DeepRapper: Neural Rap Generation with Rhyme and Rhythm Modeling

Lanqing Xue[1], Kaitao Song[2], Duocai Wu[3], Xu Tan[4*],
Nevin L. Zhang[1], Tao Qin[4], Wei-Qiang Zhang[5], Tie-Yan Liu[4]

[1]The Hong Kong University of Science and Technology
[2]Nanjing University of Science and Technology
[3]Fudan University   [4]Microsoft Research Asia   [5]Tsinghua University
{lxueaa,lzhang}@cse.ust.hk
kt.song@njust.edu.cn   dcwu18@fudan.edu.cn
{xuta, taoqin, tyliu}@microsoft.com   wqzhang@tsinghua.edu.cn

Abstract

Rap generation, which aims to produce lyrics and corresponding singing beats, needs to model both rhymes and rhythms. Previous works for rap generation focused on rhyming lyrics but ignored rhythmic beats, which are important for rap performance. In this paper, we develop *DeepRapper*, a Transformer-based rap generation system that can model both rhymes and rhythms. Since there is no available rap dataset with rhythmic beats, we develop a data mining pipeline to collect a large-scale rap dataset, which includes a large number of rap songs with aligned lyrics and rhythmic beats. Second, we design a Transformer-based autoregressive language model which carefully models rhymes and rhythms. Specifically, we generate lyrics in the reverse order with rhyme representation and constraint for rhyme enhancement and insert a beat symbol into lyrics for rhythm/beat modeling. To our knowledge, DeepRapper is the first system to generate rap with both rhymes and rhythms. Both objective and subjective evaluations demonstrate that DeepRapper generates creative and high-quality raps with rhymes and rhythms.

1 Introduction

Rap is a musical form originating from America in 1970s, and has quickly developed as one of the mainstream music genres in the world (Keyes, 2004). With the rapid development of artificial intelligence, automatic rap lyrics generation has drawn attention from academia (Potash et al., 2015; Malmi et al., 2016; Liang et al., 2018; Nikolov et al., 2020). Generally speaking, rap lyrics need to be semantically meaningful and fashionable to convey interesting stories or express feelings. Different from natural language or other artistic genres (*e.g.*,

lyrics or poetry), rap has distinctive characteristics: 1) it usually contains complex rhyme patterns among several consecutive sentences, which are the key to form a good flow; 2) it needs to align with the singing beat since rap lyrics are usually rapped according to some rhythmic accompaniments. Therefore, how to generate rap lyrics with good rhymes and rhythms is a troublesome problem.

Previous works (Potash et al., 2015; Malmi et al., 2016; Liang et al., 2018; Nikolov et al., 2020) for rap generation mainly focused on lyric generation and some of them developed strategies for rhyme modeling. Potash et al. (2015) directly added a "<endLine>" token at the end of verse lines and expected to learn rhyme patterns implicitly. Nikolov et al. (2020) applied a two-step strategy, which first generates rap lyrics and then adds rhyme tokens to the end of generated lyrics. However, these methods cannot guarantee the rhyme patterns for every lyric line and only care the rhyme on the last token. Although many works have studied rhyming modeling in other artistic genres (*e.g.*, poetry) (Li et al., 2020; Van de Cruys, 2020; Liu et al., 2020), they are not suitable for rap generation due to the complex rhyme structure in rap. For example, poetry needs to rhyme with only the last word in each sentence, while rap rhymes with multiple consecutive tokens at the end of each sentence.

No previous works have studied rhythm modeling (*i.e.*, beats in rap), to our knowledge. One of the main reasons is the lack of rap datasets with beat-lyric alignment. Consequently, the generation of lyrics without rhythmic beats cannot be regarded as a full rap generation.

In this paper, we develop DeepRapper, a Transformer (Vaswani et al., 2017) based rap generation system which can model both rhymes and rhythms. To build the system, since there is no available rap datasets with aligned rhythmic beats, we design a

---

69

data mining pipeline and collect a large-scale rap dataset for rhythm modeling. Specifically, we first crawl many rap songs, each song with both rap lyrics and audios, from the Web. For each crawled rap song, we perform a series of data preprocessing steps to extract rhythmic beats as well as beat-lyric alignment. To better model rhyme, we generate the words in a rap sentence from right to left in an autoregressive manner. Doing so we can easily identify the last few words of a sentence (now become the first words of the reverse sentence) to rhyme with. Additionally, we incorporate several rhyme-related representations into our language model to further improve the rhyming quality, and encourage the N-gram rhyme in generated rap lyrics through rhyme constraint during inference. We use a special token [BEAT] to represent the rhythmic beat and insert it into lyrics right before the corresponding word. In this way, we can model the beat in the lyric sequence both in training and generation.

Inspired by the success of pre-trained language models (Devlin et al., 2019; Radford et al., 2018; Yang et al., 2019; Song et al., 2019; Liu et al., 2019), we incorporate pre-training into our system. To obtain large-scale data for pre-training, we also use our data mining pipeline to collect another two datasets: 1) non-rap songs with aligned beats, which can be larger than rap dataset since non-rap songs are more general than rap songs; 2) pure lyrics, which can be even larger than non-rap songs. In the pre-training stage, we pre-train our Deep-Rapper model based on the above two datasets. Then we fine-tune our pre-trained model on the rap songs with aligned beats. The fine-tuned model is used for final rap generation. Both objective and subjective evaluations verify the advantages of DeepRapper in generating rap lyrics with rhymes and rhythms.

Our main contributions can be summarized as follows:

- To model rhythms in rap generation, we develop a data mining pipeline to create rap datasets with aligned rhythmic beats.

