Elevating Poetic Expression: Crafting Formal and Sentiment Constrained Poetry with PPL-MCTS

Mateo Santiago Rueda Molano
Aalto University
mateo.ruedamolano@aalto.fi

Abstract

Generating formal poetry with natural language processing (NLP) language models (LM) remains challenging due to their inability to control stringent metrics and rhyme schemes directly. Existing methods either rely on training models on specific datasets with matching meter and rhyming schemes, making data acquisition difficult for some poetic forms [5] [9] [1] [6], or employ non-poetic corpora with additional rules or modules [7] [14], risking the omission of essential poetic elements. Moreover, limited research exists on generating poetry with specific emotions or sentiments, with current methods relying on multiple fine-tuned language models for each constraint [15] [8], presenting similar challenges as formal poetry generation. This paper introduces a novel approach using Plug-and-Play Monte Carlo Tree Search (PPL-MCTS), a discriminator-based method, for formal and sentiment-constrained poetry generation [3]. PPL-MCTS offers a streamlined solution using a single language model and a discriminator for controlled text generation. In our experiments, PPL-MCTS demonstrated a significant improvement of 34% in well-formed consecutive rhyming lines compared to the baseline LM (GPT2-Poet) without controlled generation. However, achieving consecutive rhymes through LM fine-tuning remains more effective [9]. Regarding sentiment-constrained poetry, PPL-MCTS excelled in producing poems with favorable results across all emotion categories, achieving accuracy values exceeding 0.8 for each sentiment. Notably, incorporating PPL-MCTS with the discriminator into the LM did not compromise overall text quality; in some sentiment categories, it even enhanced the generated poetry. Our findings highlight the effectiveness of PPL-MCTS as a promising and efficient method for both formal and sentiment-constrained poetry generation, enabling new possibilities for creative text generation and personalized poetry composition for specific emotions or perspectives.

1. Introduction

Advanced language models like GPT-3 [2], T5 [10], and LaMDA [13] have revolutionized Natural Language Generation (NLG), excelling in tasks such as text summarization, machine translation, and question answering. However, the generation of formal poetry remains a challenge for these models, primarily due to the stringent rules governing metrics and rhyme schemes that these LMs cannot directly control. While training or fine-tuning these models on natural poems is possible, it does not guarantee their ability to generate a diverse range of rhyme and meter patterns. A study conducted by [9] revealed that fitting a GPT-2 model with natural English poems resulted in a mere 11% of consecutive rhyming lines.

The existing body of literature has proposed several solutions to address this challenge, but each approach comes with its own set of issues. Firstly, some methods necessitate training the model on specific datasets containing the same meter and rhyming scheme as the poems to be generated [5] [9] [1] [6]. This approach poses a problem, as it may be challenging to acquire sufficient data for certain poetic forms. Secondly, other studies utilize models trained on extensive non-poetic corpora and employ rules or modules to guarantee the rhyme scheme and meter in the generated lines [7] [14]. Nevertheless, employing LMs trained with non-poetic texts runs the risk of omitting words and poetic forms commonly utilized in poetic language.

Furthermore, limited research exists on generating poetry constrained to specific emotions or perspectives. The few studies in this domain present the same aforementioned problem encountered in structured poetry. That is, they employ multiple language models fine-tuned to each constraint, either for poem generation [15] or for editing the existing poem to meet the desired constraint [8].

Unsupervised constrained text generation seeks to produce text while adhering to predefined constraints without relying on supervised data [4]. This approach holds significant potential for poetry generation, as it allows the creation of poems that conform to specific stylistic, thematic, or for-

mal requirements. An effective method for generating constrained poetry is through Plug-and-Play Monte Carlo Tree Search (PPL-MCTS) [3]. Given the challenges mentioned earlier, this approach offers a streamlined solution, employing a single language model (LM) and a discriminator for controlled text generation. In this paper, we aim to assess the effectiveness of PPL-MCTS method for independently generating formal and constrained poetry. Our objective is to compare the advantages and disadvantages of this method against a basic language model without controlled generation, specifically evaluating their performance in generating both types of poetry.

