Evaluating Language Models on Hip Hop Lyric Generation

Ryan Farhat-Sabet, Moonsoo Kim, Joseph Uren
University of California, Berkeley
{ryan_farhat-sabet, myk5391, joseph.uren}@berkeley.edu

Abstract

This project explores the NLP task of next-line generation for hip-hop lyrics, which is characterized by challenges related to rhyme, rhythm, slang, and creative phrasing. Models including GPT-2, Llama-3.2, and FLAN-T5 were evaluated with baselines, training, and fine-tuning to improve performance on a set of metrics including: BLEU, ROUGE, BERTScore, SBERT similarity, rhyme rate, syllable similarity, and word diversity. For the fine-tuned versions of the models, FLAN-T5 performed best and was used for further experiments.

Experiments on the FLAN-T5 model included prompt-engineering, additional layers for syllable and rhyme, while experiments into additional input lines, and backwards generation led to significant improvements over the fine-tuned model. These results demonstrate that encoder-decoder models with bidirectional attention and targeted input strategies are more effective for creative generation in a constrained domain such as hip-hop lyrics.

1 Introduction

Song lyrics are a rich and emotionally charged form of creative expression. Much like poetry, they invoke a deep understanding of language to creatively convey meaning. Developing and fine tuning NLP models capable of interpreting and generating lyrics has promising applications in areas such as songwriting support, music therapy, and affective computing. Yet, lyrics introduce distinct challenges. Lyrics often incorporate slang, metaphor, and unconventional grammar, span multiple genres, and emotional tone can shift abruptly from line to line. These factors make both comprehension and generation more challenging than some more traditional NLP tasks.

In particular, hip hop lyrics tend to be especially complex, featuring intricate rhyme patterns, unexpected word pairings, metaphoric language, and unconventional phrasing that rarely appears in everyday speech. These challenges mean generating hip hop lyrics requires models to go beyond standard language generation strategies and engage in more creative and more flexible reasoning.

We put forward our contribution to this task: creating a model to predict the next line of a hip hop song, and figuring out which inputs assist the most in this task.

2 Background

Text generation for music is not a new concept. Since the advent of transformer-based models, creative writing generation has been a popular task, and multiple studies have looked to improve generation through all sorts of model implementations (Le, 2024). The applications for these sorts of models extend beyond music to other creative writing endeavors as well, including poetry and storytelling.

2.1 Literature Review

There have been many studies specific to work on lyric generation for music. Since this is a complex task that requires not only a deep understanding of language, but also of how music interacts with language, each paper tackles a slightly different component to improve the generation process. Early approaches focused on the creative writing task itself, using tools like masked-word prediction within the confines of a theme to predict lyrics (Oliveira, 2021). Some studies went further and would focus on the hidden structure implied through the lyrics themselves to teach the model the rules and standards of songwriting (Watanabe, 2020). Some researchers took a multidisciplinary approach, examining actual musical composition in tandem with lyric composition and how they interact, to improve their models' performance. These included providing extra inputs like musical beat and lyric positioning (Qian, 2023), melody and syllable positioning (Watanabe, 2018) (Zhang, 2024), and lyric-syllable singability (Ou, 2023), all in an effort to improve lyric generation quality.

Rap/hip hop music is particularly complex, and specific work has been done in this subdomain as well. Early attempts simply aimed to generalize the style of particular artists, essentially training a model to write like someone else and match their style (Potash, 2018). Complex line structures and rhyme patterns make generation particularly difficult, and training models to focus on certain extra inputs does often improve performance. One such approach strips down the input to the most important context before reconstructing it, allowing for increased attention on important context when it comes time for generation (Nikolov, 2020). DeepRapper attempted to model rhyme and rhythm as additional input layers to further constrain lyric generation within learned parameters (Xue, 2021).

2.2 Overview of Models

Decoder-only models

The current trend, especially in industry, is to leverage decoder-only models for creative writing tasks like this. Open-source models like GPT2 and Meta's Llama-3.2 models are readily available and easy to implement. Llama-3.2 in particular is trained on an incredibly large corpus of data recently, so it is close to the current state-of-the-art approaches. You provide a prompt before the input tokens, and the decoder's job is to generate the next words. Since there is no encoder, the model does not struggle as much when providing a longer input and prompt, and since it is less constrained by the token length (compared to encoder-decoder models), it can still generate an appropriately long output. The lack of bidirectional encodings does limit these model's ability to attend to more nuanced aspects of the input, so we used these models as our baselines while also testing models with bidirectional attention.

