Verse Generation by Reverse Generation Considering Rhyme and Answer in Japanese Rap Battles

Ryota Mibayashi[1], Takehiro Yamamoto[1], Kosetsu Tsukuda[2], Kento Watanabe[2], Tomoyasu Nakano[2], Masataka Goto[2], and Hiroaki Ohshima[1] ⋆

[1] Graduate School of Information Science, University of Hyogo
[2] National Institute of Advanced Industrial Science and Technology (AIST)
af22h007@guh.u-hyogo.ac.jp; t.yamamoto@gsis.u-hyogo.ac.jp;
k.tsukuda@aist.go.jp; kento.watanabe@aist.go.jp;
t.nakano@aist.go.jp; m.goto@aist.go.jp; ohshima@ai.u-hyogo.ac.jp

Abstract. Rap battle is a competition in which two rappers improvise rap verses alternately, and a verse is composed of multiple sentences uttered in one turn by a rapper. In this paper, we propose a method for generating response verses that are semantically related and *rhyme* with the opponent's verse in rap battles. Our approach uses a language generation model BERT2BERT to generate rap sentences and constructs a verse by appropriately arranging them using a BERT model. When generating rap sentences, it is important to include words that *rhyme* with a specific word in the opponent's verse, but it is difficult to include such words using a conventional sentence generation model that generates sentences in a forward direction from the beginning of the sentence. To address this issue, our proposed method trains the model to generate sentences in a reverse direction from the end of the sentence, which enables the model to generate rap sentences that highly likely have *rhymes* at the end. To train the model, we constructed our own rap battle corpus consisting of 6,791 verses. Our experimental results demonstrate that our proposed method outperforms a method that uses a conventional model generating sentences in a forward direction.

Keywords: Rap battle, Verse generation, BERT2BERT, Rhyme, Answer

1 Introduction

A rap battle is a competition where two rappers perform improvised rap alternately. The rap that is delivered in one turn is called a verse, and a single verse is generally composed of several rap sentences. Figure 1 shows an example of a rap battle. Rapper *A* delivers the verse "*My rhymes hit you harder than a train.*" Rapper *B* responds to that verse with "*It's like a toy train. You don't know my pain.*" The rap battle ends

⋆ This work was supported by JSPS KAKENHI Grant Numbers JP21H03775, JP21H03774, JP21H03554, JP22H03905.

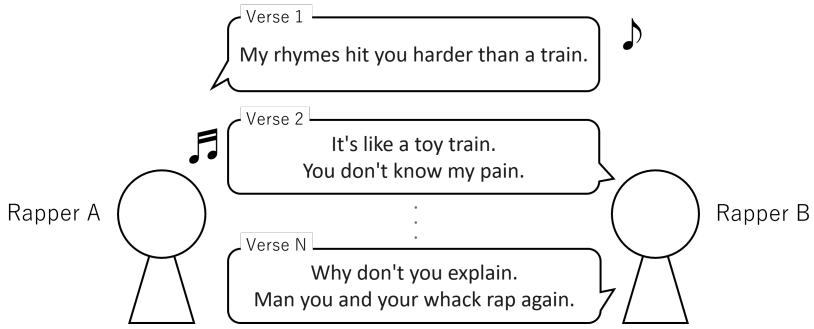

Fig. 1. An example of a rap battle.

after repeating these responses several times. In rap battles, the winner is determined by the audience or judges. The quality of the verse is one of the important factors in determining the winner.

The quality of the verse is typically determined by *rhymes*, *answers*, and *flows* [1]. *Flow* refers to the expression style, such as rhythm and accentuation, of the rapping (singing) voice by a rapper. On the other hand, *rhymes* and *answers* depend on the text of the verse, as shown below.

- *Rhyme* refers to a pair of words with the same vowel sequence. For example, the pair of "*train*" in verse 1 and "*pain*" in verse 2 of Figure 1 is a *rhyme* because both of them have the same vowel sequence "*ein*."
- *Answer* refers to a response verse that is related to the opponent's verse. For example, the verse "*It's like a toy train.*" in verse 2 of Figure 1 is an *answer* because it is semantically related to the verse 1.

