Rhyming Lyrics Generator

Capstone Project Report

Shuangmu Peng
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024
pengshuangmu@ucla.edu

March 17, 2021

Abstract

Natural language generation is a rapidly developing field. Many of the applications of the language models are focused on practical Natural Language Processing tasks such as question answering and auto-completion of search terms. In this capstone project, however, I seek to explore how existing language models, from the simplest statistical model n-gram, to state-of-the-art models such as GPT-2, perform on creative content generation. Specifically, I want to use NLG to let machine write rhyming lyrics, for a simple motivation: wouldn't it be cool if computers could sing and even rap? In this project, I will focus on rhyming lyrics generation with different language models. I will use rap lyrics as the major dataset for experiments, compare the performance of models, and finally show the general text generation capability of the best model on different genres of lyrics.

1 Introduction

Music has long been an effective way to communicate to the masses, and lyrics have played a massive role in delivering this communication. Natural language generation tasks, on the other hand, seek to mimic human's capability to use language in a creative way. In our scenario, we aim to let the machine generate lyrics like human song writers. While human song writers write with efforts and talents, the machine uses data and calculations.

During the lyrics writing process, rhyming also plays a crucial role. In most cases, rhyming lyrics sound more melodious and rhythmic. Even one genre of music, rap, is founded on rhyming lyrics. Rap is distinguished from other music genres by the formal structure present in rap lyrics, which makes the lyrics rhyme well and hence provides better flow to the music. Literature professor Adam Bradley compares rap with popular music and traditional poetry, stating that while popular lyrics lack much of the formal structure of literary verse, rap crafts "intricate structures of sound and rhyme, creating some of the most scrupulously formal poetry composed today." [1].

In this project[1], I will explore the machine's capability in rhyming lyrics writing. Since rap music lyrics are usually more complicated and have hard constraints on rhymes, I will use rap music as the major dataset for the experiments. After determining the best model for rap lyrics generation, I will demonstrate its general lyrics generation ability on dataset of other music genres. In the following section 2, I will give an introduction of the models I used for lyrics generation tests. Section 3 will be an overview of my dataset. Section 4 will present descriptions, examples and analysis of forward open-ended lyrics generation capability of all the models. For section 5, I will demonstrate the rhyming lyrics generator I built, explain in detail how the rhyming lyrics are generated, and evaluate the lyrics through comparison. Section 6 will serve as a summary of the project and future works that can be done to improve the project.

[1]All codes for the project will be included in `https://github.com/SM-P/Capstone-Lyrics-Generator`.

2 Models

2.1 N-GRAM

An n-gram model is a type of probabilistic language model for predicting the next item in a sequence of a (n - 1) order Markov model form. The main idea of generating text using n-grams is to assume that the last word x^n of the n-gram can be inferred from the other words that appear in the same n-gram $x^{n-1}, x^{n-2}, \dots x$. With this assumption, we just look back for n-1 tokens to predict the next word instead of the whole sentence. When calculating the probability, we just apply a simple conditional probability rule:

$$P(X^{(t+1)}|X^t, ..., X^{(t-n+2)}) = \frac{P(X^{(t+1)}, X^t, ..., X^{(t-n+2)})}{P(X^t, ..., X^{(t-n+2)})} \tag{1}$$

After the probabilities are computed, we select the final word given all candidates. We can either simply choose the word with the highest conditional probability, or choose the word semi-randomly, so that the words that have a higher probability will have higher chances of being produced, while other words with lower probability still have a chance to be generated[2].

2.2 LSTM

A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. Recurrent networks can in principle use their feedback connections to store representations of recent input events in form of activations with "short-term memory", as opposed to "long-term memory" embodied by slowly changing weights.

Long short-term memory (LSTM) is a successor architecture to the classic recurrent neural network. LSTMs were designed to overcome the vanishing gradient problem that plagued traditional recurrent neural networks, where gradient-based paramater updates became small enough to prevent further model training. By having the ability to retain and forget information, LSTMs are well-suited to modelling input sequences with long gaps between important events [3].

2.3 GPT-2

The transformer model is a deep learning model introduced in 2017. Like recurrent neural networks, transformers are designed to handle sequential data like natural language. However, unlike RNNs, Transformers do not require that the sequential data be processed in order.

Generative Pre-trained Transformer 2 (GPT-2) is a large unsupervised transformer-based language model that was designed to predict the next word, given the previous words, within some input text. Trained over 8 million web pages, GPT-2 has demonstrated state-of-the-art performance on many diverse tasks, such as translation and summarization. The model has been noted for its ability to adapt to the style and content of the input conditioning text [4].