Encoder-Decoder models

Encoder-decoder models, while less popular in more recent literature and industry implementations, offer an advantage over decoder-only models, as they allow for bidirectional attention which can improve their ability to understand context, rhythm, and thematic elements. FLAN-T5 is a fine-tuned variant of the T5 encoder-decoder model and was designed to follow natural language instructions for thousands of different tasks. This fine-tuning makes it highly adaptable to creative and prompt based generation tasks, such as generating song lyrics. FLAN-T5's flexibility in text-to-text generation allowed us to frame the next-line prediction task as a natural language instruction, which works well given the model's original instruction tuning. While these sorts of encoder-decoder models are more

computationally intensive and have harder limits on input/output length, next line prediction is naturally a shorter task, so the issues are less pronounced.

3 Methods

After defining our objectives and cleaning up our data, we established our baseline architecture and the experiments we wanted to run to improve our model's performance.

3.1 Task

The task for these experiments is next-line generation for hip-hop lyrics, where given a sequence of one or more previous lines from a song verse, the model is tasked with generating a line that could naturally fit. Formally, the task can be described as a conditional text generation problem, where the input sequence $X = \{L_1, L_2, ..., L_n\}$ consists of up to three preceding lines, and the goal is to generate the next line L_{n+1} such that it maintains the style, rhyme, and lyrical structure with respect to X. This task differs from standard text generation due to:

Stylistic Constraints: Hip-hop lyrics often rely on strict rhythmic patterns, internal and end rhymes, and culturally specific vocabulary. The lines generated must capture not just meaning, but musicality and genre-specific style.

Structural Balance: Generated lines must match the syllabic structure and length of preceding lines to preserve the flow of the lyrics.

Creative Language: The task involves frequent use of metaphor, slang, repetition, and other devices which challenge traditional language modeling techniques.

3.2 Data

At the start of the project a dataset using Spotify data (Huggingface, 2025) was considered, however the dataset was determined to be inadequate due to the song lyrics not having valid new line characters. We experimented with inferring the split between lines, but results were not ideal and could have affected the results of the experiments. The labeled dataset employed in the models is a subset of a dataset of lyrics for 5 million+ songs from Genius lyrics (Kaggle, 2022). The dataset was filtered down to songs tagged with the 'rap' genre, before further filtering was done to leave songs from mainstream hip-hop artists. The filtered data was then manipulated to remove tags related to song verses and producer credits, and split into pairs of consecutive song lyrics for the purpose of generating and evaluating next line lyrics.

3.3 Evaluation Metrics

Ultimately, the goal of these models is to predict the next line of the song when compared to the original next line. However, since hip hop lyrics are complex and other generations might also make sense, we wanted to capture other success metrics that give credit to generations across a variety of metrics. We broke this down into a whole host of evaluation metrics so we could determine which experiments improved which parts of the generation process, but we also examined the outputs for human readability and understanding as well:

BLEU: BLEU stands for Bilingual Evaluation Understudy and is essentially a measure of how many words produced by the model are in the reference text. It is a commonly used metric for text generation tasks, and a higher score is better (range of 0 to 1). It only counts exact words though.

ROUGE (1, 2, L): ROUGE stands for Recall-Oriented Understudy for Gisting

Evaluation and also measures the overlap of key words from the reference text. ROUGE-2 measures bigrams, and ROUGE-L measures the longest continuous sequence of words. ROUGE also only counts exact words.

BERTScore: BERTScore is a measure of the contextual embeddings of the generated output against the reference. In essence, it compares all the word embeddings against each other and calculates the cosine similarity between them. We report out the F1 score, the higher the better.

Sentence-BERT Cosine Similarity (SBERT): Whereas BERTScore measures the contextual word embeddings, Sentence-BERT compares the whole sentence embedding of the output against the reference. The higher the better here as well.

Rhyme Rate: We built our rhyme rate metric using the CMU Pronunciation Dictionary, determining the phonemes of the last words of each line and identifying both perfect and near rhymes. As a rule of thumb, higher is better.

Syllable Similarity: Syllable similarity is a measure of the difference between the total number of syllables generated vs in the reference. This acts as a guide for how closely the outputs match in length. The closer to 0, the better.

