Rock Lyrics Generation

Junzhe Wang
jwang169@stevens.edu
Stevens Institution of Technology
Hoboken, NJ

ABSTRACT

In this project, I focus on lyrics generation, which is a sub challenging task that generates a paragraph-level text. I introduce a method and discuss results of text-based RNN language model for automatic rock lyrics composition. The proposed model is designed to learn relationships within text documents, and creates lyrics with meaningful, interesting story and lyrical patterns for rock music. First, I perform analysis of different lyrics genre. Results show word frequency and the length of sentence. Second, I implement web scraping to collect lyrics of selected artists as the training data set. Then, I develop two text generation models which can be used for producing rock lyrics: a Long Short-term Memory recurrent neural network and a sequence to sequence model, and compare the result of these two models.

KEYWORDS

Seq2seq, LSTM, GRU, web scraping

1 INTRODUCTION

Rock music has embodied and served as the vehicle for cultural and social movements. My objective is to study the problem of computational creation of rock lyrics. Lyrics generation is a specific part of natural language generation, also known as text generation. Natural language generation is the automatic generation of natural language text for use in reports, speech, or websites. NLG can be used to automate the production of tedious and monotonous text to reduce human workload, as writing is a skill which can only be mastered by human beings. For example, generating summaries of the weather reports or Wall Street news on daily basis is a monotonous task that requires analysis, and the output is text that is going to be used. Text generation is a popular task in the natural language processing area. It remains challenging for generative models to generate coherent texts. In this project, I focus on lyrics generation, which takes a title content and an artist as input, and outputs a paragraph of lyrics content. Specifically, this is a conditional language modeling problem. Given the words so far, and also some other input, it predicts the next word based on the given information and generate new text.

Most recently studies on the lyric generation comes with various conditions. Watanabe [2018] presents a novel, data-driven language model that produces entire lyrics for a given input melody. DeepBeat[2016] is introduced as a technique to generate rap lyrics with rhymes. However, few work focus on generating lyrics automatically given an artist style or a topic for the lyrics. I am interested in analyzing the formal topics of rock lyrics and in developing a model that can lead to generating artistic work. Figure 1 shows a simple example of essay generation with multiple topic words. Figure 1 shows a simple example of lyrics generation with artist and title. To model the lyrics generation, I implement the LSTM recurrent neural network as the baseline. To add the artist feature to the generation model, I propose the sequence to sequence model to see whether my model can capture different features for different artists and titles.

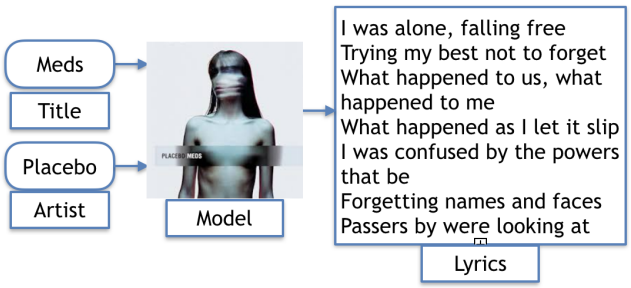

Figure 1: Example for lyrics generation with artist and title

The project is organized as follows. Related work on natural language generation and lyrics generator are listed in section 2. In section 3, I start with a brief discussion of lyrics genre analysis. Next, I introduce the data set that I have used, and describe the preparation of the training data set. Section 4 introduces Recurrent Neural Network, long short-term memory unit and gated recurrent unit as the main methods I used in my lyrics generation model. I start in Section 5 by defining the framework of the proposed neural language generator. I introduce the two main models I implemented for the experiment: LSTM RNN model and sequence to sequence model. Details of the training process and experiment results are described in section 6, followed by the discussion of the results. The final section concludes the work with a brief summary and future works.

2 RELATED WORK

The study of human-generated lyrics is relevant for various subfields of computers science. Relevant literature can be found under domains of computational creativity, information extraction and natural language processing. Additionally, methods can be found under the domain of machine learning, for instance, in the emerging field of deep learning. Recent advances in recurrent neural network-based language models [2010] has demonstrated the value of distributed representations and the ability to model arbitrarily long dependencies. Zhang and Lapata [2014] also describe an interesting work using RNNs to generate Chinese poetry. LSTM (Long Short-Term Memory) units solved this vanishing gradient problem[2001].