- To better model rhymes, we design an autoregressive language model to generate rap lyrics from right to left with rhyme constraint. As far as we know, DeepRapper is the first to explicitly model N-gram rhymes.

- We elaborately insert the beat token inside lyrics to model the rhythmic beats. To our

knowledge, DeepRapper is the first system that models rhythms for rap generation.

2 Background

Since DeepRapper generates rap lyrics with both rhyme and rhythm modeling, in this section, we briefly introduce the related background: lyric generation, rhyme modeling and rhythm modeling.

Lyric Generation Broadly speaking, lyric generation can cover rap lyric generation (Potash et al., 2015; Nikolov et al., 2020; Liang et al., 2018), song lyric generation (Watanabe et al., 2018; Lu et al., 2019; Chen and Lerch, 2020; Sheng et al., 2020), general poetry generation (Zhang and Lapata, 2014; Lau et al., 2018; Li et al., 2020; Liu et al., 2020) and *etc*. Different from previous works that leverage language model to generate lyrics similar to natural language, in this paper, we introduce a novel language model for rap generation, with well-designed rhyme and rhythm modeling to fit the characteristics of rap lyrics. Additionally, inspired by the successes of pre-trained language models (Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019; Radford et al., 2019; Song et al., 2019) in NLP applications, we also incorporate pre-training into our model to further improve the quality of rap generation.

Rhyme Modeling Rhyme modeling plays an important role in rap generation, which requires the last few tokens in consecutive sentences to have the same rhyme pattern. Existing rap generation systems either directly add a special token "<endLine>" at the end of rap lyric to encourage the model to learn rhyme structure (Potash et al., 2015), or introduce a two-step strategy for rhyme modeling that first generates rap lyrics and then adds rhyme tokens after the generated lyrics (Nikolov et al., 2020). However, these works only focused on unigram rhyme while rap appreciates more for n-gram rhyme. Although a lot of works have explored rhyme modeling in other genres, most of them cannot be directly used for rap generation. For example, poetry generation (Lau et al., 2018; Zhipeng et al., 2019; Liao et al., 2019; Li et al., 2020) usually used pre-defined format to control the rhyme pattern since poetry usually has fixed number of words and only cares the rhyme pattern for the last word. However, rap lyrics have diverse rhyme structures across multiple consecutive sentences and most importantly multiple con-
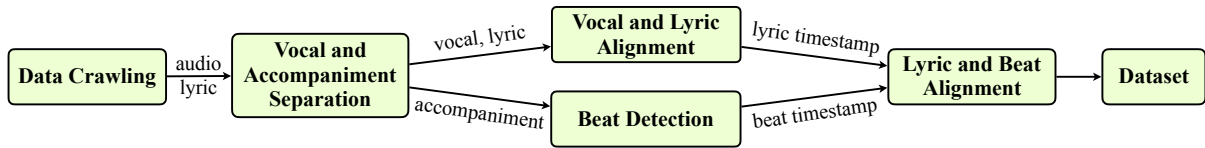
Figure 1: An overview of our data mining pipeline.

secutive words. Therefore, we introduce N-gram rhyme modeling in DeepRapper to handle the distinctive rhyme patterns in rap. Besides, we also train our language model in a reverse order (*i.e.*, right to left), similar to previous works (Van de Cruys, 2020), to better model rhymes since they always occur at the end of sentence.

Rhythm Modeling   Rhythm modeling is usually used in music generation (Zhu et al., 2018; Huang and Yang, 2020; Ren et al., 2020) which generates the duration of notes along with the note pitch to form rhythmic beats in melody and accompaniment generation. Different from music generation, rap cares more about rhythmic beats instead of note pitches (i.e. melody). In this way, the generated rap lyrics need to align with the corresponding rhythmic beats in order to be rapped, otherwise it cannot be regarded as a complete rap. However, to the best of our knowledge, none of previous works have studied the rhythm modeling in rap generation. In this paper, we introduce a novel beat modeling strategy in DeepRapper for rhythm generation.

3   Rap Dataset Mining

Previous works (Potash et al., 2015; Liang et al., 2018; Nikolov et al., 2020) for rap generation usually used rap datasets with only lyrics, without considering the rhythmic beat information. To model rhythm in rap generation, the rap dataset should contain lyrics with aligned rhythmic beats. However, beat alignments are quite difficult to obtain, since their annotations require musicians with professional knowledge to identify stressing syllable in rap songs. To handle this problem, we design a data mining pipeline to automatically extract beat-lyric alignments. In this section, we introduce the details of the data mining pipeline and our mined dataset based on this pipeline.

3.1   Data Mining Pipeline

Figure 1 overviews our data mining pipeline, which consists of 5 steps: data crawling, vocal and accompaniment separation, vocal and lyric alignment, beat detection, and lyric and beat alignment.

Data Crawling   To mine a large-scale rap dataset, we first crawl a large amount of rap songs with both lyrics and singing audios from the Web. To ensure the lyric and audio can be aligned in the sentence level which is beneficial for our later word-level beat alignment, we also crawl the start and end time of each lyric sentence corresponding to the audio.

Vocal and Accompaniment Separation   For each rap song, we utilize *Spleeter* (Hennequin et al., 2020) [1], a public music separation tool, to separate the vocal (containing rap singing) and accompaniment (containing rhythmic beats) from the crawled rap audio.

Vocal and Lyric Alignment   We split the separated vocals into the sentence level according to the crawled start and end time of each lyric sentence, and thus we can get the vocal-lyric alignments in the sentence level. We convert lyrics into phonemes via *Phonemizer* [2] and utilize *Montreal Forced Aligner* [3] to obtain vocal-lyric alignments in the phoneme level. Based on these phoneme-level vocal-lyric alignments, we obtain the corresponding timestamp of each word in the singing audio.