Word Diversity: Word diversity is a measure of the unique words generated divided by the total number of words generated. While a higher score here isn't necessarily better, a lower score is worse, as that is indicative of the model generating lots of repeated words.

3.3 Baseline and Experiments

We first established our baseline model and worked on improving it from there. We generally compared all of our experiments to our baseline model, but once we established the best model and fine-tuning approach, we proceeded with the rest of our experiments with this improved "baseline".

Baseline (out-of-the-box): GPT2 was our baseline model since it was an early pioneer in text generation, and we used it without any additional training. The prompt included only a single line of input lyrics, and generation parameters were modest (e.g., max tokens = 30, temperature = 0.8).

Other models (out-of-the-box): We tested a few more models without any additional training, most notably DistilGPT (just a smaller version of GPT2, faster to train), Llama-3.2-1B (a larger, more state-of-the-art decoder model), and FLAN-T5 (updated version of the classic T5 encoder-decoder model). The prompt included only a single line of input lyrics, and generation parameters were modest (e.g., max tokens = 30, temperature = 0.8).

Pre-training on lyric data: We trained each of these models (GPT2, Llama-3.2, and FLAN-T5) for three epochs on a curated dataset of sixty thousand hip-hop lyric line pairs to familiarize it with lyrical structure, slang, rhyme patterns, and common phrasing.

Fine-tuning: After pre-training on the lyric data, we adjusted the training hyperparameters and generation parameters to further improve coherence and performance. We once again did this for all three models, and we tested various approaches, including LoRA fine-tuning, to produce the best tuned version of our base models.

Prompt engineering: For the decoder models specifically, we also tested ten different prompts, varying in length and specificity. Without the encoder layer, we thought changing the prompt

would impact the quality of the generations produced, particularly for the decoder models.

From here, once we had identified the best combination of model and fine-tuning technique, we conducted a series of further experiments building on that setup:

Rhyme layer: Inspired by an experiment from DeepRhymes (Zhang, 2023), we added an extra decoder layer tokenizing the input line as phonemes before multiplying the output with the base model output. The goal here was to give the model phonetic information so that it could have more information to work with when it comes time for generation.

Syllable layer: Since we were already using the CMU Pronunciation Dictionary for our evaluation metrics, we drew inspiration from another paper that used it to break up input lines into syllables and group them into length buckets to feed as an added prefix to the input layer (Manjavacas, 2019). We hoped this would help with the length of the generations since the model would try and match the general length of the input.

3-Line input: The input was expanded to include three consecutive lines from a song instead of one. This experiment aimed to enhance contextual awareness and allow the model to better understand rhyme schemes, lyrical progression, and flow.

Backwards generation: Inspired by DeepRapper (Xue, 2021), we tested inverting the input line so it reads the input right to left, then generates the output right to left, or last word first. The goal here was to help the model predict the supposed rhyming word first and then let it fill in the rest of the line based on the first word it has already generated. We thought this would help with rhyme rate mainly.

Fine-Tuned Models	BLEU	BERT	SBERT	Rhyme Rate	Syllable Similarity
GPT2	0.0177	0.8268	0.2065	13.20%	6.3
Llama-3.2	0.0089	0.8212	0.2014	1%	8.2
FLAN-T5	0.0298	0.8366	0.2248	38.45%	3.8

Table comparing all three fine-tuned models

4 Results and Discussion

Looking at the three base models first, we expected the decoder-only models to perform pretty well. We had a feeling the encoder-decoder model would work well too, but we were still a bit surprised that the encoder-decoder FLAN-T5 model worked better than both GPT2 and the Llama models across essentially all metrics, regardless of how much we pre-trained and fine-tuned them. We know that the FLAN-T5 model was already fine-tuned on similar tasks, and we believe that the added bidirectional attention really helped with the model's ability to produce a coherent result. While pre-training and fine-tuning generally increased most metrics, one interesting finding was that total rhyme rate decreased. The FLAN-T5 baseline model actually had the highest rhyme rate of all of our experiments, which surprised us at first, until we realized it was often just regurgitating a very similar line

FLAN-T5 Models	BLEU	BERT	SBERT	Rhyme Rate	Syllabl Similarit
Baseline	0.0177	0.8268	0.2053	43.77%	5.
Fine-tuned	0.0089	0.8212	0.2014	38.45%	3.
Rhyme Layer	0.003	0.7966	0.1297	32.62%	13.
Syllable Layer	0.0264	0.836	0.2185	34.91%	3.
3-line Input	0.0323	0.8375	0.2265	23.06%	3.
Backwards Input	0.0324	0.8324	0.2341	42.34%	3.

