# SloGAN: Character Level Adversarial Lyric Generation

**Aidan Cookson** and **Auguste Hirth** and **Krish Kabra**
University of California, Los Angeles
{aucookson, ahirth, krish97}@g.ucla.edu

## Abstract

This paper explores the application of a deep learning models for the task of rap lyric generation. We first propose a generative LSTM character-level model that contrasts with recent word-level approaches. We rigorously compare the effectiveness of a character-level models with their word-level counterparts using BLEU, RhymeAnalyzer, and human authenticity prediction metrics. Secondly, we attempt to implement a SeqGAN for the rap lyric generation task in order to combat pitfalls of traditional maximum-likelihood estimation methods. Unfortunately, our SeqGAN model fails to converge. We provide reasons and possible solutions to improve the rap lyric generation task using adversarially trained networks.

## 1 Introduction

Natural Language Generation (NLG) for musical lyrics is an interesting task that has received relatively little attention within the research community. Traditional NLG tasks, such as data-to-text generation, prioritize conforming to strict syntactic structure and one-dimensional semantics. In stark contrast, lyric generation prioritizes creative structure and multi-dimensional semantics. Word spelling is modified for the sake of pronunciation. Sentence fragments serve as complete lines. Non-word enunciations are given meaning in the way they change how a line is sung. Repetition is abundantly used both for emphasis and verse structure. All of these quirks and many more are the result and medium of the songwriter's artistic expression. As such, recognizing and replicating them is critical to produce generated lyrics that mimic something as nebulous as "artistic style".

Because of all these quirks, we propose eschewing existing word-level text generation models and strategies. As these models seek to understand clear word definitions and sentence meaning, they would be unsuited to a dataset rife with ambiguity; like those of lyrics. Instead, we propose approaching the problem with character-level models. We seek to show that ignoring the word-level formulation will discard information that would otherwise skew our model, and the text it generates, away from something that the artist would actually write. Additionally, character level models have other benefits, like relatively low vocabulary size, which improves the speed of our training, and thus the pace of iteration and improvement on the model.

This work focuses on developing a generative language model for rap lyrics. In particular, we hope our model can mimic the style of particular rap lyricists. The rap lyric generation task has been explored by past works (Addanki and Wu, 2013; Malmi et al., 2016; Potash et al., 2015). We specifically build upon the work done by (Potash et al., 2015). In their research, a character-level LSTM model was created to generate novel rap lyrics that follow the style of a rapper without copying existing lyrics. Unlike previous RNN lyric generation models, which use a strict template to structure the lyrics, their model learns the lyrical structure implicitly.

Our first goal is to implement two identical LSTM networks trained using maximum likelihood estimation (MLE) based on (Potash et al., 2015). The networks differ in that one network generates lyrics using a character-level approach, while the other uses a word-level approach. We then evaluate the two models and their performance with regards to rap lyric generation task using the BLEU, RhymeAnalyzer, and human authenticity metrics.

Our second goal is to implement an adversarially trained network for rap lyric generation. Traditional MLE generative methods suffer from exposure bias (Bengio et al., 2015; Ranzato et al.,

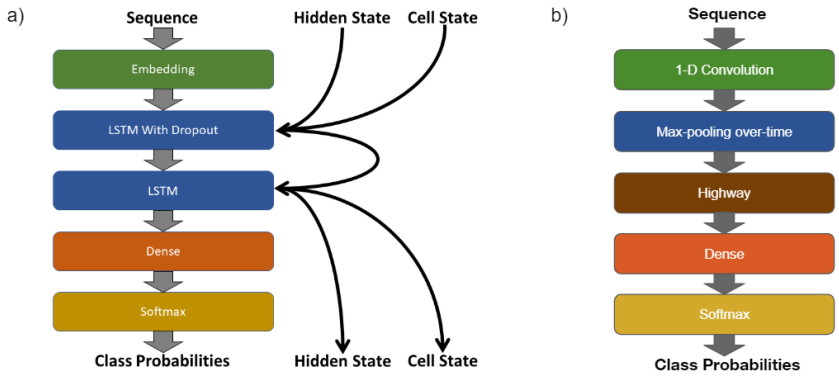


Figure 1: a) LSTM Generator Network Architecture. b) CNN Discriminator Network Architecture