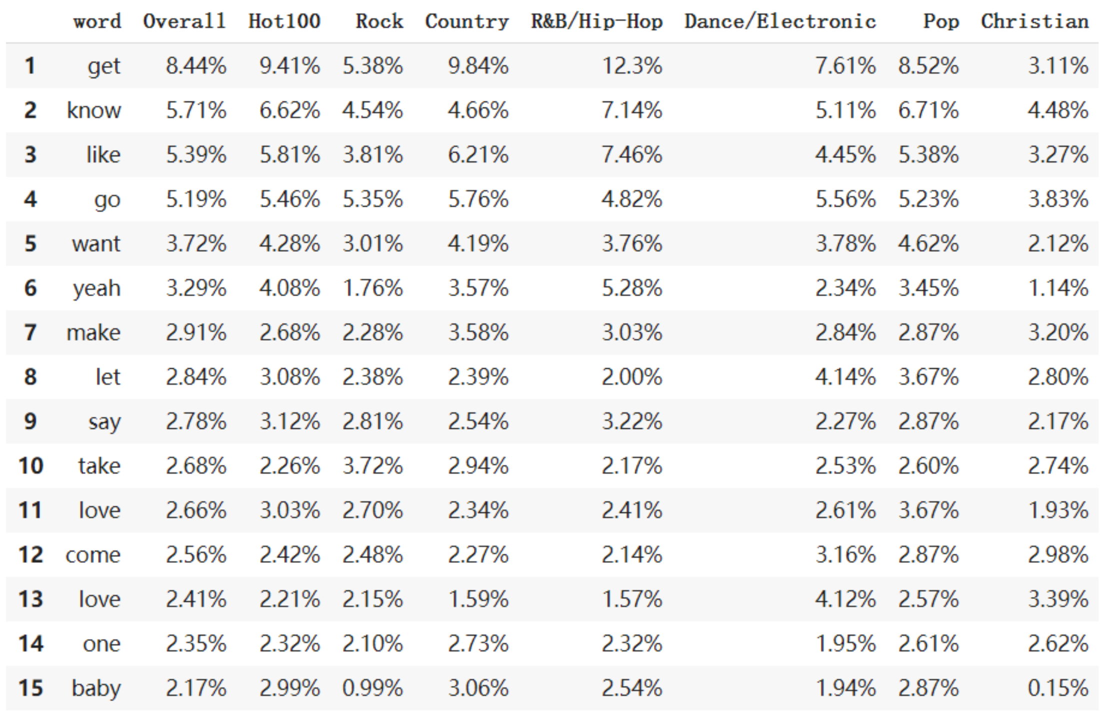
2.4 XL-NET

XL-net is a generalized autoregressive model based on the transformer architecture but it allows for bidirectional context language modeling through permutation language modeling, which means training on all or a subset of the possible permutations of the input sequence. It overcomes the limitations of previous bidirectional language through its autoregressive formulation[5].

2.5 BART

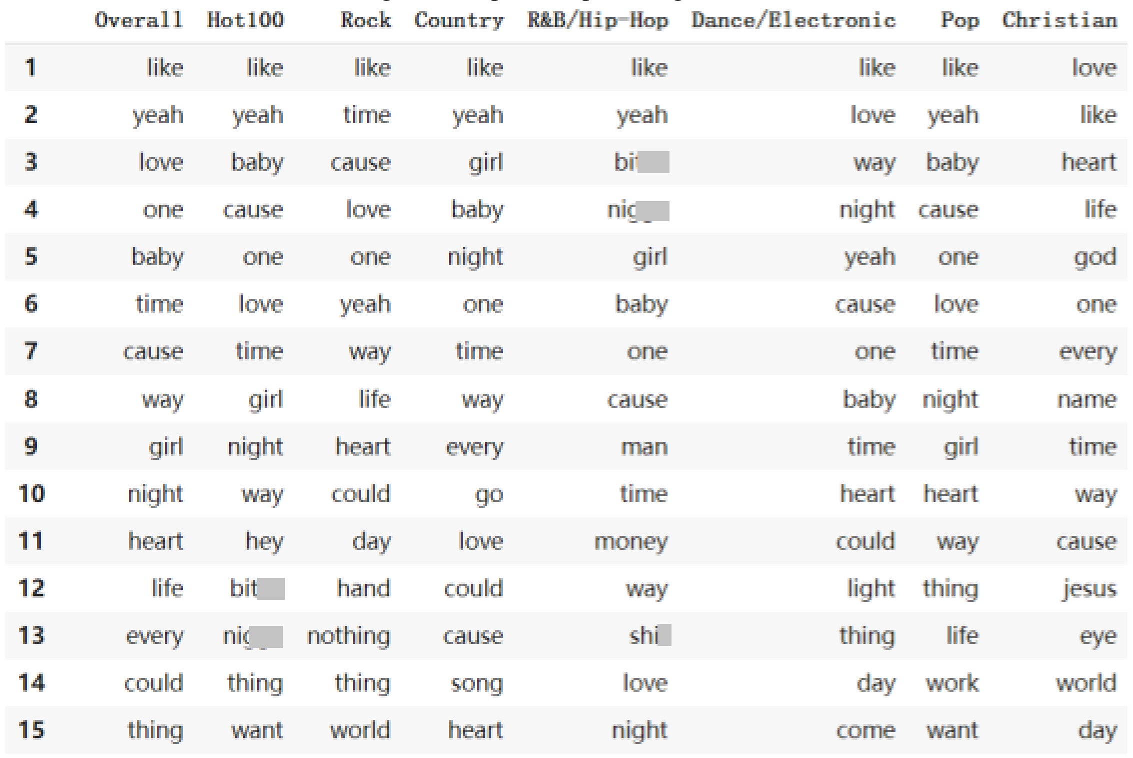
BART is a denoising autoencoder for pretraining sequence-to-sequence models. It is trained by corrupting text with an arbitrary noising function, and learning a model to reconstruct the original text. It is implemented as a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder.