Beat Detection   To obtain the alignments between lyrics and beats, we need to know the timestamp of each beat. Therefore, we use a beat track detection tool, *Librosa* (McFee et al., 2020) [4], to track the timestamp of each beat from the separated accompaniment that obtained from the second step.

Lyric and Beat Alignment   After we obtain the timestamp of each word and each beat, we can align them together according to their timestamps. However, since a rapper may not sing a word exactly following the beat, directly using the timestamp to exactly match the word and beat is inappropriate. Therefore, we propose an approximate method to align them. Denote the word sequence of a lyric

---

[1] https://github.com/deezer/spleeter
[2] https://github.com/bootphon/phonemizer
[3] https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner
[4] https://github.com/librosa/librosa

sentence as $W = \{w_1, w_2, \cdots, w_{|W|}\}$, and its beat sequence as $B = \{b_1, b_2, \cdots, b_{|B|}\}$, where w_i and b_j represent i-th word and j-th beat. We use T_{w_i} and T_{b_j} to represent the timestamps of w_i and b_j respectively. For each beat b_j, we first filter out a word set $\tilde{W} = \{w : |T_{b_j} - T_w| \le r/2, w \in W\}$, where r represents the average duration of each word in the song (*i.e.*, the total duration divides the number of words). Next, word w_i is aligned with beat b_j if it satisfies the following condition:

$$w_i = \min_w |T_{b_j} - T_w|, w \in \tilde{W}. \tag{1}$$

Table 1: The statistics of three mined datasets. The second and third column represent the number of songs and sentences for each dataset.

Dataset	#Songs	#Sentences
D-RAP	16,246	832,646
D-SONG	52,737	2,083,143
D-LYRIC	272,839	9,659,503

3.2 Mined Datasets

Using the above data mining pipeline, we obtain a rap lyric dataset with aligned beats (named as D-RAP, where D represents "dataset"), which satisfies the requirements of building a rap generation system with both rhyme and rhythm modeling. We split the D-RAP dataset into the training and validation set with a ratio of 4:1. Since rap is only one of music genres and the number of rap songs is usually smaller compared with more general songs, we also mine another two datasets to pre-train our DeepRapper model with the same mining pipeline: 1) non-rap songs with aligned beats (named as D-SONG); 2) pure lyrics without aligned beats (named as D-LYRIC). We summarize the statistics of the three datasets in Table 1 and show a rap song with aligned beats from D-Rap in Figure 2.

4 Rap Generation Model

In this section, we introduce the architecture of our rap generation model, and the details of its rhyme modeling and rhythm modeling.

4.1 Model Overview

Figure 3 illustrates the detailed architecture of our rap generation model. We use Transformer (Vaswani et al., 2017) to build an autoregressive language model (Radford et al., 2018, 2019)

LINE_[START_TIME] LYRICS

01_[00:12.04]*我长大*的地方*像一个简朴*的寨
02_[00:15.15]*简朴的*人 吃*着最简朴*的菜
03_[00:18.01]*简朴的*话 包*含着简朴的爱
04_[00:20.82]*简朴的*道理传*给一代又一代
05_[00:23.98]*难以*忘记的画*面不需相机
06_[00:26.67]难*以再闻*到的是巷*子里的香气
07_[00:29.63]*常常*想起 外*婆家的躺椅
08_[00:32.38]最*珍贵的椰*奶 往*往藏在床底
09_[00:35.24]*先填饱*肚子再想那*些背不下的书
10_[00:38.42]*外婆做*的火腿*肠比外面炸的酥
11_[00:41.36]*油烟的*香味弥漫*不那么大*的屋
12_[00:44.28]*外婆的故*事总会*让大人笑*着哭

Figure 2: An example of a rap song with aligned beats in our mined "D-RAP" dataset. '*' means a beat is aligned with the word right after '*'. Translation of the content is in supplemental materials.

for rap generation, and introduce several new designs: 1) To better model rhymes, our model generates a sentence from right to left, since rhyming words are always at the end of the sentence; 2) As aforementioned, rhythms are critical for rap performance, so we insert a special token [BEAT] for explicit beat modeling; 3) Unlike original Transformer with only word embedding and positional embedding, we add multiple additional embeddings to better model rhymes and rhythms. Next, we introduce our rhyme modeling in subsection 4.2 and rhythm modeling in subsection 4.3.

4.2 Rhyme Modeling

Rhymes are the key to form a good rap flow. In DeepRapper, we model rhymes with three components: 1) reverse-order language model; 2) rhyme representation; and 3) rhyme constraint.

