Poetry generation with ByT5

Cleo Matzken, Julien Brosseit, Sebastian Ochs, The-Khang Nguyen
Technische Universität Darmstadt
{cleo.matzken, julien.brosseit, sebastian.ochs, thekhang.nguyen}@stud.tu-darmstadt.de

Abstract

Fine-tuning token-based architectures for natural language tasks like poetry generation is common practice. While such models can create coherent texts, they often struggle to generate correct rhymes. There are explicit solutions to this problem, but these usually require additional effort, such as generating the rhyming words at the beginning. We follow the assumption of (Wöckener et al., 2021) according to which token-free models may be better at learning sublexical phenomena such as rhyming. Therefore, we investigate the character-based model ByT5 and propose a decoder-only variant of the model, which we train end-to-end on a dataset containing 110K quatrains annotated with three different rhyme schemes. Our results indicate that, while token-based models such as GPT-2 seem to generate quatrains that are more semantically coherent, our ByT5 model outperforms them in creating line endings that better conform to the rhyme schemes, while having one third less parameters.

1 Introduction

Generation of poetry is a complex task in artificial intelligence. It does not only involve the handling of the syntax, lexical choice and overall semantics, but also requires knowledge about utilizing fitting rhyme schemes and emotional expressions (Oliveira, 2009). Creating poetry is an originative task with the goal to convey its content in an aesthetically pleasing way. This is typically achieved by following a certain configuration that includes a specified length, rhyme scheme, rhythm and further subdivision into structures such as couplets or quatrains. Creativity in poetry generation can also be amplified by utilizing stylistic devices, such as alliterations, metaphors or imagery. (Bena and Kalita, 2020)

In this work, we consider the problem of conditional poetry generation, where the user can specify the desired rhyme scheme that the generated text should satisfy. This approach differs from the unconditional poetry generation process, which passes various unannotated rhymed poems to the model with the expectation to produce rhymes of the same quality as end result. [1]

GPT-2

Enclosed (abba)	
There was a spell once on either s**oul**	**-oul**
They were such strange things to beh**old**,	**-old**
But you're the children of a cloud,	
And they were things to dream and to be.	
Coupled (aabb)	
To a fair grace the brow's ivory l**ight**	**-ight**
Beamed on the eyes, and sweetly sank	
Beneath this dusky mantle's mastering l**ight**,	**-ight**
When from the wondrous servant's care	
Alternate (abab)	
And who, unblushing, flushed at the s**ight**,	**-ight**
Declares the Fates forbade a spark unfold	
Of heavenly power, or that divinely sm**iled**	**-iled**
That gifted of the mind, whose glory drew	

Figure 1: Examples of poems generated by our baseline GPT-2 model, given an enclosed, coupled and alternate rhyme scheme.

Prior poetry generation approaches often rely on Recurrent Neural Networks (RNNs) (Ghazvininejad et al., 2016; Lau et al., 2018; Van de Cruys, 2020; Wöckener et al., 2021; Zhang and Lapata, 2014) due to their proven effectiveness for text generation tasks that can remember and process past information through their high dimensional hidden states with nonlinear dynamics, according to Zhang and Lapata (2014). On the other hand, RNNs often suffer from gradient vanishing or exploding when there are long-term dependencies in sequential data, which makes learning very unstable (Sutskever et al., 2011). However, it is these dependencies that may play an important role in rhymed poetry, as there are often lengthy text se-

[1] This definition was drawn analogously to that of Cai et al. (2021) on the general unconditioned text generation process.

quences between rhyme words.

Transformers, originally proposed by Vaswani et al. (2017), overcome the problem of long-term dependencies by avoiding recursion and instead deploy a fully attention-based mechanism on the complete input sequence. Through pre-training on large amounts of textual data, transformer-based models obtain context-sensitive language knowledge that can be transferred to the text generation task (Radford et al., 2019). Variants of the GPT (Generative Pre-trained Transformer)-2 model, fine-tuned to poetry generation, are capable of outperforming RNNs in terms of semantic coherence, however, they seem to struggle to adapt proper rhyme schemes (Wöckener et al., 2021), as exemplified by Figure 1. To tackle this problem, Wöckener et al. (2021) suggest to provide more samples during training and train a model that directly operates on the character-level. (Bena and Kalita, 2020)

Following these suggestions, our research contributes to the poetry generation task in two main aspects:

1. We create a new quatrain dataset, in accordance with prior approaches (Lau et al., 2018; Wöckener et al., 2021), which contains approximately 110k samples from poems with manually and automatically annotated rhyme schemes.

