# Machine Learning Algorithms and Heuristics - Individual Report

## Introduction

### Abstract

This project looks at comparing two different machine learning approaches for generating music lyrics for different genres. Specifically we look at Bidirectional Encoder Representations from Transformers (BERT), a technique for NLP (Natural Language Processing) pre-training developed by Google and Generative Pretrained Transformer 2 (GPT2), also a technique for NLP which was developed by OpenAI. Both of these models are incredibly powerful. GPT2 was so powerful, OpenAI decided not to release the fully trained model out of fears it could be used for malicious purposes and have instead released a smaller trained model that can be used for research purposes. BERT is what Google is using for the powerful gmail predictive text and as part of their ranking algorithm in Google search. It is thanks to these developments that we can develop an application such as this lyrics generator which mimics something a human would write, which would be impossible not long ago but is now very much a reality.

## Development Pipeline

### Data Collection

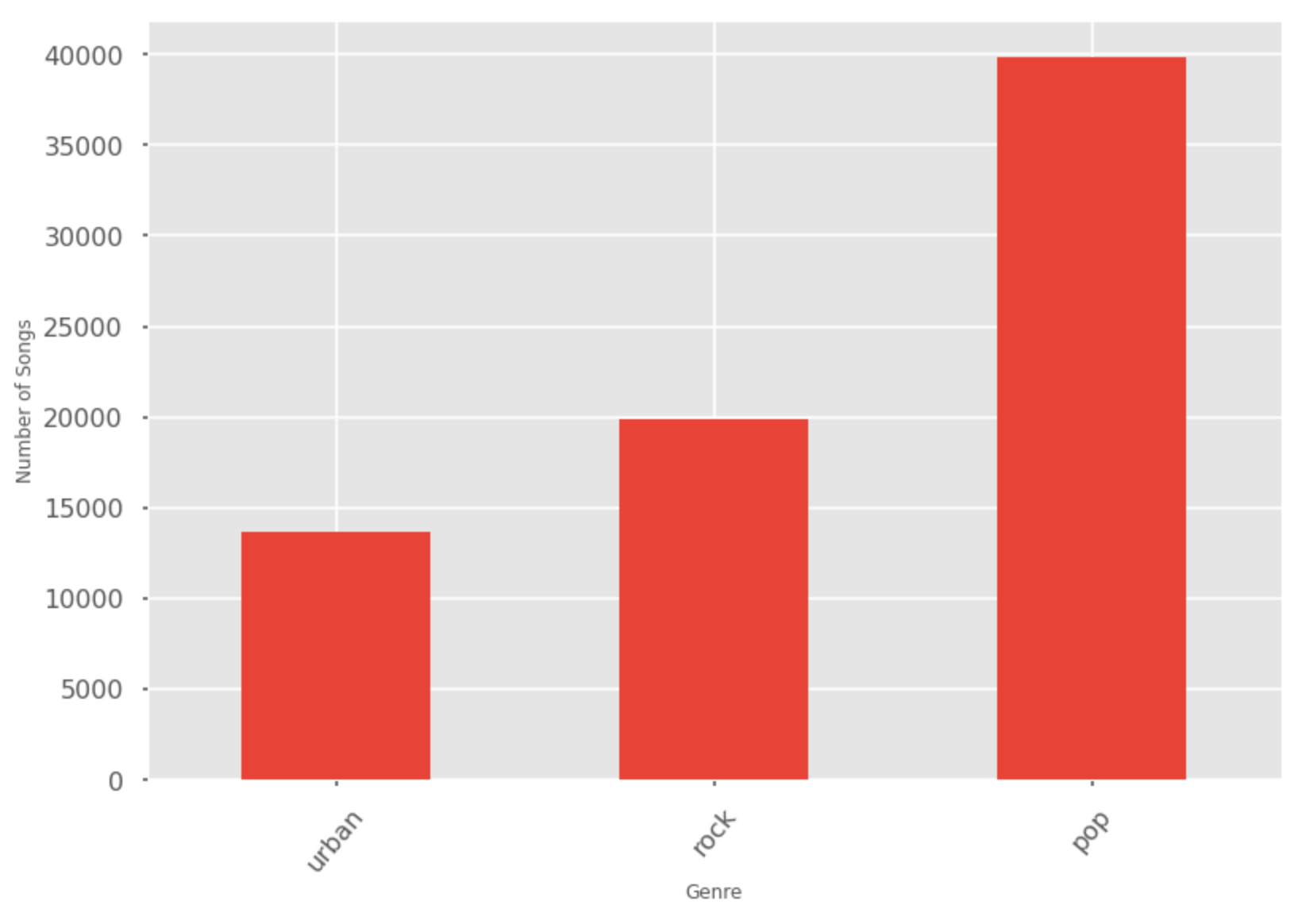
For the data collection we originally intended to scrape from the lyrics site [genius.com](http://genius.com). However doing this is not as easy as we initially thought. The major problem is that these sites employ DDOS protection, so after so many visits from an IP address, they ban your IP. This meant that it was very hard for us to grab our data without either a rotating proxy(expensive for the amount of bandwidth we required) or to have a larger cooldown (our script will have taken months to complete). Therefore we were forced to look into datasets that had already been made. We found some datasets on kaggle but unfortunately most of these datasets were of low quality and it would have been a very large job to clean. Luckily, right around the time we were exploring these datasets, a new dataset was uploaded which was a dataset of lyrics from the site [azlyrics.com](http://azlyrics.com) which we analysed and concluded it would work for our project [(*AZLyrics song lyrics*, n.d.)](https://paperpile.com/c/LSeYsr/JKMm).

