Adversarial Rap Lyric Generation

Yun-Yen Chuang
Department of Electrical Engineering
Nation Taiwan University
Taipei, Taiwan
Email: chuang.yun.yen@gmail.com

Hung-Min Hsu
Department of Electrical Computer and Engineering
University of Washington
Seattle, USA
Email: hmhsu@uw.edu

Ray-I Chang
Department of Engineering Science and Ocean Engineering
Nation Taiwan University
Taipei, Taiwan
Email: rayichang@ntu.edu.tw

Hung-Yi Lee
Department of Electrical Engineering
Nation Taiwan University
Taipei, Taiwan
Email: hungyilee@ntu.edu.tw

2022 4th International Conference on Natural Language Processing (ICNLP) | 978-1-6654-9544-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICNLP55136.2022.00077

***Abstract*—In this paper, we design a novel text generation method called RapGAN for RLG (Rap Lyric Generation). RapGAN proposes two new schemes, PRO (Phrase Roll-Out) and AREGS (Attention Reward at Every Generation Step), to achieve efficient a nd e ffective R LG. T he c onventional t ext generation methods need to complete the entire-generated sentence in GAN (Generative Adversarial Network) training. RapGAN uses PRO as selective attention to reward only meaningful phrases in every generation step. Due to the varying length of phrases segmented by PRO, we employ an attention-based neural network that can capture the local feature of phrases and the semantics of global sentences as our discriminator. Therefore, we propose AREGS that reward the alignment weights of the attention-based discriminator with feature matching. Our method can provide different weights on different words to improve RLG's diversity, originality and fluency. E xperimental r esults s how t hat our RapGAN outperforms the state-of-the-art RLG methods on many essential metrics of text generation. We also show the human evaluation to illustrate the relations between human perception and typical NLP metrics in diversity, originality and fluency. Our proposed method can facilitate real-world applications in content marketing. To facilitate future researchers, we also provide an open RLG dataset in this paper.**

***Keywords*—Language generation, Generative Adversarial Network, Rap Lyric, Machine Learning**

I. Introduction

The goal of the RLG (rap lyric generation) problem [1] is to generate rap lyrics that are similar but not identical to existing ones. As text generation, a RLG system will generate the rap lyric word-by-word including End-of-Sentence (EOS) and End-of-Lyric (EOL) to stand for the end of a sentence and the whole lyric, respectively. However, different from poem generation [2] with restricted formats of short sentences, RLG requires free formats of long sentences with diversity, originality and fluency. I t i s a h ard c hallenge t o e nsure that the generated text is coherent and semantically consistent in RLG. Early RLG methods [3]–[7] usually rely on experts' rules to generate lyrics. [3]–[7] rely on n-gram language model and hidden Markov model (HMM) with human-designed rhyme schemes to generate lyrics. These language model approaches are very efficient in computation. However, as their characteristics and styles are pre-built by some human rules, the generated lyrics are limited. To resolve this drawback, Ghost Writer [1] automatically learn the patterns of structure, rhythm, and tone for lyrics generation. Ghost Writer [1] uses a deep learning LSTM (long short-term memory) architecture to RLG. It utilized the maximum likelihood estimation (MLE) to train a sequence generative model for LSTM [8], [9]. As these MLE-trained LSTM (MLE-LSTM, for short) methods are suffered from exposure bias [10], the generated rap lyrics are similar to each other resulting in the lack of originality. Therefore, their styles [1] (such as vocabulary, rhythm density and lyrical content) are restricted, the generated lyrics are poor.

The generative adversarial network (GAN) has shown to outperform MLE-LSTM in originality of text generation [8], [11], [12] and has not been applied in RLG. However, GAN has two drawbacks. (1) GAN's roll-out process of Monte Carlo tree search (MCTS) is computationally expensive [11]–[13]. (2) GAN has the mode collapse problem [14], [15] in training [10], [15]–[17]. As the generator maps a large fraction of probable regions to only a few low-volume regions, GAN's results may be monotonous with low diversity.