Lyric generation is a specific part of text generation. Many paper focused on generating a lyric for specific author [2012] or specific style like rap . For these specific tasks, DopeLearning [2016] focuses on rhyme patterns and recombines lines from rap songs.Castro [2018] proposes a mechanism for combining two separately trained language models into a framework that is able to produce output respecting the desired song structure, while providing a richness and diversity of vocabulary that renders it more creatively appealing. These years, generative adversarial network (GAN) [2014] has shown an amazing performance in image generation and thus recently, many people began to apply GAN for text generation. LeakGAN, a new framework, is proposed to address the problem for long text generation [2018]. Lyric Generation with Style proposed a GAN-like framework with hierarchical generator and encoder to generate lyrics given a specified style and topic pair. Another direction of generating the lyrics is generating lyrics based on input melodies[2018]. The structure is a language model conditioned on a featurized melody. Choral harmonisation[2005] is generated after learning chorales by Bach using a Hidden-Markov model(HMM), which is one of the most popular methods to model and predict sequences. Choi [2016]introduces applications of character-based and word-based RNNs with LSTM units for the automatic generation of jazz chord progressions and rock music drum tracks.

3 DATASET

In this section, I first compare the difference of lyrics among different music genre. This information will be the basis for extracting useful features for generating lyrics.

I use 380,000 lyrics in different genre, analyze them by word frequencies and sentence length, and visualize the most frequent word used in word cloud.

This information will explain the major features of generating the lyrics words and sentences. For further information, the data is available from https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics.

3.1 Lyrics by genre

Since words like 'i', 'you', and 'the' are commonly used throughout all genres, all the consecutive statistics exclude them(together with some other common words). In order to see what are the differences between genres for each of these words, I plot the genre specific statistics with exact counts for each word.

'Love' is the most used of the uncommon words in most of the genres, while each genre has its own unique word, such as 'shit' and 'fuck' in Hip-pop, 'once' in R&B, and 'away' in Rock.

3.2 Rock Lyrics

Since this project is focused on the lyrics generation on Rock, a typical rock song follows a pattern of alternating verses and choruses. The length of sentences and the word frequency also varies from different topics.

3.3 Data

I compiled a list of English-speaking rock artists by my personal preference and scraped all their songs available from a lyrics sitehttps://www.lyricsfreak.com/. In total, I have around 300000 lines from 8500 songs. I then process a subset of those lyrics by removing the headers, removing unnecessary punctuation and white space, and lowercasing all the alphabet characters. I save the clean lyrics in a data frame, with columns as "artists", "titles", and "lyrics". The lyrics are stored in both line base and paragraph base for future use **??**.

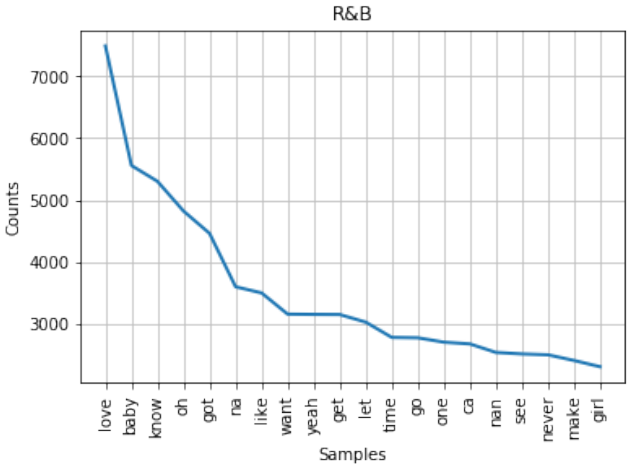

Figure 2: Most frequently used words in R&B.