Table of FLAN-T5 experiments against baseline and fine-tuning

compared to the input. Whereas we would often see all the baseline models output a slight variant of the input line before any tinkering, we would see the model generating more diverse lines with even the small amount of fine-tuned data we provided (60K line pairs is not a very large corpus in the scale of these models).

Our true experiments began with testing ten different prompts to see how that would change the generation results. We honestly believed that there would be some marked improvement with a longer, more specific prompt, so we tested a few variations with increasing instructions and restrictions. To our surprise, the metrics did not change all that much with different prompts. To the naked eye, the quality of the generations did not seem to change all that much, and the metrics didn't paint much of a better story. Our first prompt ended up with the highest BLEU and ROUGE scores, but the differences between prompts were small. We opted to go with the sixth prompt as it still had strong results when compared to the other prompts but was a bit more generic, allowing our model to be more creative, and further experiments would hopefully train our model to capture some of the instructions we had originally tried to prompt it on.

With our best fine-tuned FLAN-T5 model in hand, paired with our best generic prompt, we began experimenting with different ways to augment our input data. Our rhyme layer experiment seemed promising at first, but ultimately did not pan out. Not only were the metrics all worse than the fine-tuned baseline, the reasoning seemed to be that the model was learning to rhyme the last word of the first line with either the same word or a slightly different word and then repeating it for the majority of the second rhyme. The rhyme rate was alright, but that even performed worse than the baseline.

Adding a syllable bucket as an input prefix did have some mild success, particularly when it came to matching the expected length of the output. By defining clear buckets for the syllables and passing an extra syllable layer, the model would attend to that token and learn to correctly limit the length of the output.

The last two experiments ended up being the most beneficial. Introducing the model to 3-lines of input saw a decrease in the total rhyme rate caused by the model generating lines that matched the style, rhyme, and flow of the section, rather than the end rhyme of one line. Despite a slight decrease in rhyme density when using three input lines, overall semantic coherence, rhythm, and alignment with lyrical themes improved. The backwards input and generation further elevated the model outputs, accurately rhyming the lines while maintaining semantic consistency at both the word and sentence level. Even with the improvement from this experiment however, there was an issue with the generated outputs that wasn't captured by the metrics: the sentences did not quite make complete grammatical sense. Reading sentences backwards most likely messed with the model's understanding of grammar, because the words used were often correct and applicable in the sentence, just out of order and not in the correct position.

These results highlight the importance of task-specific training on the dataset when applying large language models to creative tasks. FLAN-T5's best performance came from either expanding the input window or reading and generating the input window backwards, suggesting that lyric generation can improve significantly with additional context from prior song structure and defining certain target aspects early in the processing. Some possible combination of the two would most probably yield the best result, given that one of the two held the best score across almost all the metrics, but we ran out of time to run that experiment.

5 Conclusion

We set out to build a model that could write hip hop lyrics that fit within a predetermined song context, and we determined that encoder-decoder models, by means of their bidirectional attention capabilities, are more effective for short-form creative generation tasks like hip hop lyric generation. The addition of extra input data and layers improved model generation across the board. Future work is still needed with regards to combining the successful experiments to see if the gains compound, along with more training data for the model to learn from.

6 Authors' Contributions

Ryan conceptualized the project and went to office hours to discuss project questions as they came up. Joseph performed exploratory data analysis and prepared the dataset for use in the model and experiments. Ryan defined and coded up the evaluation metrics. Ryan, Joseph, and Moonsoo each ran the GPT2, FLAN-T5, and Llama models (respectively) for baseline and fine-tuning. Ryan ran the rhyme layer, syllable layer, and reverse sentence input experiments. Moonsoo ran the prompt engineering experiment. Joseph ran the best prompt and 3-lines input experiments. Ryan and Joseph wrote the final report.

References

Dinh-Viet-Toan Le, Louis Bigo, Mikaela Keller, & Dorien Herremans. 2024. Natural Language Processing Methods for Symbolic Music Generation and Information Retrieval: a Survey.

Hugo Oliveira. 2021. Exploring a Masked Language Model for Creative Text Transformation.

Kento Watanabe & Masataka Goto. 2020. Lyrics Information Processing: Analysis, Generation, and Applications.

