reveals songs drawn from a wide variety of genres, giving a broad and representative overview of the popular music of the day.

2.2 Linear Regression (TF-IDF)

For our linear regression model, we chose to train over the Term-Frequency, Inverse-Document-Frequency (TF-IDF) feature of the songs. TF-IDF is a rough measure of how much more frequently a given term will appear in a document (song) than in the corpus on average. The TF-IDF score can be calculated in several ways but in general it is obtained by multiplying the count of the number of times a term appears in a song by the inverse of the average number of times it appears in the corpus. We used between 1 and 3-grams for word encodings, as recommended by common resources on the topic.

In addition to being a common and popular approach in the NLP industry as a "first pass" technique, we intuit that TF-IDF will be a fruitful feature due to the English language's tendency to incorporate new words over time, and to coin new phrases. A quick review of recent media shows many expressions in common use today that most likely would never have been used in the past. Similarly, certain vocabulary has come into fashion for a specific period (think "groovy" during the 1970s). From this, we concluded that TF-IDF might reveal interesting patterns.

2.3 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are one of the most popular tools currently in use for text analysis and Natural Language Processing [Karb]. Given its popularity, we decided to do a first pass using an RNN to predict the song year. After comparing the strengths and weaknesses of doing character vs word-level embedding analysis, we opted to go with word-level embedding [Ma]. There exists substantial literature comparing the two approaches, with the general conclusion being that for large corpuses of text and for learning meta-features such as the date a piece was written rather than semantic understanding, word-level embedding represents a much more tractable and accurate approach. We limited embeddings in the dataset to 400 and the sequence length of the input tensors to 500 to improve the speed at which our network trained (even with these improvements it took over 7 hours to train normally). We believe these were appropriate limitations as approximately 90% of the songs in the corpus had fewer than 500 words, allowing us to substantially improve the performance by cutting down on the input size.

2.4 Long Short-Term Memory Network

Recurrent Neural Networks, in spite of their clear potential, suffer from several problems which complicate our approach [Ola]. RNNs have been known in particular to suffer from the "Vanishing Gradient Problem", where networks are unable to learn after a certain point because the gradients they are computing over the training data become incredibly small. This means that during backpropagation, the weights change by only extremely small amounts due to the small gradients, and the network consequently stagnates. Adjusting the learning rate will not help to solve this, as it is typically a problem inherent to the dataset. RNNs have been known to also suffer from the

inverse "Exploding Gradient Problem", however we addressed this by simply clipping the gradient.

A common remedy for these complications is to use a Long Short-Term Network (LSTM) or a Gated Recurrent Unit (GRU). LSTMs and GRUs essentially operate by adding decision gates to an RNN architecture that allows a choice to be made to either forget (in the case of a very small gradient for example) or remember the information that is being backpropagated. This has been known to be an effective remedy for the vanishing gradient problem, so we decided to try this approach in the event that our RNN ran into issues with learning. Of the two possible approaches, we chose to go with LSTM over GRU because LSTMs enable much longer-term dependencies to persist in the data than GRUs [Ola]. In any text analysis application, there is a high potential that long-term dependencies (for example, something references several sentences or stanzas prior to the current location) will constitute an important latent feature. Consequently, we opted to use LSTM as an improved approach to this problem.

3 Results

3.1 Linear Regression (TF-IDF)

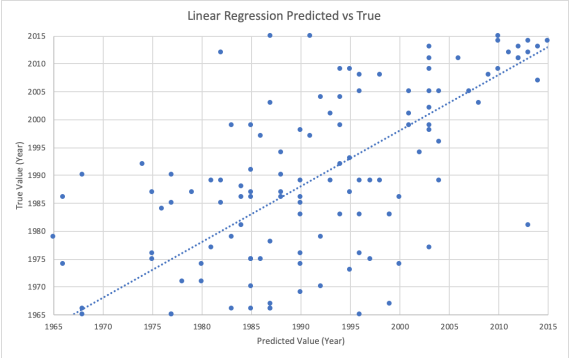

Figure 1: Linear regression results without regularization

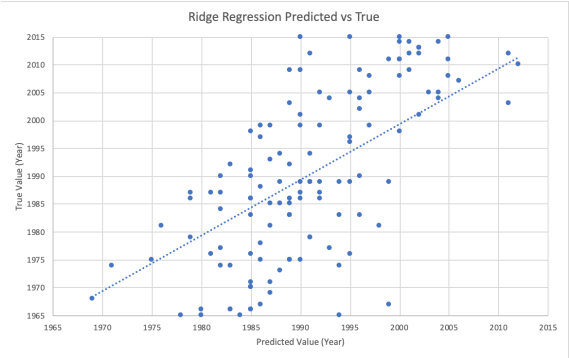

Figure 2: Linear regression results with L2 regularization

As Figs. 1, 2 demonstrate, we were able to achieve a significant (albeit noisy) predictive success using this method. Regularization over the data using the ridge regression method yielded a clear advantage in terms of accuracy, as can be seen. In particular, regularization had the predictable result of making guesses of song year "less incorrect" with more of the guesses being within the correct decade of the song being written than without employing regularization.

One interesting result obtained using regression over TF-IDF was the distribution of the errors over the year the song was written. Our analysis of the residuals clearly indicated songs written during later years (approximately 1990 to present) tended to be better predicted on average than songs written earlier. Although more research would be necessary to confirm this, we speculate

that this has to do with our choice of TF-IDF for the feature in this model. The cause behind this result, we believe, is the emergence and accumulation of terms referencing specific events over time. For example, while songs written after 2007 might very likely reference "iPods", no songs written before this year would have. This likely gives the model increasingly clear signals during later years but makes earlier years harder to predict. This could possibly be offset by extending the historical series further back in time, which would be an interesting area to research in the future.