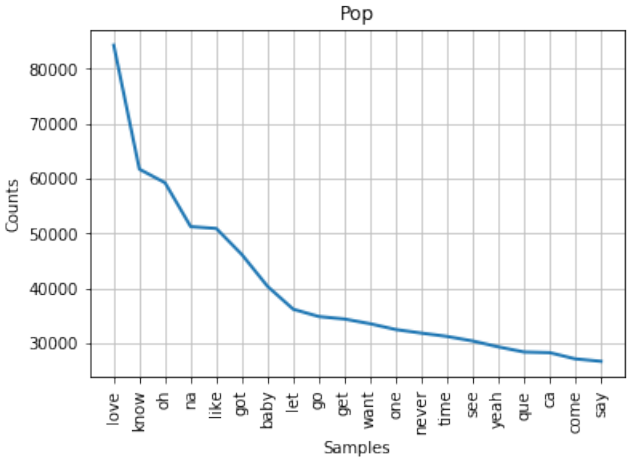

Figure 3: Most frequently used words in Pop.

title	artist	result
Girl with one eye	Florence and The Machine	I say hey girl with one eye, cut your filthy finger out of my eyes

Table 1: An example of the data frame

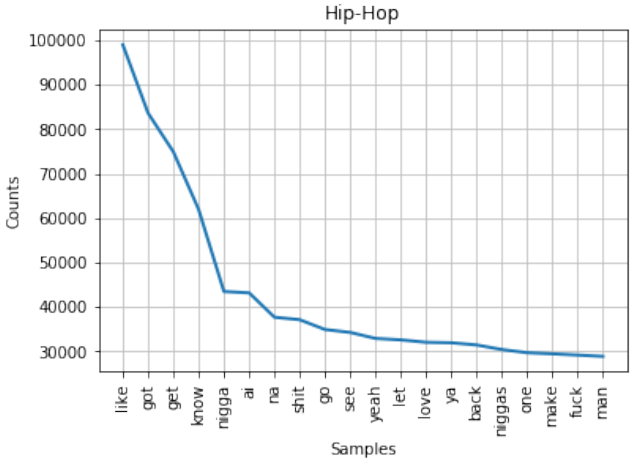

Figure 4: Most frequently used words in Hippop.

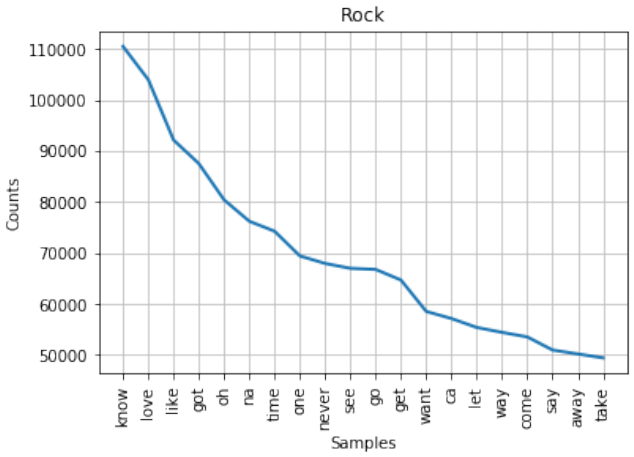

Figure 5: Most frequently used words in Rock.

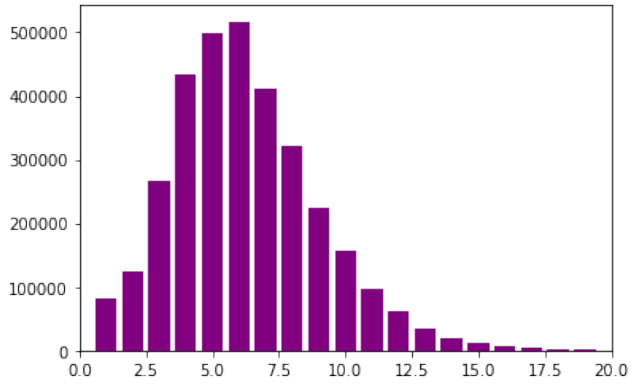

Figure 6: Sentence length of Rock

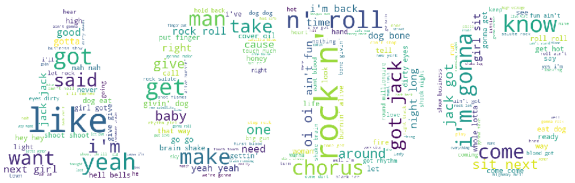

Figure 7: Most common words for AC DC

Figure 8: Most common words for Fall Out Boy

4 METHODS

4.1 Recurrent Neural Network

A recurrent neural network (RNN) is an extension of a conventional feedforward neural network, which is able to handle a variable-length sequence input. The RNN has a recurrent hidden state whose activation at each time is dependent on that of the previous time,