There were however some problems with the dataset. One problem was that the dataset was in CSV format which separates the columns by commas. However this caused conflicts with some of the lyrics which themselves contained commas, causing new cells to be created where they shouldn’t be. This was a hard problem to fix, we did find a solution to fix this problem however we did lose around 10% of our dataset in this process. In addition some of the lyrics were not in English which would of caused some problems during training. To fix this we used the python package Langdetect [(fedelopez, n.d.)](https://paperpile.com/c/LSeYsr/LqKu) to detect languages other than English and remove them from our dataset. We also had some words in the dataset which we thought could be offensive, so in order to try to stop our AI from producing these words we removed these words.

Furthermore, we also wanted to use genres in our AI. This is because there are significant differences in the language used in say, rap music, compared to pop music. Therefore we concluded that this feature would be very powerful for helping our AI create more meaningful lyrics. However we had a problem in that our dataset did not come with these genres. However it did include the artist URL. This brings us back to the original problem with scraping (cost and time) however our dataset had been reduced pretty significantly during the cleanup and we would only need to scrape each artist, rather than each song. This means we had much less urls to scrape than we had originally. To solve the problem with the IP block we purchased a rotating proxy and used a script to scrape the genre of each artist.

### Feature Analysis

In our dataset we have 73391 songs. A breakdown of the different genres can be found below.



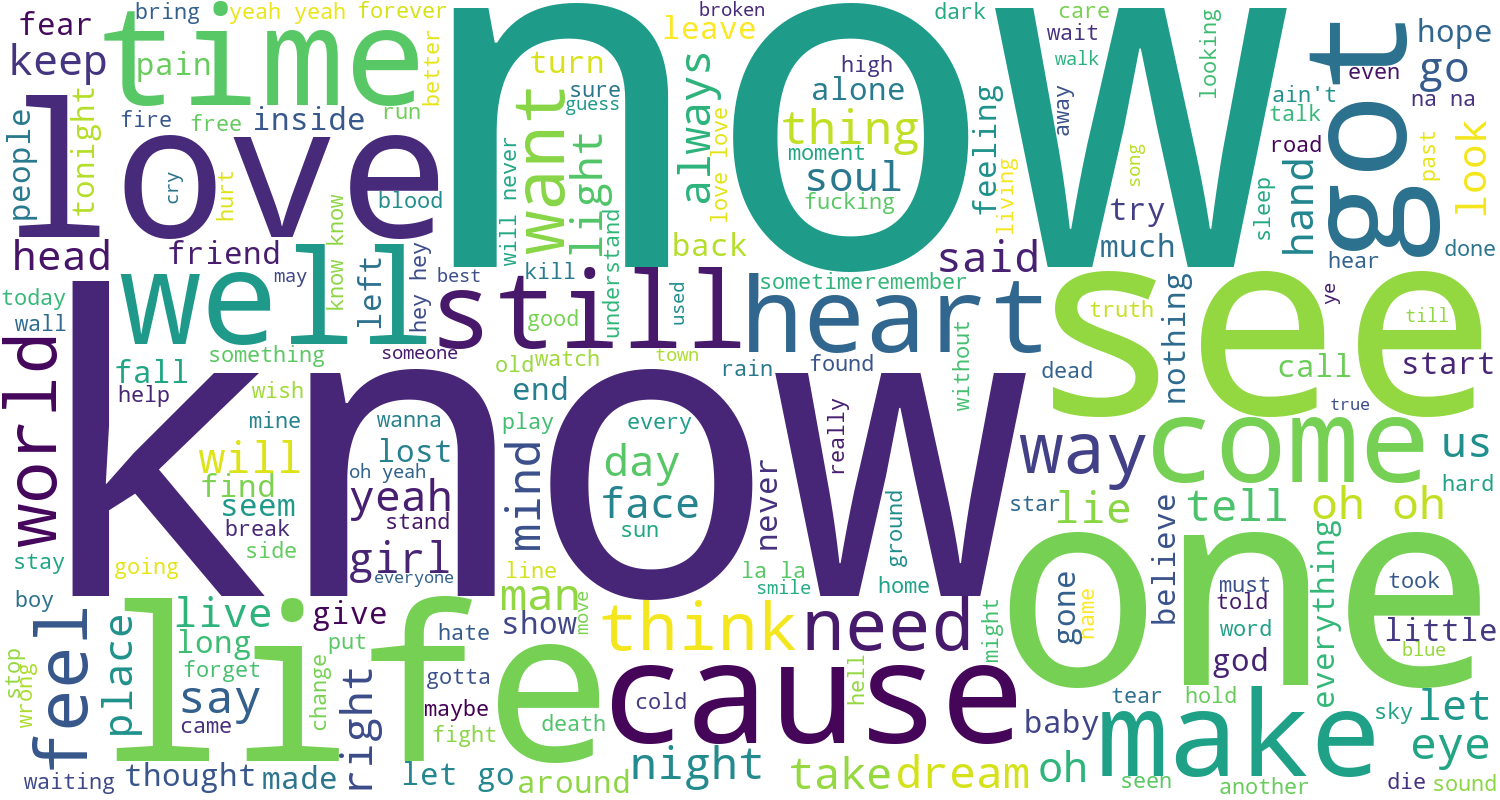
Our dataset had much more pop songs than rock and urban. In fact, there are more pop songs that rock and urban combined (33516). This result is somewhat expected. However it could mean that are models for pop could be more accurate than rock and urban, as more data is available.