3.2 Recurrent Neural Network

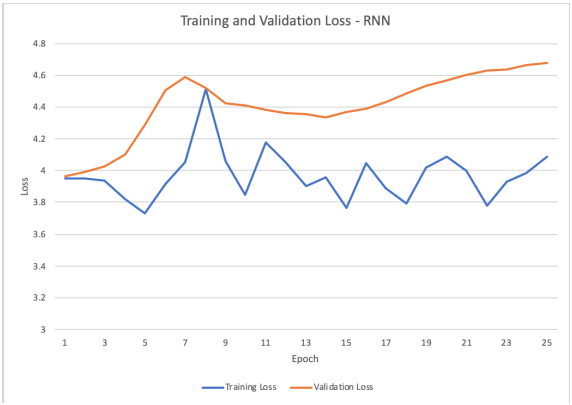

Figure 3: Training loss for RNN word-level associations with learning rate = 0.01

Our Recurrent Neural Net model had a disappointing performance, with little reduction in loss even over many training epochs [Kara]. We tried several different network architectures and learning rates but without significant improvement in the result (Fig. 3 is a typical result for our RNN). Based off of this performance, we concluded that this was likely due to the RNN suffering from the vanishing gradient problem, with too many important long-term associations being needed to learn the signal in our data. This conclusion motivated our decision to attempt to improve on this by using an LSTM network, described below.

3.3 Long Short-Term Memory Network

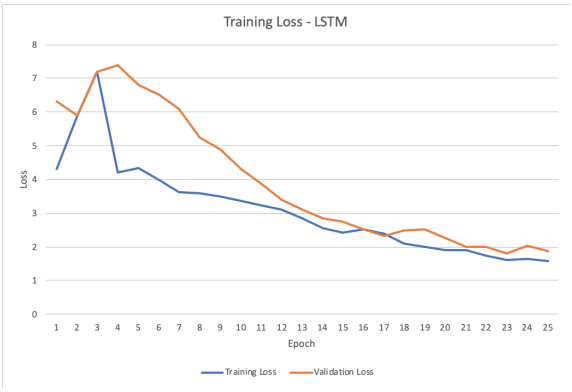

Figure 4: Training loss for LSTM network with learning rate 0.005

For the LSTM network, we used a softmax as our activation function and included a dropout layer [Agaa], and used two hidden layers with dimension 128. This approach yielded better results than the RNN approach, with the losses declining steeply after the first several epochs, leading us to conclude that the issue facing the RNN was likely a vanishing gradient problem. We found that the best results tended to be obtained when using a learning rate of 0.005 and a batch size of 16-32. Based on these results, we conclude that the LSTM approach outperformed our RNN

approach and was comparable to the TF-IDF approach as it stands. However, given the early performance of the LSTM, it is reasonable to expect further gains could be realized from trying additional network architectures and parameters.

4 Conclusion

This project has demonstrated the possibility of predicting the year a song was made based on several different analyses and adjustments made to them. The method using linear regression showed promise and with regularization in particular, the model identified a possible pattern in the increased frequency of certain words from a specific time period till present day. In contrast, the RNN by itself did not offer very promising results even after tweaking the parameters to possibly improve it's performance. The LSTM approach on the other hand, showed promise comparable to that of linear regression provided the learning rate and batch sizes were chosen carefully. A future approach would require a consideration of additional parameters with the LSTM network to make our temporal analysis more robust. Also, it would be interesting conduct the same analyses on better constructed text data including but not limited to articles, news stories and books to identify possible patterns in how these bodies of text change over time and whether or not an accurate prediction can be made as to when they were written.

Acknowledgements

We would like to acknowledge Jaan Altosaar, Rajesh Ranganath and an eclectic group of classmates for helpful discussions and contributions. Jaan provided some inspiration to pursue a topic that was dear to us and offered suggestions around data pre-processing and building a manual classifier incorporating unigram and bigram counts as a baseline. Rajesh offered valuable feedback on the model selections and offered more ideas including the analysis of non-lyric datasets. To the aforementioned and numerous individuals who provided many suggestions and ideas, we are eternally grateful.

References

[Agaa] Samarth Agarwal. Reading between the layers of an lstm network. https://towardsdatascience.com/reading-between-the-layers-lstm-network-7956ad192e58. Accessed May 10, 2019.

[Agab] Samarth Agarwal. Sentiment analysis using lstm. https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948. Accessed May 11, 2019.

[Ale] Tsaptsinos Alexandros. Lyrics-based music genre classification using a hierarchical attention network. https://arxiv.org/pdf/1707.04678.pdf. Accessed May 11, 2019.

[Geo] Kassabgi George. Text classification using neural networks. https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6. Accessed May 11, 2019.

[Kara] Andrej Karpathy. A recipe for training neural networks. https://karpathy.github.io/2019/04/25/recipe/. Accessed May 11, 2019.

[Karb] Andrej Karpathy. The unreasonable effectiveness of rnns. https://karpathy.github.io/2015/05/21/rnn-effectiveness/. Accessed May 11, 2019.

[Ma] Edward Ma. Besides word embedding, why you need to know character embedding? https://towardsdatascience.com/besides-word-embedding-why-you-need-to-know-character-embedding-6096a34a3b10. Accessed May 12, 2019.

[Ola] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed May 11, 2019.

[Rob] Chase Roberts. How to unit test machine learning code. https://medium.com/@keeper6928/how-to-unit-test-machine-learning-code-57cf6fd81765s. Accessed May 12, 2019.

[Ma] [Rob] [Agaa] [Agab] [Kara] [Ola] [Ale] [Geo]