To resolve the two drawbacks of GAN for RLG, We propose a new method called RapGAN for RLG applications. Our RapGAN proposes PRO (Phrase Roll-Out) streamline the GAN training procedure by rewarding meaningful phrases instead of roll-out complete sentences like SeqGAN [10]. For rewarding various length meaningful phrases, we employ an attention-based network which can capture both the local feature of phrases and semantics of global sentences as our discriminator [18], [19]. As shown in [8], [8], [11], [15], [20], the feature matching scheme that uses an intermediate layer in the discriminator can prevent the system from mode collapse. To protect GAN from mode collapse and increase the originality and the diversity in text generation, we propose a new feature matching scheme AREGS (Attention Reward at Every Generation Step), collocating with our attention-based discriminator. By integrating PRO and AREGS, our RapGAN can achieve good RLG with diversity, originality and fluency.

978-1-6654-9544-8/22/$31.00 ©2022 IEEE
DOI 10.1109/ICNLP55136.2022.00077

It motivates us to propose AREGS (Attention Reward at Every Generation Step) that uses alignment weights in the attention layer as a new objective function to match the statistics of real data features.

In our experiments, we compare RapGAN with three major conventional methods, including MaliGAN [17], SeqGAN [10], and Ghost Writer [1]. In this paper, we provide an open RLG dataset to facilitate the future researchers. Experimental results show that RapGAN has achieved the best performance on many important metrics of text generation. We also use the human evaluation (in diversity, originality and fluency) as the ground truth to assess the quality of RLG. The relations between the human evaluation and typical NLP metrics are also investigated to recommend the suitable RLG metrics.

II. Related Works

GAN consists of a generator and a discriminator for machine learning. Assume that x is a sampled data, and y is its next sentence in the training data. Given x, G learns to generate y in conditional sequence generation [13], [21], [22]. D takes x and y as inputs, and outputs their score $D(x, y)$. Use $\hat{y}$ and y to represent data generated by G and human, respectively. D learns to maximize score $D(x, y)$ while minimizing the score $D(x, \hat{y})$. G learns to generate $\hat{y}$ that maximizes the discriminator score $D(x, \hat{y})$. GAN [23], [24] is shown to learn the hidden characteristics of data, and can generate its original data after training . It motivates some GAN-based text generation methods proposed in past years to generate lyrics with more originality [25]–[27]. However, GAN has difficulties in dealing with discrete data (such as text) because the discrete sequences are not differentiable [10], [15]–[17]. If the generated data is based on discrete data, the "slight change" guidance from the discriminator makes little sense because there is probably no corresponding token for such slight change in the limited dictionary space [23]. To overcome the non-differentiable problem, SeqGAN [10] proposes policy gradient inspired by reinforcement learning to address this issue. SeqGAN also adopts MCTS to consider the fitness of previous tokens (prefix) and the resulting future outcome.

However, SeqGAN needs to roll out starting from the current word till the end of the sequence to get a batch of output samples in every generation step. SeqGAN is computationally expensive. Note that, as it requires generating N word sequences, SeqGAN is computationally expensive. To streamline the training procedure, [13] uses the partially decoded sequence (PDS) to assign rewards. [13] trains the discriminator with the randomly-segmented subsequences from the generator and human, respectively. Then, [13] can update the generator to prevent the time-consuming of each prefix of each sequence from the roll-out policy and the MCTS. As randomly-segmented subsequences are used in training, they may learn and generate meaningless phrases. Previous studies [13] show that its generated results are not acceptable for human.

III. Proposed Method

The drawbacks of [10], [13] motivate us to design PRO with a new PDS function to consider only meaningful phrases in training. Our PRO roll out according to meaningful phrases to reduce computing time and produce better fluency in RLG. As RLG requires not only the fluency, the originality and the diversity are essential for this application. We propose AREGS to utilize an attention layer as the discriminator to obtain originality and diversity results.