High-quality verses need to reflect both *rhyme* and *answer*, and in rap battles, it is necessary to respond immediately with such verses to the opponent's verses.

Rap battle competitions are held in many countries. For example, there is the "*True Freestyle Rap Battle*" in the U.S. and the "*UNPRETTY RAPSTAR*" rap battle TV program in Korea. In Japan, rap battle competitions such as "*Gaisen MCBATTLE*" and "*Sengoku MCBATTLE*" are also popular. In addition to such competitions in which professional rappers participate, there are also many competitions in which amateur rappers participate. In amateur rap battles, it is also important to respond to a verse with *rhymes* and *answers*, but it is not easy, especially for beginners, to come up with such a verse immediately in response to an opponent's verse. If there were a system that could automatically generate high-quality verses for any given verse, such people would be able to practice rapping by referring to the generated verses.

In this paper, as a first step toward realizing such a system, we propose a method for generating a verse with both *rhymes* and *answers* in a rap battle. We do not deal with *flows* since we focus on the text of the verse. In the proposed method, a language generation model, BERT2BERT, is used to generate the verse. More specifically, the method is composed of two steps: generating rap sentences and ordering rap sentences. In the first step, given a sentence in the opponent's verse as input, we generate rap sentences

considering the *rhyme* based on the input sentence. While sentence generation models usually generate a sentence in the forward direction from the beginning of the sentence, we generate a rap sentence in the reverse direction from the end of the sentence so that it can contain a *rhyme* at the end. Although there are a large number of words that satisfy a particular *rhyme*, our method searches for appropriate words based on the opponent's verse so that the *answer* can also be taken into account. Since multiple rap sentences are generated in this step, the second step aims to construct an appropriate verse by ordering the generated rap sentences by predicting next sentences using BERT [2].

Our contributions can be summarized as follows.

- We propose a verse generation method that takes into account *rhymes* and *answers* in rap battles and generates sentences from the end of the sentence.
- To train the model of the proposed method, we develop a training corpus consisting of 6,791 rap battle verses.
- We conduct evaluation experiments and show that the proposed method outperforms a method that generates sentences from the beginning of the sentence.

2 Related Work

2.1 Text Generation

In recent text generation methods, deep learning is commonly used. For example, in the early stages, seq2seq [3] models using autoregressive structures such as RNN [4] and LSTM [5] were employed. The seq2seq model has an Encoder that aggregates text information and a Decoder that generates texts from the aggregated information, and is also known as an Encoder-Decoder model. Subsequently, a mechanism called Attention that generates texts by selecting text information effectively was proposed. More recently, a model called Transformer [6], which uses Attention, has been proposed. Transformer is an Encoder-Decoder model with a large number of parameters and Attention.

BERT and GPT-2 are well-known models that use Transformer. BERT [2] is a general-purpose feature extractor for natural language texts and is a multi-layered Encoder-only model of Transformer. It can consider the context of the text because it takes account of the information of the text from both directions. GPT-2 [7] is a multi-layered Decoder-only model that is commonly used in text generation tasks. Different form BERT, GPT-2 considers the information of the text from one direction only. These models are generally used after having acquired general-purpose linguistic knowledge by training on a large corpus in advance. By fine-tuning a pre-trained model for a specific task, a model specialized for that task can be learned.

BERT2BERT [8] is a generative model that transfers the pre-trained parameters of BERT to the Encoder and Decoder of Transformer. It has been shown to be effective in the generation tasks to use the pre-training weights obtained by the BERT model [2]. Based on the usefulness of the BERT model, we leverage the BERT model to generate the text of rap verses.

Table 1. Differences between existing studies and ours.

	Containing *rhymes*	Generating text for rap battles	Using rap battle data
Rapformer [9]	√	-	-
GhostWriter [10]	√	-	-
DopeLearning [11]	√	-	-
Manjavacas et al. [12]	√	-	-
DeepRapper [13]	√	-	-
Wu et al. [14]	√	√	-
Shimon [15]	√	√	-
Ours	√	√	√

2.2 Rap Generation

In research aimed at generating rap lyrics, some rhyme-aware methods have been proposed that replace words in the generated lyrics with other words that *rhyme* [9], or generate rhyming lyrics by learning from lyrics that contain *rhymes* [10–13]. Although those previous methods are shown to be effective in rap generation, they are not sufficient for our purpose of rap battles because of a limited word choice. In rap battles, since the opponent's verse would contain a wide range of topics, it is important to generate a response verse using a wide range of words containing a specified *rhyme*.

