Identifying Music Artist From Song Lyrics

Abhishek Kumar abhishekk@umass.edu
Ajay Shaan Shanmugam ashanmugam@umass.edu
Siddharth Chandrasekaran schandraseka@umass.edu

1 Problem statement

Text classification is in general, one of the easier tasks in NLP in the case of binomial classification or one with a handful of labels. In that sense, classifying lyrics into genres isn't very hard. But things start getting out of control when many labels are involved and datapoints within a class don't share much similarity but datapoints in two different classes do. This explains why identifying the music artist based on lyrics is a tough problem. The fact that music artists occasionally employ songwriters to pen lyrics for songs only makes this problem of identifying the signature of a music artist in their songs harder.

But if one were to use spectograms of the music artists' songs in addition to the song lyrics in identifying the musical artist, the accuracy could be significantly improved. However, we're focussing on the NLP part of this and are experimenting with different baseline and neural network models to complete this task and saving the non-NLP (Image classification) part of the improved solution for the future.

2 What you proposed vs. what you accomplished

We were initially hoping that the RNN-based models (LSTM, bidirectional LSTM and HAN) would blow the other neural and baseline models out of the water. After studying the dataset, we sensed how difficult it would be to distinguish between artists based on recurring themes or actual word usage. Although picking on recurring themes could be a good way to narrow down the possible correct classes for the classification task, it is word usage that would be a complementing factor in deciding on the right class. For the purpose of the former we used pre-trained word embeddings Word2Vec and Fasttext and used these as inputs for our RNN-based models. This was because these embeddings were created in such a way that related or similar words were closer to each other in vector space, using which would allow the model to learn to distinguish between classes based on not just the actual words but also words in their neighborhoods. Now for our non-RNN models, we used the TF-IDF scores of words and bigrams as the inputs, this time training them to make distinctions based on actual word usage.

Although we knew that actual word usage would be the better deciding factor for this classification task, we hoped that the complexity of the RNN-based models, especially the Hierarchical Attention Network, would more than compensate for this. However, as it turned out, it was our non-RNN based model, the not-too-fancy ANN which produced significantly better results than all others. This strongly drives home how superior the TF-IDF scores are to the Word2Vec and Fasttext embeddings for this particular problem. Note that for a similar problem, say, identifying authors from text from books, Word2Vec and Fasttext embeddings could have proved to be better for multiple reasons. Firstly, there would be a significantly larger amount of data for each class for the model to learn from. Secondly, authors write their own books unless in rare instances where they hire ghost-writers to do the job for them. This is unlike in our case where it's common for music artists to not pen lyrics for their songs, leading to significant chances that multiple songs across by different artists were compiled by the same lyricist. This could seriously confuse a classification model. Thirdly, writing lyrics generally does not involve as much creativity and incorporation of singular ideas and motifs as compiling books. Therefore, using Word2Vec or Fasttext embeddings would have helped the model understand the writing styles of authors and recurring themes in their books in order to identify the author from book data. However, in our problem, the range of vocabulary and expression of ideas is limited in the domain of song lyrics as opposed to prose in books. That is why TF-IDF is the better word representation and our models that used TF-IDF (All non-neural, ANN and CNN) instead of Word2Vec and Fasttext (LSTM, bi-LSTM and HAN) generally performed better.

3 Related work

Primarily, we drew inspiration from Alex Chan's Rapnet - A neural network model for classifying hip-hop artists using song lyrics (1). The author tried a random forest, an SVM and a simple neural network with one hidden layer which were fed the TF-IDF matrix as input and compared their performances. However, they were trained to classify between just 3 hip-hop artists and it was reported that the best-performing model (the simple neural network) struggled in distinguishing between artists whose songs have a wide range of themes. We thought perhaps we could improve this by using more complex neural networks and that led to our research. We also decided to go for 100 classes instead of 3 and music artists of assorted genres in order to mimic a more realistic scenario and see how well our models can perform given that it's extremely hard even for a human to identify a music artist from song lyrics.