2. We train a ByT5 (Xue et al., 2021) model, a character-level Transformer architecture based on mT5 (Xue et al., 2020), on our collected data and compare its performance to generate poems to a GPT-2 baseline.

Our experimental results imply that GPT-2 surpasses our model in creating quatrains that are thematically coherent. However, our proposed system achieves better results in rhyming skills, leading us to conclude that character-based models learn better rhyming capabilities than token-based models.

2 Related work

As mentioned in the introduction, many researchers have concentrated on RNNs in prior approaches to the poetry generation task.

Ghazvininejad et al. (2016) introduced Hafez, a poetry generator built as a combination of a finite-state acceptor and a RNN. The finite state acceptor contains a path for every conceivable sequence of vocabulary words which obeys formal rhythm constraints and a selected rhyming word at the end of the line. The RNN then selects a coherent option from the set of paths. Using these methods, Hafez is able to generate poems that were well formed and had an appropriate level of semantic coherence. However, due to its predetermined, finite set of rhyme possibilities, the model may be limited in its capabilities to create unique poems.

Lau et al. (2018) designed Deep-speare, a joint network of long short-term memory (LSTM) architectures that separately capture language, poetic meter and rhyme information, which cooperate to generate sonnets. They achieved state-of-the-art results in terms of rhyme structure and meter, but human evaluation identified that the generated poems lack emotion and readability in comparison to human written poetry.

Van de Cruys (2020) proposed a RNN trained on standard non-poetic text in English and French, which is then constrained by prior probability distributions that model rhyme sounds and topical coherence, which boosts the appearance of words that may fit particularly well in a poem. With the help of an optimization framework, the best verse was then selected. The conditioned RNN generated poems that in half of the cases have not been distinguished from human-written poems by human evaluators. These results should nonetheless be treated with caution, as the evaluated sample size of 5 poems may not be representative enough for overall model performance.

One of the first approaches to utilize transformer-based models for poetry generation is presented in the work of Liao et al. (2019). They adapt their own GPT model based on the popular language model BERT (Devlin et al., 2018) and pre-train it on a Chinese news corpus. During fine-tuning, they train the model on Chinese poetry, providing additional knowledge about the form and theme of the poems.

Bena and Kalita (2020) based their poetry generation approach on GPT-2, but conditioned the model on emotional expression instead of rhyme schemes. They proclaim that the generated poems successfully evoked the intended emotions in readers.

Contrary to the previous approaches, Wöckener et al. (2021) implemented an end-to-end poetry generation system that does not rely on poetry-related information other than the unannotated data itself. Their proposed RNN solely learns from examples and incorporates stylistic factors such as

rhyme scheme, alliteration, sentiment, text length and time period. Furthermore, they train a GPT-2 model with the same methodology and compare the generated poems of both models. While both the RNN and GPT-2 seem to struggle with rhyming, they are able to capture the other features to a certain degree.

GPT-2-based approaches have shown better results in terms of semantic coherence than RNN approaches according to Liao et al. (2019) and Wöckener et al. (2021), however, character-based RNNs seem to adapt better to rhyme schemes (Lau et al., 2018). We therefore expect that using a ByT5 model may be able to combine the advantages of RNN and transformer-based poetry generation approaches to achieve high performances in both semantic coherence and rhyming ability.

3 Data

As most machine learning approaches benefit from larger amounts of data, we decide to combine multiple poetry data sets into a unified text corpus of four-line stanzas. During the data collection, we operate under two main assumptions:

1. Two lines of a text rhyme, if their last syllables (stressed or unstressed) have the same pronunciation.

2. All four lines of a text containing two rhymes are unique poems, if they do not share more than two lines with a directly preceding poem. E.g. a six-line stanza with rhyme scheme *AABBCC* provides two coupled quatrains, *AABB* and *BBCC*.

We limit the collection process to quatrains as they provide more variety in rhyme schemes than two-line stanzas and remain computationally feasible, given our resources, in comparison to stanzas with more lines. Additionally, we only extract alternating, enclosed and coupled four-line stanzas from each considered data set.

3.1 Poems Dataset - Kaggle

The Kaggle Poems dataset[2] consists of 20.7K poems with different poetic forms from different authors. It contains both ancient and contemporary pieces of poetry. Due to the mentioned conditions we imposed to the poems, poetic forms such as haiku, cascade and others were dropped in principle. Quatrains that belong to one of our sought-after rhyme schemes and satisfy our assumptions are manually annotated by us with the respective rhyme scheme. This process resulted in a selection of 3K poems.