2. Related Work

The dominance of NLP Language Models (LM) in poetry generation has been a prominent area of research. In this section we will present works of literature that use LMs in the generation of formal or constrained poetry.

2.1. Formal Poetry Generation

Various approaches in formal poetry generation involve fine-tuning LMs on datasets of specific poetic forms with desired meter and rhyme patterns. For instance, [9] conducted experiments by fine-tuning a GPT2 English model using a dataset of 6Mb synthetic quatrains with consecutive rhymes (AABB). The results indicated that the model successfully generated approximately 60% of the consecutive rhyming lines. However, generating alternative rhymes (ABAB) proved to be more challenging due to long-range dependencies, with the model achieving a 45% success rate in generating such rhymes. In another study, [1] utilized ByGPT5, a character-level decoder-only LM, pre-trained on a large corpus of labeled English and German quatrains. The fine-tuned model exhibited superior performance in human likeness, rhyme, and meter compared to other models such as mT5, ByT5, and GPT-2. Additionally, it outperformed ChatGPT in terms of rhyme score. Furthermore, [5] employed a Diffusion model trained on datasets consisting of Sonnets and SongCi. They introduced a novel metrical controller to manipulate and evaluate format and rhythm metrics. By incorporating a denoising process in the model, the gradual enhancement of semantics and flexible controller integration were achieved. In a joint architecture proposed by [6], a rhyme model, pentameter model, and an LM pre-trained on sonnets were combined to facilitate further generation. Interestingly, the study revealed that a vanilla LM implicitly captures the metric when trained on this poetic form. While fine-tuning models on datasets representing the desired poetic forms has been a common approach, it presents a challenge due to limited data availability for specific poems. Consequently, this approach employs fine-tuned LMs on natural poems without predefined structures. By utilizing PPL-MCTS (Predictive Procedural Language with Monte Carlo Tree Search), the decoding process of the LM can be guided towards a specific structure, enabling the generation of poetry with desired characteristics.

In contrast to the approaches mentioned earlier, some alternative methods do not require training or fine-tuning models, specifically on poetic texts. [7] introduces an unsupervised approach for generating poems that adhere to any desired metric and rhyme scheme. This method involves splitting a non-poetic corpus into sentences, each preceded by control-coded sentences describing the length and end rhyme of the subsequent sentence. Subsequently, a Transformer model is trained on this augmented corpus. During inference, control codes specifying the desired meter and rhyme pattern are created, configuring the LM to generate formal verse poetry. Similarly, [14] presents a framework for generating sonnets that does not necessitate training on preexisting poems. Instead, they propose a hierarchical framework that plans the poem's structure before decoding. The framework comprises a content planning module, a rhyming module, and a polishing module. Additionally, the authors developed a constrained decoding algorithm to impose the constraints of meter and rhyme on the generated sonnets. It is important to note that training models solely on non-poetic texts may lead to the generated outputs lacking poetic language and aesthetic resources. To address this concern, we will employ an LM that has been trained on natural poems, with the expectation that it will incorporate the desired aesthetic elements into the generated poetry. This approach aims to enhance the quality of the poetic language in the generated texts.

2.2. Constrained Poetry Generation

The area of controlled poetry generation focused on specific themes, emotions, or sentiments remains relatively unexplored. Recently proposed methods have sought to combine both formal and controlled approaches. In one such approach, [15] suggests verse-based line suggestions to assist users in composing poems. These suggestions are categorized into various styles of American classic poets. Multiple transformer models are fine-tuned to generate these verses offline using datasets comprising poetic forms associated with each poet. A dual-encoder model also recommends potential verses based on the previously provided prompt. [8] combines LMs for generating constrained poems. A language model is employed to generate the initial draft of the poem. Furthermore, classification and generation models (character-level) are fine-tuned on datasets of specific topics or emotions. These models help identify and modify words that do not align with the desired theme or emotion. In contrast, instead of fine-tuning LMs individually for each constraint, we propose utilizing PPL-MCTS to guide the LM's decoding process towards specific sentiments. This approach only requires a discriminator, often a

classifier, which assigns a score to each sentiment. Training this classifier is simpler and more efficient than fine-tuning an LM, mainly when dealing with LMs with many parameters. By employing PPL-MCTS and a constraint-aware discriminator, we can effectively steer the generation process of the LM while maintaining the desired constraints. This method offers a more streamlined and resource-efficient alternative to fine-tuning LMs for each constraint.