Our second major source of inspiration was Wallace: Author Detection via Recurrent Neural Networks (2). The authors explored the idea of using recurrent neural networks for a multinomial classification problem very similar to ours. They focussed on having a large number of classes (142) and used GloVe embeddings to train their RNN model. Their RNN model didn't perform any better than their SVM baselines and the authors attributed its poor performance primarily to the misuse of GloVe embeddings in this context. We thought it would be interesting to see if more sophisticated RNN based models like LSTM, bidirectional LSTM and HAN using Word2Vec and Fasttext word embeddings would perform better than non-neural and neural models that use TF-IDF matrices as input.

A third resource which gave us insights into text classification using HAN was the Lyrics-based music genre classification using a hierarchical attention network (3). This motivated us to use a hierarchical attention network in addition to our other neural models, for this significantly tougher problem of music artist classification. Similarly, Hierarchical Attention Networks for Document Classification (4) helped reinforce our understanding of using HANs for a text classification problem.

Other notable papers which helped us in our research include Quantifying Lexical Novelty in Song Lyrics (5), Lyrics-based Analysis and Classification of Music (6), Authorship Attribution Using a Neural Network Language Model (7), Convolutional Neural Networks for Authorship Attribution of Short Texts (8) which motivated us to explore CNNs for this problem, Hierarchical Deep Learning for Text Classification (9), Efficient Estimation of Word Representations in Vector Space (10) and Bag of Tricks for Efficient Text Classification (11). From our extensive research we couldn't find any paper where neural approaches were made to classify such a high number of music artists (100 classes) from song lyrics, and we believe we're the first to make a comparative study of different neural and non-neural models for this problem.

4 Your dataset

The dataset we used to train our models for this problem was the 380,000 Metrolyrics dataset from Kaggle - https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics. We started off by doing exploratory data analysis on this and made the following observations. The distribution of artists between genres was heavily skewed. We didn't make the distribution uniform however since artists within a genre can be expected to have more similar songwriting style than artists from different genres and this would force the model to learn the nuances better and also because genre classification isn't our goal. But it was crucial to have enough datapoints for every artist and therefore after determining the distribution of songs among music artists, we trained and evaluated our baseline models initially on all music artists having more than 200 datapoints and then on top 100 music artists in terms of datapoints. To get a tangible sense of the difficulty of the problem, we created wordclouds of most frequent words

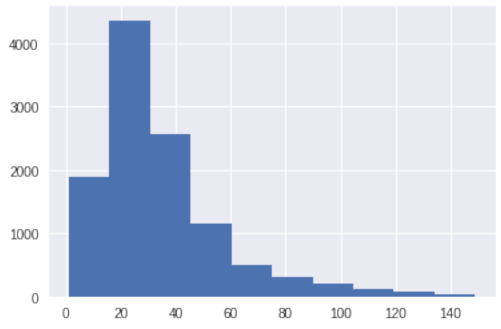


Figure 1: Distribution of sentence Count in lyrics (For top 100 artists)

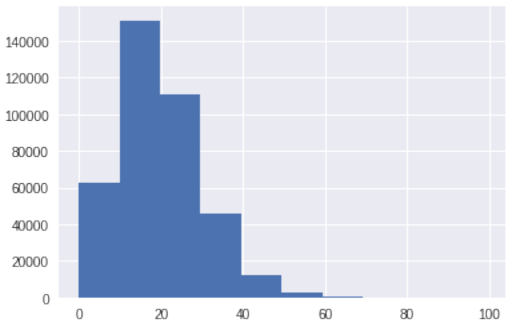


Figure 2: Distribution of Sentence Length (For top 100 artists)

used by top 3 artists in each genre and compared them between the top 3 artists within a genre for every genre and also across genres. We realized that there were a handful of words in common and many that were similar, carried similar sentiment or had a common theme. We understood that even telling a music artist apart among artists from different genres was a challenging task. If our model tries to inherently learn to eliminate possibilities using genre in order to identify the music artist, it's not going to be smooth, and this preliminary analysis made it clear.

We needed to decide on how many sentences and how many words in each sentence do we need to keep for each lyrics for training some of our neural models like HAN. To decide this we plotted the distribution of number of sentences in each lyrics and the distribution of number of words in a sentence as can be seen in figure 1 and 2. There is an average of 19 words in a sentence with standard deviation of 10.4 and an average of 35 sentences in a lyrics with standard deviation of 24. This is for the top 100 artists.