2016), making long text generation difficult. As we propose moving to a character-level model, we anticipate difficulties having the model produce longer lyrics that contain rhythm and rhyme. A common solution to mitigate the exposure bias problem is to utilize adversarially trained networks. Specifically, we utilize the SeqGAN architecture (Yu et al., 2017) with modifications for character-level generation. Adversarially trained networks for creative text generation is a topic that has been explored (Yu et al., 2017; Guo et al., 2018; Che et al., 2017; Liu et al., 2018; Saeed et al., 2019). However, to our knowledge, no other work has explored the use of such networks for rap lyric generation.

## 2 Method

### 2.1 Data Collection

We used the LyricsGenius[1] python library to scrape Genius.com for complete lyrics for songs by our chosen artists. Because we use a character-level configuration, we found during early testing that characters like capital letters, numbers, and non-punctuation symbols were interfering with the quality of the generated text. In order to solve this, and to make the data usable for our models, we performed the following preprocessing steps:

- Make all alphabetical characters lower case
- Make all numbers into their word counterpart. Eg: "64" to "sixty four".
- Filter and ignore all non-alphanumeric, non-punctuation characters.
- Only collect verses. Discard repetitive sections like the chorus.
- Add [startVerse] and [endVerse] tokens. Newline ("\n") is considered the [endLine] token. (Potash et al., 2015) performed a similar preprocessing step.

As artists have released different numbers of songs, each with different lengths, the total number of verses we got from each of them were different. As we will discuss later, this and other features caused differences in the results, like some datasets overfitting faster.

### 2.2 MLE with LSTM Generator

In order to use the network with characters, a dictionary must first be created to translate the vocabulary to integers. A histogram was used to sort the characters in the vocabulary by frequency, and integers from 0 onwards were assigned to each character. After translating the training lyrics into a sequence of integers, the sequence was split into subsequences. Each training subsequence was then indexed one character forward to generate the target data. For example, if one training subsequence was "the quick do", the corresponding target sequence would be "he quick dog".

The word-level model architecture and training procedure is identical, except the vocabulary contains words instead of characters. The hyperparameters of the model must also be tweaked to account for the shorter training subsequences and space taken up by the much larger vocabulary of unique words.

The LSTM architecture is shown in Fig. 1a below. The embedding layer converts each positive integer in the input sequence to a dense vector of a fixed size. Following the two LSTM layers is a fully connected layer to resize the output to the vocabulary size. Finally, a softmax layer calculates

[1] https://github.com/johnwmillr/LyricsGenius

the probabilities of each character class. The hidden and cell states are updated by the LSTM layers and returned.

During training, the data is split into batches and network parameters are optimized using Adam. The criteria is the Negative Log-Likelihood of the class probabilities. To prevent the gradients from exploding, they are clipped after using a L2-Norm of 5.

### 2.3 Adversarial training with SeqGAN

Traditional generative adversarial networks (GAN) (Goodfellow et al., 2014) architectures are not suitable for sequence generation tasks such as that of NLG. This is due to the non-differentiability of the argmax function used at the end of each sequence step and the inability to apply gradients to discrete tokens. Several solutions have been proposed to tackle these issues. For our work, we choose the SeqGAN model.

The discriminator network's goal is to correctly classify "real" data from "fake" data generated. We follow the authors of SeqGAN and implement a CNN classifier based on (Kim, 2014). The architecture is shown in Fig. 1b. This architecture implements several 1-D convolution filters of various window sizes optimized for the maximum sequence length. Moreover, a max-pooling over-time operation is performed for each convolution output, and the pooled features are fed into a final fully-connected softmax layer to get the probability the sequence is "real" or "fake". A highway architecture before the final fully-connected layer is also implemented.