3. Methods

This section provides a detailed description of the problem and introduces the Plug-and-Play Monte Carlo Tree Search (PPL-MCTS) method [3], which serves as the foundation for the controlled poetry generation. Subsequently, it outlines the Language Model (LM) and the individual discriminators used in the generation process to enforce constraints related to formality and sentiment. The dataset utilized for poetry generation, the baseline model, and the evaluation criteria will also be presented. Some implementation details with reference to sequence length and the selection of formal lines are described in the Annexes section.

3.1. Problem Description

The objective of formal poetry generation is to produce consecutive rhyming lines with a length of 10 syllables (AA) based on a given prompt line. These lines can serve as the building blocks for crafting sonnets and quatrains of the type (AABB). The primary goal of this task is to identify lines that exhibit a specific rhyme and meter while maintaining a high level of quality and consistency with respect to the provided prompt line. Regarding sentiment constrained poetry generation, the focus shifts to generating free verse poems of predefined lengths, characterized by one of four classes of sentiments: positive, negative, unbiased, and mixed (combining both positive and negative elements).

3.2. Plug-and-Play Monte Carlo Tree Search (PPL-MCTS)

The Plug-and-Play Monte Carlo Tree Search (PPL-MCTS) method falls under the discriminator-based category and is employed for controlled text generation. It offers a means of guiding the decoding process of the Language Model (LM) without necessitating model fine-tuning. By leveraging the Monte Carlo Tree Search (MCTS), it enables the generation of short-term decisions (selecting the next token) with a focus on promising long-term outcomes. The search space is represented as a tree structure, where the root node corresponds to the provided prompt, and each node's child consists of the parent node's sequence appended with an extra token.

In the context of constrained text generation, the primary objective is to identify the sequence of tokens $[x_{\hat{n+1}}, ..., x_{\hat{n+t}}]$ (referred to as the path of the tree) that maximizes the probability $p(x_{\hat{n+1}}, ..., x_{\hat{n+t}}|c)$ without resorting to exhaustive exploration of the entire tree in both width and depth, which becomes intractable for lengthy sequences. This probability can be calculated using the definition of marginal likelihood, as follows:

$$X_p = [x_0, ..., x_n]$$
$$X_g = [x_{\hat{n+1}}, ..., x_{\hat{n+t}}]$$
$$p(X_g|c) = p_\theta(X_g|X_p)p_D(c|X_g)$$

where X_p corresponds to the prompt tokens, and X_g the generated tokens. The likelihood $p_\theta(X_g|X_p)$ is determined by the LM, while the score $p_D(c|X_g)$ is provided by the discriminator. The authors devised a 4-step adaptation of the Monte Carlo Tree Search (MCTS) technique for text generation, comprising Selection, Expansion, Simulation, and Backpropagation.

- **Selection:** The recursive selection process begins from the root node, focusing on unexpanded children nodes. The nodes that are selected are the ones that maximize the Polynomial Upper Confidence Trees (PUCT) implementation proposed by [12]. This implementation is mathematically described as follows:

$$PUCT(i) = \frac{s_i}{n_i} + c_{puct}\, p_\theta(x_i|x_{1:t-1})\frac{\sqrt{N_i}}{1+n_i} \quad (1)$$

  where s_i represents the score assigned by $p(x|c)$, n_i is the number of simulations played after node i, N_i is the number of simulations played after the parent, and c_{puct} corresponds to the constant that influences the trade-off between exploitation and exploration.