While BART has similar features with BERT such as its bidirectional encoder and with GPT such as autoregressive decoder, there are still differences. BERT uses an additional feed-forward network before word prediction, which BART does not. In BART, each layer of the decoder additionally performs cross-attention over the final hidden layer of the encoder (as in the transformer sequence-to-sequence model)[6].



	word	Overall	Hot100	Rock	Country	R&B/Hip-Hop	Dance/Electronic	Pop	Christian
1	get	8.44%	9.41%	5.38%	9.84%	12.3%	7.61%	8.52%	3.11%
2	know	5.71%	6.62%	4.54%	4.66%	7.14%	5.11%	6.71%	4.48%
3	like	5.39%	5.81%	3.81%	6.21%	7.46%	4.45%	5.38%	3.27%
4	go	5.19%	5.46%	5.35%	5.76%	4.82%	5.56%	5.23%	3.83%
5	want	3.72%	4.28%	3.01%	4.19%	3.76%	3.78%	4.62%	2.12%
6	yeah	3.29%	4.08%	1.76%	3.57%	5.28%	2.34%	3.45%	1.14%
7	make	2.91%	2.68%	2.28%	3.58%	3.03%	2.84%	2.87%	3.20%
8	let	2.84%	3.08%	2.38%	2.39%	2.00%	4.14%	3.67%	2.80%
9	say	2.78%	3.12%	2.81%	2.54%	3.22%	2.27%	2.87%	2.17%
10	take	2.68%	2.26%	3.72%	2.94%	2.17%	2.53%	2.60%	2.74%
11	love	2.66%	3.03%	2.70%	2.34%	2.41%	2.61%	3.67%	1.93%
12	come	2.56%	2.42%	2.48%	2.27%	2.14%	3.16%	2.87%	2.98%
13	love	2.41%	2.21%	2.15%	1.59%	1.57%	4.12%	2.57%	3.39%
14	one	2.35%	2.32%	2.10%	2.73%	2.32%	1.95%	2.61%	2.62%
15	baby	2.17%	2.99%	0.99%	3.06%	2.54%	1.94%	2.87%	0.15%

Figure 1: Top words percentage overall



	Overall	Hot100	Rock	Country	R&B/Hip-Hop	Dance/Electronic	Pop	Christian
1	like	like	like	like	like	like	like	love
2	yeah	yeah	time	yeah	yeah	love	yeah	like
3	love	baby	cause	girl	bi[redacted]	way	baby	heart
4	one	cause	love	baby	nig[redacted]	night	cause	life
5	baby	one	one	night	girl	yeah	one	god
6	time	love	yeah	one	baby	cause	love	one
7	cause	time	way	time	one	one	time	every
8	way	girl	life	way	cause	baby	night	name
9	girl	night	heart	every	man	time	girl	time
10	night	way	could	go	time	heart	heart	way
11	heart	hey	day	love	money	could	way	cause
12	life	bit[redacted]	hand	could	way	light	thing	jesus
13	every	nig[redacted]	nothing	cause	shi[redacted]	thing	life	eye
14	could	thing	thing	song	love	day	work	world
15	thing	want	world	heart	night	come	want	day

Figure 2: Top words by genres

3 Dataset

The dataset for the project is scrapped from the Billboard hot 100 songs from 2013 to 2017. In total there are about 3500 songs of genres such as pop, rock, country, hip-hop, Christian, dance or electronic music with an average length of 180.5 words. In figure 1 we can see a top words percentage in term of word frequencies in all genres. For each specific genre, the top words are shown in figure 2. As we can observe, the most common words overall are also quite frequent in each genre. Nevertheless, lyrics of different genres also have a tendency to use words of their relevant topic. For example: dance/electronic music uses 'night' a lot; Christian lyrics refers to 'god' more often than any other genre; Hip-Hop music uses curse words most.

The following is an example of rap lyrics data:
Look hold up, Where my cups at?
And my homie got the tree bout to puff that
Too much lean for the seeds gotta cut back
Lookin' on my seams, she can tell where the bucks at

For our lyrics generation task, there are a few potential problems in the dataset, such as garbages created by the data scrapper, that needed to be overcome with. Instead using the raw data, we will filter and pre-process the lyrics for the generation tasks.

4 Lyrics Generation Performance

In this section I present the performance comparison I made among different language generation models to show the advantage and the deficiency of each model. While n-gram and LSTM are completely trained with our dataset[9], the state-of-the-art models, GPT-2, XLNet, and BART, are pretrained and finetuned with techniques introduced or inspired by these tutorials [10][11][12]. All texts are generated forward from left to right without the consideration of rhyming features or other constraints.

4.1 Metrics

Given that the open-ended generation is not constrained to achieve any determined goal, we can see that automatic metrics are less suited to evaluate the general performance of our models. Rather, I will first evaluate the models' performances our models qualitatively according to Oliveira's evaluation metrics [7] for automatically generated poems since song lyrics are essentially poems as well. The metrics would best capture model performance to rank various qualitative parameters about each rap by scoring each parameter with a +1 if its undeniably present, a +.5 if its somewhat present, and +0 if its non-existent. The specific parameters are:

1. Meaningfulness: convey a conceptual message, which is meaningful under some interpretation.
2. Grammaticality: obey linguistic conventions prescribed by a given grammar and lexicon.
3. Style Matching: have a coherent song structure or style.
4. Poeticness: exhibit poetic features, in rap more of provocative/stimulating/interesting points.

