Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)
16/1 (2023), 59-67. DOI: http://dx.doi.org/10.21609/jiki.v16i1.1113

Poetry Generation for Indonesian Pantun: Comparison Between SeqGAN and GPT-2

Emmanuella Anggi Siallagan*, Ika Alfina†
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia
*Email: *emmanuella.anggi@ui.ac.id, †ika.alfina@cs.ui.ac.id*

Abstract

Pantun is a traditional Malay poem consisting of four lines: two lines of deliverance and two lines of messages. Each ending-line word in *pantun* forms an ABAB rhyme pattern. In this work, we compare the performance of Sequence Generative Adversarial Nets (SeqGAN) and Generative Pre-trained Transformer 2 (GPT-2) in automatically generating Indonesian *pantun*. We also created the first publicly available Indonesian *pantun* dataset that consists of 7.8K *pantun*. We evaluated how well each model produced *pantun* by its lexical richness and its formedness. We introduced the evaluation of *pantun* with two aspects: structure and rhyme. GPT-2 performs better with a margin of 29.40% than SeqGAN in forming the structure, 35.20% better in making rhyming patterns, and 0.04 difference in giving richer vocabulary to its generated *pantun*.
Keywords: poetry generation, *pantun*, text generation, SeqGAN, GPT-2

1. Introduction

Poetry generation is one of Natural Language Processing (NLP) tasks that focuses on automatically generating poetry. Poetry generation has a unique text generation approach, combining the model knowledge of linguistics, creativity, and originality in every generated text it produces. This complexity made poetry generation one of the focused challenges in creativity computational linguistics in the last 50 years [1]. This field is not only beneficial to computer algorithm exploration in creativity but also can be beneficial in the entertainment, advertising, and education sector [2].

There are two types of poetry based on their writing rule. The first does not have controlled rules in their writing–usually modern poetry–and the other has controlled rules–usually classical poetry. While writing a modern poem meets its challenge in writing an aesthetic text, classical poetry meets its other challenge by bounding to its writing rules to be defined as good poetry.

Limerick is one of the most renowned classical poetry in English. Limerick is a five-line verse of poetry with rhyme scheme AABBA that was popularized in the early 18th century in England. Limerick is used for entertainment with its humorous content. With its writing rules, limerick is often implemented in poetry generation [3–5]. Limerick implementations in poetry generation produced good and accurate writing rules.

Besides a limerick, Chinese classical poems are also broadly implemented in the poetry generation. There are various types of classical Chinese poetry. Each type of poetry has different rules, such as words, rhyming pairing, and tone patterns. Classical Chinese poetry is also commonly implemented in poetry generation [2, 6, 7].

In writing classical poetry, the model must be able to produce a new, unique, yet bound to writing rules of poems. A generative model of poetry generation method can comply with the challenges of learning the structure of the poems by itself instead of relying on the predefined structure. The idea of generative models is to learn the distribution of a given training dataset to create new data samples by focusing on the probabilistic [8]. The model must be probabilistic, not deterministic, to produce new samples rather than repetitive ones. The approach aligns with the need for creative text generation to get an original, unique, with the least possible plagiarism.

SeqGAN was first proposed to modify the implementation of the generative adversarial method (GAN) [9] for sequential data such as text [10]. It

Kalau ada sumur di ladang
if there is a well in the field
Boleh kita menumpang mandi
could we take a bath there
Kalau ada umur yang panjang
if we still live long enough
Boleh kita berjumpa lagi
may we meet again?

Figure 1. Example of *pantun*

caters to the needs of the different problems met by implementing GAN in text data by performing reinforcement learning (RL) approach for the discriminator. In [10], SeqGAN is implemented in Chinese poetry. This poetry as well as has writing rules on its writing. SeqGAN proved to perform better BLEU and human evaluation than a model that only trained with maximum likelihood estimation (MLE).

GPT-2 is a model that is based on the transformer model [11], specifically for text generation tasks. GPT-2 model built with the decoder from the transformer model blocks and able to produce text without any specific supervised training [12]. This model has been implemented in classical poetry in various languages. From Chinese Classical Poetry [7], Vietnamese Poetry [13], English Limerick [3], to Arabic Poems [14]. All implementations have proven that GPT-2 is able to produce convincing and indistinguishable human-made poetry.

