Song Lyrics Generation and Classification by Long-Term-Short-Term Memory Networks

Samuel Kopp[1] and Davor Davidovic[2]

[1] Department of Data Science and Knowledge Engineering, Maastricht
[2] University College Maastricht

Keywords: Deep Learning · Recurrent Neural Networks· Natural Language Generation · Classfication.

1 Introduction

In an attempt to highlight the aspects that are distinctive for humans, artistry would undoubtedly be on the list. Unlike rule-based human activities, creating art is traditionally not regarded as a process which obeys a specific set of rules. In this project we will attempt to discover whether a computer can classify as well as generate genre-specific song lyrics.

With this project we want to explore the realm of computational creativity. We found it interesting to compare the approach a computer takes when generating content as opposed to our content-generation process. Our hope was to get insight into human creativity and art formation. Additionally, we wanted to see if the model can emulate not just grammar but also learn the style of a specific music genre. The second part (lyrics classification) has also an important practical use in automating the labeling of online information on music genres.

Recurrent neural networks have been used as state-of-the-art models in a variety of language modeling tasks. Their loop-like structure makes them ideal for data whose order matters and whose parts are interdependent to each other. LSTMs are special types of recurrent neural networks which are used for learning long-term dependencies between elements in a sequence. Thus, this quality fits well in the lyrics generation and classification problem.

There are several approaches for generating song lyrics. Eric Malmi et. al.[1] employed a model that predicts the next line in a verse for generating rap songs. Each line is taken from the data set it was trained on. This approach makes it easier to form lyrics with a correct structure and grammar, because in most cases a line is already a coherent part. Potash et. al.[4] , in comparison, trained their model to predict the next word given a history of words for a similar task and already showed that this produces qualitatively better results than an n-gram model. However, because available song lyrics on the internet are often susceptible to typographical errors and spelling mistakes, this can lead to extremely large vocabularies and thus large output vectors which are harder to train. This is one reason why many researchers use a character-level approach to generate song lyrics. Sutskever et. al. [5] trained a large RNN with 500 hidden layer and 1500 units each for five days on character sequences of Wikipedia and other sources. The authors showed that grammar and punctuation even over very long sequences can be learned on a character level. They pointed out that even though character level generation seems to be more difficult, because the model can make spelling mistakes or invent its own words, the preparation of the data is much easier. Furthermore, learning which characters make up a word is relatively easy compared to understanding semantic and syntactic structure. An issue with natural language generation in general is finding a good metric to evaluate the quality of the produced text. Most automated metrics to train a language model try to minimize on the deviation from the generated text and a text made by humans. Novikova et. al.[3] showed that this is problematic, because there is not a lot of correlation between the automated metric and the qualitative rating of humans. For this reason, a qualitative analysis should always be done for comparison.

2 Approach

We use a dataset[2] of more than 380,000 lyrics from different artists arranged by genre and year. Genres include pop, rock, rap, jazz, country etc. The two tasks were attempted using two separate models. In this section the generative model is explained followed by the model for the classification task.

2.1 Generative Model

We attempted both generating lyrics on a word and on a character level. In both cases a single word/character is predicted given a history of word/characters. For the first approach, punctuation was removed and non-alphanumeric characters were replaced by white space. Numbers are often part of the vocabulary, especially in rap music (e.g 420 symbolizing cannabis). The n most frequent words were used to create an indexed vocabulary for generating words. Next, the text is transformed to sequences of length m consisting of indices in the vocabulary with the index of the word of position $m+1$ as a label. These sequences are then reshaped to an array of (samples, time-steps, features), where the only feature is the word index. We normalize the indices to values from 0 to 1 to make it easier to train the network. The vocabulary of the character model consists of the English alphabet, punctuation, numbers and white spaces. In the same way as in the word model the text was transformed into sequences consisting of indices for each character. The vocabulary size is significantly smaller in this approach and we expect this to accelerate the learning process. We tried various hyperparameters and ended using for both model architectures four LSTM layers of 400 units each. Between each of these, a dropout layer with a rate of 0.2 is added to avoid over fitting. We hoped that this can help in generating sequences which are more robust to lyric variations it had not seen before. The model was trained using Adam for 50, 100 and 200 epochs. We experimented with different vocabulary sizes for the word model and different sequence length for both models on different music genres.