3.2 Chicago Rhyming Poetry Corpus

The Chicago Rhyming Poetry Corpus [3] is a collection of English and French poetry from different authors, spanning from the 15th to the 20th century. As all poems are annotated with their corresponding rhyme scheme, we automate the process of finding valid four-line poetry according to our assumptions. This results in a selection of approximately 11.5K poems.

3.3 A Gutenberg Poetry Corpus

The Gutenberg Poetry Corpus [4] concatenates hundreds of poetry books into a single collection of 3M lines of unannotated text. To nevertheless detect viable four-line poems, we apply the following method:

1. Extract the last word of each line of four consecutive text lines.

2. Convert each of these words into its phonemes using the CMU Pronouncing Dictionary [5] or the Grapheme-to-Phoneme converter (Park, 2019), if the dictionary does not contain a certain word.

3. Compute the minimal Levenshtein distance (Levenshtein, 1965) between each pair combination of the (up to) last four phonemes from each possible pronunciation of the last words. An example of the minimal phonemic Levenshtein distance (MPLD) between two words is displayed in figure 2, and a more thorough explanation of the introduced metric is presented in Section 5.2.

4. If two mutually exclusive pairs of last words have both an MPLD smaller than 2, we discovered a four-line poem and its corresponding rhyme scheme (*ABAB*, *AABB* or *ABBA*).

[2] https://www.kaggle.com/michaelarman/poemsdataset

[3] https://github.com/sravanareddy/rhymedata

[4] https://github.com/aparrish/gutenberg-poetry-corpus

[5] As the original website containing CMUdict is not reachable as of March 23rd, 2022, we refer to the repository we utilized for our approach: https://github.com/aparrish/pronouncingpy/

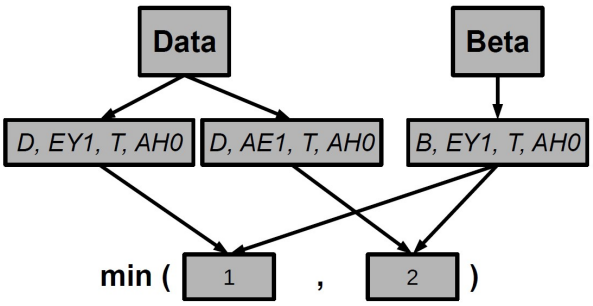

Figure 2: Visualization of the computation of the MPLD between the two words *data* and *beta*.

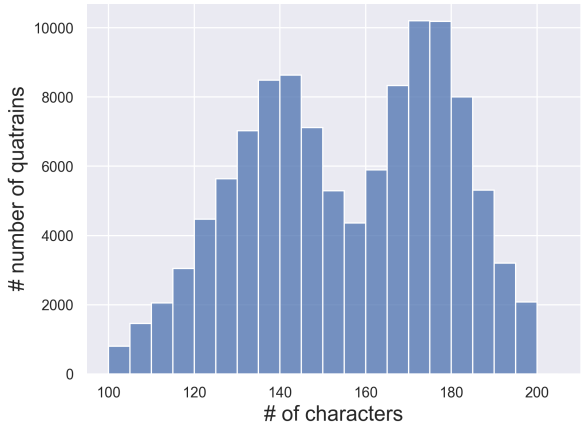

Figure 3: Histogram of the four-line poem dataset.

Furthermore, we discard any potential poem candidates if the sum of phonemes of their last line words is smaller than 12 to increase the reliability of the MPLD. As the Levenshtein distance between the same word becomes zero, and we do not seek to reward any model for using the same rhyming words twice, we also ignore all four-liners if any pair of last words contains the same word.

Using this method, we are able to automatically extract over 110K four-line poems from the Gutenberg Poetry Corpus.

3.4 Preprocessing

Since we utilize our collected data to train a character-based transformer model, we decide to remove all white-space and punctuation symbols that we deem to be less relevant for poetry generation. Also, to ensure that all poems have a sufficient length and are computationally feasible, we discard all four-line poems that have less than 100 and more than 200 characters.

Last but not least, we remove any duplicates that are detectable via string comparison, as some poetry sources may occur in multiple data sets. While this does not guarantee that our data is completely free of duplicates, the method eliminates about 4.5K samples, which certainly increases the data uniqueness.

In the end, our methods create a four-line poem dataset containing over 110K samples with 61K coupled, 41K alternating, and 9K enclosed rhymes. The distribution of the dataset according to the number of characters in a poem is visualized in figure 3.