4.2.1 Reverse-Order Language Model

Rhyming words usually occur at the end of each lyric sentence. If using a standard autoregressive language model and generating tokens from left to right, we need to identify whether the current generation step is the end of a sentence, which decides whether to generate rhyming words to be consistent with that in previous sentences. Therefore, to better model rhymes, we use a reverse-order language model to generate sentences from right to left, as shown in Figure 3. Doing so we can easily identify the last few words of a sentence (now become the first few words of the reverse sentence) to control their rhymes. Note that we only reverse
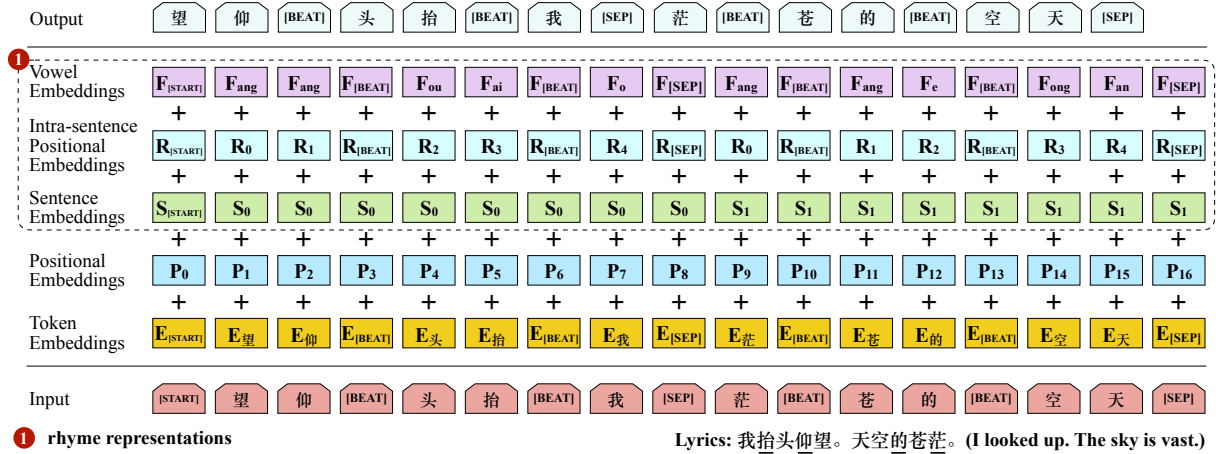
Figure 3: The architecture of the rap generation model in our *DeepRapper*. The input sequence here is a sample from a Chinese rap named 《红色》 (translated to English is *Red*). The sample contains two lyric sentences with aligned beats. Each sentence is reversed for rhyme modeling. Therefore, the original form of the sample is: 我抬头仰望。天空的苍茫。(Translation: *I looked up. The sky is vast.*) Word with underline means that a beat is aligned with this word. Sentences are separated by special token '[SEP]'. Token '[START]' represents the start of a song.

Figure 4: Comparison between sentences in left-to-right and right-to-left order. The superscript of each token represents its rhyme.

words inside a sentence, and still generate different sentences in the original order. Figure 4 compares the sentences in left-to-right order and right-to-left order, from which we can see that rhyming words of each sentence share the same relative positions (offset to the first token) in the reverse order, and are easy to model and control.

4.2.2 Rhyme Representation

Rhyming words have two important features: 1) its vowel that used for rhyming and 2) its relative position in a sentence to decide the correspondence between the rhyming words in consecutive sentences (e.g., in the reverse order setting, the first/second word of the current sentence should be rhymed with the first/second word in the previous sentence).

We use the vowel in the Pinyin [5] of Chinese characters to represent their rhymes. To this end, we

---
[5]Pinyin is the standard phoneme for Chinese.

build a vowel dictionary $\mathcal{F}(\cdot)$ to identify the vowel of each word. As shown in Figure 3, we add an additional vowel embedding $\mathbf{F}$ and an intra-sentence relative positional embedding $\mathbf{R}$ to enhance rhyme representation for each token. Besides, to better identify different sentences, we introduce a sentence embedding $\mathbf{S}$ to differentiate different sentences.

4.2.3 Rhyme Constraint

In addition to reverse-order language model and rhyme representation, we also introduce rhyme constraint to improve the quality of rhyme generation in inference. As shown in Figure 4, sentences in rap lyrics not only rhyme with the last token, but also with multiple consecutive tokens at the end. We call this phenomenon as N-gram rhymes, which mean the current sentence and the previous sentence keep the same rhyme for the last N consecutive tokens. To our knowledge, no previous work has investigated N-gram rhymes ($N > 1$), although it is important to improve rap quality. Our proposed rhyme constraint enables our model to adjust the probability of next predicted token to further encourage N-gram rhyme generation. The constraint is introduced as follows.

To generate the i-th word w_i in the standard inference procedure, we usually choose the predicted token with the maximum probability, *i.e.*, $w_i = \arg\max p(w|w_{<i}; \theta)$, where $w_{<i}$ denotes the words before position i in the reverse sentence and θ is the model. When the words before posi-

73

tion i of the current and previous sentence have the same rhyme pattern, we will use an adjusted probability distribution $\tilde{p}(w|w_{<i};\theta)$ to encourage the i-th generated word to be rhymed according to the i-th word in the previous sentence, so as to form N-gram rhymes. The adjusted probability distribution $\tilde{p}(w|w_{<i};\theta)$ is:

$$\tilde{p}(w|w_{<i};\theta) = \alpha \cdot p(w|w_{<i};\theta) + (1-\alpha) \cdot \pi(w) \tag{2}$$

where $\pi(w)$ is a vowel check function and α is a hyper-parameter to balance the two terms. Here, $\pi(w)$ is 1 if the predicted w has the same vowel with the i-th token in the previous sentence, otherwise 0. In other words, when predicting i-th token ($i \leq N$), we encourage our model to pay more attention for these words with same vowel with the i-th token in the previous sentence. In this way, the model tends to generate N-gram rhymes with large N.

4.3 Rhythm Modeling

Generating lyrics with aligned beats is necessary since rap lyrics need to be rapped with rhythmic beats. Therefore, we model and generate rhythmic beats along with the lyrics with a specific symbol: we regard beat as a special token [BEAT] and insert it into lyric sequences for model training. As shown in Figure 3, we insert [BEAT] before its aligned words like the following examples: "我[BEAT]抬头[BEAT]仰望。天空[BEAT]的苍[BEAT]茫。".

Rap usually contains different beat frequencies, i.e., the ratios between the total number of words and the total number of beats in a rap song. To explicitly model and generate rap with different beat frequencies, we use three tokens [S], [M], and [F] to represent the slow, medium and fast beat frequencies and add the corresponding tokens at the start of a rap song for training and inference. In our D-RAP dataset, the distribution of beat frequency is displayed in Figure 5. According to the distribution, we assign [S], [M], and [F] to songs with beat frequency less than 3, equal to 3, and greater than 3 respectively.

