**Machine Learning Algorithms and Heuristics – Assignment Report**

Using machine learning to generate and create forms of art is an extremely interesting way to explore the field of machine learning and to discover what computers are capable of. Our aim is to train a machine to generate its own lyrics to a song only being prompted by a genre and the first line.

One such models is BERT’s (Bidirectional Encoder Representations from Transformer) which is a pre-trained NLP (Natural Language Processing) technique developed by Google. The other model we are using is known as GPT-2 (Generative Pretrained Transformer 2), which is used to predict the next token in a sequence of tokens via unsupervised machine learning.

Both models use unsupervised machine learning, meaning the algorithm itself infers patterns from a supplied dataset. This is extremely useful in this project as we are trying to train a machine to generate its own lyrics when prompted.

A potential issue we believe we will face in this project is the struggle to find hardware powerful enough to train data using these two models. Due to both models being incredibly large, especially BERTs we believe we might not be able to finetune them as much as we would like. Generated data is also relatively hard to evaluate in the same way you would evaluate numerical data, due to opinions of lyrics being completely subjective. We will attempt to evaluate the model’s capabilities at generating lyrics by comparing both BERTs and GPT-2s outputs.

**Development Pipeline: Data Collection**

Originally, we had planned to create our own dataset by scraping known lyric websites, however we encountered several issues when trying to do it ourselves. One issue was that a lot of the websites we planned on using prevent users scraping them by tracking how much traffic is coming from a single IP. This prevented us from downloading data from most of the websites we tried as our bot was continuously spotted. Thankfully, there was already a lyric dataset on Kaggle that was open-source and suited our task extremely well. A link to the dataset can be found in the bibliography. The data was download from a famous lyric site (AZLyrics) and was stored as a CSV file containing the Artist name, Artist URL, Song name, Song URL and the lyrics themselves. The data itself was generally clean however we noticed that there were a few songs that were either not predominantly in English or contained words that could be deemed racially or otherwise offensive. Thankfully Google has a language detection library, Langdetect, to check what language given input is. Using this library, a python function was created to go through the CSV removing any songs that weren’t in English. We also manually edited the CSV to remove any problematic words from the dataset to prevent our AI producing unethical lyrics.

As an addition to our original plan, we decided to add Genres to the dataset as we wished to generate lyrics depending on a given genre. Pop songs have very different lyrics to say Rock songs and thus we wanted our AI to know the difference between genres. This was initially an issue however as the dataset we were using did not contain Genres. Spending the next few days researching how to get around the scraping issue, we realised we could pay a service to rotate IP’s thus preventing the site from catching one IP generating exceptional amounts of traffic. Using AZLyrics again for prosperity, we downloaded the genre relating to each Artist and stored this into a CSV. The two CSV datasets were then combined, assigning the correct genre to the correct Artist using a python loop, therefore allowing us to include genres in our training dataset.

**Development Pipeline: Feature Analysis**

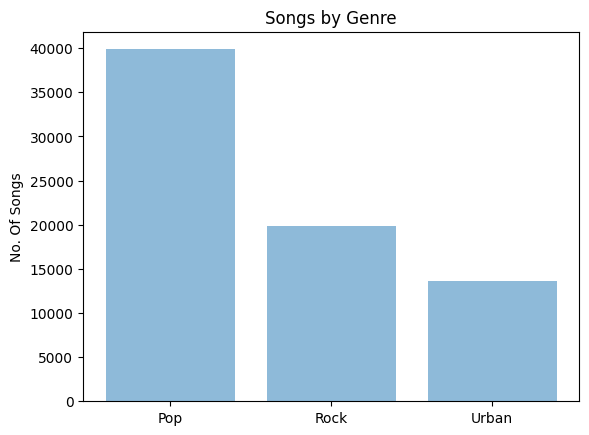
Our dataset consisted of over 72,000 songs, after cleaning, which is enough to generate an effective song generator. Unsurprisingly Pop was the most common genre, followed by Rock then Urban.