$$h_t = \begin{cases} 0, & \text{t=0} \\ \phi(h_{t-1}, x_t), & \text{otherwise} \end{cases} \quad (1)$$

where ϕ is a nonlinear function such as composition of a logistic sigmoid with an affine transformation. Optionally, the RNN may have an output $y = (y_1, y_2, ..., y_T)$ which may again be of variable length. the recurrent hidden state is updated as

$$h_t = g(Wx_t + Uh_{t-1}), \quad (2)$$

where g is a smooth function such as a logistic sigmoid function or a hyperbolic tangent function.

A generative RNN outputs a probability distribution over the next element of the sequence, given its current state h_t, and this generative model can capture a distribution over sequences of variable length by using a special output symbol to represent the end of the sequence. The sequence probability can be decomposed into

$$p(x_1, ..., x_T) = p(x_1)p(x_2|x_1)...p(x_T|x_1, ..., x_{T-1}), \quad (3)$$

However, it is difficult to train RNNs to capture long-term dependencies [1994] because the gradients tend to either vanish or explode. More sophisticated activation function such as LSTM[1997]and GRU [2014] are designed to solve the problem.

4.2 Long Short Term Memory Unit

Long Short-term Memory is a recurrent neural network architecture which uses a vector of memory cell $c_t \in R^n$, and a set of element-wise multiplication gates to control how information is stored, forgotten, and exploited inside the network. Of the various different connectivity designs for an LSTM cell[2013], the architecture used in this paper is illustrated and defined by the following equations,

$$i_t = \sigma(W_{\omega i} w_t + W_{hi} h_{t-1}) \quad (4)$$

$$f_t = \sigma(W_{\omega f} w_t + W_{hf} h_{t-1}) \quad (5)$$

$$o_t = \sigma(W_{\omega o} w_t + W_{ho} h_{t-1}) \tag{6}$$
$$\hat{c}_t = \tanh(W_{\omega c} w_t + W_{hc} h_{t-1}) \tag{7}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \hat{c}_t) \tag{8}$$
$$h_t = o_t \cdot \tanh(c_t) \tag{9}$$

where σ is the sigmoid function, $i_t, f_t, o_t \in [0,1]^n$ are input, forget, and output gates respectively, and $\hat{c}_t$ and $c_t$are proposed cell value and true cell value at time t.

4.3 Gated Recurrent Unit

A gated recurrent unit (GRU) was proposed by Chung[2014] to make each recurrent unit to adaptively capture dependencies of different time scales. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cells. The activation h_t^j of the GRU at time t is a linear interpolation between the previous activation h_{t-1}^j and the candidate activation $\hat{h}_t^j$:

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j \tag{10}$$

where an update gate z_t^j decides how much the unit updates its activation, or content. The update gate is computed by

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j \tag{11}$$

The candidate activation $\hat{h}_t^j$ is computed as follows,

$$\hat{h}_t^j = tanh(W x_t + U(r_t \odot h_{t-1}))^j \tag{12}$$

where r_t is a set of reset gates and $\odot$ is an element-wise multiplication. The reset gate r_t^j is computed similarly to the update gate:

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \tag{13}$$

5 NETWORK ARCHITECTURE

I propose two models for this task. First, I implement a traditional long short-term memory recurrent neural network to generate complex sequences with long-range structure. This approach is considered as the baseline of the text generation tasks. Secondly, I develop a generation model based on the sequence to sequence model [2014]. It contains two components, an encoder, and a decoder.

5.1 LSTM Model

The prediction architecture of the recurrent neural network is presented in Figure 9 [2013] .

An input vector sequence $x = (x_1, ..., x_T)$ is passed through weighted connections to a stack of N recurrently connected hidden layers to compute first the hidden vector sequences $h^n = (h_1^n, ..., h_T^n)$ and then the output vector sequence $y = (y_1, ..., y_T)$. Each output vector y_t is used to parameterise a predictive distribution $Pr(x_{t+1}|y_t)$ over the possible next inputs x_{t+1}.