A. Phrase Roll out

Given x, the generator learns to generate y in conditional sequence. y is the next sentence of x. Assume that $x = x_{1:M} = (x_1, x_2, ..., x_t, ..., x_M)$ and $y = y_{1:T} = (y_1, y_2, ..., y_t, ..., y_T)$, where x_t and y_t are words at time t. M and T are the numbers of words in these two sentences.

1) Training the Discriminator: The discriminator, denoted as D, aims to distinguish real data and generated data, while the generator is trained to fool the discriminator by sampling high-quality generated data. For a sequence y, the PDS function $\rho(y)$ randomly splits y into subsequences. Given $\rho(y)$, the model keeps sampling tokens from the distribution until the decoding is finished. For example, if y is the sentence "I am doing wonderful, thank you." with 8 words ("I", "am", "doing", "wonderful", ",", "thank", "you", "."). Its subsequence $\rho(y)$ can be one of 8 possible subsequences ("I", "I am", ..., "I am doing wonderful, thank you."). However, updating the discriminator with a meaningless phrase like ("I am") from $\rho(y)$ is not reasonable.

In past years, some tools are proposed to segment sentences into meaningful phrases [28]–[30]. However, they are not applied to RLG. Our PRO uses a tool [30] to segment the sentence y into a sequence of meaningful phrases $\rho'(y) = (p_1, p_2, ..., p_{t'}, ..., p_{T'})$ where T' and t' are the number and index of meaningful phrases, respectively. It can filter out the meaningless phrases for better training. We train the discriminator as follow:

$$\min_D -\mathbb{E}_{(x,y)\sim P(x,y)}[\log D(x, \rho(y)]- \mathbb{E}_{x\sim P(x),\hat{y}\sim P(y|x)}[\log(1 - D(x, \rho(\hat{y})))] \quad (1)$$

Our discriminator learns meaningful phrases from TextRank [30]. It can avoid meaningless phrases for maintaining the fluency of generated sentences.

2) Training the Generator: In SeqGAN, the purpose of sampling is to estimate a complete and optimal value function with roll-out. SeqGAN produces the step-dependent value $Q(x, \hat{y}_{1:t})$ by $D(x, \hat{y})$ with the roll-out policy and the MCTS. SeqGAN roll-out from current word till the end of the sequence. The current step of the word is n and the step of end of the sequence is N. To complete the training [9], [10], SeqGAN samples N word sequences $\{\hat{y}^1_{t+1:T}, \hat{y}^2_{t+1:T}, \ldots, \hat{y}^N_{t+1:T}\}$ to

make $\hat{y}_{1:T} = (\hat{y}_{1:t}, \hat{y}^n_{t+1:T})$ for n = 1 to N. The reward term $Q(x, \hat{y}_{1:t})$ can be written as follows.

$$Q(x, \hat{y}_{1:t}) = \begin{cases} \frac{1}{N} \sum_{n=1}^{N} D(x, \hat{y}^n_{t+1:T}) & \text{for } t < T \\ D(x, \hat{y}_{1:t}) & \text{for } t = T. \end{cases} \quad (2)$$

After obtaining the $\rho'(y)$, it is unnecessary to roll out in every generation step in our PRO like [13]. However, the training procedure of training without roll-out produces worse performance [13]. Note that, We propose PRO roll-out samples with the same length as the segmented meaningful phrases without roll-out complete sentences like SeqGAN [10]. Each $p_{t'}$ from $\rho'(y)$ have start and end step corresponding to steps of y. We denote the start step and end step of $p_{t'}$ as t'_{start} and t'_{end}, respectively. The reward function of our PRO can be written as follows:

$$Q(x, \hat{y}_{1:t}) = \begin{cases} \frac{1}{N} \sum_{n=1}^{N} D(x, (\hat{y}^n_{t+1:t'_{end}})) & \text{for } t'_{start} \leq t < t'_{end} \\ D(x, \hat{y}_{1:t'_{end}}) & \text{for } t = t'_{end}. \end{cases} \quad (3)$$

Where $t'_{start} \leq t < t'_{end}$ which means the y_t is segmented to belong $p_{t'}$, we make generator rolls out from the current step t until the end step t'_{end} to estimated reward by D.