Figure - Genre Distribution of our Dataset

We initially wanted to use more than 3 genres, however due to the scraping issues we faced we could only use AZLyrics, which only has those 3 genres in its dataset. It was also interesting to see which words were the most common in our dataset, as this could be compared to what lyrics are commonly generated by both models. To do this we generated a word cloud.

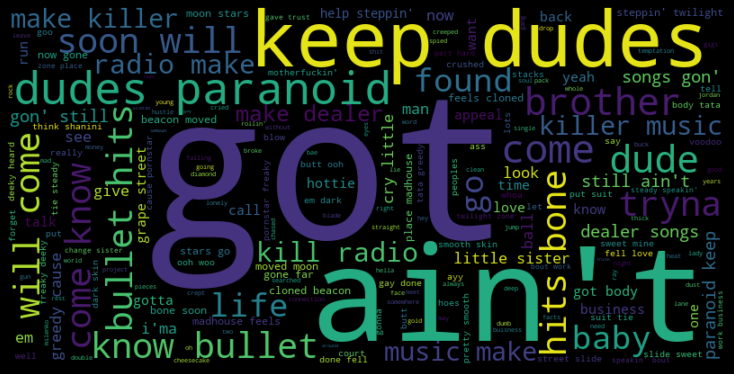


Figure - Word Cloud for Most Common Word in Our Dataset

Some of these words are surprising, but this is perhaps due to how many Urban and Pop songs featured in the dataset. As this project only focuses on the lyrics and the genre, we decided it wasn’t necessary to analyse the artists themselves.

**Development Pipeline: Data Learning**

All our code was done via Google Colab, a free cloud service created by Google specifically for machine learning. This decision came about when we realised that neither of our personal computers were powerful enough to run BERTs or GPT-2.

We decided to split the models up between us, I predominantly focused on finetuning GPT-2 to use our dataset and Matthew on using our dataset in conjunction with BERTs to generate a verse. Although we mainly focused on our chosen one, we both had a large amount of input in both models.

**GPT-2**

For GPT-2 we finetuned it to use our lyrics dataset using a smaller version of GPT-2 known as GPT-2 Simple. To do this we firstly had to load one of 4 GPT models from small to extra-large. Upon further research of GPT-2 Simple we discovered it wasn’t possible to finetune models larger than the medium one thus medium was best suited for our purposes. An issue we faced with using GPT-2-Simple was that it requires a txt to finetune whereas our dataset was a CSV. It also wouldn’t allow us to use a single dataset containing all the genres due to how GPT works. To fix this we created a separate CSV for each genre and converted these to .txt files using Excel.

After installing TensorFlow, gpt-2-simple and datetime and with our 3 txt datasets we were ready to begin training. GPT-2 simple defaults to using the Adam optimizer, which is a common optimizer that combines the advantages of both Adagrad and RMSprop. ("Adam — latest trends in deep learning optimization.", 2020). As you can see from the graph below, Adams performance can be extremely fast in terms of training speed.

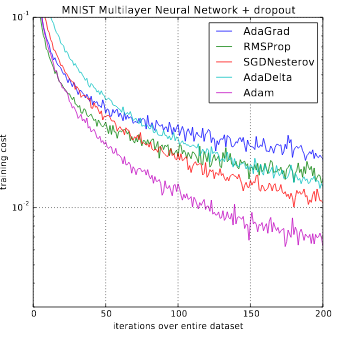


Figure - Comparison of optimizers on the MNIST Dataset

"Adam — latest trends in deep learning optimization.", 2020

One of the big reasons Adam is so powerful is that it adapts learning rate methods to find the most suitable learning rate for each given parameter. This is great for our lyrical dataset due to the variety of lyrics it contains. To train the 3 new datasets we created, one for each genre, we set up 3 finetuning algorithms, all using the same training parameters, the only difference being the dataset used. We chose to iterate for 1000 steps, as this is what the developer of GPT-2 Simple recommends. (Woolf, 2020) This is to prevent both overfitting of the data and to allow the model to learn our data well enough to present meaningful results. When finetuning we saved the model after every 500 steps, this was to prevent losing any progress we made during training. We also included a step to show a sample verse it could generate after every 100 steps to track the differences as it learned our dataset.