4 Methods

One of the biggest problems in poetry generation is the correct selection of rhyming words. Fine-tuning token-based transformer models such as GPT-2 (Radford et al., 2019) often result in poetry that appears correct at first glance, but does not adhere to the given rhyme scheme. While other approaches address this problem explicitly by e.g., generating the rhyming words first and then predicting the rest of the text depending on them, we, on the other hand, want the model to learn this characteristic by itself, which eliminates the need for special methods. In (Wöckener et al., 2021), it is suggested that model architectures which are not token-based but work at the character level instead might be able to learn such phenomena implicitly. This assumption is supported by results from (Xue et al., 2021), which presents ByT5, a token-free transformer model for text-to-text generation tasks. In this work, the model is compared to the token-based mT5 (Xue et al., 2020) architecture which it was able to beat on word-level tasks such as grapheme-to-phoneme conversion. Based on these results, we want to fine-tune the ByT5 model on the poetry generation task. The ByT5 model is a pre-trained generative language model that can work directly with character sequences and also predict them. We want to condition this model on the respective rhyme schemes. Since we only have three rhyme schemes, the encoder has limited influence on the rhyme generation, while having three times as many layers than the decoder. This causes a lot of computational effort, especially during inference and training, which is why we propose to remove the encoder. This effectively gives us a decoder-only model with improved inference time and significantly fewer parameters, allowing higher batch sizes to be trained. In order to still provide the decoder with a meaningful hidden state

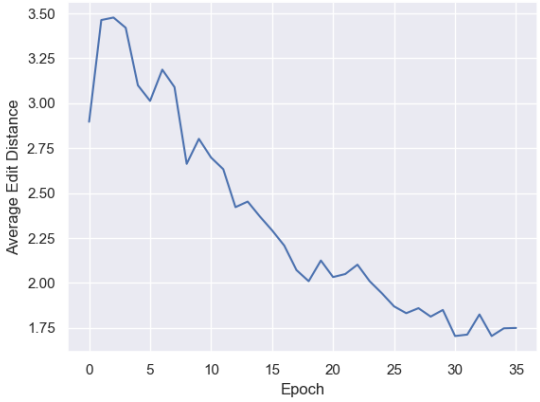

Figure 4: The average MPLD over the entire training of the decoder-only variant. A steady decline in this metric can be seen, making the model increasingly capable of generating correct rhymes. Towards the end of the training, this value begins to stagnate.

of the encoder, we generate a hidden state for each rhyme scheme, which we save and give to the decoder as a condition during training. These hidden states were generated by the pre-trained encoder, which received the respective rhyme schemes as input as a string. This proved to be a sufficiently good representation for our project. Despite significantly fewer parameters, we can also show that the decoder is still able to learn qualitative rhymes.

We use this decoder-only architecture and train it on the 110k dataset of four-line rhymes from the Gutenberg Corpus. The ByT5 model comes in many different variants with various number of parameters. Due to technical limitations on our side, we use the smallest variant with about 300 million parameters in total, of which 218 million fall on the encoder. Therefore our decoder-only variant has only 82 million parameters. While training, we use cross entropy as the loss and Adafactor as the optimizer, which automatically adjusts the learning rate but uses very little memory, which is very beneficial for transformer models that usually have many parameters. We train with a batch size of 8 and accumulate 32 gradient steps before doing an update, which effectively gives us a batch size of 256. Using this procedure we train over a duration of 35 epochs.

During the training, the Minimal Phonemic Levenshtein Distance is tracked after each epoch on 50 generated examples from each rhyme scheme, which serves us as validation to some extent, as visualized in Figure 4. This metric is explained in more detail in Section 5.2. Before training, the hidden states of the encoder are generated for each rhyme scheme and saved. They are not changed afterwards and are directly used for training and inference.

5 Experiments & Results

To evaluate the performance of our ByT5 model presented in section 4, we implement a baseline approach for comparative purposes.

5.1 Baseline

We choose GPT-2 as our baseline model and fine-tune it on our quatrain dataset due to the following reasons:

1. GPT-2 is a general-purpose model, which can even be used without domain-specific training. To achieve this, it has been pre-trained on an enormous corpus of eight million web pages, also known as *WebText*. Due to the unprecedented exposure to large amounts of human-written texts, the transformer-based model is able to compile a language representation that achieves state-of-the-art performances on different natural language tasks, including, but not limited to text generation, reading comprehension and question answering.