I also did some word analysis on the lyrics. This is helpful in spotting trends in the data and can help to make sense of outputs.

Below is the word cloud for the urban genre



Below is the word cloud for rock



And finally the word cloud for pop



Interestingly, there seems to be much less repeating words in pop, that is it is more diverse than rock and urban which seem to be more concentrated towards the more common words. It means that when training the urban and rock models we will need to look out for it repeating words too much and adjusting parameters if need be, considering the fact that there is also less data for these genres.

### Data learning

For our project we used two different models, BERT and GPT-2. We ran all of our applications on Google colab which is a free service that allows you to use powerful GPUs in the cloud.

#### BERT

BERT is a very advanced NLP. It’s a transformer, meaning that it learns contextual relations between words and sub-words in a text. In the most basic sense, it is made up of an encoder that reads the text input and a decoder that produces the prediction. The inputs are converted into a series of tokens which are first embedded into vectors before they are fed to the neural network. The difference with BERT compared to other transformers is that it is bidirectional. Meaning that instead of reading left to right or right to left the transformer reads the entire sequence all at once. This means that it learns based on the context of the surrounding language.

The way we have designed our application with BERT is we feed it with a batch of randomly selected sentences, then use BERT’s next sentence prediction to choose from this batch which sentence is most likely to come after the previous line. Then we take this line and feed it back in so it generates the next most likely and so on and so forth.

For this, we are using the pretrained model with no fine tuning. We planned to fine tune BERT but we ran into some issues. BERT is very powerful and trained on a large set of data and so fine tuning requires a lot of resources and we had issues where we kept running out of RAM. Even though Google colab provides 32GB it was not enough. We tried TPU and GPU versions but sadly we still ran into the same issue therefore we decided to use the pretrained model instead without fine tuning.

In addition, next sentence prediction also uses a lot of RAM. So if you feed it 1000 sentences and ask it to predict the next, the RAM usage will rapidly get full. This is why for optimal performance we stick to a number of choices of 100 in our application. This gives us the best results while still being possible to run. We also experimented using a lite version of BERT, ‘bert-base’ which does allow us to feed it more choices but rendered much worse results hence we decided it would be better to stick with the standard BERT model.