In Malay literature, there is a similar type of classical poetry with bounded writing rules named *pantun* as shown in Figure 1 which has similar writing rules as limerick and classical Chinese poetry. *Pantun* commonly has four lines and forms an ABAB rhyme scheme. The purpose of *pantun* is very broad, from formal to informal. Its content consists of two parts: deliverance and message. The deliverance usually has no logical connection with the message [15]. The rhyme in words aimed to ease the audience to understand the message.

Classical Malay literature is also inherited from the countries that use the language, officially and unofficially. As a language of Austronesian, the Malay language is spoken in Indonesia, Malaysia, Singapore, and Brunei, while also spoken–unofficially–in East Timor and some parts of Thailand) [16]. As an official language in several countries, Malay is called differently. In Malaysia, it's called Malaysian Language. In Brunei and Singapore, it's called the Malay Language. And in Indonesia, it's called the Indonesian Language.

As far as we know, *pantun* generation in Indonesian has not been studied yet. However, for modern poetry generation, it has been implemented in Indonesian [17]. The implementation was done by a constraint satisfaction approach. This was done by utilizing the writing framework of poetry and the slot-fillers method.

In this work, with each good performance on generating controlled creative text such as classical poetry, we compare the two notable poetry generation methods: Sequence Generative Adversarial Nets (SeqGAN) and Generative Pretrained Transformer 2 (GPT-2) to generate Indonesian *pantun*. We also propose how to evaluate the accuracy of the rhyming pattern of Indonesian *pantun*. Therefore, the contributions of this work are:

- Comparing the performance of SeqGAN and GPT-2 in generating Indonesian *pantun*.
- Propose how to evaluate the accuracy of the rhyming pattern of Indonesian *pantun*.
- Provide the first *pantun* dataset that consists of 7.8K *pantun* that we share for public[1].

This work is organized as follows. Section 2 discusses relevant works on poetry generation. In Section 3, we explain the SeqGAN and GPT-2 model we adopted. We explain the process of building the dataset in Section 4. The experiments and results are presented in Section 5. Finally, Section 6 discusses the conclusion and future works.

2. Related Works

Poetry generation has many approaches. In early works, it uses a framework-based approach like using a corpus with templates [1, 17]. Then the approach shifts to using neural network model [18, 19], variational autoencoder [2, 6], and modified GAN [4, 10].

Full face poetry implemented the approach of creating poetry that follows a single author's writing style [1]. The implementation of neural networks introduced to be capable of forming infinite poets based on the topic and rules of the poetry [18]. The use of more advanced models other than frameworks then became more common. Implemented in limerick and sonnet with long short-term memory neural network [19], adversarial generative set up [4], and for non-thematic Chinese poetry for semi-supervised conditional variational autoencoder [2].

With the rise of implementation of GAN, for sequential data, SeqGAN was introduced to comply with the difference in approach. SeqGAN is a modified GAN model that is used to handle the generative task of sequential data like text [10]. GAN

[1]https://github.com/ir-nlp-csui/sampiran

is a generative model that is capable of producing new samples from a similar distribution of data. The main structure of GAN is the generator and the discriminator. The generator's role is to produce a sample, and the discriminator's role is to classify which are the fake-generated samples and which are the samples from the dataset. However, GAN was proposedly built for image generation, and the approach of the discriminator model for sequential data is totally different. The discriminator in GAN's model only classifies and returns loss for the entire sequence. However, a sequential dataset cannot be classified as a whole. Instead, it requires partial scoring. These concepts of partial feedback align with the concept of reinforcement learning (RL). This combination of approaches is the idea of SeqGAN.

In [10], SeqGAN was implemented in Chinese poetry [10]. SeqGAN has proven to perform better BLEU and better human evaluation than a model that is only trained with maximum likelihood estimation (MLE).