- **Expansion:** Non-terminal nodes, identified during the selection phase, undergo expansion utilizing the Language Model (LM).

- **Simulation:** One of the children is sampled based on the LM's probabilities, and a pattern such as random walk is followed to traverse the tree until reaching a terminal node.

- **Backpropagation:** The score $p(x|c)$ obtained at the terminal node is propagated up the tree, being added to each of the parent nodes' scores until reaching the root node.

Upon reaching the budgeted number of iterations, the tree is created for that step. Importantly, the token selected at each step of the algorithm is the one most frequently visited among the root's children nodes. Furthermore, it's worth mentioning that this adaptable model can be applied to any Language Model (LM) and discriminator with ease.

3.3. Language model

To create the Language Model (LM) for this text generation approach, a GPT-2 Transformer model underwent fine-tuning on an extensive corpus of English poems. This fine-tuning process followed a self-supervised approach, exclusively relying on the raw texts without any human labeling. Crucially, during training, the model employed a target sequence that matched the input sequence, but with a one-token right shift. Additionally, a masking mechanism was employed to ensure that the prediction of each token i solely took into account the preceding tokens (from 1 to $i-1$). The resulting model, known as GPT2-Poet, has been made publicly available on the *HuggingFace* hub as an open-source resource.

3.4. Discriminator

In the context of generating both formal and sentiment constrained poetry, discriminators play a crucial role. The discriminators used for these specific tasks are detailed below.

3.4.1 Formal Poetry Discriminator

For formal poetry generation, a specialized function was employed, considering both the number of syllables and the rhyming pattern. This function evaluates the candidate line based on two criteria:

- Rhyming Pattern: If the last word of the candidate line rhymes with the last word of the prompt line, a score of 0.75 is assigned, indicating a strong rhyme.
- Number of Syllables: The score for the number of syllables is generated using a normal distribution centered at 10 syllables, with a peak score of 0.25.

Thus, if the candidate line rhymes and contains exactly 10 syllables, the discriminator function will output a score of 1. By utilizing this function, PPL-MCTS aims to select and expand tokens at each step in a way that leads to candidate sequences with the desired metric and rhyming pattern.

3.4.2 Sentiment constrained Poetry Discriminator

To enforce sentiment constraints during poetry generation, we employed a BERT model, which is based on a Bidirectional Encoder Transformer and trained with the MLM (Masked Language Modeling) objective. Initially, this model underwent self-supervised training on a large corpus of English texts. For the specific classification task, the model was extended by adding a SoftMax layer on top of BERT's output to map the final hidden state to a set of sentiment classes.

The subsequent fine-tuning was conducted on the Sentiment Poems dataset, sourced from Project Gutenberg and described in detail in [11]. Through this fine-tuning process, poems were categorized into four distinct classes: positive, negative, unbiased, and mixed, serving as essential constraints for our generation process. Moreover, the normalized logit score for each class was utilized as the score of the discriminator $p(c|x)$, further enhancing the model's capability to enforce sentiment constraints. This classification model is accessible through *HuggingFace* as bert-base-uncased-poems-sentiment.

3.5. Dataset

To acquire prompts we will use the Poem Dataset available in the open hub Kaggle. These poems are categorized by the form (e.g. haiku, sonnet, etc.) or topic (love, nature, joy, peace, etc.). We will select the first line of 100 Italian sonnets as a prompt for the formal and sentiment constrained poetry generation.

3.6. Baseline

To provide a comprehensive comparison, we employ the same LM described in the methodology as the baseline model. This LM serves as the reference for evaluating the effectiveness of our approach, without any constraints imposed on formal poetry or sentiment. The generated number of tokens from the baseline model will be kept consistent with that of PPL-MCTS to ensure a fair and unbiased comparison between the two methods.

3.7. Evaluation

3.7.1 Formal Poetry

To gauge the quality of formal poetry generated, we will calculate the percentage of consecutive rhyming lines that align with the appropriate number of syllables. Additionally, we will evaluate perplexity calculated from the GPT2 medium LM (355M parameters), which serves as an indicative measure of the overall text generation quality.