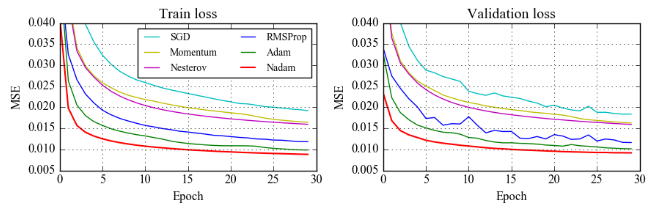
##### GPT2

The release of GPT-2 was a huge advancement for machine learning. It was famous for it’s human-like talking abilities, being able to generate content from fake news to anti-recycling articles. One of the reasons why is the sheer scale of the data it was trained on, 1.5 billion parameters and 8 billion websites. However something to bear in mind is that this model was not publically released, only a much smaller model was. GPT2 is essentially a transformer combined with a pre-trained language model. But because it’s generative, they threw away the encoder part of the transformer and instead it is just a decoder with references to the encoder.

In order to fine tune GPT-2 to our dataset we used a version of GPT-2 called GPT-2 simple. The largest model in GPT-2 we could fine tune was the medium sized model, therefore we decided to use this one.

We decided to use tensorflow GPU rather than CPU or TPU. CPU would have been much, much slower and the TPU in theory would be faster but in practise GPT is badly optimised for TPU and we found that we ran out of memory fast. Therefore the best option was clearly to use GPU.

For our optimiser we decided to use an Adam optimizer which is also the default in GPT2 simple. We chose Adams for our model because it is one of the most efficient optimisers for large datasets and many studies have shown that Adams works very well in practice [(Brownlee, 2017; n.d.)](https://paperpile.com/c/LSeYsr/LqKu+M3wB)



In order to fine tune our dataset we created 3 different models, one for each genre (pop, rock and urban). We trained at 1000 steps as this is what was recommended by the developer of GPT-2 simple [(Woolf, 2019)](https://paperpile.com/c/LSeYsr/Xw3z). We set up our training notebook so that checkpoints were saved at regular intervals, this was very useful in case there were crashes or outages.

We experimented using both the TPU and GPU to train. Interestingly we found that the GPU there was a significant speedup when using the GPU over the TPU, by 1200%. Once again, this is most likely because of GPT-2 optimization for GPUs over TPUs.

### 

### Performance Evaluation

Due to the subjective nature of our project it is hard to perform an objective evaluation and so we couldn’t use methods such as confusion matrices or ROC curves. Therefore we opted for a human based approach. We gathered some participants and gave them 3 different sets of lyrics. One of them was real lyrics, one one lyrics generated using BERT and the other lyrics generated using GPT-2. We found that while it was possible for our participants to distinguish between the human generated lyrics and AI generated lyrics, they will still be impressed by the capabilities of our system. We then asked them what they thought was better, GPT-2 or BERT, we found that most of them chose GPT-2 over BERT. GPT-2 seemed to be more consistent at making decent songs but sometimes BERT did produce better outputs.

While BERT was less consistent, it has the advantage that we were able to implement rhyming, that is, a higher weight is assigned to pairs of lyrics that rhyme together. This resulted in songs that “sounded” more like actual lyrics. We found that with GPT-2 it was much less likely to rhyme and instead it seems to assign higher weight to repetition. Repetition with GPT-2 actually became an issue at one point, especially in the pop genre where there is a lot of repetition, this can be solved by increasing the temperature value greater than 0.7. The higher this value, the more of the original pretrained model it remembers.

The advantage GPT-2 had is that unlike BERT it completely generated its own unique lyrics whereas with BERT it is reusing lyrics. This makes the songs much more unique from GPT-2 and we were impressed by how it was able to reproduce the structure of the language. However the problem is that not all of these lyrics made too much sense. This could have been improved if we had more lyrics data to use in fine tuning and also if we had access to the larger GPT-2 model. But still the results were impressive.

## Bibliography

[*AZLyrics song lyrics*. (n.d.). Retrieved March 23, 2020, from](http://paperpile.com/b/LSeYsr/JKMm) <https://kaggle.com/albertsuarez/azlyrics>

[Brownlee, J. (2017, July 2). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning - Machine Learning Mastery*. Machine Learning Mastery.](http://paperpile.com/b/LSeYsr/M3wB) <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

[fedelopez. (n.d.). *fedelopez77/langdetect*. GitHub. Retrieved March 23, 2020, from](http://paperpile.com/b/LSeYsr/LqKu) <https://github.com/fedelopez77/langdetect>

[Woolf, M. (2019, September 4). *How To Make Custom AI-Generated Text With GPT-2*. Max Woolf’s Blog.](http://paperpile.com/b/LSeYsr/Xw3z) <https://minimaxir.com/2019/09/howto-gpt2/>