After choosing models with acceptable performance, I will use automatic evaluation metric, BLEU, to qualitatively assess the models. BLEU is often used for evaluations of machine translations. In our scenario, however, BLEU scores are calculated for individual generated sentences by comparison between them and the original dataset. Those scores are then averaged over the whole corpus to reach an estimate of the generation's overall quality[8]. For machine translation, the specific metrics of BART scores are:

1. < 0.1: Almost useless.
2. 0.1 - 0.19: Hard to get the gist.
3. 0.2 - 0.29: The gist is clear, but has significant grammatical errors.
4. 0.3 - 0.4: Understandable to good translations.
5. 0.4 - 0.5: High quality translations.
6. 0.5 - 0.6: Very high quality, adequate, and fluent translations
7. > 0.6: Quality often better than human

4.2 High-level Evaluation

In the following subsections I will give a representative example data and generation for each of our chosen models, evaluate the quality of the generation, rate the model overall based on my parameters, describe the main challenge faced by each model, and provide a solution if possible.

4.2.1 N-GRAM

To demonstrate how a statistics based language model perform on lyrics generation, I include a n-gram model in the test. Basically it builds a probability dictionary and choose the most probable combinations when generating texts.

Example output with input *This rap*:

> *This rap is destroyed and ironic*
> *Life's go let's now, it solve problem no be follower, a in look wasted, get gotta*
> *I scars from return the with position f*****-up a without*
> *All clappin be won't why baby acres 40 a*

High-level evaluation:

1. No theme, +0
2. Grammatically incorrect, +0
3. No song structure, +0
4. No poeticness in any way, +0

Overall score: 0 / 4. As we can see, a statistics based language model completely fails in lyrics generation tasks. Compared with N-gram, we can see how great other models' generated lyrics improve in quality.

4.2.2 LSTM

The basic idea of the LSTM model is to train the RNN with many sequences of words and the target "next word". For our model, about 90 percent of the dataset goes to training set and the rest test set.

Example output with input *I am who i am* :

> *I am who i am like the vault and my friends*
> *You know what you heard that you're doing to you*
> *You ain't a shooter if 'm living it up in the hood*
> *You know that 'm a gangsta you ain't sayin nothing*

High-level evaluation:

1. An obscure theme about gangsters, +0.5
2. Grammatically correct in phrases but not in sentences, +0.5
3. A structure of a main sentence followed by a clause, yet not suitable for a song, +0.5
4. Not very interesting, overly repetitive +0

Overall score: 1.5 / 4. The main challenge faced by this model is to overcome repetition and to understand the global structure. While repetitiveness could be fixed for the LSTM by penalizing repetition with a custom loss function, the lack of a global structure seems like an inherent problem of the model.

4.2.3 GPT-2

To train the GPT-2 model, we first inject control code to the dataset and do the encodings. Then we pass our input text into the transformer model and train the model to get the text back as output. Thanks to its large scale pre-training, we expect GPT-2 to show advantages of the inherent grasp of the English language.

Example output with input *Roses are red*:

> *Roses are red and blue, you're in love*
> *Now just because we don't do it doesn't mean we don't care*

All night long I still love you
And even after you come home
Girl I'll still care about you

High-level evaluation:

1. A cogent theme about love throughout, +1
2. Grammatically correct, +1
3. Good love song structure, +1
4. First few lines are interesting, rest are repetitive, +.5

Overall score: 3.5 / 4. It can be seen from the example output that GPT-2 is capable of generating fluent text with a consistent theme throughout, the lines flowing with somewhat of a poetic style. When generating lengthy texts, however, the model may fall in a repetitive loop with only the first few lines consisting of novel output.

4.2.4 XLNet

For our XLNet model, we use XLNetForSequenceClassification as a sentence classifier. Rather than training every layer in a large model from scratch, we only need to train the top layer, with a bit of tweaking going on in the lower levels to accommodate to our task. I expected that the bi-directional context that XL-NET incorporates in its training would enable the model to create songs more fitting to the global context and having coherent song structure.

Example output with input *All my friends*:

All my friends are all the men who are bad
I don't think about the things that my friends are not bad
Or who I know I want to be
Or how I need to make things happen

High-level evaluation:

1. Somehow logical but meaningless as song lyrics, +0.5
2. Grammatically correct, +1
3. A structure of overly verbose sentences, +0.5
4. Not very interesting, overly repetitive +0

Overall score: 2 / 4. As a model more often used for text comprehension, XLNet does not show superiority in lyrics generation. In fact in other tests the model often falls into repetition. The structure of the generated texts is more prolonged sentences than lyrics. Yet one note-worthy thing is that the generated text maintains a logical in a good length, as seen in that the meaning of the first sentence is preserved in the second sentence. Still, the generated texts are ill-suited for song-lyrics.

4.2.5 BART

For our implementation of rap generation with BART, we pair the sentences in our training data with one another so that the previous sentence has connection with its next. When generating the lines after the first one, we add noise to the lines thus creating more variations in the outputs of the model.

Example output with input *I see the money*:

I see the money, I get it
*I don't give a f*** about it*
*I don't give a f***, I just give it*
I got the money, I get it

High-level evaluation:

1. A coherent theme, but partly due to repetitions, +.5
2. Grammatically correct, +1
3. Good structure for a rap song, +1

4. First lines are interesting and provocative but the rest not so much, +0.5

Overall score: 3 / 4. The model work pretty well when the number of lines to be generated is small. The generated lyrics are unexpectedly suited for a rap song thanks to a coherent theme and a certain level of repetitions. Yet if the desired number of outputs gets larger, the model loops over one sequence and the situation of repetition occurs more often. The lyrics also become less coherent among one another.