Tao Qian, Fan Lou, Jiatong Shi, Yuning Wu, Shuai Guo, Xiang Yin, & Qin Jin. 2023. UniLG: A Unified Structure-aware Framework for Lyrics Generation.

Longshen Ou, Xichu Ma, Min-Yen Kan, & Ye Wang. 2023. Songs Across Borders: Singable and Controllable Neural Lyric Translation.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A Melody-Conditioned Lyrics Language Model.

Zhe Zhang, Karol Lasocki, Yi Yu, & Atsuhiro Takasu. 2024. Syllable-level lyrics generation from melody exploiting character-level language model.

Peter Potash, Alexey Romanov, & Anna Rumshisky. 2018. Evaluating Creative Language Generation: The Case of Rap Lyric Ghostwriting.

Nikola I. Nikolov, Eric Malmi, Curtis Northcutt, and Loreto Parisi. 2020. Rapformer: Conditional Rap Lyrics Generation with Denoising Autoencoders.

Lanqing Xue, Kaitao Song, Duocai Wu, Xu Tan, Nevin L. Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021. DeepRapper: Neural Rap Generation with Rhyme and Rhythm Modeling.

Bessie Zhang, Catherine Kung, and Ivan Villa-Renteria. 2023. DeepRhymes: Efficient End-to-end Conditional Rap Lyrics Generation.

Enrique Manjavacas, Mike Kestemont, and Folgert Karsdorp. 2019. Generation of Hip-Hop Lyrics with Hierarchical Modeling and Conditional Templates.

Li Tian & Xiaoli Yang. 2023. DeepLyrics: GPT2 for lyrics generation with finetuning and prompting techniques

Appendix

A Full table of all experiments and results

Experiment	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERT	SBERT	Rhyme Rate	Syllable Similarity	Word Diversity
GPT2 Baseline	0.0063	0.08	0.009	0.07	0.8177	0.1523	3.04%	16.37	0.863
DistilGPT Baseline	0.0071	0.0735	0.0105	0.0661	0.7985	0.1616	4.60%	13.5	0.824
Llama-3.2 Baseline	0.0065	0.076	0.0127	0.0699	0.8172	0.1792	1%	11.52	0.318
FLAN-T5 Baseline	0.0177	0.1006	0.032	0.0963	0.8282	0.2053	43.77%	4.95	0.951
GPT-2 Pre-Training	0.0156	0.1027	0.0262	0.0954	0.8262	0.2038	12.71%	6.34	0.724
Llama-3.2 Pre-Training	0.0089	0.084	0.016	0.079	0.8212	0.2014	1%	8.2	0.366
FLAN-T5 Pre-Training	0.0252	0.1149	0.0358	0.109	0.8354	0.2142	30.96%	3.84	0.913
GPT-2 Fine-tuning	0.0158	0.1035	0.0262	0.0967	0.8268	0.2065	13.20%	6.28	0.712
Llama-3.2Fine-tuning	0.0089	0.084	0.016	0.079	0.8212	0.2014	1%	8.2	0.366
FLAN-T5 Fine-tuning	0.0298	0.1239	0.0425	0.118	0.8366	0.2248	38.45	3.77	0.883
GPT2 Prompt Engineering	0.0268	0.0909	0.0365	0.0899	0.8166	0.2378	5%	10.51	0.14
FLAN-T5 Rhyme Layer	0.003	0.0354	0.0043	0.0344	0.7966	0.1297	32.62%	13.13	0.795
FLAN-T5 Syllable Layer	0.0264	0.1184	0.0376	0.1123	0.836	0.2185	34.91%	3.76	0.91
FLAN-T5 3-line Input	0.0323	0.1277	0.0452	0.1213	0.8375	0.2265	23.06	3.79	0.91
FLAN-T5 Backward Generation	0.0324	0.1251	0.0463	0.1193	0.8324	0.2341	42.34%	3.66	0.902

B Table of all prompts tested during prompt engineering and results from running them on GPT-2