B. AREGS

By protecting GAN from mode collapse [8], [11], [12], [15], [23], we propose AREGS to increase the originality and the diversity of the generator. AREGS use attention alignment weights to guide the generator training instead last layer output of the discriminator $D(x, y)$ directly as the generator reward. [18], [19], [31] indicates the attention-based neural network can capture both the local feature of phrases as well as global sentence semantics. The attention-based discriminator can score the phrases in various chunk sizes in our PRO. First, we propose to apply the attention-LSTM [18] in our discriminator. Attention-LSTM consists of three components: hidden dimensions from LSTM (H), and attention alignment weights (α) and representation (r). Given an input sequence x, we define H as the output vectors of the LSTM layer $\{h_1, h_2, ..., h_T\}$ where T is the sentence length. The attention alignment weight $\alpha = [\alpha_1, \alpha_2, ..., \alpha_T]$ is obtained by

$$\alpha = \text{softmax}(w^T * \tanh(H)), \quad (4)$$

where w is the network parameters to be learned. The representation r of the input sequence is formed by a weighted sum of the output vectors H.

$$r = \alpha^T H. \quad (5)$$

We assign context sentence x and predicted sentence y to the attention-LSTM layer. Then, we obtain the representations rep_x and rep_y from the attention-LSTM layer. rep_x and rep_y are the representations of the context sentence x and the predicted sentence y through the attention-LSTM layer. The two representations rep_x and rep_y are concatenated as the final representation $\{rep_x, rep_y\}$ to obtain the final output scalar $D(x, y)$.

Second, different from the traditional approaches [10], [17] in training the generator, we do not use the last layer output of the discriminator $D(x, y)$ directly as the generator reward. Inspired by feature matching [15], [16], we use attention alignment weights α to guide the generator training. It matches the statistics of real data to prevent the system from mode collapse. The generator learns to generate sequences with α similar to real sequences. Given the real data (x, y^*) and generated data $(x, \hat{y})$, assuming that the attention weights obtained in D are α^* and $\hat{\alpha}$ respectively, we compute the similarity of α^* and $\hat{\alpha}$ using cosine similarity [15], denoted as $S(\alpha^*, \hat{\alpha})$. $S(\alpha^*, \hat{\alpha})$ is then used to replace $D(x, \hat{y})$ when training the generator. In this paper, we combine these two new schemes, PRO and AREGS, and introduce a novel text generation method called RapGAN to achieve efficient and effective RLG.

IV. Evaluation

A. Rap lyrics data sets

We train and evaluate the models using an Nvidia K80 GPU and a 32-core Intel CPU. The applied training dataset is collected from Internet with 160,000 Chinese songs, and provided as an open RLG dataset to facilitate the future researchers. Our dataset and code are released on [1] in this paper. We collected all lyrics on the Nestease website that have a "rap" tag. In addition to the release albums of hip-hop singers themselves, it also contains many musical works produced by hip-hop music programs. There are 3,214 creators in total and each creator has about 5 songs in our dataset. To filter out the uncommon words, we set the threshold of word frequency as 3 to obtain 89,736 unique words in our corpus. The average length of each song is 813 characters and 126 words; the maximum length of each sentence of each song is 20 sentences and the average word length in each sentence is 9. Our experiments split these 160,000 songs into two parts. There are 80,000 songs for training and the other 80,000 songs for testing.