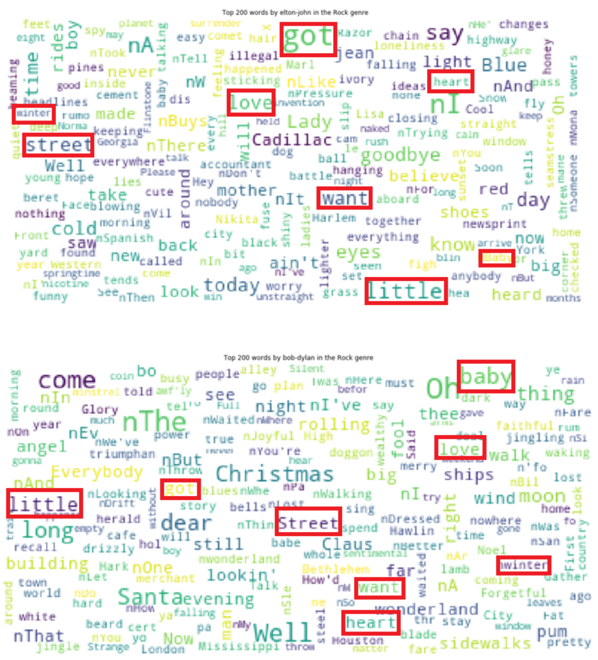


Figure 3: Word clouds of top 200 words used by top artist in Pop and Jazz genres, with common and synonymous words highlighted.

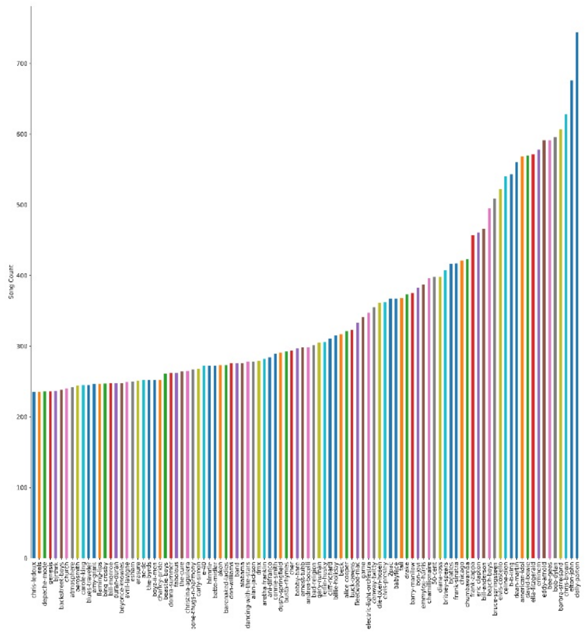


Figure 4: Song counts of top 100 artists

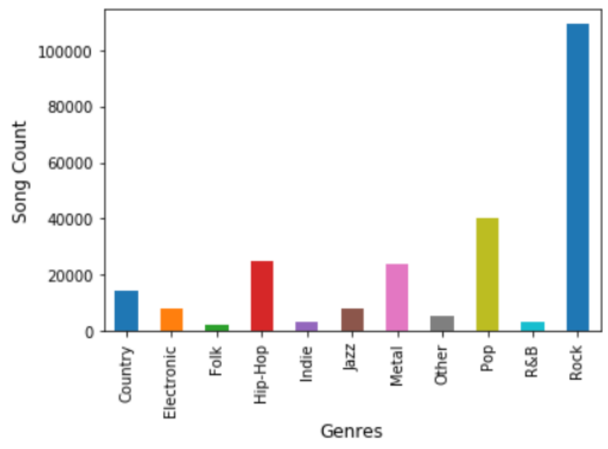


Figure 5: Genres and song counts

4.1 Data preprocessing

As part of preprocessing, we performed the following steps. First we dropped all datapoints that had missing values. Then we extracted datapoints pertaining only to the top 100 artists in terms of number of songs. We earlier went with an extract containing artists having 200 or more datapoints. It didn't yield good results since the number of labels in that case was 140 and 200+ datapoints for each weren't good enough for the model to discern the music artist from the lyrics. After getting that extract in a dataframe, we performed the typical preprocessing steps of transforming all lyrics to lowercase, removing stopwords, removing frequent words and words that occur in rare occassions and punctuations etc. Then we created features using count vectorization, TF-IDF vectorization at word level and bigram level, pretrained Fasttext embeddings and pretrained Word2Vec embeddings.