Generative pre-trained transformer 2 (GPT-2) is a pre-trained language model that is built over transformer model [12]. GPT-2 modifies the Transformer model [11] with an adversarial model. In this work, we implemented the Indonesian Small GPT-2 with fewer parameters[2]. The general model consisted of 12 layers of decoders. In every layer, there are 12 independent attention heads and passed to 144 distinct attention patterns. This attention-based model makes GPT-2 suitable for longer text dependencies and lets the model focus on the encoding of the input sequence (see Figure 2).

As a leading model in text generation, GPT-2 has also been implemented for poetry generation, especially in classical poetry with writing rules. There are various languages and types of poetry, such as Chinese classical poetry [7], English limerick [3], and Arabic poems [14], and with custom loss in the pretraining model of Vietnamese poetry [13].

In measuring the goodness of each poetry, poetry generation has its own unique approaches by correlating the metrics with the characteristics of the poetry generated. Referring to recent works in GPT implementation in Poetry Generation. The implementation used various metrics to prove the quality of generated Poems. [7] measures the quality with model conciseness, poetry diversity, and artistry with the correctness of complex and different Chinese poetry. Meanwhile, in the implementation of GPT in limerick [5], the evaluation used is the diversity statistics, syntactically correctness, and consistency in subject and topic measured by lexical diversity,

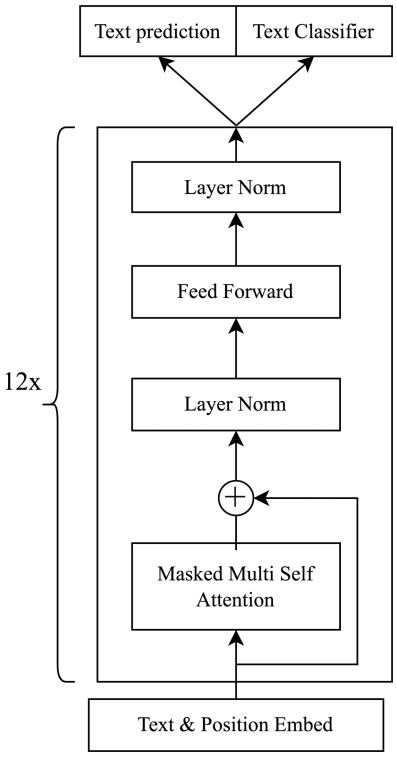


Figure 2. GPT-2 Architecture

subject continuity, BERT-based embedding distance, and content classification.

With much research that focuses on poetry generation, there is not yet implementation of poetry generation that uses Indonesian literary heritage. By *pantun*'s unique structure, it is a challenging task on generative models. To study the implementation of poetry generation, especially with generative models in Indonesian, we modestly implement notable models, SeqGAN and GPT-2, on the *pantun* dataset.

3. The Models

In this section, we explain two generative models we compare for *pantun* generation: SeqGAN and GPT-2.

3.1. SeqGAN

For SeqGAN, we adopted the implementation by [10][3]. Figure 3 shows the flow of SeqGAN implementation.

Initially, the *pantun* dataset was altered to sequences with a tokenizer, and each sequence will

[2]Pretrained model is openly available in https://huggingface.co/cahya/GPT-2-small-indonesian-522M

[3]https://github.com/LantaoYu/SeqGAN

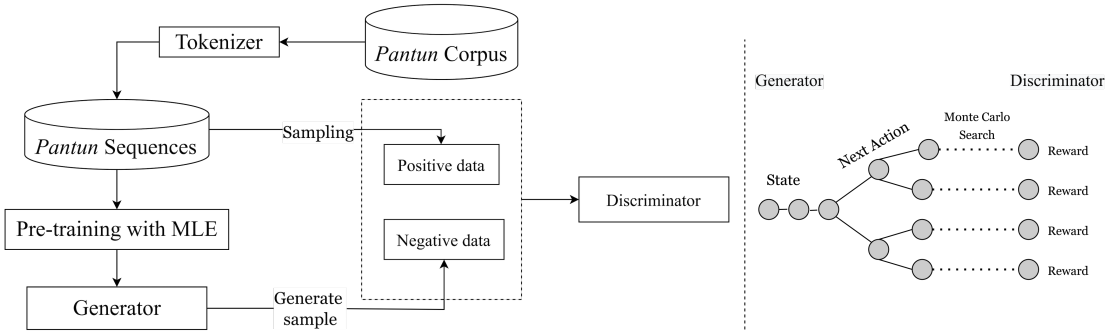


Figure 3. SeqGAN Implementation Diagram

consist of a same-length sequence of 32 words. These sequences will then get into the pre-training phase with MLE. This step creates the generator model.