In order to train the generator in an adversarial fashion, SeqGAN considers the generator to be an RL agent that takes an action (i.e. outputting a character) based on a policy and its current state (i.e. the previous characters outputted) with the hopes of maximizing a reward signal (i.e. fooling the discriminator) at the end of the sequence. The generator policy is optimized using a policy gradient method. Specifically, the REINFORCE algorithm is utilized (Williams, 1992).

To estimate the action-value function, a Monte-Carlo rollout search is performed at each time step $t$ to sample the unknown future $T - t$ tokens. The final action-value reward is equal to the probability output of the discriminator predicting the sequence as being "real".

### 2.4 Evaluation Metrics

To get varied perspectives when analyzing the results, we evaluate the performance of our models in three ways: (1) applying the BLEU metric, (2) using the RhymeAnalyzer software, and (3) using a by-hand analysis.

The BLEU metric is a sentence to sentence comparison metric designed for translations (Papineni et al., 2002). As it is evaluates word by word, and is intended to compare very similar sentences like translations, the original version is not immediately suitable to our task of comparing each generated sentence to the sentences of the artist's corpus. Luckily, with the addition of smoothing functions described in (Chen and Cherry, 2014), we can get reasonable results, even in this extreme case. We use this metric as a means to compare the quality of our generated text to that of other works in a subject-agnostic way.

The RhymeAnalyzer System is a metric and classification system used for identifying pairs of rhyming syllables in lyrics (Hirjee and Brown, 2009, 2010). It divides sentences not into words or characters, but into syllables, then considers them in terms of how often they occur sequentially with others. We use it for the aggregate statistics it provides about the identified rhyming syllables. As these metrics can be considered representative of an artist's style, we use them to check whether our generated lyrics can replicate the style present in the original baseline lyrics.

Finally, we take the Turing test into our own hands and act as individual discriminators, analysing lyrics and attempting to identify which source they came from. This is the most convincing of the three metrics but, due to time constraints, we have limited data for this particular experiment. We report our accuracy on text generated with both methods in order to compare the two. If the generated text is able to fool us, that is, to often pass as real lyrics, we conclude that they are well generated lyrics.

## 3 Results

### 3.1 Character vs. Word Level Generation

#### 3.1.1 BLEU Score

Our BLEU score results (Fig. 2) show the difference in BLEU scores between the word and character level models, as a percentage of the lower score. We find that, in six of seven cases, the

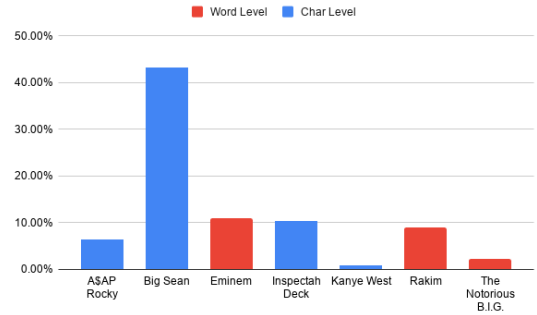


Figure 2: Smoothed BLEU - $\Delta\%$ from lower score

BLEU scores for both the word level and character level models are within 12% of each other. This indicates that, at least as translations of the baseline lyrics, the word and character level models are performing similarly.

If we consider the BLEU score to be indicative of the overall "goodness" of the generated lyrics, then we can note that, for the Big Sean case, our character level model outperforms the word level model by 40%. However, because we are not working with a translation task, this conclusion lacks the support necessary to be considered meaningful.

### 3.1.2 RhymeAnalyzer Metrics

For demonstration, we discuss three of the RhymeAnalyzer metrics. Syllable Variation measures the variance in the number of syllables per lyric line. Novel Word Proportion indicates the percent of words used which were unique. Rhyme Density measures the average frequency of rhyming syllables.