2.2 Classification Model

In this task, classification of song lyrics into their respective genres using an LSTM approach was attempted. Since the networks expect numbers as input, a vocabulary was built for the songs considered which transformed the words into integers (identical to the generative approach). The labels (the genre of each song) were preprocessed in the same way. A maximum sequence length of 500 words was considered and songs above that threshold were truncated while those below were padded with zeroes.

The first architecture was the same as in the generative model. In the second architecture, a vector embedding layer with a length of 32 was inserted as the first layer of the network. Other values for these hyperparameters were considered but these ended up being the best choices. To simplify, the two architectures were first trained on a binary classification problem between two genres: rap and pop. Both architectures were trained using the RMSprop optimizer for 100 epochs with a batch size of 1024.

The architecture with the embedding layer was then chosen to attempt a multi-class classification problem with five genres: pop, rap, rock, jazz and country. Unlike the previous task, the classes here were less balanced rock, rap and pop had around 24,000 songs each while jazz and country 7,000 each. The training settings were identical to the ones in the binary case. To compare the results with more conventional classification approaches, a multinomial naive bayes model was also used for this multi-class task.

3 Results and Discussion

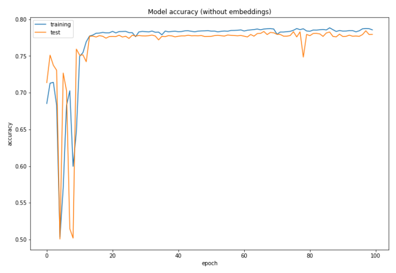
3.1 Generative Model

The results for the generative model will be shown in the presentation. There were some issues with running the experiments over night and the saved files got lost.

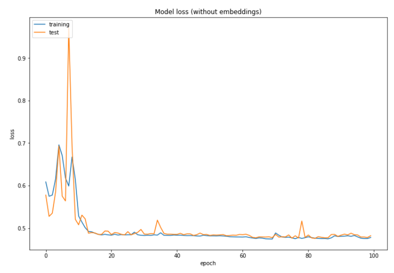
3.2 Classification Model

In the binary classification task, the network without the embedding layer gave an accuracy of 0.78 on the test set. As can be seen on figure 1 below, the accuracy and loss are fluctuating at the early epochs (when the network is probably predicting instances on random), and gradually converge later on. Both metrics are static after the 20th epoch which suggests that the network achieved some local minimum which it cannot get out from.

Using an embedding layer gave a big accuracy boost with a final accuracy of 0.85 on the test set. The word embedding maps the words as multidimensional vectors indicating the similarity between them. This might help the network with extracting meaning from the songs and associating certain words with a specific genre since the two genres use a substantially different vocabulary. However,

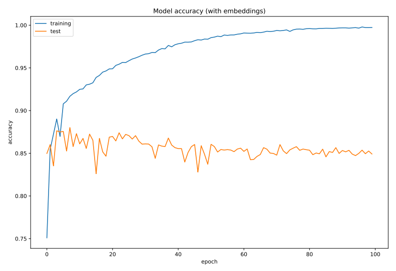


(a) Accuracy plot

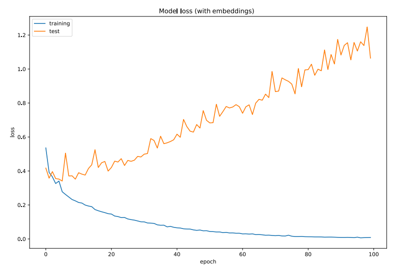


(b) Loss plot

Fig. 1: Performance of LSTM model without a vector embedding layer



(a) Accuracy plot

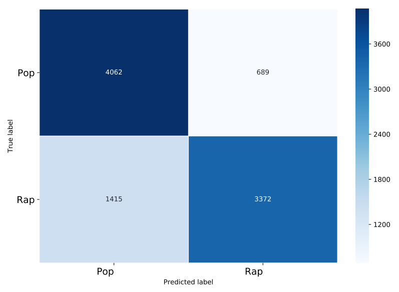


(b) Loss plot

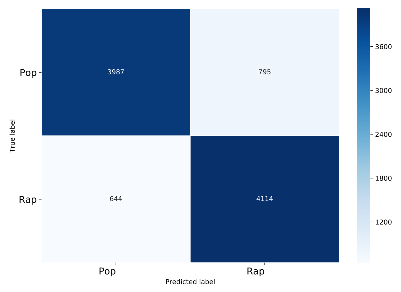
Fig. 2: Performance of LSTM model with a vector embedding layer