5 Baselines

After pre-processing, we proceeded to build the baseline models. We started off with using the TF-IDF word embeddings with a multinomial Naive Bayes classifier. We tried different alpha values for smoothing and also experimented with a model where the class prior probabilities are learned instead of being set uniformly. Then we read about an adaptation of the multinomial Naive Bayes model called the Complement Naive Bayes which was empirically proven to correct severe assumptions made by its multinomial couterpart. This technique uses statistics from the complement of each class when calculating the model's parameters and is particularly useful for imbalanced datasets. Despite our imbalanced dataset used for training, the standard multinomial Naive Bayes model still performed better, with our best accuracy being 0.3299.

Then we tried solving this problem with an ensemble of trees - a random forest model using the TF-IDF word embeddings. We experimented with different number of trees to be used in the ensemble such that the model was powerful and there was no overfitting. This was ultimately set to 100. We also tried using different split quality criteria - the gini impurity and entropy. This didn't have much effect on the result though. We also tried using log2(num features) instead of the usual sqrt(num features) for the maximum number of features the model should consider from which the best split is calculated. Our best performing random forest model resulted in an accuracy of 0.3669, which was quite impressive. We then tried an eXtreme Gradient Boosted model which gave a rather insignificant accuracy boost from the random forest's.

The models above were experimented with using both word-level TF-IDF scores and bigram level scores, and the former led to better results.

6 Your approach

However, not fully satisfied with the non-neural baseline models, we then tried different ANN architectures and finalized with a 3-layer neural network with two fully connected layers containing 5000 units and the final dense layer with 100 units, with dropout and batch normalization layers in between and a softmax activation at the end, and could improve the accuracy to 0.3907. Note that for all neural models, we used the Adam optimizer with categorical cross entropy as the loss to be minimized. This model performed the best and we attribute its success to using TF-IDF as input instead of word embeddings. After that, we experimented with simple LSTM and bidirectional LSTM networks and finalized the LSTM and bidirectional LSTM, both having 1000 units and dropout enabled that take in Word2Vec embeddings as input. We also tried the same with Fasttext embeddings as input. We knew that using TF-IDF scores for words was the better technique for this problem than using Word2Vec and Fasttext, but we had a sliver of hope that perhaps LSTM or bidirectional LSTM would do better given that this is a text classification problem and such RNN based models are better at learning dependencies between sequential information like words in sentences. The results weren't too surprising though. LSTM and bidirectional LSTM did worse and took longer to train. We even tried using word sequences of different lengths, adding dense layers of varying complexities, but to no avail. We realized that using the TF-IDF matrix instead of the Word2Vec and Fasttext embeddings was the way to go! This makes sense because unlike a typical text classification like genre classification based on lyrics where the word embeddings pertaining to similar words will be closer in vector space, which will in turn lead to much better classification results than when TF-IDF vectorizer is used, in our case, intuitively, the classifier should try to distinguish between artists based on the actual words they use, not based on all words in the neighborhood of the actual words (the word vector values will be similar for these words and thus the model will tend to miss the importance of signature words corresponding to an artist). The lyrics pertaining to artists of the same genre tend to have many overarching themes and therefore many words frequently used by an artist will tend to be in the neighborhood of frequent words used by another artist. This will confuse the classifier, more so when the model picks up the fact that even artists of different genres sometimes have similar themes of lyrics. This problem is even exacerbated because we don't have enough datapoints for every artist such that the model may correctly classify not a handful but a hundred artists based on their song lyrics. We then tried out different architectures of CNN and finalized on a 3 layer CNN with two convolutional layers having 64 and 32 filters, along with maxpooling and dropout and a dense layer at the end followed by softmax activation. This network used the TF-IDF matrix as input and could achieve slightly lower accuracy than the ANN we experimented with earlier. We were very hopeful that HAN would do better with the word embeddings despite RNNS, LSTMs and bidirectional LSTMs performing badly. This is because HAN has two components, the bi-directional LSTM and an attention model.