4.3 Automatic Evaluation

According to our qualitative assessment, GPT-2 and BART are probably the most suitable models for lyrics generation. Here, I compare them with the other state-of-the-art model XLNet with automatic evaluation. I will let each of the model generate 50 lines and use BLEU scores to quantify their performance. All their scores will be recorded in the plots from figure 3 to figure 5.

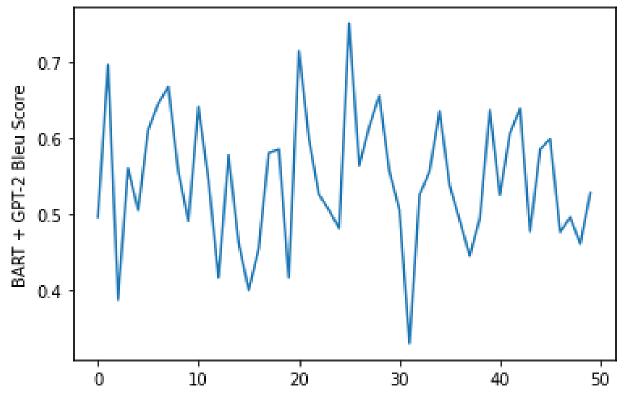


Figure 3: GPT-2 Average Score: 0.5963

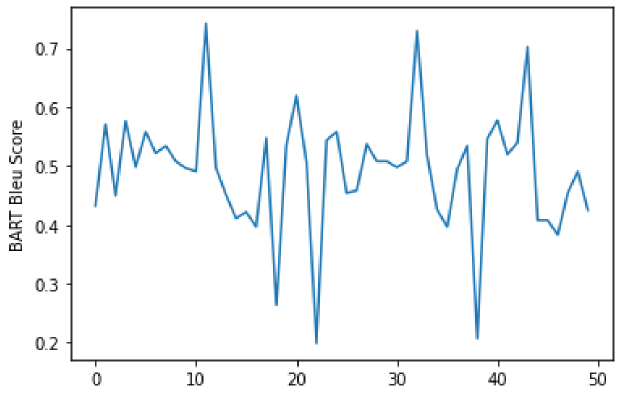


Figure 4: BART Average Score: 0.5752

As we can observe, almost all GPT-2's outputs are in the range of 0.4 to 0.7, from "good" to "exceptional". As for BART, although there are more aberrations, in general the generated texts have scores from 0.4 to 0.6. In contrast, XLNet's sentences are mostly from 0.35 to 0.45, from "understandable" to "good". There are also greater variations among the scores compared with GPT-2 and BART. In terms of average BLEU scores, the ranking among the 3 models would be GPT2>BART>XLNet, which should also be the ranking overall.

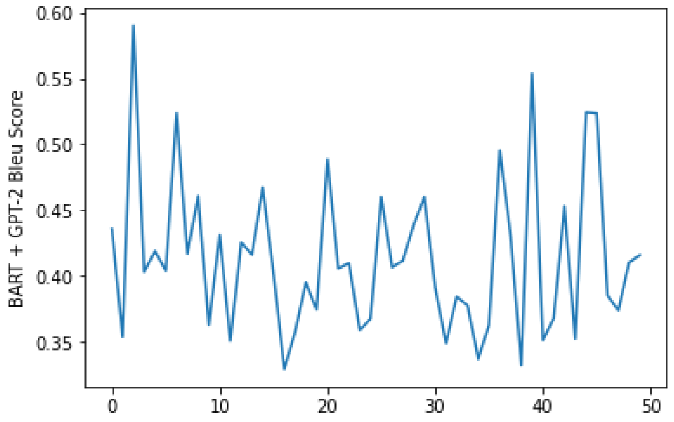


Figure 5: XLNet Average Score: 0.4079

4.4 Lyrics of other genres

Based on our previous tests, BART and GPT2 outperform other language models on the forward rap lyrics generation task. Their text generation capability is expected to function as well on other genres' lyrics. Here, I choose these 2 models to generate lyrics of other genres such as country and pop.

GPT-2 Pop Lyrics Example:

> *Why you act surprised*
> *Don't you know it's crazy*
> *You need a vacation*
> *Ain't no vacation for a girl like me*

GPT-2 Country Lyrics Example:

> *There's a truck in ainity*
> *With a rattlesnake high*
> *And I'm sitting on a bench right beside her*
> *Drawn to the silver spoon*

BART Pop Lyrics Example:

> *I want you to know*
> *That you're the only one*
> *E-mail me your name*
> *What you're doing to me*

BART Country Lyrics Example:

> *Country life is good*
> *I'm a country boy, country boy*
> *I've got a lot of things to say*
> *and it's hard to know where to start*

As we can see, both BART and GPT-2 managed to generate lyrics in other genres with some quality. The overall scores would not change a lot if our previous metrics are applied to these lyrics. One difference we can find between the two models, however, is that GPT-2 often generates lyrics less repetitive but less fluent, while BART more consistent in theme but more repetitive in vocabulary.

5 Rhyming Lyrics Generation

This section will be mainly about the generation of rhyming lyrics and again will mainly use the rap dataset. To generate rhyming lyrics we basically need a rhyme finder and a forward and backward lyrics generator, which I will explain in detail in the subsections.