5 Experimental Setup

5.1 Model, Data, and Training Configuration

Our DeepRapper model is built on the autoregressive Transformer decoder (Vaswani et al., 2017; Radford et al., 2018, 2019), where the hidden size, the number of attention heads and the number of
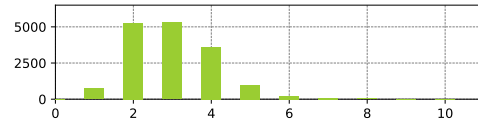

Figure 5: The distribution of beat frequencies in D-RAP. X-axis: beat frequency (the ratio between the total number of words and the total number of beats in a rap song). Y-axis: the number of songs.

Transformer layers are set as 768, 12, 12. The dimension of all different kinds of embedding in DeepRapper is set as 768. Considering there is no existing pre-trained language model in reverse order, we do not utilize any pre-trained language models for initialization. Instead, we first pre-train our model on D-LYRIC and D-SONG for 2 millions steps, and then fine-tune our model on D-RAP with 3K steps as the size of D-RAP is smaller than our pre-training corpus. We convert each song to a sequence with a length of 1024 tokens by cutting longer sequence or padding shorter sequence. Our model is trained with a batch size of 8 songs on 4 NVIDIA TITAN V GPUs. We use Adam optimizer with a learning rate of 0.00015, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-6}$. We set the maximum value of N-gram rhyme as 3 and the hyper-parameter α in Equation 2 as 0.95. Samples are generated conditioned on a given sentence in reference.

5.2 Evaluation Metrics

In this subsection, we introduce the objective and subjective metrics to evaluate the quality of the generated raps.

Objective Evaluation We evaluate the generated raps in terms of the quality of language, rhyme and rhythm. We choose five metrics to evaluate our model: 1) **Perplexity** (PPL), a standard metric to evaluate the quality of a language model; 2) **Rhyme Accuracy** (RA), the ratio of sentences that have correctly predicted rhymes; 3) **Rhyme Density** (RD), the longest rhyme of a song, averaged over all songs, which is introduced by Malmi et al. (2016) to measure the quality of rhyming fluency; 4) **Combo-N**, the maximum number of consecutive sentences with the same N-gram rhyme in a rap song, averaged over all songs, where we study $N = 1, 2, 3$; 5) **Beat Accuracy** (BA), the accuracy of our model in beat prediction, under the teacher-forcing mode.

Table 2: Results of objective and subjective evaluations. "+PT" means using pre-training. Since the two baselines do not include beat information, we only compare in perplexity (PPL), rhyme accuracy (RA) and rhyme density (RD) for objective evaluation. For subjective evaluation, we report the average annotation score of theme, fluency, rhyme quality, and rhyme diversity.

Model	Objective Evaluation			Subjective Evaluation			
	PPL ↓	RA ↑	RD ↑	Theme ↑	Fluency ↑	Quality ↑	Diversity ↑
Baseline	24.65	32.29	0.23	3.13	2.55	3.10	3.46
Baseline + PT	13.47	39.59	0.35	3.41	2.69	3.25	3.65
DeepRapper	**5.65**	**66.95**	**1.65**	**3.67**	**3.57**	**4.16**	**4.14**

Subjective Evaluation Similar to previous works (Zhang and Lapata, 2014; Nikolov et al., 2020) in artistic creation, we also use human evaluation to accurately evaluate the quality of the generated raps. We invite 10 participants with professional knowledge in music as human annotators to evaluate 100 sampled raps. Each annotator is required to score from 1 (Poor) to 5 (Perfect) on the following perspectives: 1) the clearness of the theme of the rap lyrics; 2) the fluency of the rap lyrics; 3) the quality of the rhyme; 4) the diversity of the rhyme. The averaged score of all annotators on all sampled raps is used as the evaluation score for each perspective.

6 Experimental Results

Results Table 2 shows the objective and subjective results of DeepRapper compared with two baselines: 1) Baseline: a standard autoregressive language model with the same model configuration with DeepRapper but without our proposed rhyme and rhythm modeling; 2) Baseline + PT, using pre-training on Baseline. We have several observations from Table 2: 1) DeepRapper achieves better perplexity, rhyme accuracy and rhyme density than the two baselines, which demonstrates the advantages of our method in generating high-quality rap lyrics with accurate and diverse rhymes. 2) DeepRapper achieves better scores in all subjective metrics, demonstrating that DeepRapper can generate high-quality and rhyming raps that accord with human taste. 3) Pre-training improves the performance of baseline in both objective and subjective metrics, which indicates the importance of pre-training. However, its performance is still worse than DeepRapper.

Ablation Studies To further validate the necessity of each component in DeepRapper, we conduct a series of ablation studies, including remov-

Table 3: The ablation studies on each component in DeepRapper. "-" means removing the corresponding component. "Rhyme", "Rhythm" and "PT" represent rhyme modeling, rhythm modeling and pre-training. "RO", "VE", "IPE" and "SE" mean reverse-order, vowel embedding, intra-sentence position embedding and sentence embedding.