The structure of HAN that we are using follows the one used by (4). We are using bidirectional LSTM instead of the bidirectional GRU which was used in the original model. We focus on lyrics level classification where we assume each lyrics to be composed of multiple sentences. Given a sentence we first map the words to word embeddings and run it through a bidirectional LSTM which summarizes the sentence. To account for the fact that not all words are equally important, we then use an attention mechanism to give more weight to the important words. We find the weighted average of the output of the bidirectional LSTM where the weight is the attention computed as described in (4). This generates a sentence vector and then we repeat the process of passing sentence vectors through a bidirectional LSTM and applying attention to them to get a final vector which summarizes the document. This vector is used for the final classification task using softmax. The model architecture can be seen in figure 6.

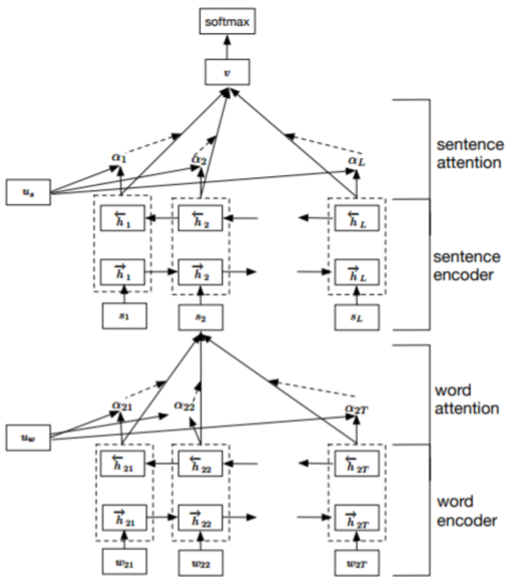


Figure 6: HAN

The figure 7 shows the keras model implementation for the HAN (info of nested models is hidden. See figure 8 for details of nested model). The model takes each lyrics in the form of a 2D matrix of size $35 * 12$ where 35 is the max number of sentences and 12 is the max length of each sentence. For the final model we chose 45 sentences each with at max 14 words for each lyrics. The guess was based on the initial analysis of data where

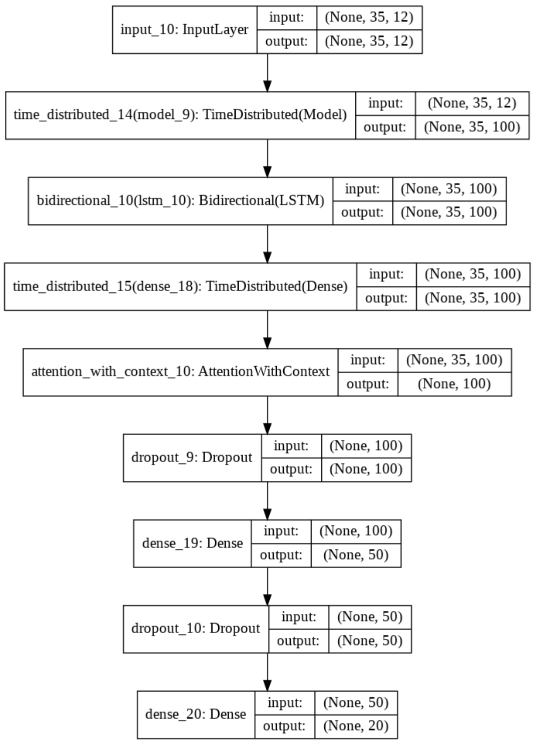
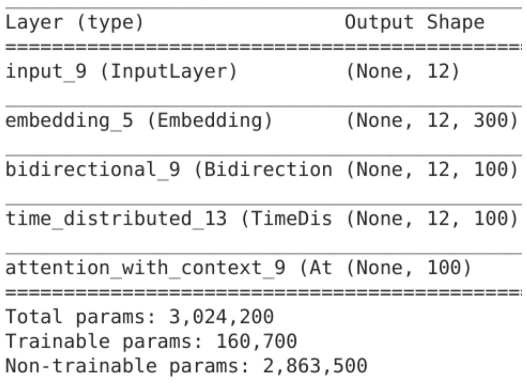

input_10: InputLayer | input: (None, 35, 12) | output: (None, 35, 12)
time_distributed_14(model_9): TimeDistributed(Model) | input: (None, 35, 12) | output: (None, 35, 100)
bidirectional_10(lstm_10): Bidirectional(LSTM) | input: (None, 35, 100) | output: (None, 35, 100)
time_distributed_15(dense_18): TimeDistributed(Dense) | input: (None, 35, 100) | output: (None, 35, 100)
attention_with_context_10: AttentionWithContext | input: (None, 35, 100) | output: (None, 100)
dropout_9: Dropout | input: (None, 100) | output: (None, 100)
dense_19: Dense | input: (None, 100) | output: (None, 50)
dropout_10: Dropout | input: (None, 50) | output: (None, 50)
dense_20: Dense | input: (None, 50) | output: (None, 20)

