Predicting mood of a song using transfer learning

Maria Espinoza

December 1, 2019

Abstract

There are plenty of ways to predict the mood of a song, many reports have shown that the lyrics can play a vital row in doing so, in addition to genre, title, and audio analysis. This project will classify a song as one of two options: happy or sad depending on its lyrics using transfer learning.

1 Introduction

Transfer learning is a machine learning method in which a model developed for a task is reused as the starting point for a model on a second task.[1] In this project, a pre-trained model from Google's AutoML tutorial[2] was used to get a sentiment analysis of a song from the dataset.[3] Upon further investigation it was realized that this could become a transfer learning problem if applied to different algorithms as those used during the course, such as Binary Classification.

2 Planning for the Project

2.1 Music Mood Project using sentiment analysis

Originally, deciding on a project was quite hard, and how to approach the problem was equally challenging. The problem as with all machine learning problems is that there are many ways to handle the problem. In this case, the problem was how to determine the mood of a song with a model and a simple data set with lyrics and name and genre. As this project is something I'd like to dig futher into in the future, in terms of both research as to what data can be used to determine a mood, both in the person, how the hardware can be used to aid in that, and what qualities in a song can help as well. In addition to that, there is another project I'd like to work on that also deals with sentiment analysis, so getting some exposure to it through this project will help with that one as well!

2.2 Limiting project scope

For the purpose of applying machine learning, I decided to shrink the project down significantly at first to using a pre-trained model from an API and using a music API[4] on the frontend to type a random song you'd like to find the mood for. Once starting that part, I deemed it much too simple and decided to make it more complex.

2.3 Expanding project scope

Upon completing the basic tutorial about sentiment analysis, it became apparent that simply using the Natural Language Processing API was much too simple, so I decided to try something else. Currently, the dataset found on Kaggle[3] does not have a sentiment label. But if there was one, I could create my own model and use transfer learning to append the sentiment using the Natural Language Processing API and then create my own model to classify a random song as happy or sad instead of using the API directly. This would allow for more exposure to creating a model and getting some more experience and practice using tensorflow. Therefore, a script was created to append the sentiment to each song in the dataset and creating my own model to classify the songs being typed in the frontend.

3 Executing the Project

3.1 Creating script to add sentiment label

Throughout the course, tensor flow with javascript has been used, and the tutorial I was following was in python, so has been a bit of a different experience using it when writing up the script to append the column containing the sentiment. It will become a bit messier once the entire dataset has that label. Perhaps, the dataset will be limited, so as to not overflow in API calls and hardware resources, as 38,000 songs is a significant amount. Not only was it helpful to add the sentiment label so I could expand the project, doing so also helped me be more aware of how to prep the data in order to get it ready for use in a model. Throughout the course we've talked about data-prep, but haven't had much practice with doing so. So preparing the data in this way allowed me to get a more real-life experience.

3.2 Using formatted dataset to create model

Using the newly outputted dataset with the sentiment label, we can now create a model using tensorflow. Currently, this work is being done in the browser in javascript since I will be using an Html page to render the searchbar for the songs. Time permitting, I will create an API in python and render the page in Django instead, so as to get more practice with python and tensorflow. As of now, the project will use binary classification and standardize the labels to be either happy or sad (sentiment ¿ 0.5 = happy, sentiment ¡ 0.5 = sad). The only feature being used in this project is the lyrics, making it a very shallow neural network.

3.3 Predicting song's mood

The user will have an interface to search for a song they'd like to find the mood for. Upon clicking submit or search or something similar, the song API call will get executed, returning song details, including lyrics. These lyrics will get passed through the model and run a prediction using that model that is already trained. The output of the prediction will be either happy or sad. It would be ideal to host the model with the API, but that will be done time-permitting.

4 Conclusion and afterthoughts

This project introduced me into sentiment analysis and just tensorflow and machine learning over all. I was able to apply the information from the course and from the lectures to something tangible and exciting that can be used in the future for my long term project goal. In terms of building the model, I utilized tensorflow with python and used the keras tokenizer API in order to be able to convert the words into a matrix of digits. The next layer of the neural network was a GRU model and then passing through a sigmoid activation. There was a good example I used that described using keras with text embedding and explained how you can build a model like that using a movie reviews dataset. [5] Django REST framework was used as a connection between the frontend in basic HTML and CSS and the tensorflow libraries. In the end, only a dataset of about 100 labeled songs was used due to issues with data prep and AutoML crashing every few iterations of the script that labeled the song. These were used inside the Django REST API in order to train the model and use that to make predictions.

4.1 Prediction Accuracy

Ultimately, the model did not render very accurate results. There are various things that might've contributed to this inaccuracy, such as: small dataset, embedding method used, weights used in the model, and of course, using different loss or activation functions in the model could also help improve the accuracy of the model. In future implementations, more time shall have to be dedicated to simply finding where the model went wrong!

References

[1] Brian Curry *An Introduction to Transfer Learning in Machine Learning.* `https://medium.com/kansas-city-machine-learning-artificial-intelligen/an-introduction-to-transfer-learning-in-machine-learning-7efd104b6026`

[2] Sentiment Analysis Tutorial `https://cloud.google.com/natural-language/docs/sentiment-tutorial?authuser=1`

[3] 380,000+ lyrics from MetroLyrics `https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics`

[4] Musixmatch API `https://github.com/musixmatch/musixmatch-sdk`

[5] Machine Learning — Word Embedding Sentiment Classification using Keras `https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classificatio`