Model	PPL ↓	RA ↑	BA ↑	RD ↑
DeepRapper	5.65	66.95	79.00	1.65
- Rhyme	7.13	41.25	79.42	0.35
- RO	3.70	44.58	82.83	0.41
- VE	6.06	66.11	78.19	1.46
- IPE	5.81	63.86	77.29	1.50
- SE	6.69	66.52	80.37	1.63
- Rhythm	4.66	66.08	–	1.56
- PT	28.69	40.77	76.40	1.98

ing rhyme modeling, rhythm modeling and pre-training, respectively. The results are reported in Table 3. We have several observations: 1) Removing rhyme modeling affects rhyme quality a lot as it results in a dramatic drop in rhyme accuracy and rhyme density; 2) Removing each specific design in rhyme modeling (i.e., RO: reverse order language model, VE: vowel embedding, IPE: intra-sentence position embedding, SE: sentence embedding) causes worse rhyme accuracy and rhyme density. Specifically, while removing RO leads to a better PPL since left-to-right order can be more easily modeled than right-to-left order according to the analysis in Wu et al. (2018), it causes large accuracy drop in rhyme quality. 3) Apparently, DeepRapper without rhythm modeling cannot produce any beat information; 4) DeepRapper without pre-training affects the perplexity and rhyme accuracy a lot, however, obtains a higher rhyme density. The reason is that without pre-training, DeepRapper tends to copy previous rhyme tokens due to the

lack of generalization (larger PPL). To verify this, we count the repetitive rate of rhyming words and found that the rate of DeepRapper is 23.8% while without pre-training is 42.5%, which is higher than using pre-training. The above results verify the effectiveness of each component in DeepRapper.

N-gram Rhyme   To highlight the advantage of DeepRapper in modeling N-gram rhyme, we use Combo-N to measure the ability of each design in DeepRapper to model N-gram rhyme. The results are reported in Table 4. We can find that 1) The model without rhyme modeling can hardly generate good rhyme, regardless of the value of N in N-gram; 2) Removing rhyme constraint also weakens the capacity of generating N-gram rhyme. These results further demonstrate the importance of our rhyme modeling and rhyme constraint in generating multiple consecutive rhymes.

Table 4: Quality of N-gram rhyme in terms of Combo-N . "- Rhyme" means removing rhyme modeling and "- RC" means removing rhyme constraint during inference.

Model	Combo-1	Combo-2	Combo-3
DeepRapper	**87.10**	**18.16**	**9.10**
- Rhyme	7.37	2.85	1.71
- RC	70.08	13.73	3.77

Beat Frequency   To better measure the beat quality, we randomly generate about 5,000 samples by DeepRapper and DeepRapper with beat frequency control. We propose the First Order Distribution (FOD) and the Second Order Distribution (SOD) and measure the distance (via Wasserstein Distance (Vallender, 1974)) of these distributions between the generated samples and our DRAP dataset. We define the interval of the current [BEAT] as the number of words between the current [BEAT] and the next [BEAT]. Therefore, the FOD is defined as the distribution of the interval of the current [BEAT]. Similarly, the SOD is defined the distribution of the difference between the interval of the current [BEAT] and the next [BEAT]. The results of the distance are normalized into $[0, 1]$ and are reported in Table 5. It can be seen that DeepRapper with beat frequency control achieves better performance in beat modeling, which indicates the importance of beat frequency control in beat modeling.

Table 5: Measurement of beat generation. "+ Beat Frequency" represents DeepRapper with beat frequency control.

Model	FOD	SOD
DeepRapper	0.1107	0.0514
+ Beat Frequency	**0.1091**	**0.0502**

Case Analyses on Generated Raps   We list a sample case from our generated raps in Figure 6 to demonstrate the good quality of the raps generated by DeepRapper. The sample is generated by feeding the first sentence of the example in Figure 2 to DeepRapper. As we can see, the generated sample exhibits good theme, fluency and rhyme. The sample is a rap with a number of 1-gram, 2-gram, 3-gram, and even 4-gram rhyme. The generated lyrics depicts the fond memories

Figure 6: A rap generated by *DeepRapper*. For each example, we provide the corresponding vowel of each word. Vowels in red color represents that the word rhymes with previous sentence. Bold word means a beat is aligned with the word. The translation of the example is attached in supplemental materials.

of childhood and the beautiful visions for the futures. We also provide a group of samples generated with beat frequency control. To save space, we put them and the translation of all the samples to Appendix. More samples are provided in https://deeprapper.github.io.

7 Conclusions

In this paper, we develop *DeepRapper*, a novel Transformer-based rap generation system, which leverages rhyme modeling, rhythm modeling and pre-training for rap generation. Considering there is no available rap dataset with aligned rhythmic beats for rhythm modeling, we propose a data mining pipeline to mine a rap dataset with beat-lyric alignments. We leverage right-to-left generation, rhyme representation and rhyme constraint to better model rhyme and encourage N-gram rhyme, and explicitly model beat information by insert beat token beside the corresponding word in the lyric sequence. To our knowledge, DeepRapper is the first system to generate rap with both rhymes and rhythms. Both objective and subjective evaluations demonstrate that DeepRapper generates high-quality raps with good rhymes and rhythms. Thanks to the design of DeepRapper, we can further build another rap singing system to sing out the raps according to the rhymes and rhythms, which we leave as future work. We also leave Multilingual *DeepRapper* as future work.

Acknowledgements

Ethical Considerations

The proposed framework can be considered a novel language model for rap generation in automatic artistic creation. Specifically, the proposed framework has been configured with novel rhyme modeling as rhyme is quite important in music genres. Therefore, our proposed framework is also beneficial for generating other music genres. On the other hand, although we collect large-scale lyric data for pre-training, it still cannot fully utilize the potential of pre-training. In the future, we expect to employ more large-scale data in the open domain plus the music domain for pre-training to improve the capacity of the language model. In addition, our training datasets may have biases, which may bring some potential risks of model bias. Hence, we encourage future works to study how to apply other techniques in mitigating similar problems in our framework.

References

Yihao Chen and Alexander Lerch. 2020. Melody-conditioned lyrics generation with seqgans. In *arXiv*.