Prompt #	Prompt Text
Prompt 1	"Given this rap line, generate the next line: {line1}"
Prompt 2	"This is a line in a hip hop song. I want you to generate the next line in the same song: {line1}"
Prompt 3	"This is a line in a hip hop song. I want you to generate the next line in the same song. Try to match the style from the first line: {line1}"
Prompt 4	"This is a line in a hip hop song. I want you to generate the next line in the same song. Try to match the style and length from the first line: {line1}"
Prompt 5	"This is a line in a hip hop song. I want you to generate the next line in the same song. Try to match the general length and style and emotion from the first line. I also want the next line to rhyme: {line1}"
Prompt 6	"Given this song lyric line, generate the next song lyric line: {line1}"
Prompt 7	"Take this line from a hip hop song and generate the next line, matching the style, rhyme, and flow: {line1}"
Prompt 8	"Take this line from a rap song and generate the next line, matching the style, rhyme, and flow: {line1}"
Prompt 9	"Take this line from a rap song and generate the next line, matching the style, rhyme, and flow. Keep the length and meter consistent as well: {line1}"
Prompt 10	"Keep the rhyme going. Given current rhyme, write the next rhyme: {line1}"

Prompt #	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERT	SBERT	Rhyme Rate	Syllable Similarity	Word Diversity
Prompt 1	0.0274	0.0946	0.0346	0.0939	0.8184	0.2352	0.07	10.99	0.1385
Prompt 2	0.0266	0.0854	0.0326	0.0855	0.816	0.2415	0.06	11.32	0.1411
Prompt 3	0.0261	0.0873	0.0347	0.0869	0.8179	0.2368	0.05	10.64	0.1506
Prompt 4	0.0263	0.0855	0.0345	0.0852	0.8071	0.241	0.04	10.68	0.1436
Prompt 5	0.0258	0.0887	0.0347	0.0884	0.8078	0.2431	0.04	10.93	0.1439
Prompt 6	0.0268	0.0909	0.0365	0.0899	0.8166	0.2378	0.05	10.51	0.1398
Prompt 7	0.0269	0.0864	0.0326	0.086	0.8072	0.2361	0.07	10.69	0.1429
Prompt 8	0.0267	0.0873	0.0331	0.0874	0.8097	0.2385	0.04	11.69	0.148
Prompt 9	0.0265	0.0843	0.034	0.0844	0.8078	0.2383	0.05	11.02	0.1454
Prompt 10	0.0263	0.0878	0.0329	0.0878	0.8156	0.2388	0.05	10.01	0.1312

C Output examples from various experiments

Experiment	Given Line(s)	Generated Next Line
FLAN-T5 Finetuned	And put my head on his bible	And I'm a hero, to the mercy of God
	And living single, all my diamonds seem clear	He got me talking, he got me on a plane
FLAN-T5 Best Prompt	And put my head on his bible	And I'ma see what's to come
	And living single, all my diamonds seem clear	And I'm just a living single, like that girl
FLAN-T5 3-Line Input	I'm gnarly, and I'm wavy Salute me, I'm going crazy I get blowed until I'm faded	Then I got all the money I asked for (That's all the money)
	We ain't gonna talk about drip I'm in Chanel, you in Guess right now (Pussy) I hit the road for the shows	And I made it, I am better than you
FLAN-T5 Rhyme Input	Had this bitch on a ride, like roller coasters, Coney Island	"I tamed the Earth and listened and listened and listened and listened and listened and listened and listened and
	She so mo'fuckin' wet use my dick as a squeegee	country cookie origami baby origami baby origami baby origami baby oriami baby oriami baby oriami baby
FLAN-T5 Syllable Input	And I know we getting down, getting down to your last ounce of patience	That means we gon' be together
	Yes I'll be right here on the ground when you want me	I can't see you in my room when I leave
FLAN-T5 Backwards Generation	You might not have been the only o ne that could've did that	And you can't do that say they doing that
	The motherfuckin' law, in every way	I'll make your shit, it pull it up
GPT2 Baseline	and put my head on his bible	. Also don't throw anything if you wanna win from this line.. (It must be nice to hear you disagree with me, eh?)
	dont nun round dis bitch move unless you get approval	You, I'm not telling you to stop. You, on . Also don't throw anything if you wanna win from this line.
GPT2 Finetuned	and put my head on his bible	and we'll kill a bitch, just like she did it in the streets of the streets of the
	dont nun round dis bitch move unless you get approval	i need to move back to the past, i'm in my time, i'm in my time
DistilGPT	and put my head on his bible	. Then, my mother was there and I just did not know how the man would take the sentence.
	dont nun round dis bitch move unless you get approval	I then went back to the same for the line, get the job done. This line isn't even the first one, this line is just the