2. GPT-2 is able to maintain the overall context of its generated text, making it the ideal choice for the poetry generation task, as the coherency between every poetry line in a stanza plays an important role. Therefore, with further fine-tuning on our dataset, the model should be able to produce meaningful poems.

3. As stated in Section 2, GPT-2 has already been deployed to generate poetry in previous research (Liao et al., 2019), (Bena and Kalita, 2020), (Wöckener et al., 2021). Given the size of our dataset, we may observe the emergence of stylistic features, such as rhyming, in the generated poems of a GPT-2 baseline, as suggested by Wöckener et al. (2021).

As our ByT5 model only contains 82 million trainable parameters, we decide to fine-tune a pre-trained GPT-2 small version with 124 million parameters for a better performance comparison. To simplify the process of fine-tuning, we utilize *gpt-*

2-simple[6], which allows us to train GPT-2 without complex text preprocessing.

To provide information about the rhyme scheme of the quatrains, we modify each input sequence in the following way:

$$< \text{rhyme scheme} >:< \text{poem} > \quad (1)$$

By starting each input sequence with the rhyme scheme, similarly to our ByT5 model, GPT-2 may adapt better to the rhyming constraints.

We fine-tune the baseline using the Adam optimizer with a learning rate of 0.001 and utilizing the sparse softmax cross entropy loss as loss function. Additionally, we set the batch size to 2 and accumulate gradients over 32 batches. In each epoch, we fine-tune the baseline for 400 training steps on the quatrain dataset. We finish the model training after 24 hours, as the cross-entropy loss stopped decreasing.

5.2 Evaluation

We let each model generate 1000 samples for each rhyme scheme (*AABB*, *ABAB*, *ABBA*) and compare both approaches using the following metrics:

Minimal Phonemic Levenshtein Distance (MPLD):

As mentioned in Section 3.3, we apply the Levenshtein distance on a phoneme-level to determine the rhyme scheme of a quatrain. As two words w_0 and w_1 may have more than one correct pronunciation, MPLD computes the Levenshtein Distance (LD) for each pair of the Cartesian product of the different pronunciations phon(w_0), phon(w_1) and returns the minimum. We only compute the MPLD using the (up to) four last phonemes of each pronunciation. The reasoning behind this decision is that we argue that the last syllable is sufficiently captured by these (up to) four last phonemes, and we assume during the data collection that two words rhyme, if their last syllable has the same pronunciation. Equation 2 displays our utilized MPLD as Python-style formula:

$$\text{MPLD}(w_0, w_1) = \min(\big[\text{LD}(x\,[-4:], y\,[-4:]) \text{ for } x, y \text{ in } \text{phon}(w_0) \times \text{phon}(w_1)\big]) \quad (2)$$

Additionally, as we provide both models with the rhyme scheme they should replicate, and each generated poem is supposed to have four lines, we average the MPLD between two rhyme pairs. For example, given the rhyme scheme *ABBA* and the four last line words w_0, w_1, w_2, w_3, we calculate:

$$\text{MPLD}_{ABBA}(w_0, w_1, w_2, w_3) = \frac{\text{MPDL}(w_0, w_3) + \text{MPDL}(w_1, w_2)}{2} \quad (3)$$

While the MPLD provides an estimate of the rhyme quality of the generated poetry, we additionally compute the perplexity of the poems to evaluate their lexical cohesion.

Perplexity (PPL):

We use PPL to estimate how unexpected the generated poems of the baseline and our proposed ByT5 model are to a pre-trained GPT-2 small model (Radford et al., 2019). We calculate the PPL of a poem x by raising e to the power of the cross-entropy loss (CEL) of GPT-2, given x:

$$\text{PPL}(x) = e^{\text{CEL}(x,x)} \quad (4)$$

Maximal Text Similarity (MaxSim):

One important feature we want to have in our model is originality of generated poems. Our method should not simply copy the poems from the dataset, which would also lead to a very low MPLD. Therefore, we want to measure how closely the generated poems match those from the dataset. For each poem, we calculate a similarity to every dataset poem and calculate the maximum similarity with the assumption that one of these poems could be copied. We then take the average of all these max similarities of the poems to determine the metric. To compare two poems, we use the Universal Sentence Encoder (Cer et al., 2018), which computes efficient sentence level embeddings for the texts and has pre-trained models publicly available.

5.3 Results

We present the results of our experiments in Table 1.