Testing GPT-2 with our entire dataset at once did cause an Out of Memory Issue due to the size of our combined dataset. The notebook simply didn’t have access to enough RAM to train using our entire dataset, which didn’t have much of an effect due to how we split up the data depending on genre, but it was interesting to test. We also experimented with training it on both the GPU and the provided TPU, noticing a rapid speed up when using the GPU over the TPU. Each step took over 12x longer when using the TPU which was extremely interesting to see. This is perhaps due to GPT-2 Simples optimization for GPU training. Once the training was complete, a model was saved for each genre, thus allowing us to generate songs for a genre of our choosing. We measured the time taken per every 50 steps using the python datetime library.

**BERTs**

BERTs can be considered one of the biggest and best techniques to use when it comes to generating and predicting text. Thanks to Google, BERTs is completely open source allowing anyone to generate their own text-based AI. BERTs converts sequences into pairs of tokens which are embedded into vectors, it does this by reading the entire sequence unlike other transformers as it is bidirectional. BERTs was our first choice when it came to which technique to use, however it wasn’t without its issues. Due to BERT’s being exceptionally powerful and trained on a large set of data, to finetune it requires an extremely powerful machine, something we struggled to access. Due to this, we were not capable of finetuning BERTs in the typical way. BERT consists of 4 different model sizes, all 4 however seemed to be too large for us to finetune. When we attempted to use BERT-Base Cased, the smallest of the models, our computers ran out of memory as it was using over 32GB’s of RAM. Neither using a GPU or TPU seemed to circumvent this issue as they too rapidly ran out of memory. The only solution to this we discovered was changing the amount of lines the machine could access to generate its verse was 100. Both BERTs and GPT-2 use tokenizers to generate ID’s for the words by splitting the lyrics into individual words. This allows the network to discover patterns it wouldn’t be able to due to a computer’s preference for numbers over words. This performance ceiling could also have been caused due to us choosing to use the BERTs word rhyming library.

**Development Pipeline: Evaluation**

Due to text-based projects being completely open to interpretation we decided to evaluate our outcome by comparing the two models we used and the songs they generated. Both used different methods to generate the lyrics and thus we found it interesting to compare the differences. Doing this we realised that unless the temperature (Higher temperature means the network has more control over what it generates) was higher than 0.7 when generating using GPT-2, repetition became an issue. This is due to how a song is structured, and the network attempting to replicate this. This was especially common in Pop songs where the structure is typically unique verses with a repeated chorus.

When it comes to GPT-2’s generation, it’s lyrics are entirely generated by the network itself. This is exceptionally different to how BERTs reuses lyrics from the dataset and combines different ones to make a verse. GPT-2 uses the words and the song structure to generate a song, as opposed to reusing lyrics from the dataset, thus generating a much more unique song than BERTs. BERTs however has a method of checking for rhymes using tokens, therefore the songs generated by BERTs consistently had a rhyming scheme, which the GPT-2 songs lacked. It could then be argued that songs generated by BERTs are perhaps closer to human created lyrics, but not all songs have a rhyming scheme.

To evaluate the model’s outputs, we also asked a few friends what their opinions on the songs outputted were and whether they could pass as human written songs. It was quite clear to tell which songs were generated by our network and a majority of those we asked agreed. They did however say they were surprised at both BERTs and GPT-2’s ability to generate songs only given a prompt. We decided to ask the participants to rank the model’s songs, with most ranking GPT-2 above BERTs.

Overall, we believe the project was a success and we are happy with both model’s song generation outputs. We would do some tweaking If we were to release this publicly as sometimes it takes multiple generations to produce a consistent verse, but even humans sometimes struggle to write those.

**Bibliography**

*Adam — latest trends in deep learning optimization.* Medium. (2020). Retrieved 15th March 2020, from <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>.

Woolf, M. (2020). *How To Make Custom AI-Generated Text With GPT-2*. Max Woolf's Blog. Retrieved 15th March 2020, from <https://minimaxir.com/2019/09/howto-gpt2/>

The Dataset we modified for our training - <https://www.kaggle.com/albertsuarez/azlyrics>