Given the hidden sequences, the output sequence is computed as follows:

$$y\hat{}t = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n \tag{14}$$

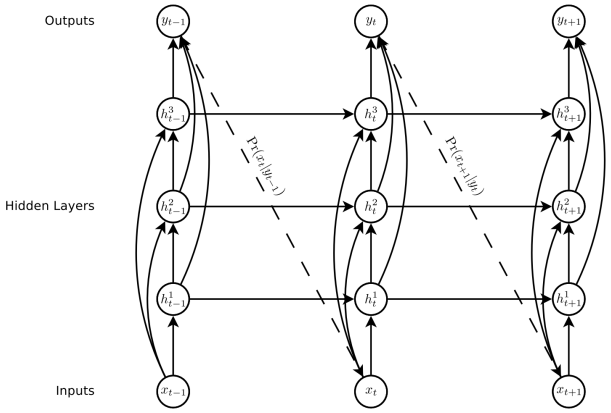

Figure 9: Deep recurrent neural network prediction architecture

$$y_t = \mathcal{Y}(\hat{y}_t) \tag{15}$$

where $\mathcal{Y}$ is the output layer function.

The probability given by the network to the input sequence x is

$$Pr(x) = \sum_{t=1}^{T} Pr(x_{t+1}|y_t) \tag{16}$$

The sequence loss $\mathcal{L}(x)$ used to train the network is the negative logarithm of Pr(x):

$$\mathcal{L}(x) = -\prod_{t=1}^{T} logPr(x_{t+1}|y_t) \tag{17}$$

The partial derivatives of the loss with respect to the network weights can be efficiently calculated with backpropagation through time[1995] applied to the computation graph shown in Figure 9, and the network can then be trained with gradient descent[2013].

5.2 Seq2seq Model

The architecture is presented in Figure 10 and Figure 11 [2013] .

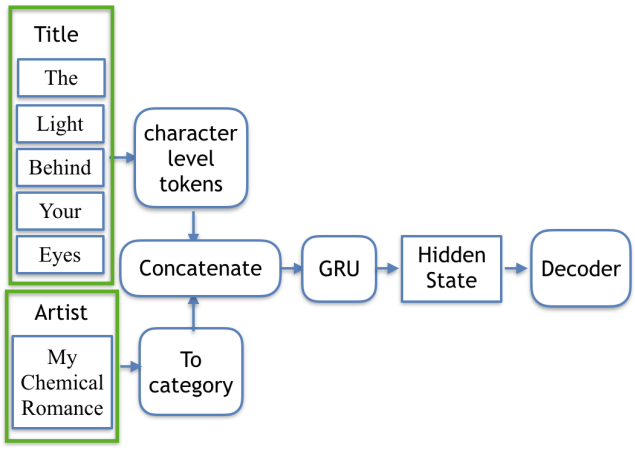

Figure 10: Encoder architecture

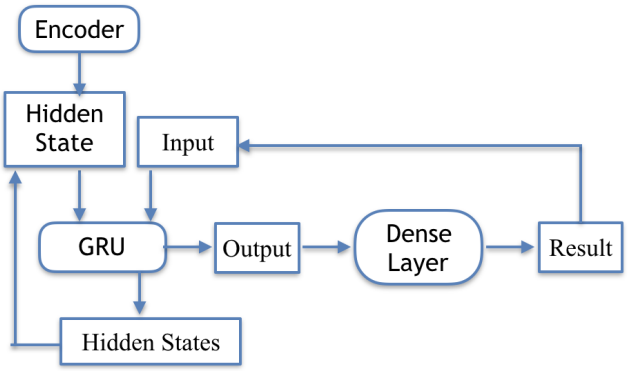

Figure 11: Decoder architecture

Model	Output
Seq2seq	i can see you and i want you this is my life in your eyes you cant hear you make me love you this is the same where you leave me baby <pad>
LSTM	hat the back in the kidd so taat i cont liow what i dont lese yhu ut iedo reat dare ceneed ie a coowt oeke the same to tae od io oe kine to the eon dndnt and the simet that i have to se m man in dyen marem mer mo men ies iosd befone

Table 2: Output Sample

Model	LSTM	Seq2seq
Rouge Score	0.18	0.26

Table 3: Rouge Score of Prediction Sample

In this model, I focus on generating a song lyric given an artist and a title. Different from the previous LSTM model, which only has one input vector sequence, the input of Seq2seq model is consisted of two parts: a title as a text content, and an artist as a category. This model has two components, a character level encoder, and a decoder. Both of the encoder and the decoder contain a RNN. The encoder maps the input sequence to a fixed-sized vector using one RNN, and the decoder maps the vector to the target sequence with another RNN.[2014]. To get better result with long dependency, I choose GRU as the RNN unit, and each hidden layer contains 128 hidden units.