Rapformer [9], for example, is a method that first uses the Transformer model [6] to generate a lyric sentence and then replaces the last word of the generated sentence with another rhyming word. Since this replacement cannot change the other words in the generated sentence, the replaced rhyming word must be semantically consistent with the other words, thus limiting word choice.

The same limitation exists in other rap generation methods that use deep learning models to learn *rhymes* in lyrics [10–12]. Potash et al. [10] proposed a method called Ghostwriter that uses LSTM models to generate new rap lyrics from existing rap lyrics data. DopeLearning proposed by Malmi et al. [11] selects the most appropriate rap sentence by comparing the similarity between the generated rap sentence and the previous one. Manjavacas et al. [12] proposed a method for generating rap lyrics by using LSTM as well. Those methods generate words in order starting from the first word to reach the last word. The choice of the last word is limited since it should be semantically consistent with the previous words. On the other hand, Xue et al. [13] proposed a method called DeepRapper that uses a Transformer-Decoder to consider the beat when generating rap verses. DeepRapper generates a sentence in the reverse direction and the above limitation is mitigated. However, since DeepRapper generates multiple lyric sentences at once, the choice of rhyming words in the second and subsequent sentences is still limited by the context of the previous sentence.

With a focus on rap battles, there have been a few studies on generating rap battle verses. Wu et al. [14] developed a chatbot system for rap battles that uses a modified version of RAAM (Recursive Auto-Associative Memory) called TRAAM (Transduction Recursive Auto-Associative Memory) for generating verses. Shimon proposed by Savery et al. [15] generates rap sentences by using LSTM based on existing rap lyrics

data, and then rearranges them to create a verse based on keywords in the input verse. These studies used their own hip-hop lyric corpora, but did not use a rap battle corpus since there was no existing corpus for rap battles.

Table 1 summarizes the studies introduced above. Our study differs from them in that our method can generate a response verse using a wide range of rhyming words because it generates a rap sentence in the reverse direction starting from a rhyming word at the end of the sentence. Moreover, we develop our own corpus specialized for rap battles and use it for training our sentence generation model.

3 Verse Generation Method for Rap Battles

An overview of the proposed verse generation method specialized for rap battles is shown in Figure 2. Since rhyming is indispensable in rap battles, our method takes into account the following process through which rappers typically create a rhyming response verse to the opponent's verse. First, based on the last word in the opponent's verse, rappers decide on a word to use as a *rhyme*. It is also important to choose a word that can serve as an *answer* to the opponent's verse. Then, they create a rap verse that includes that word.

With reference to this process, our method first finds candidate words that have *rhyme* vowels based on the opponent's verse and selects a set of semantically related words from them (section 3.2). It then generates a rap sentence by using each of the selected words, resulting in a set of the rap sentences (section 3.3). Finally, following the previous approach of arranging the sentences to generate rap lyrics [11], the method constructs a response verse by appropriately arranging (deciding the order of) the generated rap sentences (section 3.4).

3.1 Rap Battle Corpus

Rap Battle Corpus Our rap battle corpus was created by transcribing videos of rap battles in Japanese by using the crowdsourcing service Lancers [3]. The videos of rap battles were selected from the following three popular YouTube channels with over 100,000 subscribers: *UMB* [4], *Gaisen MCBATTLE* [5], and *Sengoku MCBATTLE* [6]. Workers hired through Lancers were native Japanese speakers and were not required to be familiar with rap battles. They accessed a web page that we created for the task and transcribed all the verses in the rap battles while watching the specified videos on YouTube. In total, 691 rap battles were transcribed by 194 workers.

To expand the size of the corpus, we also transcribed rap battles that were broadcasted on two TV shows: *High School Rap Championship* and *Freestyle Dungeon.* Although these videos were publicly available on the web, viewing them required payment. Hence, instead of using crowdsourcing, we had one university student transcribe 596 rap battles.