3.7.2 Sentiment Constrained Poetry

For the sentiment constrained poetry, we utilize accuracy classification as our evaluation metric. This enables us to assess the performance of PPL-MCTS in generating poems aligned with each of the four sentiments: positive, negative, unbiased, and mixed. In addition, we will also evaluate perplexity to assess the quality of the constrained poems generated, providing further insights into the coherence and fluency of the sentiment-aware poetry.

Through this rigorous evaluation process, we aim to gain valuable insights into the efficacy and limitations of both PPL-MCTS and the baseline model in their respective poetry generation tasks.

4. Results & Discussion

4.1. Formal Poetry Generation

Approach	No. of prompts	Generated tokens	% rhyming lines
PPL-MCTS	100	6	28%
		7	**38%**
		8	34%
GPT2-Poet	100	6	2%
		7	4%
		8	4%

Table 1. Rhyming Lines Analysis: PPL-MCTS vs. GPT2-Poet

Approach	No. of prompts	Generated tokens	Perplexity
PPL-MCTS	38	7	**3518,75**
GPT2-Poet			**3574,41**

Table 2. Comparing Perplexity of lines: PPL-MCTS vs. GPT2-Poet

This section investigates the effectiveness of PPL-MCTS approach in achieving well-formed consecutive rhyming lines with the appropriate metric of 10 syllables. To assess the performance of the approach, we analyzed the percentage of consecutive lines generated with the same rhyming pattern (AA) and identified the optimal number of tokens for generating rhyming lines.

Table 1 displays the outcomes of our experiments, presenting the percentages of consecutive lines following the desired rhyme pattern and metric for various token generation lengths. Among the different token configurations tested, the generation of 7 tokens yielded the most favorable results, demonstrating the highest percentage of well-formed consecutive rhyming lines. The controlled generation approach using PPL-MCTS showcased a significant improvement in the percentage of well-formed consecutive rhyming lines, with a notable 34% increase compared to the LM (GPT2-Poet) without any controlled generation. This enhancement in performance indicates the efficacy of incorporating the Monte Carlo Tree Search (MCTS) technique alongside the discriminator in our controlled generation process.

It is noteworthy that the perplexity metric, which measures the model's uncertainty in predicting the next token, exhibited favorable results for the lines generated by PPL-MCTS compared to the baseline model, as shown in Table 2. This observation suggests that the controlled generation approach does not compromise the overall text quality generated by the language model.

Despite the encouraging outcomes, it is essential to consider the limitations of our approach. The achieved percentage of consecutive rhyming lines (AA) remains lower than what could be attained through fine-tuning the language model on quatrains with the rhyming pattern (AABB) as previously explored in the literature [9]. The aforementioned study reported a higher percentage of 60% for consecutive rhyming lines.

Furthermore, our attempts to manually influence the last tokens to force rhyming on the generated lines, as detailed in the Annexes, were not entirely successful. Even with a budget of 2000 iterations, the algorithm proved insufficient to identify the precise tokens required to generate the desired rhyming words. Although increasing the number of iterations could potentially enhance the rhyming percentage, this improvement comes at the cost of a significant slowdown in the algorithm's execution. With the current settings, generating a single line with 7 tokens takes approximately 8 minutes.

4.2. Sentiment Constrained Poetry Generation

Approach	Sentiment	No. of poems	**Accuracy**
PPL-MCTS	Positive	100	0,75
	Negative		1,00
	Impartial		0,82
	Mixed		0,98

Table 3. Performance evaluation of sentiment-constrained poetry generation using PPL-MCTS. The table displays the accuracy of generated poems for different sentiments, including positive, negative, impartial, and mixed.

The performance of sentiment-constrained poetry generation using PPL-MCTS is presented and analyzed in Tables 4 and 5. The results demonstrate the effectiveness of the PPL-MCTS approach in generating poems with specific sentiments while maintaining text quality.