The goal of the GRU is to estimate the conditional probability $p(y_1, ..., y_{T'}|x_1, ..., x_T)$ where $x_1, ..., x_T$ is the input text sequence concatenated by the artist category, and $(y_1, ..., y_{T'}$ is its corresponding output sequence. The GRU computes this conditional probability by first obtaining the fixed dimensional representation v of the input sequence $(x_1, ..., x_T)$ given by the last hidden state of the GRU, and then computing the probability of $(y_1, ..., y_{T'}$ with a standard Language Model [2003] formulation whose initial hidden state is set to the representation v of $x_1, ..., x_T$ [2014] :

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \sum_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1}) \quad (18)$$

In this equation, each $p(y_t|v, y_1, ..., y_{t-1})$ distribution is represented with a softmax over all the characters in the vocabulary. To decode the sequences of all possible length, each target sentence starts with a special start-of-sentence symbol \t and ends with an end-of-sentence symbol \n.

6 EXPERIMENTS, RESULTS, AND DISCUSSION

6.1 Experimental Settings

For the LSTM recurrent neural networks, I load the first 150,000 characters as tge training set. I split the lyrics text up into subsequences with a fixed length of 100 characters, an arbitrary length. Each training pattern of the network is comprised of 100 time steps of one character (X) followed by one character output (y). Next, I to rescale the integers to the range 0-to-1 to make the patterns easier to learn by the LSTM network that uses the sigmoid activation function by default. Finally, I convert the output patterns (single characters converted to integers) into a one hot encoding, so that I can configure the network to predict the probability of each of the different characters in the vocabulary. To define the LSTM model, I use a single hidden LSTM layer with 256 memory units with a dropout of 0.2. The output layer is a Dense layer using the softmax activation function to output a probability prediction for each of the characters between 0 and 1.

Due to the time and memory limitation of the computation, I used 5000 lyrics of the whole data set for the sequence to sequence model. Considering this problem as a single character classification problem with around 35 classes, my goal is to optimize the log loss, which is the categorical cross entropy loss. For the optimization algorithm, I use ADAM for its speed. I use a modest number of 30 epochs and a batch size of 64 patterns.

6.2 Results

Figure 2 is one output sample for two models. From the output, we can see that there are more spelling mistakes in the LSTM model. The sequence to sequence model performs better than LSTM model.

6.3 Recall-Oriented Understudy for Gisting Evaluation

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially of a set of metrics for evaluating automatic summarization of texts as well as machine translation. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced). ROUGE is based on n-gram overlap, and the scores are reported separately for each n-gram:

$$ROUGE\text{–}N = \frac{\sum_{S \in ReferencesSummaries} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in ReferencesSummaries} \sum_{gram_n \in S} count(gram_n)}$$

Table 3 compares the rouge score of two models.

7 DISCUSSION AND FUTURE WORK

For future work, I can try tuning the hyper-parameters of the model to improve the result. From my experiments, I find that it is easier

to implement text generator on character-level than word-level. I only trained the word-level generator with a small vocabulary size due to the memory limitation. I can train the generator with larger vocabulary size and bigger decoder output length to get better results for the word-level generator. Besides, more time is needed for training the models.

Text generation is still a challenging task in the natural language processing area. There are other learning methods, such as SeqGAN [2017], an adversarial learning method using reinforcement learning. New structures need to be proposed and improved for the text generation task. In a generative adversarial networks(GANs) framework, there are two adversaries, a discriminator and a generator. The generator forges samples that are intended to be from the same distribution as the training data. The discriminator is a binary classifier striving to distinguish the generated fake samples from the real ones. In this project, different recurrent architectures of the generator and discriminator can be investigated. Considered GANs have been successfully applied in computer vision to generate high-resolution and realistic images, but not so successful in text generation models, this approach will be an attempt.