[3] https://www.lancers.jp/
[4] https://www.youtube.com/user/umbofficial
[5] https://www.youtube.com/channel/UCe_EvY8GrvYgx8PbwRBc75g
[6] https://www.youtube.com/user/senritumc

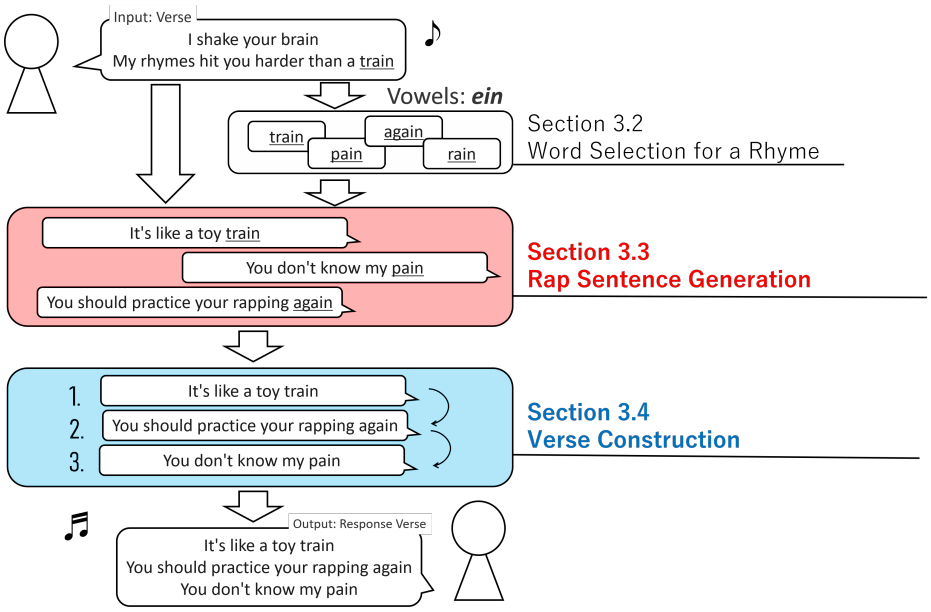

Fig. 2. Overview of the proposed verse generation method.

The final statistics for the corpus are as follows: 1,287 rap battles, 6,791 verses, 52,422 sentences in verses, an average of 7.71 sentences per verse, and an average of 15.40 tokens (words) per sentence. All the raps in the videos are performed in Japanese.

Rap Sentence Pair Data To train a model for generating rap sentences, we need pairs of input and output rap sentences. Therefore, we created pairs of rap sentences by taking two consecutive sentences in a verse. For example, if a verse consists of three sentences, two pairs of rap sentences are created (a pair of the first and second sentences and a pair of the second and third sentences). In total, we created 22,125 pairs of rap sentences from the 6,791 verses in the corpus (hereafter we refer to the pair data "rap sentence pair data").

3.2 Word Selection for a Rhyme

As described in section 3.1, the proposed method first selects words to use as *rhymes* based on the opponent's verse. This process consists of the following three steps.

First, a word that *rhymes* is extracted from the opponent's verse. Rhyming with the final word in the opponent's verse is highly valued in rap battles because it requires a rapper's high ability to respond quickly right after listening to it. Therefore, in this study, the Japanese morphological analysis module MeCab[7] is used to divide the opponent's

[7] https://taku910.github.io/mecab/

verse into morphemes, and the last noun in the verse is extracted as the target word to *rhyme* with.

Next, words that have the same vowel sounds as the target word are searched as candidate words that *rhyme*. We use the words included in the AWD-J dictionary[8] for this search. All the words in the dictionary are converted to vowel sequences in advance, and words with the same vowel sequence as the target word are searched. If the number of vowels in the target word is less than four, a word that has the same vowel at the end is searched. If the number of vowels is four or more, at least four vowels matching from the end are searched to relax the search constraints.

Finally, words that are semantically related to the target word are selected from the searched words. To compute semantic relationships, Japanese word embeddings learned with fastText[9] are used, and the top seven words (i.e., the most semantically related seven words) in terms of the cosine similarity with the target word are selected. In addition, to ensure that one of the generated rap sentences is a definite *answer* to the opponent's verse, the target word itself is also added, resulting in a total of eight words.