The results indicate that our ByT5 approach generates significantly better line-ending words than our baseline. Additionally, the ByT5 model is not too far off the measured MPLD of the dataset, but achieves its best performance for coupled (*AABB*) quatrains, which form the largest class in our dataset. Our baseline, on the other hand, does not seem to follow the rhyming constraints specified

[6] https://github.com/minimaxir/gpt-2-simple

	AABB			*ABAB*			*ABBA*		
	MPLD	PPL	MaxSim	MPLD	PPL	MaxSim	MPLD	PPL	MaxSim
Dataset	0.95	304.36	-	1.02	295.25	-	0.97	281.95	-
GPT-2	3.33	**165.87**	0.55	3.36	**160.51**	0.55	3.38	**151.88**	0.55
ByT5	**1.53**	388.06	0.54	**1.77**	370.30	0.55	**1.89**	353.48	0.54

Table 1: Minimal Phonemic Levenshtein Distance (MPLD), Perplexity (PPL) and Maximal Text Similarity (MaxSim) measured on the generated samples from our baseline GPT-2 model and proposed ByT5 system.

GPT-2		**ByT5**	
Enclosed (abba)			
Among the graves and mountaineers b**are**,	**-are**	Rich in the south and east to **one**,	**-one**
And countless chiefs, if patriots say,		That where we saw the maples w**eep**,	**-eep**
'Tis nothing to be prepared for thee		Sure one cheerful beauty w**ept**,	**-ept**
To shatter from the victor's branches and c**are**.	**-are**	The savor of a Mayor was d**one**.	**-one**
Coupled (aabb)			
When I am dead, he said, I have gr**own**	**-own**	Far from the grass he look'd and dared to b**end**,	**-end**
A good and pious man, after I am dead.		And when within him those arms were well am**end**,	**-end**
He spoke to me then and said This, you must kn**ow**,	**-ow**	And plac'd behind him the wond'rous cens**er**,	**-ser**
This man was an excellent king and knight.		His hand the quiver with the mother then**cer**	**-cer**
Alternate (abab)			
From the first moment when we f**ind**	**-ind**	The fluted cabin stands and gl**ides**,	**-ides**
That the dear Lord within our m**ind**	**-ind**	The clay and the current b**eat**,	**-eat**
Tries to be with us to give,		The burdens to the sky are shivered t**ides**,	**-ides**
We lay aside our joys and play		The cannons to the eaves are s**eat**.	**-eat**

Figure 5: Comparison between samples with a low MPLD generated by GPT-2 and ByT5

by the input, as it does not deviate much from the worst possible MPLD of 4 for all rhyme schemes.

The GPT-2 model significantly surpasses our ByT5 system in the measured perplexity, however, the high perplexity on the quatrains of the dataset may indicate that the metric is biased towards our baseline, as both stem from pre-trained GPT-2 model, which decreases the validity of the metric for this specific comparison.

The MaxSim is consistently low for both models and all rhyme schemes. This shows that both approaches do not simply copy poems from the dataset, but rather generate unique quatrains that may thematically overlap with the data.

Further manual analysis of the generated poems let us conclude that, while the perplexity values may not correctly quantify lexical coherence, the quatrains of our GPT-2 baseline appear to be thematically and semantically more consistent than poems from our ByT5 model. This is illustrated by the side-by-side comparison of samples from both models in Figure 5, where ByT5 mostly adapts to the rhyme scheme, but contains thematic jumps that may disrupt the readability.

We present more examples of generated quatrains in the appendix 7.

6 Discussion

Our ByT5 model is able to more reliably generate quatrains that follow a specified rhyme scheme than our GPT-2 baseline, while having approximately $\frac{2}{3}$ of its parameters. But as the observable difference in semantic coherence between the two systems indicate, our approach is limited in a few aspects.

6.1 Limitations

The Minimal Phonemic Levenshtein Distance (MPLD) may be insufficient to correctly identify rhymes due to its indiscrimination of vowels and consonants. Especially for short line-ending words, MPLD becomes less reliable. For example, the words “fall” and “tall” have the same MPLD as “fall” and “fell”, yet the second pair does not rhyme. While this does not affect the data collection process as much, as we set a lower bound to the sum of phonemes of the last words, MPLD may overestimate the performance of our proposed model. This issue may be preventable by automatically detecting the last syllables of the last words of each line, or by giving more weight to vowels and their following consonants.

As hypothesized in Section 5.3, our computed perplexity metric may be heavily biased towards our baseline, as both rely on the same pre-trained

GPT-2 model. This may be fixable by deploying a different architecture that computes the cross entropy loss, such as mT5 (Xue et al., 2020). In addition to that, MaxSim is too computationally expensive to track during training. Other metrics based on n-gram overlap may be less costly and may better indicate if models plagiarize the dataset during poetry generation.