Tim Van de Cruys. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2471–2480.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. 2020. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154. Deezer Research.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, page 1180–1188.

Cheryl Lynette Keyes. 2004. *Rap music and street consciousness*, volume 501. University of Illinois Press.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958.

Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Rigid formats controlled text generation. In *ACL*, pages 742–751.

Hongru Liang, Qian Li, Haozheng Wang, Hang Li, Jin-Mao Wei, and Zhenglu Yang. 2018. Attae-rl[2]: Attention based autoencoder for rap lyrics representation learning. In *Companion Proceedings of the The Web Conference 2018*, pages 7–8.

Yi Liao, Yasheng Wang, Qun Liu, and Xin Jiang. 2019. Gpt-based generation for classical chinese poetry. *arXiv preprint arXiv:1907.00151*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yusen Liu, Dayiheng Liu, and Jiancheng Lv. 2020. Deep poetry: A chinese classical poetry generation system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13626–13627.

Xu Lu, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A syllable-structured, contextually-based conditionally generation of chinese lyrics. In *PRICAI*, volume 11672, pages 257–265.

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*, pages 195–204.

Brian McFee, Vincent Lostanlen, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, Keunwoo Choi, viktorandreevichmorozov, Josh Moore, Rachel Bittner, Shunsuke Hidaka, Ziyao Wei, nullmightybofo, Darío Hereñú, Fabian-Robert Stöter, Pius Friesch, Adam Weiss, Matt Vollrath, and Taewoon Kim. 2020. librosa/librosa: 0.8.0.

Nikola I. Nikolov, Eric Malmi, Curtis Northcutt, and Loreto Parisi. 2020. Rapformer: Conditional rap lyrics generation with denoising autoencoders. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 360–373.

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*, pages 1919–1924.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1198–1206.

Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. 2020. Songmass: Automatic song writing with pre-training and alignment constraint. *arXiv preprint arXiv:2012.05168*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

SS Vallender. 1974. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A melody-conditioned lyrics language model. In *NAACL*, pages 163–172.

Lijun Wu, Xu Tan, Di He, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. Beyond error propagation in neural machine translation: Characteristics of language also matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3602–3611.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Guo Zhipeng, Xiaoyuan Yi, Maosong Sun, Wenhao Li, Cheng Yang, Jiannan Liang, Huimin Chen, Yuhui Zhang, and Ruoyu Li. 2019. Jiuge: A human-machine collaborative chinese classical poetry generation system. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 25–30.

Hongyuan Zhu, Qi Liu, Nicholas Jing Yuan, Chuan Qin, Jiawei Li, Kun Zhang, Guang Zhou, Furu Wei, Yuanchun Xu, and Enhong Chen. 2018. Xiaoice band: A melody and arrangement generation framework for pop music. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 2837–2846.

A Comparison with GhostWriter

We provide a comparison between DeepRapper and GhosterWriter (Potash et al., 2015) in Table 6. The results show that both DeepRapper and baselines outperform GhosterWriter in terms of PPL, rhyme accuracy, and rhyme density on rap generation tasks.

B Samples with Beat Frequency Control

Fast Figure 7 provides a rap generated by DeepRapper with fast beat frequency, which the frequency is 4.3. The rap express ones beat wished

Table 6: The comparison of DeepRapper with Ghoster-Writer in perplexity (PPL), rhyme accuracy (RA) and rhyme density (RD). "+PT" means using pre-training.

Model	Objective Evaluation		
	PPL ↓	RA ↑	RD ↑
GhostWriter	58.95	7.28	0.17
Baseline	24.65	32.29	0.23
Baseline + PT	13.47	39.59	0.35
DeepRapper	**5.65**	**66.95**	**1.65**

to his/her lover. The following is the translation of texts in Figure 7.

我长大的地方像一个简朴的寨
The place where I grew up is like a simple village
遥望远方镜子里的海
Looking into the distance, the sea is in the mirror
看见理想很实际的说一句拜拜
See my dream and say goodbye
这世界在我眼里和千里之外
The world is sometimes in my eyes and sometimes thousands of miles away
穿过河流沙漠和人海
Across rivers, deserts and crowds
一山万水渡过岁月冲洗我的爱
A mountain and a million rivers wash my love through the years
和那亲爱的你一起去踩一路的尘埃
Step on the dust all the way with dear you
我一生中最重要的你我壹一都在
The most important you in my life, I'll always by your side
点点轻地落在你那冰封的大门外
Little by little, it falls outside your frozen gate
我在默默的祝福你
I am blessing you silently
我在无数个夜里默默地祝福你
I have secretly blessed you for countless nights
远远地保护着你我的爱
Protecting you and my love from a distance

Medium Figure 8 provides a rap generated by DeepRapper with medium beat frequency, which the frequency is 2.6. The rap praises the times we live in. The following is the translation of texts in Figure 8.

Figure 7: Rap generated of fast beat frequency. Vowels in red color represents that the word rhymes with previous sentence. Bold word means a beat is aligned with the word.