3.3 Rap Sentence Generation

After selecting eight words to use as *rhymes*, the next step is to generate a rap sentence that includes each of these words. In general, when generating sentences using a Decoder (forward generation model), words are output from the beginning to the end of the sentence (Figure 3 (a)). However, it is difficult to generate a sentence that must end with the word selected as a *rhyme*. We therefore propose a method for generating a sentence by outputting words in reverse order from the end to the beginning of the sentence. For this method, we first train a rap sentence generation model (reverse generation model) using the reversed sentences (Figure 3 (b)). The method then uses the trained model to generate a rap sentence using the selected word as the starting token (Figure 3 (c)). The details are described below.

For rap sentence generation, we used a generation model BERT2BERT [8], which transfers the pre-trained parameters of the BERT model to the Encoder and Decoder of the Transformer. As the pre-trained BERT model for both the Encoder and Decoder, we used the model publicly available from Tohoku University[10], which were pre-trained on Japanese Wikipedia text data.

To fine-tune BERT2BERT, we used the rap sentence pair data created in section 3.1. First, as shown in Figure 3 (b), each sentence in the pair is divided into tokens using a tokenizer[11]. Next, the first sentence in the pair is used as the input sentence for the Encoder by arranging the tokens in forward order, and the second sentence is used as the output sentence for the Decoder by arranging the tokens in reverse order. This enables

[8] https://sociocom.naist.jp/awd-j/

[9] https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.ja.300.vec.gz

[10] https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking

[11] https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking

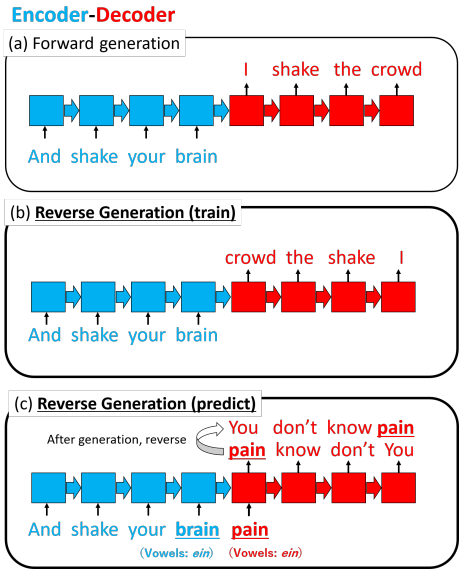

Fig. 3. Differences between forward and reverse generation models.

the model to learn to generate sentences in reverse order. The following hyperparameters were used to train the Encoder and Decoder: a batch size of 32, Adam optimizer, Cross Entropy Loss function, a learning rate of 2e-7, a dropout rate of 0.1, max length of 128, and early stopping with a patience of 10. To train the model, we divided the rap sentence pair data into train, validation, and test data with an 8:1:1 ratio. The training was completed in 224 epochs.

Finally, we used the trained model to generate a token sequence by inputting the last sentence of the opponent's verse to the Encoder and specifying one of the words selected in section 3.2 as the initial token for the Decoder. Then, by reversing and concatenating the generated token sequence, a rap sentence is generated for each of the eight selected words. The parameters used for generation were as follows: max length of 30, top k of 10, top p of 0.95, and no repeat n-gram size of 2.

3.4 Verse Construction

In section 3.3, because eight rap sentences are independently generated, we need to arrange them in an appropriate order to construct a verse. To do this, we first train a BERT model through the next sentence prediction task in which the model predicts whether two given sentences are appropriate as consecutive sentences (Figure 4). The sentence pairs in the rap sentence pair data was used as positive examples, while negative examples were created by replacing the second sentence of each sentence pair in the rap sentence pair data with a randomly selected second sentence of another pair. We created 10,748 pairs of sentences (5,374 positive examples and 5,374 negative examples) and divided them into train, validation, and test data at a ratio of 8:1:1. We again used the