B. Evaluation Metric

In this paper, we use the following four metrics to evaluate the rap lyrics. In our experiments, we compare RapGAN with three major conventional methods, including MaliGAN [17], SeqGAN [10], and Ghost Writer [1]. Four well-known metrics, including Sim-EL (similarity to existing lyrics) [1], BLEU [9], [32], Self-BLEU (bilingual evaluation understudy) [9], [32], and RD (Rhythm-Density) [1]), are applied to make the performance evaluation. In these metrics, RD is applied to measure the fluency of the generated sentences.

- **Sim-EL** [1]: evaluates the originality by the inverse document frequency and max similarity. Sim-EL is based on the well-known Inverse Document Frequency and uses cosine similarity to calculate distance. Lower scores are

[1] https://github.com/deepseqgan/RAPGAN

better. First, we build the Term-Document Matrix with weights for each term in each song.

$$w_{ij} = f_{ij}\frac{N}{n_j} \tag{6}$$

where N is the total number of documents; n_j is the number of verses that contains term j; f_{ij} is the frequency of term j in the ith verse. By using this matrix, we then can use the cosine similarity to calculate the distance between verses. The lower value of w_{ij} represents the higher novelty of the generated lyrics. In our experiment, we calculate the Sim-EL between the 80k songs from training data and generated rap lyrics by each rap approach. The lower Sim-EL score is better.

- **BLEU** [9], [32]: is generally applied to measure the quality (including the fluency and the diversity) of sentences in neural machine translation and natural language generation [10], [15], [32], [33]. The definition of BLEU is to calculate the number of n-gram matches sentence by sentence. Then, the total number of clipped n-grams counts are aggregated and divided by the total counts of all possible number of n-grams to represent the precision P_n of this model. The higher BLEU score is better. In our experiment, we adopt the BLEU with 3-gram to represent the precision of each model. The reference sentences are from the 80k songs of the training data.
- **Self-BLEU** [9], [32] is generally applied to measure the diversity of sentences in machine translation and natural language generation [10], [15], [32], [33]. We employ Self-BLEU to evaluate the diversity of the generated rap lyrics. Lower scores are better. Self-BLEU is to evaluate how one sentence resembles the rest in a generated collection. By treating one sentence as the hypothesis and the others as a reference, the BLEU score can be calculated for every generated sentence. A higher Self-BLEU score means less diversity of the generated rap lyrics. Therefore, the lower self-BLEU score is better. In our experiment, we adopt the Self-BLEU with 3-gram to show the performance of each model, and then the reference sentences are selected from the 80k songs of the testing data to calculate self-BLEU.
- **Rhythm Density (RD)** [1]: evaluates the fluency of the generated rap lyrics. RD is defined as the total number of rhymed syllables divided by the total number of syllables. RD means that how effective we are in modeling an artist's style. The higher RD score is better.

C. Training Setting

Our method is generic and can be applied for different languages. In this paper, we use Chinese songs as an example and adopt Jieba [28], [29], [34] for word segmentation. Wiki with skip-grams is applied for our pre-trained word weights. For all comparison methods, we use the generator with 128 neurons and a maximum length of 20 in each sentence. There are maximum length of 10 in each songs. In the discriminators, we also use 128 neurons in hidden layers. The generators in all GAN-based methods [10], [13], [17] are pre-trained with MLE and adversarial training for 10 epochs.

D. The different numbers of keywords

Our RapGAN applies TextRank in PRO for phrase extraction. It uses the top k keywords to extract key phrases. The more keywords we extract, the bigger chunks of the sentence are segmented. In this paper, we experiment with different values of k to segment sentences to various chunk sizes.

As shown in Table I, we tune k from 2 to 10. We find that k = 6 is the best overall. Self-BLEU scores are low until the keywords exceed 6. The reason is that the excessive keywords cause tiny sentence bits leading to mode collapse in the generator. Breaking sentences into smaller segments, however, causes the generator to learn poorly for Chinese rap lyrics with too little partial information. Breaking sentences into larger segments cause the excessive roll-out steps to be suffered from error-propagation [35]. Breaking sentences into meaningful phrases in proper length can effectively improve performance and reduce computing time.