REFERENCES

[1] allan2005harmonisingAllan, M. Williams, C. 2005. Harmonising chorales by probabilistic inference Harmonising chorales by probabilistic inference. Advances in neural information processing systems Advances in neural information processing systems (25–32).

[2] inproceedingsBarbieri, G., Pachet, F., Roy, P. Degli Esposti, M. 201208. Markov Constraints for Generating Lyrics with Style Markov constraints for generating lyrics with style. (242). 10.3233/978-1-61499-098-7-115

[3] bengio2003neuralBengio, Y., Ducharme, R., Vincent, P. Jauvin, C. 2003. A neural probabilistic language model A neural probabilistic language model. Journal of machine learning research3Feb1137–1155.

[4] bengio1994learningBengio, Y., Simard, P., Frasconi, P. . 1994. Learning long-term dependencies with gradient descent is difficult Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks52157–166.

[5] castro2018combiningCastro, PS. Attarian, M. 2018. Combining Learned Lyrical Structures and Vocabulary for Improved Lyric Generation Combining learned lyrical structures and vocabulary for improved lyric generation. arXiv preprint arXiv:1811.04651.

[6] cho2014learningCho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[7] choi2016textChoi, K., Fazekas, G. Sandler, M. 2016. Text-based LSTM networks for automatic music composition Text-based lstm networks for automatic music composition. arXiv preprint arXiv:1604.05358.

[8] chung2014empiricalChung, J., Gulcehre, C., Cho, K. Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[9] goodfellow2014generativeGoodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S.Bengio, Y. 2014. Generative adversarial nets Generative adversarial nets. Advances in neural information processing systems Advances in neural information processing systems (2672–2680).

[10] graves2013generatingGraves, A. 2013. Generating sequences with recurrent neural networks Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

[11] guo2018longGuo, J., Lu, S., Cai, H., Zhang, W., Yu, Y. Wang, J. 2018. Long text generation via adversarial training with leaked information Long text generation via adversarial training with leaked information. Thirty-Second AAAI Conference on Artificial Intelligence. Thirty-second aaai conference on artificial intelligence.

[12] hochreiter2001gradientHochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J. . 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A field guide to dynamical recurrent neural networks. IEEE Press.

[13] hochreiter1997longHochreiter, S. Schmidhuber, J. 1997. Long short-term memory Long short-term memory. Neural computation981735–1780.

[14] malmi2016dopelearningMalmi, E., Takala, P., Toivonen, H., Raiko, T. Gionis, A. 2016. Dopelearning: A computational approach to rap lyrics generation Dopelearning: A computational approach to rap lyrics generation. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (195–204).

[15] mikolov2010recurrentMikolov, T., Karafiát, M., Burget, L., Černockỳ, J. Khudanpur, S. 2010. Recurrent neural network based language model Recurrent neural network based language model. Eleventh annual conference of the international speech communication association. Eleventh annual conference of the international speech communication association.

[16] sutskever2014sequenceSutskever, I., Vinyals, O. Le, Q. 2014. Sequence to sequence learning with neural networks Sequence to sequence learning with neural networks. Advances in NIPS.

[17] watanabe2018melodyWatanabe, K., Matsubayashi, Y., Fukayama, S., Goto, M., Inui, K. Nakano, T. 2018. A melody-conditioned lyrics language model A melody-conditioned lyrics language model. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers) (163–172).

[18] williams1995gradientWilliams, RJ. Zipser, D. 1995. Gradient-based learning algorithms for recurrent Gradient-based learning algorithms for recurrent. Backpropagation: Theory, architectures, and applications433.

[19] yu2017seqganYu, L., Zhang, W., Wang, J. Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient Seqgan: Sequence generative adversarial nets with policy gradient. Thirty-First AAAI Conference on Artificial Intelligence. Thirty-first aaai conference on artificial intelligence.

[20] zhang2014chineseZhang, X. Lapata, M. 2014. Chinese poetry generation with recurrent neural networks Chinese poetry generation with recurrent neural networks. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp) (670–680).