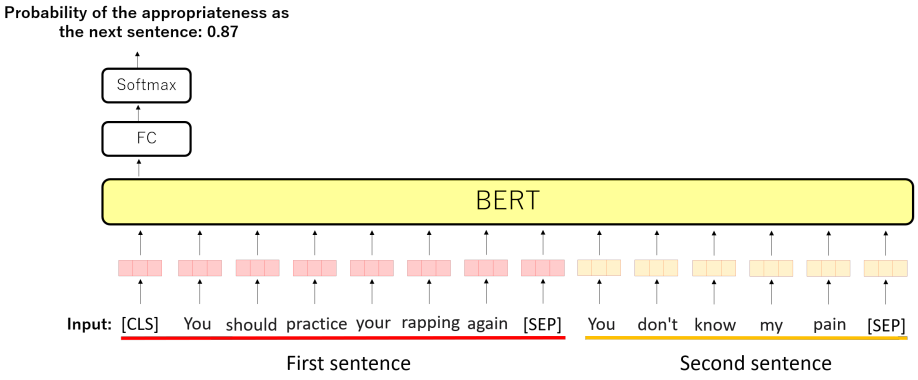

Fig. 4. The next sentence prediction task.

Japanese pre-trained BERT model released by Tohoku University[12]. The hyperparameters during training were as follows: a batch size of 8, Adam optimizer, Cross Entropy Loss function, a learning rate of 2e-7, a dropout rate of 0.1, max length of 128, and early stopping with a patience of 10. The training was completed in 30 epochs, and the accuracy on the test data was 0.60.

Using the trained BERT model, we construct the response verse. Our method first selects the rap sentence containing the target word used at the end of the opponent's verse as the first sentence of the response verse. It then uses the trained BERT to select the most appropriate sentence (i.e., the sentence with the highest estimated probability of being suitable) among the remaining seven sentences as the second sentence of the verse. It repeats this selection process: it selects the most appropriate remaining sentence as the $n+1$th sentence next to the nth sentence ($2 \leq n \leq 7$). It thus constructs the response verse consisting of the eight sentences.

4 Evaluation

Quantitative and qualitative evaluations were conducted to present the effectiveness of the proposed method. The quantitative evaluation was based on three aspects: the naturalness of rap, the quality of *rhyme*, and the quality of *answer*. In the qualitative evaluation, we compared the forward and reverse generation results.

4.1 Quantitative Evaluation

To verify the usefulness of generating sentences in reverse order from the end of the sentence, a comparison was made with the method of generating sentences in forward order from the beginning of the sentence. In the comparison method, all processes except for the direction of sentence generation were the same as the proposed method. The

[12] `https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking`

Table 2. The average scores of the three aspects for the proposed and comparison methods.

method	Naturalness of rap	Quality of *rhyme*	Quality of *answer*
Forward generation (comparison method)	2.56	1.77	2.48
Reverse generation (proposed method)	**3.20**	**3.58**	**2.92**

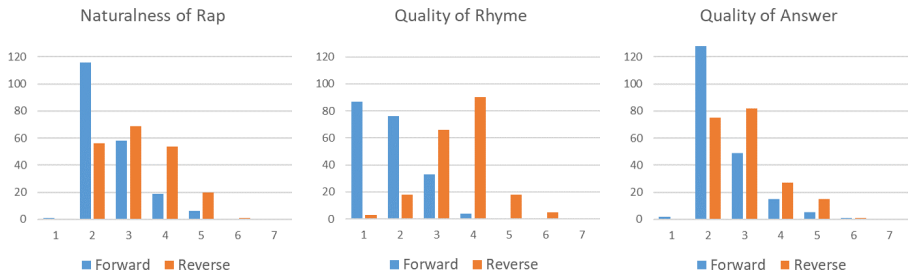

Fig. 5. The score distribution of the three aspects for the proposed ("Reverse") and comparison ("Forward") methods.

hyperparameters used for training the comparison method were also the same. First, we randomly selected 100 of the 1,287 rap battles in the rap battle corpus, and used the last sentence of each rap battle as the input sentence (i.e., the last sentence of the verse). Then, for each input sentence, one verse was generated using the proposed method and the comparison method, resulting in a total of 200 verses generated for the 100 input sentences. Note that the last sentence of each rap battle was not input to the Encoder during the training, so this experiment treated the sampled 100 sentences as unknown data. Two evaluators with over five years of experience in watching rap battles evaluated each of the 200 verses based on a 7-point scale (1: very poor to 7: very good) for each of the three aspects: the naturalness of rap, the quality of *rhyme*, and the quality of *answer*.