Figure 7: Sentence level HAN. model_9 refers to word level HAN. This currently shows prediction over top 20 artists.

there is an average of 19 words in a sentence and an average of 35 sentences in a lyrics as mentioned in dataset section. The model takes lyrics of size $35*12$, passes each sentence to a nested han model which generates a 100 dimensional summary of each sentence formed using attention mechanism. The nested word level han model uses an embedding layer to map words to word embedding and these embedding were kept trainable during training as we didn't find pretrained embedding for several words found in lyrics like 'whoaaa', 'settlin' etc. These sentence summaries are later used to create attention based summary of lyrics which is used for prediction. We used a learning rate of 0.002 with Adam optimization (same lr and optimizer was used for the final HAN model also). Categorical Crossentropy was used as the loss function.

Despite having a separate attention model to focus on important words critical to identifying the artist information, the HAN model didn't perform as well as our best model, a 3-layer neural network which uses TF-IDF. We got accuracy of 0.3417 using HAN. We think this is because the dataset is not adequate for the HAN to get trained well, and



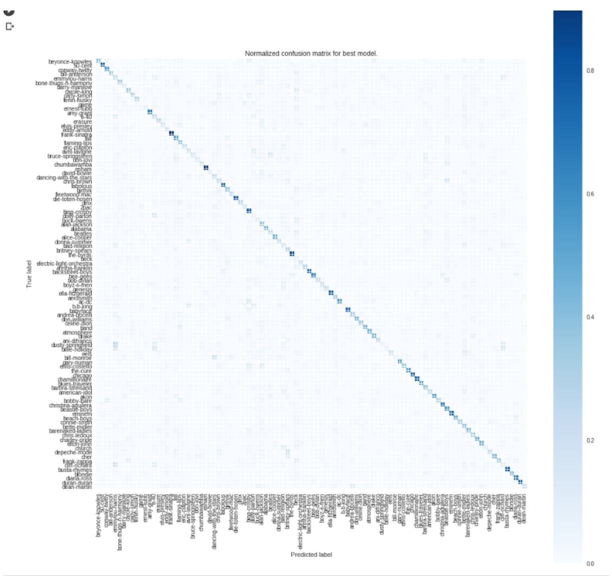
```
Layer (type)                 Output Shape
=============================================
input_9 (InputLayer)         (None, 12)

embedding_5 (Embedding)      (None, 12, 300)

bidirectional_9 (Bidirection (None, 12, 100)

time_distributed_13 (TimeDis (None, 12, 100)

attention_with_context_9 (At (None, 100)
=============================================
Total params: 3,024,200
Trainable params: 160,700
Non-trainable params: 2,863,500
```

Figure 8: Details of Word level HAN. Here we are using 12 as the max number of words in a sentence of lyrics.

TF-IDF word embeddings weren't used. All of our implementations were done using Keras with Tensorflow as backend in Google Collab environment. For our baseline models, we used sklearn's inbuilt libraries and just tuned the hyperparameters using gridsearch.

7 Error analysis

The non-neural baseline models that we tried, despite their good performance, weren't suited for the real world because they weren't as good as a neural model in non-linear classification. Unlike a typical text classification problem like sentiment analysis or genre classification based on lyrics where the number of labels are somewhere in the range of 2 and 10, expecting a classifier to accurately tell apart a music artist from 99 others solely based on song lyrics every single time is unrealistic. It is very hard even for a human being. However, so far from all our experiments with different models, we could achieve an accuracy around 40 percent. The LSTM and bidirectional LSTM models that used Word2Vec or Fasttext embeddings didn't do well because of the reason stated in the preceding section. We tried to formally investigate this by drawing out a few misclassified examples in the validation dataset. The following is a notable example. Beyonce's song 'Be with you' was misclassified as a song by Avril Lavigne. We inspected the lyrics of the song and found it to be composed of mushy, lovey-dovey language with words like 'love', 'whisper', 'feel', 'care', 'thinking', 'baby' etc. These words, their synonyms and similar words are commonly used in love songs in the pop genre. We found the ex-



