Elijah Stein & Dagmawi Zerihun
12/12/2023
CS 401

Final Project Report: Comparison of Machine Learning Sentiment Analysis – Naive Bayes vs. Recurrent Neural Networks

Overview:

In this report, we analyze the efficacy of our Python program which classifies songs into one of 2 genre categories based on their lyrics. The program utilized the Million Song Dataset (MSD) for training and testing data, NLTK for the Naive Bayes algorithm implementation, and TensorFlow to create a Recurrent Neural Network (RNN).

Code Structure:

For the Naive Bayes model, the implementation is fairly abstracted by NLTK's algorithm. Our process then was as follows: First, we got the data read into a suitable format for NLTK's Naive Bayes, specifically a list of tuples where the 0^{th} element of the tuple represents a dictionary bag of words, and the 1^{st} element is a genre tag. We then ran the formatted data through NTLK's built-in Naive Bayes model after splitting data into training and testing sets (80 / 20). This gave us access to the top 10 most informative features and the model's accuracy on the test data.

As for building the RNN, the process was also fairly abstracted by TensorFlow but making a good model relied on using the right type of layers and understanding their function. Before building the model, however, we extracted the features (lyrics) and labels (genre) from a Pandas Dataframe we built by reformatting the NLTK data, where each record is essentially a vector with the presence of each lyric as a boolean. We then split the data into training and testing features (80 / 20).

Once the data was ready, we initialized a RNN (Sequential Model) object. Then, we added an embedding layer to the model. Word embeddings are vectors that represent the meaning of words as the model learns, so similar words have similar embeddings. We tuned the "embedding_dim" variable to 50, representing the dimensions returned by the embedding layer,

or the level of detail the model will extract from a given feature. Essentially the embedding layer creates 50 dimensional vectors for each word that capture in some way its semantic meaning.

Then we added a Long Short Term Memory (LSTM) layer. LSTM layers process sequences. They maintain a 'memory' of prior inputs in a sequence. We set the variable “units” = 100, meaning are 100 neurons in the layer. This was tuned for a balance of efficiency and speed.

Finally, we added a dense layer. Via TensorFlow documentation, dense layers "do some final processing, and convert from this vector representation to a single logit as the classification output." Essentially, the 100 LSTM layer neurons feed into 2 neurons, one per genre from the “units” parameter, which we set as the computation of distinct genres in the dataset.

We then compiled the model and added an early stopping clause, essentially saying that if the error parameter stops changing significantly, we stop training the model on an earlier epoch. An epoch is a cycle of training with all training data.

Results:

The accuracy results were as follows for the 6 genre combinations we tried:

Genre Combination:	NB Accuracy (%):	RNN Accuracy (%):
Rock & Metal	64	78
Rock & Pop	57.6	65.8
Rap & Rock	90.5	94.7
Rap & Pop	83.2	89.8
Rap & Country	68	79.5
Metal & Gospel	83.9	93.4

First, we saw that the RNN was always significantly more accurate than the Naive Bayes implementation. We also observed that rap music seems to be very distinct. We hypothesized

based on the most informative features that abbreviations play a role. Words like “cuz” were often very informative features for rap relative to another genre. It seems very difficult to distinguish pop and rock, on the other hand. We think this might be because these are both very mainstream genres.

The differences in accuracy between the Naive-Bayes (NB) and Recurrent Neural Network (RNN) models across various music genre combinations can be attributed to the intrinsic qualities of the two types of algorithms. NB models are generally simpler and based on the assumption that all features (in this case, aspects of the lyrics) contribute independently to the probability of belonging to a particular genre. This can limit their performance when the relationship between words is important, as is often the case in song lyrics. RNNs, however, are designed to recognize patterns in sequential data, which is highly relevant for processing lyrics where the context and order of words can significantly affect meaning. Consequently, RNNs often perform better on tasks that require understanding the nuances of language, such as distinguishing between the lyrical styles of different music genres. Their typically higher accuracy in these results reflects their ability to capture the sequential dependencies and contexts characteristic of lyrical content, leading to a better distinction between genres.

The observed differences in performance across genres may also reflect the distinct linguistic and stylistic elements inherent to each music genre. For instance, genres like rap often employ complex rhyme schemes and regional slang, which may offer more sequential patterns and contextual cues for RNNs to learn from, hence their higher accuracy in the Rap & Rock and Rap & Pop combinations. In contrast, the lyrical content of genres like metal or gospel may be more formulaic or repetitive, which could be sufficiently captured by the NB model's simpler probabilistic approach, as seen in the relatively high NB accuracy for Metal & Gospel. The less

pronounced differences in genres that have more in common, such as Rock & Metal or Rock & Pop, could be due to overlapping vocabulary and themes, which pose a challenge to both models in distinguishing between the genres, leading to closer accuracy rates. These nuances underscore the importance of understanding genre-specific characteristics when applying machine learning models to lyrical analysis.

Improvement / Future Research:

To start, both methods (Naive Bayes and RNNs) are supervised learning - i.e. the genres are provided to train with. We could further compare the efficacy of unsupervised learning models in the future.

Additionally, we could implement testing with more than one type of NN – possibly a simple self-coded one, and a more complex TensorFlow model with more layers or back-propagation, etc.

Next, there are inefficiencies in the code made for readability and ease. One such example is reading the data for Naive-Bayes and then reformatting it for TensorFlow, rather than reading the data once and making it work for both formats in one pass.

Another area for improvement would be equalizing the number of songs of each genre in a given run. This could be engineered through the join_sql statement, or by manually chopping off tracks for the genre with more songs.

Finally, we could modify the code such that we take a series of more than one genre at a time. It would require looking at a larger subset of the data to achieve suitable sample sizes but would provide more insight into the capabilities of the models to make complex predictions.