The discriminator training will maximize the log-likelihood of making the correct prediction between positive and negative data. Positive samples are sampled from the *pantun* corpus, whereas the generator model from the previous step will generate the negative samples. The discriminator model will calculate the action-value function reward from the Monte Carlo search and return reward. The policy gradient will update the generator parameters.

The model will repeat this step based on the epoch stated. The current model in every phase generates the negative samples. The discriminator will train the combined generated and sample *pantun* until the model converges. When the model achieved its convergence, we implemented the model to create a 1,000 generated sample of *pantun*.

3.2. GPT-2

The approach of GPT-2 that we added in this work is the fine-tuning method. The model was trained on Indonesian dataset implementation by [12] that already could produce Indonesian text.

The implementation of *pantun* generation with GPT-2 can be seen in Figure 4. The pantun dataset transforms to sequences with a tokenizer. The model will then undergo fine-tuning training. The pre-trained GPT-2 model adapts to the pantun style by training the dataset above the existing model. The fine-tuning purpose will maximize the generated text's output based on the pantun dataset's style.

To get the generated text, it minimized the negative log-likelihood loss with the formula shown in Equation 1. Where model parameters θ and token input is x_i.

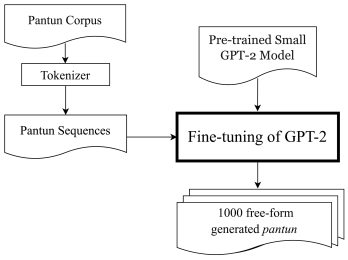


Figure 4. *Pantun* Generation with GPT-2 Model

$$P(\theta) = -\sum_{i=1}^{I} logP(x_i|x_1, ..., x_{i-1}) \qquad (1)$$

After training, 1,000 *pantun* will be generated with sampling methods of Top-K and Nucleus Sampling (Top-p). Top-K took *k* numbers of words with the highest probability. At the same time, shown in Equation 2, nucleus sampling takes the smallest probable combination of a set sequence while making sure that the sum of $V \geq p$.

$$p = -\sum_{x \in V^{(p)}} P(x_{1:i-1}) \geq p. \qquad (2)$$

4. Development of *Pantun* Dataset

In this section, we discuss the development of *pantun* dataset from collecting, removing, and annotating. Finally, we will present the dataset statistics.

4.1. Collecting *Pantun*

Initially, we collected 15,667 *pantun* from various sources that are available publicly, such as blogs, social media accounts, and forums [20–22].

Figure 5. An annotated *pantun*.

4.2. Cleaning *Pantun* Dataset

All *Pantun* collected will processed to be cleaned. We clean the pantun based on the ground rules that build a pantun which are its deliverance and content by its count of lines, rhyming pairs, and syllables [23].

First, the dataset will be filtered so there would be no duplicate entries. Second, each *pantun* will be validated, the *pantun* in our dataset only contains 4 lines-*pantun*. Third, all *pantun* with wrong rhyming pair will be removed. Lastly, *pantun* data that didn't have eight to twelve syllables was removed. After cleaning the data, we have dataset of 7,879 *pantun*.

4.3. Annotating *Pantun*

The *pantun* dataset was annotated by adding the tags for structure identifiers. For every *pantun*, there will be identifiers for a start with a BOS tag, the end of *pantun* with an EOS tag, the end of the line with a CLS tag, and starting of content lines with a CONTENT tag. Each *pantun* will be annotated like this:

$<$***BOS***$>$ $[first\_line]$ $<$***CLS***$>$ $[second\_line]$ $<$***CLS***$>$ $<$***CONTENT***$>$ $[third\_line]$ $<$***CLS***$>$ $[last\_line]$ $<$***EOS***$>$

An example of a sequence from the dataset can be seen on Figure 5.