Our approach correctly generates rhyming patterns, however, the textual content of the poems sometimes lacks lexical cohesion. This partially may be caused by the small decoder structure of the utilized ByT5 model. Given the scale of our dataset, it should be possible to train larger variants of ByT5, which may increase the textual coherence. Unfortunately, training larger models was infeasible with our available computational resources.

As previously mentioned in subsection 3.4, our dataset may not be completely free of duplicates. Additionally, a small number of the automatically collected four line poems are of another language than English, such as 150 poems extracted from a book that contained both English and Spanish poetry. We also purposefully assigned all four-line monorhymes (*AAAA*) to the class of coupled rhymes (*AABB*), as the number of samples that are monorhymes is significantly smaller than the amount of poems in the other rhyme schemes. Last, but not least, our automatic data collection process only works well, because poetry texts often contain rhyme schemes. When applied to other text corpora, our method may often misidentify four consecutive lines as poems, even though the last words do not rhyme.

6.2 Future Work

We were able to show in our work that character aware models like ByT5 are implicitly able to learn rhyme schemes correctly with enough training data. The question now is whether this is really due to the fact that ByT5 is token-free, or whether it is due to design choices of the architecture. Therefore, one could also test other token-free models such as PerceiverIO (Jaegle et al., 2021) or Charformer (Tay et al., 2021), which even outperforms ByT5 on several benchmarks. This could build a deeper understanding of how models can learn internal phenomena in text, such as rhyming.

Furthermore, one could incorporate other information into the model such as the sentiment of the poem (e.g. happy/sad poems) as also done in (Wöckener et al., 2021), or specify the theme, length, meter etc. of the poem. If there are only a finite number of possible conditions, one could also consider to remove the encoder network for gaining faster performance and working with saved hidden states. Replacing the encoder network with a smaller network that returns the same output for the given conditions would also be possible. This could be achieved, for example, through knowledge distillation.

7 Conclusion

Poetry generation is an intriguing challenge in the area of natural language generation, as writing poems usually demands creativity while conforming to rhythmic, stylistic and phonetic specifications.

Prior research mainly focused on architectures based on recurrent neural networks (RNNs) that are modified to some extend for better adaptation of poetic meter and rhyming constraints (Ghazvininejad et al., 2016; Lau et al., 2018; Van de Cruys, 2020). Recent approaches rely on transformer-based models to generate poetry by appending additional information about poems to the input sequences, instead of adjusting model architectures (Liao et al., 2019; Bena and Kalita, 2020). The comparison of style-conditioned RNNs and GPT-2 for poetry generation by Wöckener et al. (2021) shows that subword-based Transformer architectures struggle with learning rhyme schemes from data, however, their output is semantically more coherent than the output of RNNs.

We therefore base our approach to poetry generation on the character-level ByT5 model (Xue et al., 2021), as we estimate that it may adjust better to rhyming constraints than GPT-2 by simultaneously sustain its capability to generate coherent texts. To train the ByT5 architecture on sufficient data, we assemble quatrains and their corresponding rhyme schemes from three different poetry collections, using manual annotation, exploiting already annotated data and utilizing a method based on phonemes to automatically annotate four-line stanzas. During fine-tuning, we only train the decoder stack of ByT5, as the encoder is only used to compute a representation for a given rhyme scheme, while we set the respective poem as desired output. We evaluate our approach by comparing it to a GPT-2 baseline by computing the minimal phonemic Levenshtein distance, perplexity and maximal text similarity on generated samples.

Our results show, while the baseline seems to be semantically more consistent, our proposed system generates quatrains that conform surprisingly well to specified rhyme schemes. We expect that the lexical cohesion of our approach may improve, if adapted to larger character-level Transformer variants.

As our approach demonstrates, it is possible for transformer-based models to learn rhyming constraints through the means of end-to-end training. Future research may investigate the incorporation of additional poetic features to our proposed system, such as emotional expressiveness (Bena and Kalita, 2020), poetic meters or generating poems with more lines.

References

Brendan Bena and Jugal Kalita. 2020. Introducing aspects of creativity in automatic poetry generation. *arXiv preprint arXiv:2002.02511*.

Ping Cai, Xingyuan Chen, Peng Jin, Hongjun Wang, and Tianrui Li. 2021. Distributional discrepancy: a metric for unconditional text generation. *Knowledge-Based Systems*, 217:106850.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

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. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. 2021. Perceiver IO: A general architecture for structured inputs & outputs. *CoRR*, abs/2107.14795.