TABLE I
EVALUATION WITH DIFFERENT VALUES OF TEXTRANK PARAMETER k TO CONTROL THE SENTENCE SEGMENT LENGTH.

TextRank parameter k	**Sim-EL**	**BLEU**	**Self-BLEU**	**RD**
k=2	0.2777	0.6527	0.9697	1.0172
k=4	0.2773	0.6824	0.9697	1.0213
k=6	**0.2770**	**0.7211**	**0.9697**	**1.0892**
k=8	0.2771	0.6923	0.9765	1.0244
k=10	0.2770	0.6641	0.9823	1.0072

Our RapGAN is inspired by feature matching to guide the generator training. In this paper, we compare 3 different feature matching (FM) schemes to select the suitable one. They are FM1 (AREGS), FM2 (context-LSTM hidden dimensions [21]), and FM3 (predicted-LSTM hidden dimensions [22]). FM1 is our AREGS with attention alignment weights from context sentence. FM2 and FM3 use the LSTM hidden dimensions proposed from either the context sentence or the predicted sentence, respectively.

For each metric, we do the experiments 10 times. For each time, we randomly choose 8000 start sentences for these feature matching schemes to generate rap songs. Experimental results are shown in Table II with the average value of each metric. All our test methods have improved the Sim-EL score where comparing with the method that calculates the score directly with the discriminator. They degrade the self-BLEU scores. Among all their intermediate layers, FM2 obtains the best self-BLEU among all the methods, which shows the total quality of the generated lyrics. Therefore, our RapGAN uses FM2 (context-attention alignment weights) as the feature matching scheme for RLG.

E. Comparing with Other Methods

After exploring the experimental settings of our RapGAN, we compared the proposed method with other text generation methods in Table III. All GAN-based methods outperform the

TABLE II
EVALUATION WITH DIFFERENT FEATURE MATCHING SCHEMES.

Intermediate layer	Sim-EL	BLEU	Self-BLEU	RD
FM1 (AREGS)	0.2656	**0.6999**	**0.9632**	1.0259
FM2 [21]	**0.2639**	0.5808	0.9766	1.0254
FM3 [22]	0.2649	0.6603	0.9774	**1.0262**

TABLE III
EVALUATION WITH DIFFERENT TEXT GENERATION METHODS FOR RLG.

Model name	Sim-EL	BLEU	Self-BLEU	RD
RapGAN	**0.2656**	**0.6999**	**0.9632**	**1.0259**
Mali-GAN [17]	0.2708	0.6447	0.9802	1.0042
SeqGAN [10]	0.2747	0.6262	0.9952	1.0254
Ghost Writer [1]	0.3107	0.5929	0.9921	0.8976

baseline Ghost Writer approach [1]. For the previous experiment, our RapGAN adopts the FM1 (AREGS) with attention alignment weights from context sentence. For each metric, we do the experiments 10 times to show the average value. Since we use TextRank to extract PDS, we perform partially decoding at the phrase level instead of the word level. We also take advantage of this to streamline our approach. Results show that our RapGAN (takes 224.8k seconds) is much faster than Mali-GAN (takes 656.2k seconds) and SeqGAN (takes 741.5k seconds) for the training time. Our methods can achieve good RLG by enabling the discriminator to score arbitrary-length samples with some bias.

To prove that the results are statistically significant, we compare RapGAN with the all other models by the t-test verification [15]. Results show that all their P-values are smaller than 10^{-6} on all the metrics. They are statistically significant. RapGAN can improve the performance and effectiveness to have the best Sim-EL, BLEU, Self-BLEU and RD.