Table 2 shows the average score of the three aspects for the proposed (reverse generation) and comparison (forward generation) methods. As shown in the table, the proposed method outperformed the comparison method in all the aspects. To analyze the differences in the results between the methods in more detail, the score distribution of the three aspects is shown in Figure 5. As the proposed method considerably outperformed the comparison method in the quality of *rhyme* in Table 2, Figure 5 also shows that the proposed method generated many verses with scores of four or higher, and that the comparison method rarely generated verses with scores of four or higher. Since all processes except for the generation direction suitable for *rhymes* were the same in the proposed and comparison methods, as expected, their score differences were smaller in the naturalness of rap and the quality of *answer* than in the quality of *rhyme* in Table 2. However, as shown in Figure 5, the comparison method generated a large number of

Opponent's verse:	Response verse generated by the proposed method:
リズム倍で乗ってはめて (Riding on the rhythm, fitting in.) 遊ぶだけは俺もできる (I can play just for fun too.) こんな奴のスキルならば更に上を行く (My skills go beyond you.) ニガリを越すため３連覇 (To surpass Nigari, I'll aim for a triple crown.) お前全然足りない鍛錬が (You haven't trained nearly enough.) おいおい俺にアンサーがない (Hey, you say I don't have an answer.) ってアンサーがないのは (But the one without an answer is you.) あんたじゃない勝ちたいなら韻 (If you want to win, use rhyme.) 沢山踏もうコイツはｆａｋｅ拡散希望 *(KIBOU)* (Let's use a lot of rhymes, this guy is fake, I hope it spreads.)	俺のネタにしてもマジで腐って俺はここに拡散希望 *(KIBOU)* (My jokes are seriously rotten, and I want them to spread here.) そうじゃない俺はイケてるマジで政治志望 *(SIBOU)* (Not like that, I'm seriously interested in politics.) 何度も言って俺を否定をしてるんじゃないよみんなが見てきた理想 *(RISOU)* (I'm not saying it over and over again, everyone has seen the ideal.) 口だけは教えてくれると思ったんでなくて俺は否定して見てるだけの未来志向 *(SIKOU)* (I don't expect anyone to teach me with just words, I'm just looking towards the future.) 言ってぇって言っただけじゃねぇ俺が持ってるだけの魅力 *(MIRYOKU)* (I didn't just say it, I have the charm.) そんなことは絶対に言わないって言われても興味はないな俺の利用 *(RIYOU)* (Even if you say you'll never say something like that, I'm not interested in using it.) ラップして確かに無限の意欲 *(IYOKU)* (Rapping with infinite motivation for sure.) お前が負けるのは俺は自分の意向 *(IKOU)* (You'll lose because I have my own intentions.)
	Response verse generated by the comparison method:
	俺が勝ち上がることはない (I won't be able to climb to the top.) 俺が勝つのを見たぜ (But I saw myself win.) 俺は俺が優勝した (I won the championship.) 俺も俺のやり方ではない (I'm not doing it my way either.) 俺はそうやってるのは (I'm doing it like this.) この場でやってるぜ (I'm doing it here.) 俺がラップはこのままでやってきてるんだぜ (I'll keep rapping like this.) 俺はラップは上手いんだ (I'm good at rapping.)

Fig. 6. Examples of response verses generated by the proposed and comparison methods. Since verses are generated in Japanese, the English translation is shown in parentheses.

verses with a score of two, while the proposed method generated fewer verses with a score of two, indicating that the proposed method has the advantage that low-quality verses are less likely to be generated. From these results, it was demonstrated that generating verse sentences in reverse order from the end of the sentence is indeed effective.