Table 4 showcases the accuracy of the generated poems for different sentiments. It is evident that PPL-MCTS excels in generating poems with sentiment constraints, achieving favorable results across all categories. The accuracy values surpass 0.8 for all sentiments, with particularly remarkable outcomes of 100% for negative sentiment poems and a closely approaching value for mixed sentiment poems. These results illustrate the capability of PPL-MCTS in producing constrained poetry with high fidelity to the intended sentiment. Selected examples of poems generated for each sentiment category can be found in the Annexes section, further highlighting the potential of this method in controlled poetry generation.

Approach	Sentiment	No. of poems	**Perplexity**
Poem Dataset	NA	100	3354.54
GPT2-Poet	NA		3560,41
PPL-MCTS	Positive		4050,67
	Negative		3617,07
	Impartial		3509,46
	Mixed		3557,06

Table 4. Perplexity analysis of sentiment-constrained poetry generation, comparing PPL-MCTS with the GPT2-Poet and the original poem dataset.

In Table 5, we examine the perplexity of the generated poems compared to the original poem dataset and the baseline GPT2-Poet. The perplexity scores of the PPL-MCTS model and the baseline LM without controlled generation are lower than those of the poem dataset. This highlights the limitations of the small GPT2-based model, indicating its inferior text generation quality when compared to human-created poems. However, the PPL-MCTS model outperforms the GPT2-Poet in certain sentiment categories, displaying significantly lower perplexity values for the same number of tokens Specifically, the "Impartial" sentiment category stands out with a perplexity value of 3509.46, indicating an improvement in text quality compared to the baseline. These results align with the findings from formal poetry, suggesting that incorporating PPL-MCTS with the discriminator does not compromise the overall text quality and may even enhance it in specific sentiment categories.

5. Conclusion

The formal poetry generation approach utilizing PPL-MCTS yielded positive results, showcasing improved percentages of consecutive well-formed rhyming lines compared to the GPT2-Poet model. However, the observed percentage falls short of results achieved through fine-tuning the model on specific rhyming patterns. The complexity of finding precise tokens for generating rhyming words highlights the challenges in enhancing rhyming performance without compromising the algorithm's efficiency.

One possible direction is to investigate character-level tokenization that can better capture rhyming patterns. By operating at the character level, the model can gain a more fine-grained understanding of the text, potentially leading to improved recognition and utilization of rhyme.

Moving on to the results obtained from the free-verse sentiment-constrained poem generation using PPL-MCTS, the outcomes are highly promising. The approach demonstrates its effectiveness in generating poems with desired sentiments while maintaining favorable text quality. These findings represent a significant contribution to the field of controlled text generation techniques, as they open new possibilities for generating expressive and sentiment-specific poetry.

However, there remain opportunities for further exploration. Investigating how sentiment constraints interact with aspects of formal poetry generation, such as rhyme and meter, could lead to even more sophisticated and harmonious poetic compositions.

6. Annexes

6.1. Implementation Details

6.1.1 Sentence Generation Length

When employing PPL-MCTS, it is necessary to determine the number of tokens to generate. In the case of formal poetry generation, we conducted experiments using different number of generated tokens. Conversely, for the generation of constrained sentiment poetry, a consistent approach was adopted, generating 60 tokens for all poems. This decision aimed to ensure that the evaluation metrics align with the specified token count.

6.1.2 Selection of formal lines

In our approach, PPL-MCTS generates $n-2$ tokens within $t-2$ steps of the algorithm. We selected lines that both rhyme with the prompt line and meet the desired metric from the candidate series that differ in their last two tokens. This was essential because we observed a significant decrease in the percentage of rhyming lines when allowing the method to select all tokens. To maintain coherence, we chose to allocate 2 "free" tokens, considering that, in most cases, 2 tokens form a complete word. The process is visually depicted in the following figure:

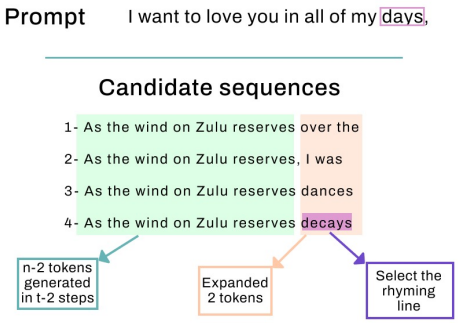

Figure 1. Selection process of PPL-MCTS generated rhyming lines.