4.4. Dataset Statistics

The final dataset contains a total of 7,879 pantun that consists of total 133,137 words with 13,149 unique words, and 36,813 total of syllables. The distributions of twenty most common words in the *pantun* dataset's words can be found in Table 1. Note that we do not show function words such as conjunction, personal pronouns, or determiners on this table.

5. Experiments and Results

In this section, we will explain the experiment process. Then, we will analyze the result from each metric used.

Table 1. Top Words Distribution

Word	Quantity	Word	Quantity
pergi "go"	817	*lupa* "forget"	392
hati "heart"	677	*cinta* "love"	381
anak "580"	881	*hidup* "life"	371
orang "people"	579	*kota* "city"	343
ilmu "knowledge"	552	*sekolah* "school"	340
buah "fruit"	510	*pohon* "tree"	339
belajar "study"	509	*burung* "bird"	338
makan "eat"	427	*suka* "like"	331
pagi "morning"	424	*membeli* "buy"	321
bunga "flower"	415	*jalan* "road"	307

Table 2. SeqGAN Hyperparameters

Hyperparameters	Value
Batch Size	64
Sequence Length	32
Epoch on Pre-training MLE (Generator)	120
Discriminator Epoch	50
Training Set	12,896

Table 3. GPT-2 Hyperparameters

Hyperparameters	Value
Batch Size	24
Sequence Length	32
Epoch	10
Learning Rate	5e-5
Training Set	10,340
Testing Set	2,585
Top-K	50
Top-p	0.92

5.1. Experiments

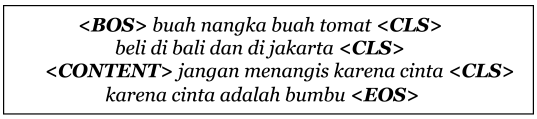
The training was done with one NVIDIA P100 GPU and took 6 hours for SeqGAN training and 2 hours for fine-tuning GPT-2.

Details of hyperparameters for SeqGAN can be seen Table 2 and for GPT-2 can be seen Table 3. Batch size is the amount of data that are used for one forward and backward pass, while the sequence length is the length of each word in one data.

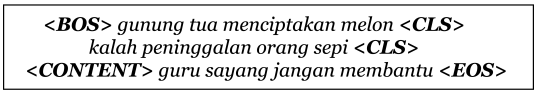
For SeqGAN, the training and its epoch were differentiated into two processes: Generator and Discriminator. Whereas the GPT-2 was only done with one fine-tuning process. Dataset for SeqGAN will not be separated between training and testing set as this approach of SeqGAN validates from the generated text and training set in training. While for GPT-2 implementation, the dataset will be divided into training and testing sets, randomly with a ratio of testing compared to training set 1:5.

5.2. Evaluation Metrics

We used three evaluations to evaluate the generated *pantun*. We assessed the diversity of the vocabulary with lexical richness. Then in the *pantun*



a. Correct Structure of *Pantun*



b. Wrong Structure of *Pantun*

Figure 6. Example Evaluation for Structure Correctness

formedness with structure and rhyme scheme correctness.

5.2.1. Lexical Richness. Lexical richness counts the ability of the model to produce text with rich vocabulary. This metric is used to know if the model is able to use diverse and non-repetitive vocabulary. This is calculated with type-token ratio (TTR) with the Herdan formula in Equation 3, where lexical richness (C) is counted with the log of every word's appearance divided by all vocabulary.

$$C = \frac{log(w)}{log(N)} \tag{3}$$

5.2.2. Structure Correctness. Correct *Pantun* should consist of four lines separated by two lines of Sampiran and two lines of Isi. These rules were taken from common Pantun's rules [23] and ones that applied on the dataset. With a predefined token that is inputted before the fine-tuning, it is expected that the model trained can generate accordingly. The example of a correct and wrong structure of *pantun* in this metric can be found in Figure 6. As shown in Equation 4, where S is the correct structure from *pantun* and N is all *pantun* generated from each model.

$$StructureAccuracy = \frac{S}{N} \tag{4}$$

5.2.3. Rhyme Correctness. To evaluate the rhyme correctness, we counted how many *pantun* with the correct rhyme that was generated by each model. For each *pantun*, first, the ending word in each line is taken, after that we compare the word in line 1 with line 3 and line 2 with line 4. A successful pairing is ones with the same rhyme.