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. *arXiv preprint arXiv:1807.03491*.

Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

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

Hugo Oliveira. 2009. Automatic generation of poetry: an overview. *Universidade de Coimbra*.

Jongseok Park, Kyubyong & Kim. 2019. g2pe. `https://github.com/Kyubyong/g2p`.

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

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Prakash Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *CoRR*, abs/2106.12672.

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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jörg Wöckener, Thomas Haider, Tristan Miller, Thanh Tung Linh Nguyen, Minh Vu Pham, Jonas Belouadi, Steffen Eger, et al. 2021. End-to-end style-conditioned poetry generation: What does it take to learn from examples alone? In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 57–66.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *arXiv preprint arXiv:2105.13626*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

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.

Appendix

GPT-2	
Coupled (aabb)	
I know it well, by looking on the sea,	**-ea**
By looking on the waves, and by walking at the doors	
In short, I have seen yon old farm once and for all	
Of goodly sons who made good, clean	**-ean**
For it is well that with this one party	**no rhyme**
There should be no access to the Pope's papers.	
The Magistrates' House, whose high duty	
Was emulation of the common pleas	
Dreams are but mirrors for our delight,	**no rhyme**
And when our summons is blown away,	
The outlet is but a heart's pleasure,	
Some sweet treasure is ours to miss,	
Alternate (abab)	
On the one side was a book of shining hew,	**no rhyme**
On the other was a pen of liquid shining.	
And both its eyes were carved of the bech.	
And the thick black leaves that lay at either end	
'The world's own flower, the world's own flower,	**no rhyme**
That, honour'd by no impulsive force,	
Advancing to wider reach,	
Soon seeks, with those 'tis o'er us prevails,	
I saw the tear trembling in the giant's hand,	**-and**
A tear that Dewey\'s heart had consecrated	
To Him, the glory of the earth's hushed band,	**-and**
His cry came False. He is gone!	
Enclosed (abba)	
Of the world's sorrows, and the many woes	**no rhyme**
Which the spirit of the Ages bedew'd,	
And the rose despondency, and the tear,	
In the cure of their cruel lives,	
They loved thee much, and brought thee home	**no rhyme**
The many flowers that lie within the hall,	
But now thine eyes have seen not for years,	
And thou art come to remember whence came	
The want of happiness and health is flaring	**-ing**
In all the country far and wide,	
And country bumpkin country is singing,	**-ing**
Pushing off in even chattering measure	

(a) Three random samples for each rhyme scheme generated by our GPT-2 baseline

ByT5	
Coupled (aabb)	
Where there reposes from that old man's schemes,	
And waters, garlands, sleeps and grows and greets	
But see!the waters range about its ends	**-ends**
And dews beneath its misty shadow ends.	**-ends**
With a hot thread of jet,	
The waves conceal'd his hair,	
Whilst Betty Maud still more fair	**-air**
Was warm and sunk to pair.	**-air**
Dances for ever. Oh, you heart to me	**-e**
When you play at your pipe with your beads and sea,	**-ea**
Wi' the pawrest fate you drink to sob,	
Wi' usquabae, weel, weel, waraway on,	
Alternate (abab)	
Near Westminster's golden cave	**-ave**
He turned to fold,	
To mate his house was there strave,	**-ave**
From buried field to burned his cave.	
She tilled the book, and saw the Gods had said,	**-aid**
I thought Anne of the Makers had made	**-ade**
To whom I once as one slipped off in my head,	**-ead**
Yet her heart then smiled, and, in the maid	**-ade**
But when you will say that your hand	**-and**
Has missed a pleasant lover,	**-lover**
When you loved me well and sadly stand,	**-and**
And filled you with your lover	**-lover**
Enclosed (abba)	
'Three times here on Christmas Eve takes your way,	
Or comfort yourself to light.	**-ight**
Each field of sun and sea, from might,	**-ight**
With cottagers and lovers, stands the sad tale while we call	
our gaze.	
Beyond Youth's fall of life, when earthly joys are hoar,	**-oar**
And hopes that never cease,	
Though transient spheres the ocean roar,	
Be partial with the winds, that blow no more,	**-ore**
Full well I know, the Neville wrote his way	**-ay**
To play a damsel who made a second priest	**-iest**
Set it together, put out in a measure eased	**-eased**
Of death, beloved by the levels of a day.	**-ay**

(b) Three random samples for each rhyme scheme generated by our ByT5 model