Our Syllable Variation data (Fig. 3a) details the percentage difference in syllable variation for the generated lyrics, relative to the original baseline lyrics. In every case, the word level models produced lyrics with syllable variation at least 72% higher than the syllable variation of the baseline lyrics, with a maximum of 157% higher. In contrast, the character level models exceeded a difference of 26% in only one of the seven cases. Even in the single case where the character level model had significantly higher Syllable Variation (at 93% for Inspectah Deck), it is still closer to baseline than the associated word level model (at 112%). These results indicate that the character level models tend to have Syllable Variation much closer to the baseline than the word level models.

Almost all generated lyrics from both model types used unique words at roughly half the rate that the baseline lyrics do (Fig. 3b). This is to be expected as both models would be unlikely to reuse a word seen only once in the learning corpus. However, we find that our character level models perform very similarly in this metric to the word level models. This is a surprising result because it shows that our character level models can perform similarly to their word level counterparts for *word level metrics*. This hints that representing the text as words may not provide a significant advantage for improving the usage of unique words in generated text, at least in the LSTM model we used. For our Kanye West character level model, we observed a significantly higher rate of unique word usage than the word level model, on par with the baseline lyrics. We suspect this is an outlier caused by the model misspelling words, but do not know why this was observed in only one of the models.

In terms of Rhyme Density, all of the models performed admirably, with the character level models tending to lag somewhat behind both the baseline and word level lyrics (Fig. 3c). However, even in the worst case, the character level lyrics had about over 60% of the Rhyme Density of the other lyric sets. More typically, they had only ten to twenty percent less rhyme density. We believe the word level models were able to rhyme as or more often than the baseline lyrics due to the comparatively fewer unique words used, meaning that rhyming words may be reused more often. This effect may be not have been replicated as strongly by the character level models due to misspelled words.

### 3.1.3 Hand Evaluation

Finally, our by-hand lyric evaluation data (Fig. 4) shows that the lyrics generated by our character level model tended to be predicted as real less often than the word level lyrics. However, a generated lyric is only around 15% less likely to be predicted as a real lyric if it is a from a character level model, compared to if it was from a word level model. Additionally, both sets of lyrics were predicted to be real more than half the time, meaning we produced lyrics that were passable more often than not. It is important to note that this data is based off hand evaluation of less than 250 lyrics in total. It is possible that, with significantly more data, the character level and the word level models would be shown to produce more similar results.

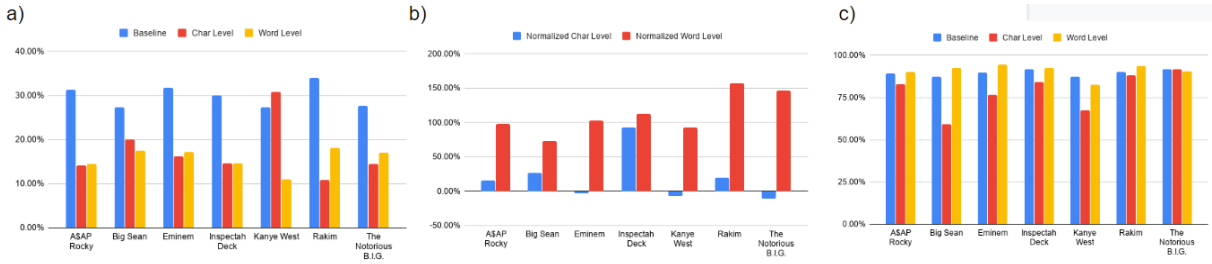


Figure 3: RhymeAnalyzer Results. a) Syllable Variation - $\Delta\%$ from baseline. b) Novel Word Proportion per Artist. c) Rhyme Density per Artist