Pantun rhyme can be categorized into two: perfect and imperfect rhyming rules. Perfect rhymes pairing have the same last syllables. Imperfect rhyme pairing has the same vocalization but partly on its syllable form. For example, in Figure 7 (a), perfect rhyme, the last syllable of the first and second lines with third and fourth lines (*lu* and *an*) are the same. As for the imperfect rhyme in Figure 7 (b), both pairings last syllables of the last word in each line are only the same in the vocalization (Indonesian words commonly sound the same as how it's written). The false rhyming is shown in Figure 7 (c), where the rhyming pairs are not qualified for perfect nor imperfect rhyming. In this work, for the script effectiveness, we use the validation using the imperfect rhyme scheme.

Table 4. Experiment Results

Metrics	SeqGAN	GPT-2
Structure Accuracy	73.09%	99.50%
Rhyme Correctness	14.49%	49.70%
Lexical Richness	0.82	0.78

Algorithm 1 Rhyme Evaluation

```
function RHYME CHECK(pantun generated)
    Extract last words from every lines
    Break down each words to syllables
    Get last vowel of each lines' syllables
    if vowel line 1 == vowel line 3 and vowel line 2 == vowel line 4 then
        return 1
    else
        return 0
    end if
end function
```

5.3. Result and Analysis

In this section, we applied each metrics to 1,000 *pantun* generated from SeqGAN and GPT-2. We then compare both models' performances. Table 4 shows the experiments results.

5.3.1. Lexical Richness. By only a 0.04 difference, GPT-2 produces richer vocabulary in the generated texts compared to SeqGAN. The result describes the spread usage of each word from the dictionary. The less the score means each word does not repetitively appear in the text of 1,000 generated *pantun*. The score of both models proves that both models perform equally well in creating the text with words from the dataset.

5.3.2. Structure Correctness. GPT-2 performs significantly better than SeqGAN in producing the correct structure in the text produced. It has the ability to mimic the structure from the dataset and right-formed *pantun* by a significant 26.40%. This is seen

