Exploring Music Genre Lyrics With Transformers

Idan Kringel (ID. 208390831), Itai Kaplan (ID. 205411739)

Submitted as final project report for the NLP course, IDC, 2022

1 Introduction

Music is very close to our hearts, and for that reason we decided to focus our project on music, and specifically music lyrics. In this paper we explore approaches for generating song lyrics given a genre and a title, and explore different language generation techniques using transformers.

We focused our efforts in producing songs that make sense, fit the genre and are correlated to the song title, and for that matter we tried various approaches, using HuggingFace pre-trained GPT-2 model, which we fine-tuned on dataset that we scraped ourselves.

1.1 Related works

Text generation - There are numerous works on text generation using transformers, and specifically GPT-2. Among them there are also some works on generating song lyrics. One example we found for a model that generates song lyrics is HuggingArtists [1] by Aleksey Korshuk. In this work the model was trained for text generation, on a specific artist, and the model generates results that resemble the specific artist.

The work has awesome results, but our aim is to generate lyrics that are genre specific and not artist specific, and we wish to generate lyrics that are related to the title. Those differences pushed us to take different approaches - different dataset, different input to the model, and different training procedure.

Conditional text generation - There are several works on conditional text generation using GPT-2 [2]. It seems our case is a bit challenging, as the generated text is relatively long and consists of specific structure. We couldn't find any satisfying one on conditional text generation for song lyrics.

PPLM - A work by Uber Research, "Plug And Play Language Models: A Simple Approach To Controlled Text Generation" [3], discusses the difficulty of controlling the attributes of text generated through large scale language models and proposed an approach which we integrated in our model in order to steer the model to create song lyrics that fit our given title.

2 Solution

2.1 General approach

We chose to explore 5 different genres - Hip-hop, Metal, Rock, Pop, and Reggae. Our assumption was that training on all songs of several bands for each genre would generalize well. For that reason we scraped by ourselves 5 datasets for the 5 chosen genres from several chosen bands for each genre.

We decided to use an auto-regressive language model for generating the song lyrics. Auto regressive models are a natural choice for text generation, and GPT-2 is a state of the art auto-regressive model that trained on a huge corpus and achieved amazing results. For that reason, we fine-tuned a HuggingFace GPT-2 model with language model head, for each genre dataset.

We structured the data fed to the model in order to try to fine tune the model for conditional text generation - as GPT-2 model with a language model heads is generating text based on previous words, we want to make the model to generate song lyrics based on a previous structured section that represents the title.

We later added a PPLM component to steer the lyrics generation to better match to music title, using the title as a bag-of-words. PPLM stands for Plug and Play Language Model, and as the name implies, it is a component that is "plugged" on a language model such as GPT-2 and is intended to steer the text generation.

In short, the PPLM tries to maximize $P(x|a)$

Where x is the generated text and a is some attribute, in our case, a bag of words. The component is trained on our frozen model in a technique that is using the fact that $P(x|a) \propto P(a|x)P(x)$

So it trains a discriminator for $P(a|x)$

and steers the text generation by using both the result of the language model and the discriminator. A broader explanation can be found in the original paper.

2.2 Design

We scraped the songs ourselves from lyricsfreak.com which had an easy HTML structuring that enabled us to scrape songs pretty easily. After choosing musical genres, we chose multiple bands to represent each genre and scraped all their songs from the website. We did very simple clean-up code for the data in the scraping module, and enhanced that with more clean-up in the data-set building pipeline in our Jupyter notebook. We saved the song lyrics in a CSV file that consist two column - title and lyrics. We had to deal with inconsistent structures of songs - some songs had more new-lines between each phrase then others, and some songs had markings for song parts (Verse / Chorus / etc) while others didn't. Also some songs had parts marked with parenthesis while others didn't. We decided to get rid of all of the above, in order to create a unified structure for all songs to make training easier.

As mentioned, for each genre we decided to scrape all songs of several representing bands. This method turned out good and generalized pretty well, but the thing we didn't take into consideration is that some bands/artists had songs in languages other than English, which we had to clear up from out data.

The genres we decided to work with are - Pop, Hip-hop, Rock, Metal and Reggae. In the way explained above we created 5 different data-sets - one for each genre.

We used pre-trained medium HuggingFace GPT-2 models with a language model head and a pre-trained tokenizer. We used a pytorch Dataloader in order to initialize and iterate the data-sets, and added further preprocessing to the data in the initialization.