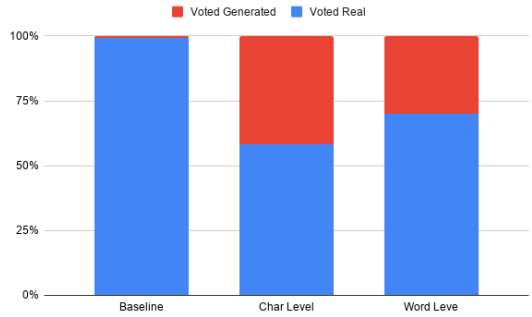


Figure 4: Vote Percentages for Hand Evaluated Lines

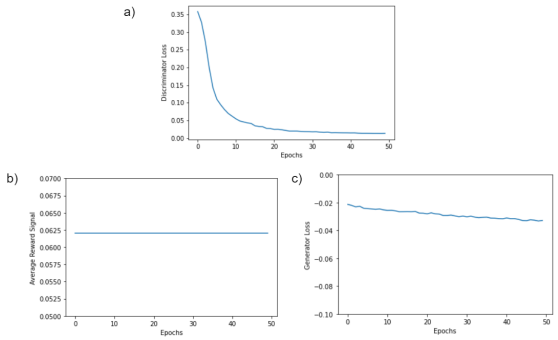


Figure 5: a) Discriminator Loss. b) Average Reward Signal. c) Generator Loss

### 3.2 Adversarial Lyric Generation

Unfortunately, the SeqGAN architecture failed to converge to an adequate solution as the generator mode-collapsed. After 50 epochs, the generator begins to continuously repeat the word "the". We suspect this failure is largely due to the very sparse average reward signals used to guide the generative model (see Fig. 5b). We attempt to alleviate this issue by introducing a bootstrapped rescaled activation function (Guo et al., 2018), to no avail.

## 4 Conclusion

Our character level models performed well, both relative to the baseline lyrics and to the word level models. More often than not, they produced metric results either very similar to those of the word level models, or only slightly worse. In the case of the Syllable Variation metric, the character level lyrics outperformed the word level ones significantly. These results show that character level models can be competitive with word level models, for our task of lyric generation. Unfortunately they do not show improvement across the board, but it may be possible for them to, in future work.

The SeqGAN architecture failed to converge, suffering from mode-collapse. This behavior is prevalent with adversarially trained models, with practitioners consistently citing the multitude of instabilities and failures to converge (Caccia et al., 2018). Nevertheless, several works have proposed new adversarial frameworks that show promising results for stabilizing training for the text generation task (Che et al., 2017; Guo et al., 2018; Fedus et al., 2018). For future work, we propose implementing these new frameworks for the lyric generation task.

## References

Karteek Addanki and Dekai Wu. 2013. Unsupervised rhyme scheme identification in hip hop lyrics using hidden markov models. In *Statistical Language and Speech Processing*, pages 39–50, Berlin, Heidelberg. Springer Berlin Heidelberg.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1171–1179. Curran Associates, Inc.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon

Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 362–367. The Association for Computer Linguistics.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. pages 5142–5148.

Hussein Hirjee and Daniel Brown. 2010. Using automated rhyme detection to characterize rhyming style in rap music. *Empirical Musicology Review*, 5.

Hussein Hirjee and Daniel G. Brown. 2009. Automatic detection of internal and imperfect rhymes in rap lyrics. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, pages 711–716. International Society for Music Information Retrieval.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Bei Liu, Jianlong Fu, Makoto P. Kato, and Masatoshi Yoshikawa. 2018. Beyond narrative description: Generating poetry from images by multi-adversarial training. In *2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018*, pages 783–791. ACM.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. 2016. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 195–204. ACM.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: Using an LSTM for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1919–1924. The Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Asir Saeed, Suzana Ilic, and Eva Zangerle. 2019. Creative gans for generating poems, lyrics, and metaphors. *CoRR*, abs/1909.09534.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2852–2858. AAAI Press.