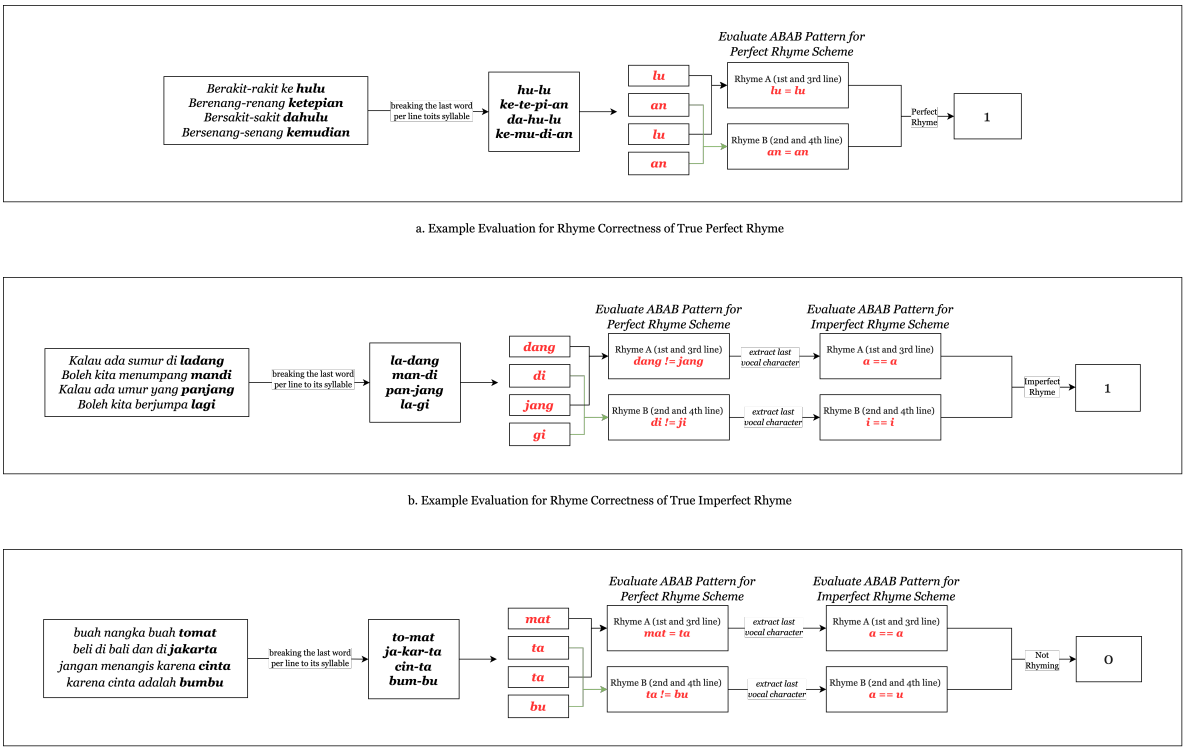


Figure 7. Example Evaluation for Rhyme Correctness

that GPT-2 successfully generates *pantun* text with four lines and gives the token identifier before the third line.

5.3.3. Rhyme Correctness. Both models produce low accuracy. GPT-2 model produced 49.70% accuracy on the generated *pantun*. Meanwhile, with a significant difference, the SeqGAN model produced 14.49% with all perfect rhyme and imperfect rhyme.

6. Conclusions and Future Works

In this work, we compare two generative models, SeqGAN and GPT-2, to generate Indonesian *pantun*. We measure the success by *pantun* characteristics in structure, rhyme, and lexical richness. We also built a new *pantun* dataset of 7.8K *pantun* that will available publicly. Moreover, we also propose a rule to evaluate the rhyme correctness of Indonesian *pantun*.

Experiments show that GPT-2 performs significantly better than SeqGAN in all of the metrics. However yet, both models proved to fail in capturing the rhyme scheme, as the best model, i.e. GPT 2, only achieves around 50% for rhyme correctness.

Unfortunately, we haven't conducted an evaluation of the quality of the sentences of the *pantun*. Therefore, it is possible that a *pantun* generated qualified formedness, but sentences between lines are not coherent.

In future works, we recommend addressing three issues: 1) the accuracy of rhyme pattern should be improved; 2) evaluation of sentences coherence and its meaning should be done; (3) moreover, human evaluation of the meaning of the text will also be needed to improve and validate the linguistic quality of the generated *pantun*.

References

[1] S. Colton, J. Goodwin, and T. Veale, "Full-face poetry generation," in *Proceedings of ICCC-2012, the 3 rd International Conference on Computational Creativity*, 2012.

[2] H. Chen, X. Yi, M. Sun, W. Li, C. Yang, and Z. Guo, "Sentiment-controllable chinese poetry generation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019,

Example of *Pantun* Generated SeqGAN	
Accurate Structure & Wrong Rhyming	Accurate Structure & Accurate Rhyming
sayur pare dijual di laut membuat kuaci bitter melon sold in sea to make sunflower seed *batiknya lilin berkarat* the batik is candle corroded *hormati olehmu menuntut ilmu* respect by you pursuing knowledge *agar tak dalam rakyat bicara* so that not deep citizen talks	*paman libur tetap terang* uncle vacays still bright *hanya bintang di balik* only stars reversed *sudah diratapi di lain banyak orang* already mourned by a lot of people *kuulangi dari kelas cilik* i repeat the early class