5.1 Rhyme Search

To generate a sentence with an ending word that shares the same rhyme with the previous sentence, we first need to find such an ending word by its pronunciation pattern. For the rhyme search process, I used Carnegie Mellon University Pronouncing Dictionary, which contains over 134,000 English words and their pronunciations. The dictionary has a phoneme set of 39 phonemes, with vowels carrying a lexical stress marker. All words' pronunciations are composed by the 39 phonemes. For example, the pronunciation of "LOVE" is denoted by "L AH1 V", where "1" denotes stressing and "AH1" is vowel sound with the primary stress. To find a rhyming word we then simply search the dictionary for words whose pronunciations end exactly with AH1 V.

In order to avoid situations such as "superman" has the same rhyme with "man", a double check is implemented to ensure that one word is not a subset of the other. Meanwhile, since many rarely used words are also included in the dictionary, the language model may get stuck when generating sentences from rare words. Thus, a filter is implemented to exclude rare words from the candidate word pool.

5.2 Forward and Backward Generation

Since we have hard constraints on the ending word, the common practice of generating sentences from left to right cannot meet our requirement. It is theoretically possible to select a sentence that has a certain ending word from a very large candidate pool, yet such an approach is ineffective. Instead, I used the language models to predict the text in reverse from right to left, from a fixed rhyme word to the whole sentence. As language models essentially use connections among word tokens to predict texts, reversing the connections only influence the direction but not the quality of the predictions. Thus, I reverse all the word sequences, including line sequences, in the dataset, and then train the backward generation models with the reversed dataset to generate reversed lyrics. In the end, to get sentences in the normal direction all I need to do is to flip all the words the generated text .

As inspired by Mou's backward and forward language RNN model[13] for conditional text generation, I use BART model for forward generation and other models, including LSTM, XLNet and GPT-2, for backward generation. The BART model will take the responsibility in producing the first line and generating later lines based on past lines, while the backward models focus on corresponding rhyming sentence generation. For the seeding line to be fed to BART, I use rapid automatic keyword extraction algorithm[14] to extract keywords in the former lyrics, with an expectation that the model would have a better understanding on previous contents.

To conclude, a complete workflow of the rhyming lyrics generator is as the follows: after the forward generator yields a sentence, the rhyme searcher finds a rhyming word for the sentence's ending word. Starting from the rhyming word, the backward model generates a reversed sentence. The reversed sentence will then be added to the output lyrics. For later lines, the similar procedures are implemented, except for that the keywords in the output lyrics will be extracted and used as guiding sentences for later generation. In the end, all outputs are flipped and we have a normal song lyrics.

5.3 Results

In this section I present the actual performance comparisons of the forward and backward models for generating rhyming rap lyrics. For each of the four line verses, the second and the fourth sentences are generated by BART, while the first and the third sentences are generated by a corresponding backward model. The input sentence would actually be the fifth sentence in each verse in the normal direction. For evaluation, I will continue to use the metrics in the previous section, but will focus more on the sentences that are generated backwards.

5.3.1 BART + LSTM

Example with input *You and me forever* :

From wage alert mount down angeles degree
I'm the best of the best I can be
Champagne dudes wonderful for wherever
I'll be there for you forever

High-level evaluation:

1. No theme, +0
2. Despite the lack of meaning and some mistakes, grammar rules can still be found in LSTM sentences, +0.5
3. Almost no structure can be seen, +0
4. Not vert interesting due to the lack of meaning, +0

Overall score: 0.5 / 4. While LSTM manages to output some understandable texts when generating forward and without restrictions, it fails as a backward conditional lyrics generator. Its weakness in understanding the global context is further worsen by the requirement to generate from a given word. Although the words it chooses are somehow connected with one another, together they do not form a meaningful sentence. Still, BART manages to function by producing a sentence about "best" based on keywords "champagne" and "forever".

5.3.2 BART + XLNet

Example with input *You and me forever* :

The little bit lame
But I'm still the same

Smart and don clever
It's all about me forever

High-level evaluation:

1. Each sentence is meaningful but overall not cogent, +0.5
2. Most of the time grammatically correct, but mistakes exist, +0.5
3. Acceptable song structure, +1
4. Not very interesting as a song, +0

Overall score: 2 / 4. The sentences generated by XLNet are much more understandable than LSTM sentences. Some good thing we can also see is that thanks to the forward and backward model, the problem of repetitions that often appears in BART and XLNet models is solved. Still, the problem is that while each sentence in the lyrics has some meaning, together they do not form a fluent verse.

5.3.3 BART + GPT2

Example with input *You and me forever* :

I think I'ma kill you, have you with me, the same
I'm at the top of the game
I'm at your team whichever
You and I will be together forever

High-level evaluation:

1. A theme on "you and me" but a sudden change in attitude, +0.5
2. Mostly grammatically correct but minor mistakes exist, +0.5
3. Good song structure, +1
4. Very interesting, no repetition, +1

Overall score: 3 / 4. The lyrics seem OK in general and the rhymes have good coordination with the contents of the sentence. While the theme is maintained throughout, the models seem to have wrong focus when generating later lines, which results in the change of the attitude. Further improvements can be made to enhance the coherence among sentences.

5.4 Automatic Evaluation

According to our previous tests, the combination of BART and GPT-2 is the most promising model for rhyming lyrics generation. Here, we compare this model with the second best BART+XLNet using automatic evaluation. For each model, a total of 100 lines are generated, where 50 lines are generated by BART and the rest the corresponding backward model.