我长大的地方像一个简朴的寨
The place where I grew up is like a simple village
简朴的看着简朴的海
Simply looking at the simple sea
爸爸拿着一个简朴的麦
Dad holding a simple wheat
有人真实的努力就有人背负着爱
Someone takes effort, somebody is carrying love
那生活的美好让人人们热爱
The beauty of life makes people love
这世界的美好纯粹是意外
The beauty of this world is pure accident
而我长大的地方是个简朴的寨
And the place where I grew up is a simple village
让我们在这里开心的喝彩
Let's cheer happily here
伟大母亲怀抱着爱
Great mother embrace love
看着幸福的人们敞开淳朴的怀
Watching happy people open their simple arms
我们最美好的这个快乐海
We are in the most beautiful happy sea
唱出我们的时代
Sing our time

o ang a e i ang ang i e an u e ai
我 长 大 的 地 放 像 一 个 简 朴 的 寨

an u e an e an u e ai
简 朴 的 看 着 简 朴 的 海

a a a e i an u e ai
爸 爸 拿 着 一 个 简 朴 的 麦

ou en en i e u i ou ou en ei u e ai
有 人 真 实 的 努 力 就 有 人 背 负 着 爱

a eng e ei ao ang en en en e ai
那 生 活 的 美 好 让 人 人 们 热 爱

e i ie e ei ao en ei i i ai
这 世 界 的 美 好 纯 粹 是 意 外

er o ang a e i ang i i e an u e ai
而 我 长 大 的 地 放 是 一 个 简 朴 的 寨

ang o en ai e i ai in e ai
让 我 们 在 这 里 开 心 的 喝 彩

ei a u in ai ao e e ai
伟 大 母 亲 怀 抱 着 爱

an e in u e en en ang ai en u ai
看 着 幸 福 的 人 们 敞 开 淳 朴 的 怀

o en ei ei ao e e e ai
我 们 最 美 好 的 这 个 快 乐 海

ang u o en e i ai
唱 出 我 们 的 时 代

Figure 8: Rap generated of medium beat frequency. Vowels in red color represents that the word rhymes with previous sentence. Bold word means a beat is aligned with the word.

Slow Figure 9 provides a rap generated by Deep-Rapper with slow beat frequency, where the frequency is 2.1. The rap express ones relief from life. The following is the translation of texts in Figure 9.

我长大的地方像一个简朴的寨
The place where I grew up is like a simple village
快到有一天看见了父母的爱
Almost one day I saw the love of my parents
我的时间你实在不用去考虑自己多坏
You don't have to think about how bad you are in my time
当我脚步在外从没过的这么可爱
I've never been so cute when I'm out
我只是一次旅行
I'm just a trip to your life
你现在的校服我也想换
I want to change your current school uniform
我曾经追你
I used to chase you
你的运气也不摔
Your luck won't fall
毕竟上次
After all last time

o ang a e i ang ang i e an u e ai
我 长 大 的 地 方 像 一 个 简 朴 的 寨

ai ao ou i an an an e u e ai
快 到 有 一 天 看 见 了 父 母 的 爱

o e i an i i ai u ong i ao i i i o ai
我 的 时 间 你 实 在 不 用 去 考 虑 自 己 多 坏

ang o ao u ai ai ong ei ou o e e e ai
当 我 脚 步 在 外 从 没 有 过 的 这 么 可 爱

o i i i i ing
我 只 是 一 次 旅 行

i an ai e ao u o ie an an
你 现 在 的 校 服 我 也 想 换

o eng ing ei i
我 曾 经 追 你

i e in i e ai
你 的 运 气 也 不 摔

i ing ang i
毕 竟 上 次

i ai e u ang i e an an
你 爱 的 姑 娘 你 也 想 看

a en e i i ing ao en i
她 们 和 你 一 定 要 分 离

i iu e ang i i ao er i
你 就 这 样 子 一 笑 而 去

Figure 9: Rap generated of slow beat frequency. Vowels in red color represents that the word rhymes with previous sentence. Bold word means a beat is aligned with the word.

C Translation of Chinese Examples in the Paper

Words in red are rhymes.

Translation of Chinese in Figure 2

我长大的地方像一个简朴的寨
The place where I grew up is like a simple village
简朴的人吃着最简朴的菜
Simple people eat the simplest dishes
简朴的话包含着简朴的爱
Simple words contain simple love
简朴的道理传给一代又一代
Simple principles are passed on from generation to generation

难以忘记的画面不需相机
Unforgettable picture do not need camera to capture
难以再闻到的是巷子里的香气
What is hard to smell is the aroma in the alley
常常想起外婆家的躺椅
I often think of grandma's recliner
最珍贵的椰奶往往藏在床底
The most precious coconut milk is often hidden under the bed
先填饱肚子再想那些背不下的书
Fill your stomach first, then think about the books that you can't remember
外婆做的火腿肠比外面炸的酥
Grandma's ham sausage is crispier than fried outside
油烟的香味弥漫不那么大的屋
The smell of lampblack pervades the not so big house
外婆的故事总会让大人笑着哭
Grandma's stories always make adults laugh and cry

几百公里我们相约到未来
Through hundreds of kilometers, we meet in the future
在那无尽的沙漠和海
In the endless desert and sea
看着温暖花开
Watching the warm flowers bloom
花一样的在
Like flowers be there
写动人的天籁
Write moving sounds of nature 跟着自由自在
Feeling the freedom
消沉在那片海
Sometimes depressed in the sea
不懂儿时的他们不懂什么是爱
I don't understand their childish. I don't know what love is
到现在你看来
Till now you see
最真的迷彩
It is The most true fantasy

Translation of Chinese in Figure 3

我抬头仰望。天空的苍茫。
I looked up. The sky is vast.

Translation of Chinese in Figure 4

是这座城市的气象
It is the weather of this city
让你感受生命的力量
makes you feel the power of living

Translation of Chinese in Figure 6

我长大的地方像一个简朴的寨
The place where I grew up is like a simple village
公里也许大的远方简直是个小小的寨
A small far away village
偶尔穿件毛衣那样子很可爱
It is cute to wear a sweater occasionally
远方可单纯的姑娘还是单纯的孩
Is it a simple girl or a simple child far away
是放不下的故事大声的喝彩
Cheers loudly for the unforgettable story
像快乐的小孩莫名的敞着怀
Happy kids like happy kids