6.2. Poetic examples from PPL-MCTS

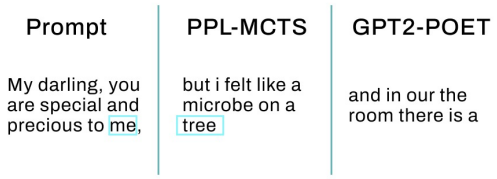

Figure 2. Rhyming Line Generation Example: PPL-MCTS vs. GPT2-Poet. This figure showcases the successful rhyming line generation using the proposed method PPL-MCTS (left), while the baseline method GPT2-Poet (right) failed to produce a rhyming counterpart for the given prompt.

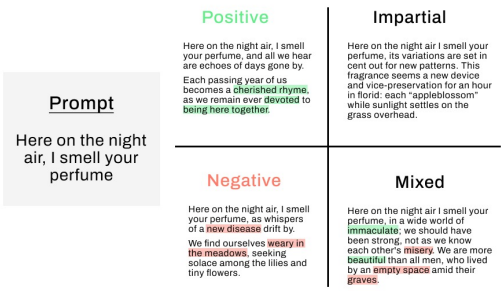

Figure 3. Free-verse poem generated by PPL-MCTS, incorporating various sentiments - negative, positive, impartial, and mixed - within a single poetic prompt. The poem's sentiments are visually enhanced with color-coded words, with positive words highlighted in green and negative words accentuated in orange.

References

[1] J. Belouadi and S. Eger. ByGPT5: End-to-end style-conditioned poetry generation with token-free language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7364–7381, Toronto, Canada, July 2023. Association for Computational Linguistics.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[3] A. Chaffin, V. Claveau, and E. Kijak. Generating texts under constraint through discriminator-guided MCTS. *CoRR*, abs/2109.13582, 2021.

[4] Y. Fu, W. Ou, Z. Yu, and Y. Lin. Effective unsupervised constrained text generation based on perturbed masking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1417–1427, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[5] Z. Hu, C. Liu, Y. Feng, and B. Hooi. Poetrydiffusion: Towards joint semantic and metrical manipulation in poetry generation, 2023.

[6] J. H. Lau, T. Cohn, T. Baldwin, J. Brooke, and A. Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[7] A. Ormazabal, M. Artetxe, M. Agirrezabal, A. Soroa, and E. Agirre. PoeLM: A meter- and rhyme-controllable language model for unsupervised poetry generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3655–3670, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[8] A. Popescu-Belis, À. Atrio, V. Minder, A. Xanthos, G. Luthier, S. Mattei, and A. Rodriguez. Constrained language models for interactive poem generation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3519–3529, Marseille, France, June 2022. European Language Resources Association.

[9] A. Popescu-Belis, À. R. Atrio, B. Bernath, E. Boisson, T. Ferrari, X. Theimer-lienhard, and G. Vernikos. GPoeT: a language model trained for rhyme generation on synthetic data. In *Proceedings of the 7th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 10–20, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[11] E. Sheng and D. C. Uthus. Investigating societal biases in a poetry composition system. *CoRR*, abs/2011.02686, 2020.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, Oct 2017.

[13] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. A. y Arcas, C. Cui,

M. Croak, E. H. Chi, and Q. Le. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022.

[14] Y. Tian and N. Peng. Zero-shot sonnet generation with discourse-level planning and aesthetics features. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3587–3597, Seattle, United States, July 2022. Association for Computational Linguistics.

[15] D. Uthus, M. Voitovich, and R. Mical. Augmenting poetry composition with Verse by Verse. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 18–26, Hybrid: Seattle, Washington + Online, July 2022. Association for Computational Linguistics.