Figure 8. Example for SeqGAN Model Results

Example of *Pantun* Generated GPT-2	
Accurate Structure & Wrong Rhyming	Accurate Structure & Accurate Rhyming
buah nangka buah tomat dragon fruit tomato *beli di bali dan di jakarta* bought in Bali and in Jakarta *jangan menangis karena cinta* don't cry because of love *karena cinta adalah bumbu* because love is a seasoning	*ibumu menjadi bidadari* your mother becomes angel *mendampingimu sepanjang hari* accompanies you all day long *ketika engkau menjadi bidadari* when you becomes angel *mendampingimu sepanjang hari* *accompanying you all day*

Figure 9. Example for GPT-2 Model Results

pp. 4925–4931. [Online]. Available: https://doi.org/10.24963/ijcai.2019/684

[3] J. Wang, X. Zhang, Y. Zhou, C. Suh, and C. Rudin, "There Once Was a Really Bad Poet, It Was Automated but You Didn't Know It," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 605–620, 07 2021. [Online]. Available: https://doi.org/10.1162/tacl_a_00387

[4] H. Jhamtani, S. V. Mehta, J. G. Carbonell, and T. Berg-Kirkpatrick, "Learning rhyming constraints using structured adversaries," *CoRR*, vol. abs/1909.06743, 2019. [Online]. Available: http://arxiv.org/abs/1909.06743

[5] K.-L. Lo, R. Ariss, and P. Kurz, "Gpoet-2: A gpt-2 based poem generator," 2022. [Online]. Available: https://arxiv.org/abs/2205.08847

[6] X. Yang, X. Lin, S. Suo, and M. Li, "Generating thematic chinese poetry with conditional variational autoencoder," *CoRR*, vol. abs/1711.07632, 2017. [Online]. Available: http://arxiv.org/abs/1711.07632

[7] Y. Liao, Y. Wang, Q. Liu, and X. Jiang, "Gpt-based generation for classical chinese poetry," *CoRR*, vol. abs/1907.00151, 2019. [Online]. Available: http://arxiv.org/abs/1907.00151

[8] D. Foster, *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media, 2019.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[10] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," *CoRR*, vol. abs/1609.05473, 2016. [Online]. Available: http://arxiv.org/abs/1609.05473

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[13] T. Nguyen, H. Pham, T. Bui, T. Nguyen, D. Luong, and P. Nguyen, "SP-GPT2: semantics improvement in vietnamese poetry generation," *CoRR*, vol. abs/2110.15723, 2021. [Online]. Available: https://arxiv.org/abs/2110.15723

[14] M. E. G. Beheitt and M. B. H. Hmida, "Automatic arabic poem generation with gpt-2," 2022.

[15] R. Milyartini, "Singing keroncong and the values behind it," in *Proceedings of the International Conference on Arts and Design*

Education (ICADE 2018). Atlantis Press, 2019, pp. 136–139. [Online]. Available: https://doi.org/10.2991/icade-18.2019.31

[16] Wardhana and D. E. Chandra, "Indonesian as the language of asean during the new life behavior change 2021," *Journal of Social Work and Science Education*, vol. 1, 2021.

[17] F. Rashel and R. Manurung, "Pemuisi: a constraint satisfaction-based generator of topical indonesian poetry." in *ICCC*, 2014, pp. 82–90.

[18] M. Ghazvininejad, X. Shi, Y. Choi, and K. Knight, "Generating topical poetry," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1183–1191. [Online]. Available: https://aclanthology.org/D16-1126

[19] J. H. Lau, T. Cohn, T. Baldwin, J. Brooke, and A. Hammond, "Deep-speare: A joint neural model of poetic language, meter and rhyme," *CoRR*, vol. abs/1807.03491, 2018. [Online]. Available: http://arxiv.org/abs/1807.03491

[20] Anonymous, "Twitter Pantun Gombal," [Accessed 2022-07-06]. [Online]. Available: https://twitter.com/PantunGombal

[21] ——, "Klak klik bermutu," [Accessed 2021-07-06]. [Online]. Available: https://pantuncinta2000.blogspot.com

[22] ——, "Pantun senipedia," [Accessed 2021-07-06]. [Online]. Available: https://www.senipedia.id/

[23] N. Nurhayati, "The use of language in malay pantun (traditional poetry) of bangka: A stylistic study," 08 2011.