act and similar words in Avril Lavigne's song vocabulary. Thus the model failed at correctly identifying the music artist. This is the reason TF-IDF trumps Word2Vec and Fasttext word embeddings and model relying on the former are much superior than the ones based on the latter. As for the simple 3 layer artificial neural network with three dense layers sandwiching dropout and batch-norm layers returning slightly better results than the CNN architectures we tried so far, it's hard to pinpoint what exactly is causing this, but we suspect that it's because convolving the TF-IDF matrix provided as input with kernels perhaps doesn't effectively summarize the matrix. We also tried using the bigram level TF-IDF matrix in addition to word level TF-IDF matrix but they were either equally good as using word level TF-IDF scores with some neural architectures or worse. We also plotted the confusion matrix of the best performing model using a heat map and it told us that artists such as Frank Sinatra and Chumbawamba have a unique writing style and vocabulary since their songs were almost always classified correctly back to them as shown above. Whereas there are a lot more artist's whose lyrics were almost always mis-classified either meaning that they don't have an artistic signature in their lyrics or because they hired other lyricists to write their songs.

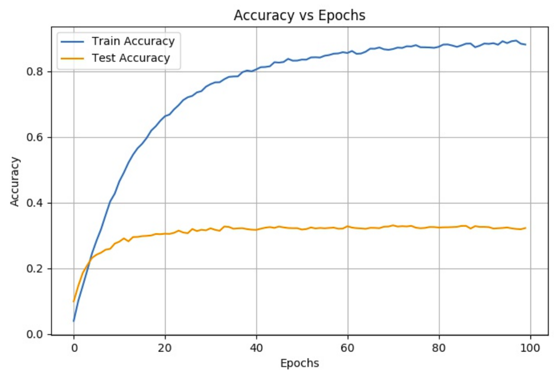


Figure 9: Train and test accuracy of un-regularized HAN for top 100 artists over epochs

8 Results

Baselines	TF-IDF I	TF-IDF II
Naive Bayes	0.3299	0.2480
Random Forest	0.3669	0.2734
XGBoost	0.3680	0.2656

TF-IDF I stands for TF-IDF done at word level
TF-IDF II stands for TF-IDF done at bigram level

Figure 9 shows the test and train accuracy for HAN over the epochs where we can see that the test accuracy flattens after around 30 epochs. To get a better sense of understanding of what our attention model was actually learning, we visualized the attention scores of words for some sentences of a Beyonce song as shown in figure 10. We can see that in line 12 of the song, 'hustle' has the highest weight of around 98% whereas word 'rather' has 0% and 'hard' has around 2% weight. This may be because 'hustle' is a rare word to be used in songs and it's unique to the style of Beyonce's writing. The model indentifies this and puts most of the weight on it rather than distributing weightage over common words. It can be seen that the same word 'rather' has full 100% weight in the first line (line 8) as it's the only word in that line (these are after preprocessing). We did a similar visualization for the weight of these sentences from the sentence level attention as can be seen in figure 11. The line with the word 'hustle' has around 12% weight among all sentences (total of 45 for final model).

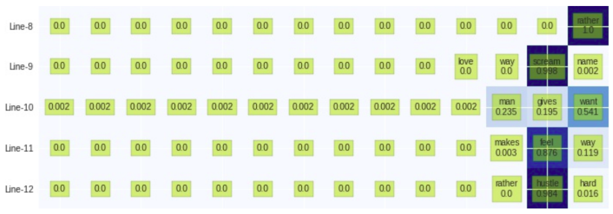


Figure 10: Visualization of the attention for words of some sentences from a correctly classified Beyonces song where we show weight of each word. The higher intensity of blue color denotes more weight for that word. (The green color for text background is for readability.) Here we allow max sentence length to be 14.

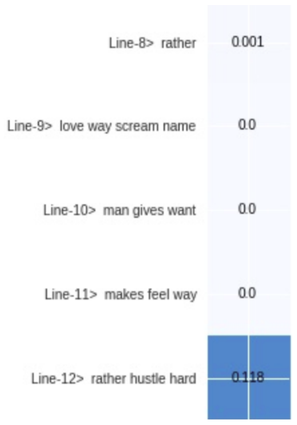


Figure 11: Visualization of the attention for some sentences from a correctly classified Beyonces song where we show weight of each sentence. The higher intensity of blue color denotes more weight for that sentence.