4.2 Qualitative Evaluation

The upper and lower rows of Figure 6 show examples of response verses generated by the proposed and comparison methods, respectively. In the example in the upper row, the vowels of the words that need to *rhyme* with the opponent's verse are "*iou*" and highlighted with red color. The proposed method reflects this *rhyme* by ensuring that each sentence in the response verse ends with a word that has the vowels of "*iou*." Despite imposing the constraint of rhyming in each sentence, meaningful sentences can be generated and are suitable as *answers*. In contrast, none of the sentences in the comparison method's verse end with a word that has the vowels of "*iou*." These results also clearly demonstrate the usefulness of learning and generating sentences in reverse order.

5 Conclusion

In this paper, we proposed a verse generation method that takes into account *rhymes* and *answers* in rap battles and verified its effectiveness. Although we used a Japanese

rap battle corpus, the proposed method itself is language-independent, and we would like to verify its usefulness in other languages such as English in the future. To construct response verses more flexibly, future work will also include the extension of our method to find a rhymed word that is different from the last noun in the verse and is strongly related to the opponent's verse, or to take consonant similarity [16] into consideration in the case of Japanese rap [17]. Finally, as mentioned in section 1, since our future goal is to support people who are unfamiliar with rap to practice it, we would like to develop an interactive verse generation system equipped with the proposed method and verify its usefulness in training support.

References

1. Venla, S.Interactive Oral Composition: Resources, Strategies, and the Construction of Improvised Utterances in a Finnish Freestyle Rap Battle. The Journal of American Folklore, vol.132, no.523, pp. 3–35 (2019).
2. Devlin, J., Chang, M., Lee, K., and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL'19, pp. 4171–4186 (2019).
3. Sutskever, I., Vinyals, O., and Le, Q.: Sequence to Sequence Learning with Neural Networks. Proceedings of NIPS'14, pp. 1–9 (2014).
4. Rumelhart, D., Hinton, G., and Williams, R.: Learning Internal Representations by Error Propagation. tech. rep., University of California, San Diego (1985).
5. Gers, F., Schmidhuber, J., and Cummins, F.: Learning to Forget: Continual Prediction with LSTM. The Journal of Neural computation, vol.12, no.10, pp. 2451–2471 (2000).
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, U., and Polosukhin, I.: Attention Is All You Need. Proceedings of NIPS'17, pp. 6000–6010 (2017).
7. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I.: Language Models Are Unsupervised Multitask Learners. tech. rep., OpenAI (2019).
8. Sascha, R., Shashi, N., and Aliaksei, S.: Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. Proceedings of TACL'20, pp. 264–280 (2020).
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J.: Distributed Representations of Words and Phrases and Their Compositionality. arXiv preprint arXiv:1310.4546 (2013).
10. Potash, P., Romanov, A., and Rumshisky, A.: Ghostwriter: Using an LSTM for Automatic Rap Lyric Generation. Proceedings of EMNLP'15, pp. 1919–1924 (2015).
11. Malmi, E., Takala, P., Toivonen, H., Raiko, T., and Gionis, A.: DopeLearning: A Computational Approach to Rap Lyrics Generation. Proceedings of SIGKDD'16, pp. 195–204 (2016).
12. Manjavacas, E., K., , and Karsdorp, F.: Generation of Hip-Hop Lyrics with Hierarchical Modeling and Conditional Templates. Proceedings of INLG'19, pp. 301–310 (2019).
13. Xue, L., Song, K., Wu, D., Tan, X., Zhang, N., Qin, T., Zhang, W., and Liu, T.: DeepRapper: Neural Rap Generation with Rhyme and Rhythm Modeling. Proceedings of ACL'21, pp. 69–81 (2021).
14. Wu, D. and Addanki, K.: Learning to Rap Battle with Bilingual Recursive Neural Networks. Proceedings of IJCAI'15, pp. 2524–2530 (2015).
15. Savery, R., Zahray, L., and Weinberg, G.: Shimon the Rapper: A Real-Time System for Human-Robot Interactive Rap Battles. Proceedings of ICCC'20, pp. 212–219 (2020).
16. Kawahara, S.Half rhymes in Japanese rap lyrics and knowledge of similarity. The Journal of East Asian Linguistics, vol.16, no.2, pp. 113–144 (2007).
17. Manabe, N.Globalization and Japanese creativity: Adaptations of Japanese language to rap. The Journal of Ethnomusicology, vol.50, no.1, pp. 1–36 (2006).