We trained each model using AdamW optimizer, for 3-5 epochs, and also added linear scheduler with a warmup phase, which schedules the learning rate decay. Some models were overfit after 5 epochs and tend to produce words that were seen more then others in that genre very frequently. After training each model we saved it and cleared it from memory since having multiple large models such as GPT-2 in memory causes an Out-Of-Memory problems

We initially got non-satisfying results - the lyrics contained words that seemed as they were Gibberish, and there was very little connection between each phrase of the songs that we generated.

After better cleanup of our data, and hyper-parameters tuning, we were able to generate better lyrics. Here is example of "Stairway to Heaven" generated by the hip-hop model using these steps:

Yo!

D-assup with me

Yall love me!

Im going you say

Oh

Look at it!

Got me

We have it, I know that I love me! I love it this joint, look

Uh, the game, we want it,

Put it with my world

Your, is, and I just like this little doubt it?

We can see that the lyrics in this step make sense, for the most part, and seem strongly correlated to the hip-hop genre, as we expected from a model that trained using only hip-hop and rap songs. The obvious problem here - there doesn't seem to be a correlation between the song title and the song lyrics. It's possible that more data, resources and training time would have done the trick, yet since we have limited resources and time, we had to find another approach.

Other than further tuning the hyper-parameters, we have tried various approaches to force the lyrics to be correlated to the title, or force the lyrics to include words from the title.

First approach - emphasize the title when it appears in the song itself (case insensitive), by wrapping it with special tokens. This approach didn't yield good results - the generated text consisted of the special tokens in random places, or alternatively wrapping song titles from the data set. We assume the cause for the failure of this approach is lack of data and training time for the model to "grasp" this pattern.

Second approach - replace the title as it appears in the lyrics with a special token, then replace back the title token with the title itself in the generated text. This approach felt "hacky" but yielded interesting results, for example, Stairway to Heaven with pop model yielded -

Boy why Ive

I can let me dont hearin at time

Ive know you

Stairway to Heaven

Stairway to Heaven

You cant win

And no time

The problem with this approach - the location of the title in the generated songs didn't always match the parts of speech in the title, as would be expected, and sometimes the text appeared random as the language model only generated the location of the title, and not the title itself.

Final approach - PPLM. We understood we need to somehow steer the generation algorithm to generate words that are in the title of that are correlated to the title. Fortunately enough, the paper about PPLM discusses the difficulty of controlling the attributes of text generated through large scale language models, and proposes and approach to tackle the problem.

We used the PPLM component as designed by Uber Research. The component had to be trained separately for each song/topic, however fortunately enough the training requires a relatively short time and can be done when we generate a sample song.

We had to change the PPLM code slightly to support additional tokens, and trained it on a bag-of-words of the title. When the title consisted of stop-words ('to', 'the', etc) or personal pronouns ('I', 'he', etc) it seemed to steer generation entirely to that stop-word. For that reason we removed stop-words and pronouns from the bag-of-words, either manually or using NLTK tokenizer, which led us to better results. The PPLM is very sensitive to learning rate so we had to be careful - we wanted the model to use the words of the title, but not too much. Results (some introduced in the 'Experimental Results' section) were interesting and did consist of words of the song title as we tried to achieve, though not necessarily of the whole title.

We also tried to use the PPLM gneration with a bag of words of a song from certain genre and observe the result on another genre model. For example, using the hip-hop model and a bag-of-words of the song stairway to heaven. We got the best results in this approach when we left only nouns in the bag-of-words.

3 Experimental results

We got interesting results from the models. We noted that when hving stop-words such as "to" and "the" and using PPLM, the generation algorithm generated mainly those words. The same applies for pronouns, like "I" and "he", which generated mainly those pronouns. Therefore we removed stop-words and pronouns from the input and entered the title as a bag of words that consists of mainly of verbs and nouns.

We got the next song section from the pop model with the title "Run To The Hills" -

Im not get on your side for all in the way

Im in my hills

Im getting it up

Im gonna run and run

Im gonna run and

Feel my heart in the run my life was

We also get the next section from the metal model with the song title "Wrecking Ball":