as can be seen on the figure 2 the model clearly overfits the training data and performs worse on the test set in later epochs.

It still outperforms the previous approach not only in accuracy but also by significantly decreasing the number of rap songs classified as pop as can be seen on the confusion matrices below. Interestingly, the other approach is slightly more robust to classifying pop music correctly.



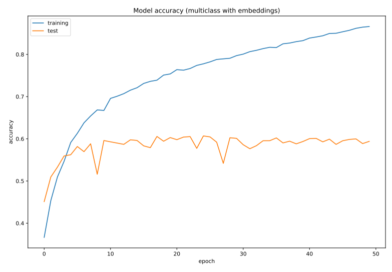
(a) Without embedding



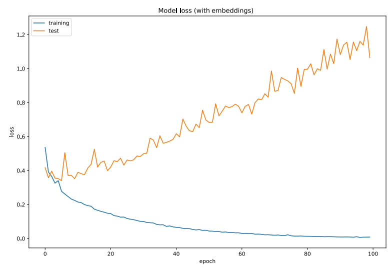
(b) With embedding

Fig. 3: Confusion matrices for binary classification of the two architectures

The latter architecture was then used for the multi-class classification problem. The model achieves 0.59 accuracy across the five classes. In a binary classification task this would not be a good result. But in a task with five-classes it is a solid performance. For reference, the multinomial bayes approach achieved a 0.49 accuracy, which is 10 percent worse than the LSTM model.



(a) Model accuracy



(b) Model loss

Fig. 4: Performance of multi-class classification LSTM model with vector embedding layer

There was several challenges with performing genre classification. First, there was a significant imbalance in the number of instances per genre. For example, the dataset contains more than 100,000 rock songs while only 7,000 jazz songs. This had to be solved by sampling only a quarter of the songs from certain genres. Secondly, similar as in the generative task, spelling inconsistencies caused many words which are essentially identical to vectorized as separate words (for example, "watcha" and "wat-cha"). This decreased the frequency count of some words which influences their predictive power. Additionally, the dataset contained a considerable amount of foreign lyrics (moslty German and Spanish) which added a lot of noise to the model.

4 Conclusion

The classification model performed reasonably well with 0.85 accuracy in the binary case and 0.59 on the multi-class case. Performance significantly benefited from a vector embedding layer in the LSTM network.

There were several challenges in the two approaches. The main one was the noise in the dataset. This is expected and unavoidable since lyrics are mostly crowdsourced and spelling errors are common. Also, the scraping method has not considered only English lyrics, but also Spanish and German among other languages, which introduced a lot of uninformative words with low frequency that just complicated the problem space. One word having many different forms had an influence on the vector embedding layer as well, since such words were considered as separate despite being identical in meaning. Lastly, long training times significantly slowed down the writing process which made experimentation difficult.

As a possible exploration for future work, research should be put into pre-trained vector embeddings. Additionally, more extensive preprocessing should also be applied on the lyrics in order to reduce noise. Lastly, we recommend stronger computational resources (such as Google Cloud) so that experiments would go a lot faster.

Bibliography

[1] Malmi, E., Takala, P., Toivonen, H., Raiko, T., and Gionis, A. (2016). Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 195–204. ACM.

[2] Mishra, G. (2017). 380,000+ lyrics from metrolyrics. https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics.

[3] Novikova, J., Dušek, O., Curry, A. C., and Rieser, V. (2017). Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*.

[4] Potash, P., Romanov, A., and Rumshisky, A. (2015). Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924.

[5] Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.