F. Human Evaluation

In past years, some NLP methods [10], [13], [15], [16], [36] adopt the human evaluation to verify the quality of text generation. Different from the metrics applied in the previous subsection, the human evaluation is based on the perception of the text application. For example, in the text application of Chatbot, the fluency is more important than the originality and the diversity. However, for the text application of RLG, the originality and the diversity are more important than the fluency [37]. To discuss the human evaluations, we invite 40 participants to vote on the lyrics generated by different methods. These participants are 22 to 26 years old including 27 males and 13 females.

Our human evaluation uses multi-turn voting [3], [36] where the results from a pair of methods are given to decide which one is better. In Fig. 1, we summarize the ranking scores of multi-turn voting for different methods to show their fluency, diversity, and originality from the human perception. Randomly-generated lyrics are evaluated to have the medium diversity, but the fluency is very low. From the human perception, we would not accept this kind of lyrics. The diversity and the originality will be considered only when the lyric is readable with at least the medium fluency. On the other hand, the generated rap lyrics of Ghost Writer are evaluated to have the medium fluency. But these lyrics are almost the same and make participants feel of lack originality.

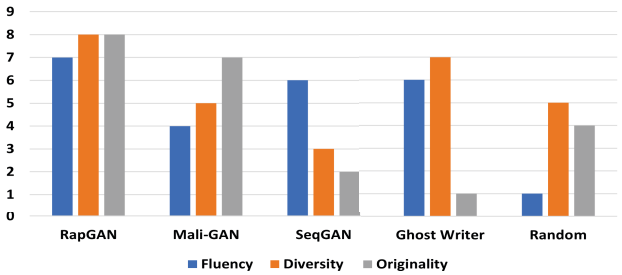

Fig. 1. The human evaluation of originality, diversity, and fluency. The x-axis is ranking scores of multi-turn voting. Higher scores are better.

Previous GAN-based methods, as Mali-GAN and SeqGAN, can yield more original lyrics than Ghost Writer. However, with randomly-generated subsequence in training, they may have consecutive generation of non-meaningful words. The meaningful words is the key to fluency. In typical NLP metrics, Sim-EL and RD are related to the originality and the fluency of text generation. Furthermore, by investigating the relations between human evaluations and typical NLP metrics, the most obvious commonality is the Self-BLEU score. We infer that the Self-BLEU score is a good metric to represent the quality (especially, the diversity) of RLG in human perception. In this case, RapGAN outperforms the previous methods in Sim-EL, BLEU, Self-BLEU and RD scores to show the superior performance of RLG.

V. CONCLUSION

In this paper, we propose RapGAN - a new GAN-based text generation method designed for RLG. In addition, a RLG dataset with 160000 Chinese songs is introduced to facilitate future researches for RLG. We propose and compare different RapGAN methods, and show that the introduced RapGAN scheme can improve the performance of RLG. In our RapGAN, We propose PRO that rolls out with meaningful phrase to streamline the roll-out process and AREGS to prevent RapGAN from mode-collapsed. Moreover, we investigate the relations between human evaluations and typical NLP metrics in RLG. We also discuss the trade-offs between the metric and the human perception. Finally, we achieve a new state-of-the-art performance by comparing to the state-of-the-art RLG methods. There are other special corpus emphases on rhyme and rhythm besides rap lyrics. In the future, we hope to combine our method with speech processing to improve the possibility of generating corpora that emphasize rhyme and rhythm.

REFERENCES

[1] P. Potash, A. Romanov, and A. Rumshisky, "Ghostwriter: Using an lstm for automatic rap lyric generation," in *Proceedings of the Conference on*