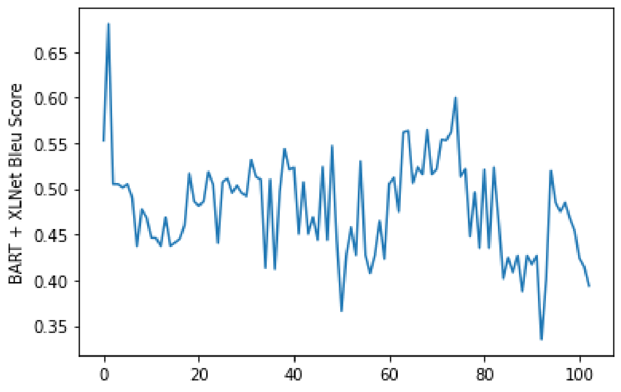


Figure 6: BART+GPT-2 Average Score: 0.4768

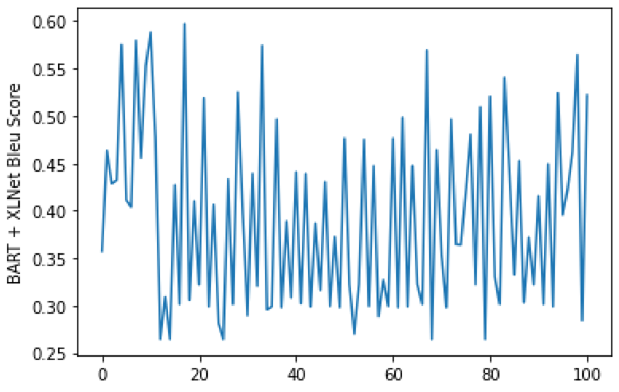


Figure 7: BART+XLNET Average Score: 0.4153

In general, all models' performance is deteriorated more or less due to the limitation of rhyming endings, as compared with the previous open-ended generations. As we can see in figure 6 and figure 7, BLEU scores for BART+GPT-2 mostly range from 0.4 to 0.5, with an average of 0.4768. The corresponding metric would be "high quality translations". In contrast, BART+XLNET's scores vary from 0.3 to 0.45 and thus basically "understandable to good". In BART+GPT-2, we can also observe that there are relatively few variations in the lines, meaning the quality among lines is more identical. Meanwhile, the quality of texts generated by BART+XLNET varies greatly. There are also many turning points between a previous line and a next line.