I dont know that I can see

I want to see what is a way of the world and I know Im gone

I dont have a world

I am a dream

Ive got to see it

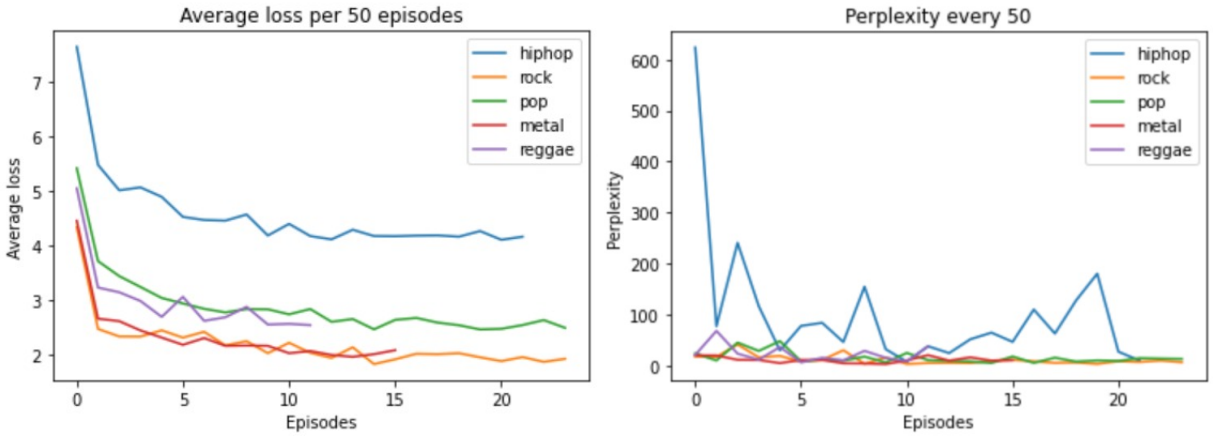
Im a dream I want to get it all you can know

I am no ball

I want to be real

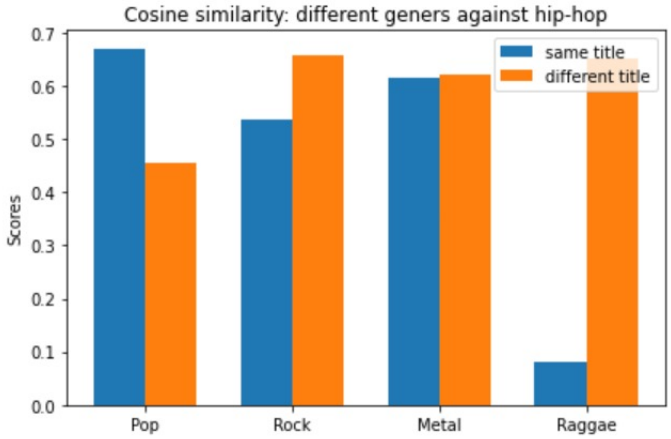
Seems like our model wants to break out of the Matrix! Amusing results aside, we got interesting outputs from our models. We wish to analyze them by other metrics as well.

We decided to do so by using three different methods, first, as a sanity check, we wanted to make sure the training is efficient, by observing whether the loss and perplexity of our models are decreasing. We plotted graphs showing the average loss of the model every 50 iterations, as well as batch perplexity every 50 iterations, and indeed as can be seen in Figure [3], the loss and the perplexity in each model were decreasing. Perplexity in the hip-hop model is more noisy then the others, we assume it is related to the increased number of slang words in hip-hop songs.



Second, we wanted to check how similar are songs across different genres. For that we used Cosine similarity, which is a measure for similarity between two normalized vectors. We tested the cosine similarity between several generated songs, two from each genre. We expected to see high similarity between songs from the same or "close" genres (rock and metal for instance), and low similarity between "far" genres (metal and reggae). For the purpose of this comparison we had to find a way to embed a whole song. For that we used a pre-trained Universal Sentence Encoder by Google. On Figure [3] you can see the results of the cosine scores of every two songs with respect to a hip-hop generated song, and on Figure [1] you can view a table measuring all possible cosine results between two songs.

We also tested our generated songs similarity with real songs, in the hope that a generated song with some title will be as similar as possible to the real song with the given title. For that we used songs that are not in our training data, in order to make the model create a brand new song. Results for this experiment were far from perfect, especially in genres with little data, and can be seen in Figure [2],