Empirical Methods in Natural Language Processing, 2015, pp. 1919–1924.
[2] Z. Wang, W. He, H. Wu, H. Wu, W. Li, H. Wang, and E. Chen, "Chinese poetry generation with planning based neural network," *arXiv preprint arXiv:1610.09889*, 2016.
[3] F. Satoru, N. Kei, S. Shinji, N. Takuya, and S. Shigeki, "Automatic song composition from the lyrics exploiting prosody of the japanese language," in *Proc. 7th Sound and Music Computing Conference (SMC)*, 2010, pp. 299–302.
[4] D. Wu, K. Addanki, M. Saers, and M. Beloucif, "Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 102–112.
[5] D. Wu and K. Addanki, "Learning to rap battle with bilingual recursive neural networks." in *IJCAI*, 2015, pp. 2524–2530.
[6] E. Malmi, P. Takala, H. Toivonen, T. Raiko, and A. Gionis, "Dopelearning: A computational approach to rap lyrics generation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 195–204.
[7] T. Wengert, "Generating poetry via an n-gram approach."
[8] G. H. de Rosa and J. P. Papa, "A survey on text generation using generative adversarial networks," *Pattern Recognition*, p. 108098, 2021.
[9] S. Lu, Y. Zhu, W. Zhang, J. Wang, and Y. Yu, "Neural text generation: past, present and beyond," *arXiv preprint arXiv:1803.07133*, 2018.
[10] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.
[11] T. Iqbal and S. Qureshi, "The survey: Text generation models in deep learning," *Journal of King Saud University-Computer and Information Sciences*, 2020.
[12] C. Garbacea and Q. Mei, "Neural language generation: Formulation, methods, and evaluation," *arXiv preprint arXiv:2007.15780*, 2020.
[13] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017.
[14] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," 2016.
[15] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[16] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Advances in Neural Information Processing Systems*, 2017, pp. 3155–3165.
[17] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-likelihood augmented discrete generative adversarial networks," 2017.
[18] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, 2016, pp. 207–212.
[19] G. Liu and J. Guo, "Bidirectional lstm with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, 2019.
[20] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 4006–4015.
[21] W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: better text generation via filling in the_," *arXiv preprint arXiv:1801.07736*, 2018.
[22] Y.-L. Tuan and H.-Y. Lee, "Improving conditional sequence generative adversarial networks by stepwise evaluation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 788–798, 2019.
[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.
[24] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[25] K. Gayadhankar, R. Patel, H. Lodha, and S. Shinde, "Image plagiarism detection using gan-(generative adversarial network)," in *ITM Web of Conferences*, vol. 40. EDP Sciences, 2021, p. 03013.
[26] D. Yang, S. Hong, Y. Jang, T. Zhao, and H. Lee, "Diversity-sensitive conditional generative adversarial networks," *arXiv preprint arXiv:1901.09024*, 2019.
[27] A. H. Nobari, M. F. Rashad, and F. Ahmed, "Creativegan: editing generative adversarial networks for creative design synthesis," *arXiv preprint arXiv:2103.06242*, 2021.
[28] Y. Zhang and S. Vogel, "Competitive grouping in integrated phrase segmentation and alignment model," in *Proceedings of the ACL Workshop on Building and Using Parallel Texts*. Association for Computational Linguistics, 2005, pp. 159–162.
[29] N. Xue, F. Xia, F.-D. Chiou, and M. Palmer, "The penn chinese treebank: Phrase structure annotation of a large corpus," *Natural language engineering*, vol. 11, no. 2, pp. 207–238, 2005.
[30] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the conference on empirical methods in natural language processing*, 2004.
[31] Y. Wang, M. Huang, L. Zhao *et al.*, "Attention-based lstm for aspect-level sentiment classification," in *Proceedings of the conference on empirical methods in natural language processing*, 2016, pp. 606–615.
[32] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, "Texygen: A benchmarking platform for text generation models," 2018.
[33] W. Nie, N. Narodytska, and A. Patel, "Relgan: Relational generative adversarial networks for text generation," 2018.
[34] J. Sun, "'jieba'chinese word segmentation tool," 2012.
[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," 1985.
[36] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *arXiv preprint arXiv:1606.01541*, 2016.
[37] R. Shusterman, "The fine art of rap," vol. 22, no. 3. JSTOR, 1991, pp. 613–632.