Neural Models	Accuracy
Best ANN w/o regularization	0.3762
Best ANN with regularization	0.3942
Best CNN	0.3271
Best LSTM	0.2756
Best Bidirectional LSTM	0.2792
Best HAN	0.3417

9 Future work and Extensions

Unlike a typical text classification problem, classifying genre based on lyrics where the word embeddings pertaining to similar words will be closer in vector space has not led to better classification results than when TF-IDF vectorizer is used. For the problem statement here, the classifier should try to distinguish between artists based on the actual words they use (signaturizing) and not based on all words in the neighborhood of the actual words (word vector values will be alike for similar words and thus the model will tend to miss the importance of signature words corresponding to an artist). So the best bet to classify the artists is by capturing the syntactic signatures hidden in the lyrics. Syntactic tree embedding (12) can be of great help here. They eliminate the issues Word2Vec, GLoVe and other common Fasttext embeddings are prone to. (12) proposes proposes a novel new strategy to encode syntax parse tree of sentences into learnable distributed representations. And the impressive part is that, if there are enough dimensions in the vector representation, the syntax tree information is encoded in a loss-less fashion. We believe that syntax tree embedding can cause significant improvements in accuracy especially with the HAN model.

10 Contributions of group members

We brainstormed ideas and worked on each of the steps sequentially together (such as data collection, clean up, data analysis visualizations, building models, hyperparameter tuning to get the best results, training models, error analysis, visualizations for error analysis and the written report). We believe in discussing the whole section of work thoroughly, making a list of micro-tasks which each of us then pick up. We found this technique to be very effective especially for a tough problem such as this, where what to do next was decided by the error analysis of the previous model.

11 Conclusion

We were able to achieve a decent accuracy at this task of predicting the artistś name based on lyrics. We got to explore a lot of different techniques of NLP and were able to discover the advantages and disadvantages of each of the models. We learnt to visualize the intermediate results and were able to get valuable insights as to what the model has learnt and what it hasn't yet. We have also realized the importance of error analysis which gave us some of our best leads for improving our models. The most difficult part was when we tried to improve the accuracy of the HAN model. We had to try out various sentence and song lyrics sizes, visualize the attention networks etc to understand where the model was lacking and where the bottleneck was. We were surprised when our neural model with TF-IDF outperformed our best HAN model. But we were not caught off guard though since we understood what TF-IDF captured which the other commmon word embeddings missed out on and this information was a crucial feature for the Authorship attribution problem. As mentioned in the future work and extensions section, we would place our bets on syntax tree encoding as the input for our HAN model.

References

[1] Chan A *Rapnet : A neural network for classifying hip-hop artists based on song lyrics.*

[2] Leon and Derrick *Wallace: Author detection via recurrent neural networks.*

[3] Tsaptsinos *Lyrics-based music genre classification using a hierarchical attention network.*

[4] Yang Z, Yang D, Dyer C, He X, Smola AJ *Hierarchical attention networks for document classification.*

[5] Robert J Ellis, Zhe Xing, Jiakun Fang, and Ye Wang *Quantifying Lexical Novelty in Song Lyrics.*

[6] Michael Fell, Caroline Sporleder *Lyrics-based Analysis and Classification of Music.*

[7] Zhenhao Ge, Yufang Sun and Mark J.T. Smith *Authorship Attribution Using a Neural Network Language Model.*

[8] Prasha Shrestha, Sebastian Sierra, Fabio A. Gonzlez, Paolo Rosso, Manuel Montes-y-Gmez, Thamar Solorio *Convolutional Neural Networks for Authorship Attribution of Short Texts.*

[9] Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, Laura E. Barnes *Hierarchical Deep Learning for Text Classification.*

[10] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean *Efficient Estimation of Word Representation in Vector Space.*

[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov *Bag of Tricks for Efficient Text Classification.*

[12] Richong Zhang, Zhiyuan Hu, Hongyu Guo, Yongyi Mao *Syntax Encoding with Application in Authorship Attribution.*