	Hiphop title 1	Hiphop title 2	Pop title 1	Pop title 2	Rock title 1	Rock title 2	Metal title 1	Metal title 2	Raggae title 1	Raggae title 2
Hiphop title 1	1.000000	0.733657	0.671043	0.454392	0.535874	0.656859	0.613963	0.619837	0.082217	0.650906
Hiphop title 2	0.733657	1.000000	0.663693	0.434340	0.477527	0.598044	0.487460	0.641406	0.229049	0.531143
Pop title 1	0.671043	0.663693	1.000000	0.505663	0.332158	0.720787	0.434583	0.746987	0.199062	0.623911
Pop title 2	0.454392	0.434340	0.505663	1.000000	0.325579	0.522256	0.403279	0.417626	0.096067	0.585911
Rock title 1	0.535874	0.477527	0.332158	0.325579	1.000000	0.484318	0.584743	0.441273	0.086552	0.428768
Rock title 2	0.656859	0.598044	0.720787	0.522256	0.484318	1.000000	0.548003	0.763380	0.208984	0.670855
Metal title 1	0.613963	0.487460	0.434583	0.403279	0.584743	0.548003	1.000000	0.494891	0.125154	0.582291
Metal title 2	0.619837	0.641406	0.746987	0.417626	0.441273	0.763380	0.494891	1.000000	0.211476	0.620479
Raggae title 1	0.082217	0.229049	0.199062	0.096067	0.086552	0.208984	0.125154	0.211476	1.000000	0.142381
Raggae title 2	0.650906	0.531143	0.623911	0.585911	0.428768	0.670855	0.582291	0.620479	0.142381	1.000000

Figure 1: Cosine similarity between every two generated songs

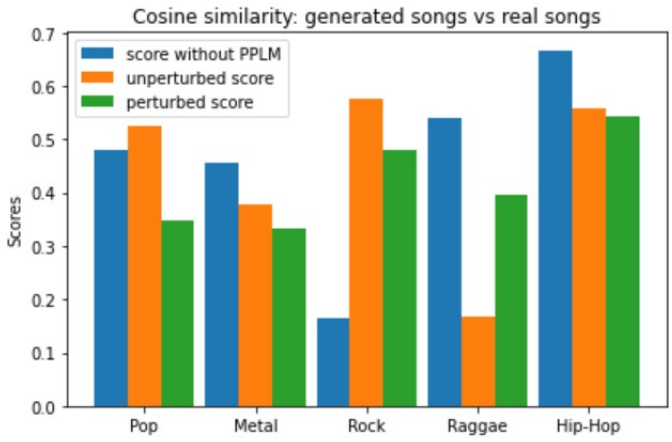


Figure 2: Cosine similarity between real songs vs generated ones

4 Discussion

In this work we described and evaluated approaches for music lyrics generation for different genres and titles. It seems as though the GPT-2 conditional text generation approach requires more data and longer fine tuning - after several epochs the models seemed to generate mainly words that appeared frequently in their input, such as pronouns ('I' / 'Im') and words that seemed to appear a lot in that genre ('yeah' in rock, 'love' in pop, 'death' in metal and some non politically correct words in hip-hop). We have seen how important it is to clean the data and have an accurate data-set, as it seemed that some songs that contained other languages and esoteric slang words had bad implications on the performance of the model.

By fine tuning multiple models, one for each genre, we were able to observe differences between the genres language models, both by looking at the plain-text and by using more advanced methods. The models can generate unique song lyrics that were correlated to each genre.

We also explored the work of PPLM by Uber Research on our models and reached interesting results, that incorporated our requested words and also possibly a slight shift of the lyrics towards a different meaning, as observed in the cosine similarity metrics. This approach also had its problems - when there were certain parts of speech in the input bag-of-words it made the model generate almost

exclusively this word, and also when not being careful the model was steered to generate only words from the bag-of-words. This work seems to achieve the best results with a bag of words that consists exclusively of nouns and lemmatized verbs.

We expected that an embedding of the songs from distinct genres will represent to some extent the 'distance' between those genres. The results we got from experimenting with song embeddings were far from perfect, which might be related to the embedding method we used, or maybe lack of data and training time.

To conclude, we got lyrics that make sense for the most part and that fit the genre from the different approaches we used, and the results would probably be better with larger amount of data and training time. We were able to get results which are correlated to the title mostly by manipulating the text generation itself.

It will be interesting to see how using newer state of the art models such as GPT-3 would have affected this task.

5 Code

Our code is provided in the following link: https://github.com/idankri/FinalProjectNLP

References

[1] Aleksey Korshuk. "HuggingArtists". In: *Github* (2021). URL: https://github.com/AlekseyKorshuk/huggingartists.

[2] Ivan Lai. "Conditional Text Generation". In: *Towards Data Science* (2021). URL: https://towardsdatascience.com/conditional-text-generation-by-fine-tuning-gpt-2-11c1a9fc639d.

[3] Uber Research. "Plug and Play Language Models: A Simple Approach to Controlled Text Generation". In: *Arxiv* (2020). URL: https://arxiv.org/abs/1912.02164.