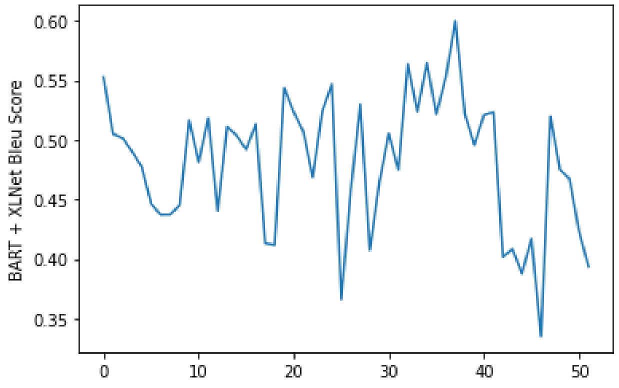


Figure 8: BART in BART+GPT-2 Average: 0.4664

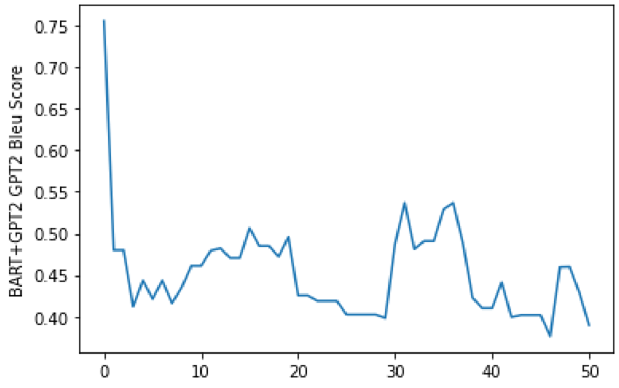


Figure 9: GPT-2 in BART+GPT-2 Average: 0.4672

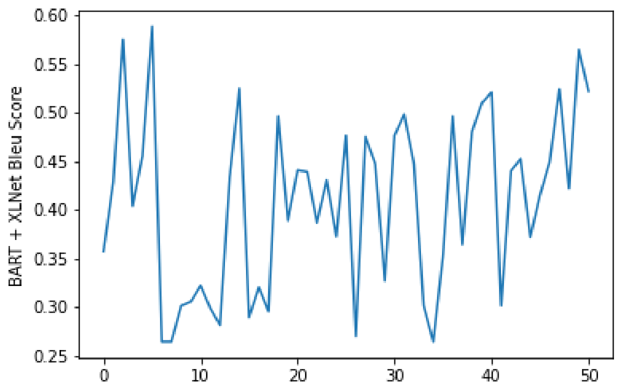


Figure 10: BART in BART+XLNET Average: 0.4490

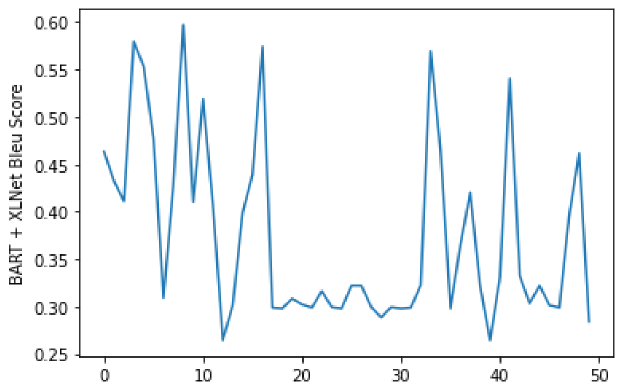


Figure 11: XLNET in BART+XLNET Average: 0.3816

If we take a look at figure 8 to figure 11, we can discover more details about how each model performs in the generation. For BART model in BART+XLNet, it seems to be producing OK sentences with score from 0.4 to 0.5 with an average of 0.4490, but still generated some sentences with less quality. XLNet, on the other hand, has great variations in its generated texts. While there are about 8 very good sentences, for a lot of the time it just generates barely understandable lyrics. For BART+GPT-2, BART's stability is observably higher than in XLNet, while the range of the scores remains the almost same. As for GPT-2, given the constraints of the ending rhyme, it still manages to output good sentences with scores higher than 0.4 for most of the time and with an average of 0.4672. The quality of BART and GPT-2 sentences is also almost the same, as seen by their average scores 0.4664 and 0.4672. In general, the combination of BART and GPT-2 is the winner in terms of both the stability and the overall scores.

5.5 Lyrics of other genres

While there is still space for further improvements, the combination of BART and GPT-2 performs relatively well in our previous tests. To show its rhyming lyrics generation capability in other genres, again I made the combined models to produce country and pop lyrics.

BART + GPT2 Pop Lyrics Example:

I want to be with the crew
And i'll be there with you
I'm gonna make it ignite
And i'm gonna make it right

BART + GPT2 Country Lyrics Example::

You'll bring me to one seven
I'm gonna go to heaven
I'm gonna have a good time
The taste of a sublime

Through these two examples, we can see that the combined models are able to write general rhyming lyrics without too many problems. Nevertheless, we also have a better view on where improvements can be made. One point is that although rapid automatic keyword extraction algorithm helped with contextual information, the coherence among lines can yet still be enhanced. Meanwhile, the restrictions of the rhyming word seems like too much a burden for the models. For example, for the second sentence in the country lyrics example, the generator had to write the sentence based on the ending word "seven", while there apparently are better options, like "feather". To solve that, we can either include more words for considerations as a slant rhyme candidate, or allow the generator to change the ending word a bit. Still, the best approach is to build a big word net based on how relevant two words are and let the generator select only the most relevant rhyming word. This approach may take a lot of efforts but should be very effective.

6 Summary and Future Work

In this project, I explored the capabilities of various language models towards song lyrics generation and built a basic rhyming lyrics generator based on my findings. While some models like GPT2 and BART are more competent in the specific task of lyrics generation, other models are less fitted. The 2 most notable problems in generating lyrics are the failure in capturing a proper lyrical global structure and the failure in avoiding repetitions. GPT2 and BART, on the other hand, manages to produce new and understandable texts at a reasonable length, thus more suited for lyrics generation models.

As for rhyming lyrics generation, in general the combined model of BART and GPT-2 can write lyrics in different genres with an acceptable quality. The combination of different models effectively solves the problem of repetitiveness. Nevertheless, we can see there is a lack of coordination between the rhyming word and the whole sentence and between the sentence and the whole verse. To produce a better rhyming word, we can either allow tense or noun/verb change, or include slant rhymes in the rhyme dictionary, or just build a word net based word relevance. As for the coordination between a sentence and the whole verse, rapid automatic keyword extraction algorithm helped with the situation. Yet, an approach that can further enhance the fluency of lyrics is to use keywords in the training period. When trained to find connections between keywords and sentences, the model would have better understanding when asked to generate texts of a certain topic.

References

[1] Eric Malmi, et al. Dopelearning: A computa-tional approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Con-ference on Knowledge Discovery and Data Mining.* 2016.

[2] Shashank Kapadia. "Language Models: N-Gram A step into statistical language modeling." *https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9*. 2019.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In *Neural Computation Vol 9 Num 8. Pages 1735-1780.* 1997.

[4] Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya "Language Models are Unsupervised Multitask Learners" 2019.

[5] Zhilin Yang, et al. "XLNet: Generalized autoregressive pretraining for language understanding." In *Advances in neural information processing systems.* 2019.

[6] Mike Lewis, et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* 2019.

[7] Hugo Oliveira. Automatic generation of poetry: an overview. In *Proc. 1st Seminar of Art, Music, Creativity and Artificial Intelligence*. 2009.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*. 2002.

[9] Enrique Dans. Word-level LSTM text generator: Creating automatic song lyrics with Neural Networks. *https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb*. 2018.

[10] Chris McCormick. XLNet Fine-Tuning Tutorial with PyTorch *https://mccormickml.com/2019/09/19/XLNet-fine-tuning/*. 2019.

[11] Ian Pointer Text Generation With GPT-2 And (only) PyTorch *https://snappishproductions.com/blog/2020/03/01/chapter-9.5-text-generation-with-gpt-2-and-only-pytorch.html.html*. 2020.

[12] Neil Sinclair. Fine-tuning Hugging Face's BART Model *https://morioh.com/p/c116e6c5fb09*. 2020.

[13] Lili Mou, Rui Yan, Ge Li, Lu zhang, Zhi Jin. Backward and Forward Language Modeling for Constrained Sentence Generation. 2016.

[14] Stuart Rose, Dave Engel, Nick Cramer, Wendy Cowley. Automatic Keyword Extraction from Individual Documents. In *Text Mining: Applications and Theory* 2010.