CharPoet: A Chinese Classical Poetry Generation System Based on Token-free LLM

Chengyue Yu[*][†], **Lei Zang**[*][†], **Jiaotuan Wang, Chenyi Zhuang, Jinjie Gu**

Ant Group

{yuchengyue.ycy, zanglei.zl, yunting.wjt,chenyi.zcy, jinjie.gujj}@antgroup.com

Abstract

Automatic Chinese classical poetry generation has attracted much research interest, but achieving effective control over format and content simultaneously remains challenging. Traditional systems usually accept keywords as user inputs, resulting in limited control over content. Large language models (LLMs) improve content control by allowing unrestricted user instructions, but the token-by-token generation process frequently makes format errors. Motivated by this, we propose CharPoet, a Chinese classical poetry generation system based on token-free LLM, which provides effective control over both format and content. Our token-free architecture generates in a character-by-character manner, enabling precise control over the number of characters. Pruned from existing token-based LLMs, CharPoet inherits their pretrained capabilities and can generate poetry following instructions like "Write me a poem for my mother's birthday." CharPoet achieves format accuracy above 0.96, outperforming Jiuge-GPT-2 (0.91) and GPT-4 (0.38). In terms of content quality, CharPoet surpasses traditional systems including Jiuge, and is comparable to other LLMs. Our system is open source and available at https://modelscope.cn/models/CharPoet/CharPoet. A video demonstration of CharPoet is available at https://youtu.be/voZ25qEp3Dc.

1 Introduction

Chinese classical poetry, one of the most valuable heritages of human culture, conveys rich connotations through its concise and exquisite form. Chinese classical poetry can be classified into two primary categories: SHI and CI, both of which have strict format requirements (Hu and Sun, 2020). For example, Wuyanjueju, the simplest form of SHI, requires four lines with each line containing exactly
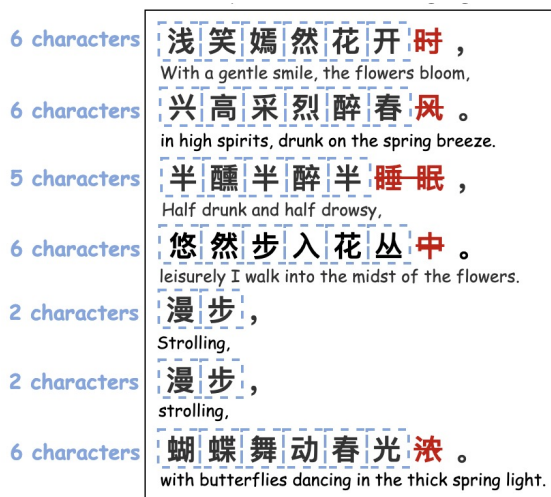

Figure 1: Poem generated by GPT-4. The poem violates the format requirement of *Rumengling* with 6 excess characters.

five Chinese characters. CI is more complex: there are nearly one thousand forms in total, each with different requirements for the number of lines and characters.

Automatic generation of Chinese classical poetry has attracted much research interest. However, achieving effective control over both format and content simultaneously remains a challenge.

Traditional systems in this field usually take keywords as user inputs (Guo et al., 2019; Hu and Sun, 2020; Wang et al., 2016; Yan, 2016; Yi et al., 2017, 2018; Zhang and Lapata, 2014; Zhang et al., 2017). However, it is often insufficient for users to fully describe the theme or emotion they expect with just one or several keywords. This inability to process complex inputs has reduced the diversity and quality of the generated poetry. In contrast, Large Language Models (LLMs) can accept unrestricted user prompts and allow more control over the content. LLMs are capable of generating diversified texts following complex user instructions (OpenAI, 2022, 2023; the Qwen team, 2023). Nevertheless,

---

[*]Equal contribution.
[†]Corresponding authors.

token-based LLMs face challenges in strictly adhering to the expected format of poetry, occasionally producing lines with an excess or insufficient number of characters.

An example of a GPT-4-generated poem is given in Figure 1. In this example, GPT-4 is asked to write a poem in the *Rumengling* form, with the keyword *cheerful*. The generated poem performs well in terms of content, but it clearly violates the format requirements. The redundant characters are marked in red with a strikethrough.

We argue that the problem is partly due to the token-based nature of LLMs. Standard token-based LLM systems split text into word pieces before feeding them into the model. These text pieces are known as *tokens*, and they usually contain more than one character (Sennrich et al., 2016; Schuster and Nakajima, 2012). The system must generate text in a token-by-token manner. Under such a setting, if a model needs to control the number of characters precisely, it must know exactly how many characters are contained in each token. We have conducted a simple test that shows LLMs clearly lack such knowledge. The results are provided in Appendix A.

Motivated by this, we propose CharPoet, a Chinese classical poetry generation system based on a token-free LLM, which achieves effective control over both format and content simultaneously. "Token-free" here means that our model operates only on characters or bytes, in contrast to regular tokens. As shown in Figure 2, our system generates poems in a character-by-character manner. With the token-free architecture, our system can precisely control the number of characters. Instead of being trained from scratch, our token-free LLM is pruned from existing token-based models. We remove long tokens from the tokenizer and the language model head, keeping only character-level and byte-level tokens, and then finetune on a poetry dataset. Through this pruning process, our system inherits capabilities from existing token-based LLMs, and can generate poetry following complex instructions such as "Write me a poem for my mother's birthday."

Without any post-processing, our token-free system achieves a format accuracy of 0.96, outperforming Jiuge-GPT-2 (0.91) and GPT-4 (0.38). In addition, our system performs comparably to existing LLMs in terms of the content quality.
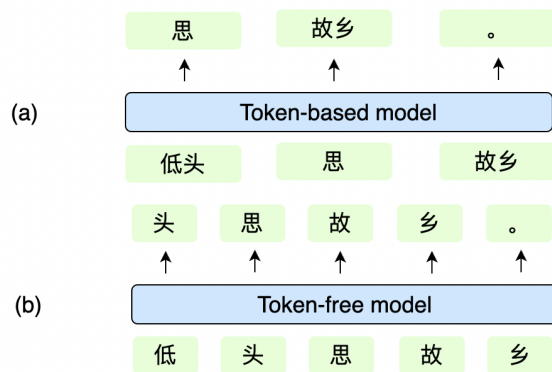

Figure 2: Generation process of a token-based model vs. a token-free model: (a) In a token-based model, the system may output more than one character at a time, resulting in difficulty in exerting precise control over the number of characters. (b) In a token-free model, the system outputs at most one character at a time, making control over the number of characters easier.

2 Related work

Traditional systems in this field (Zhang and Lapata, 2014; Wang et al., 2016; Yan, 2016; Yi et al., 2017, 2018; Guo et al., 2019) have demonstrated that RNNs and LSTMs can generate high-quality poetry. However, these systems usually accept keywords as user inputs, resulting in poor control over content. Moreover, they often have complex architectures or special modules designed to handle the strict format and content constraints inherent in poetry. For example, Yi et al. (2018) imposes a working memory mechanism; Guo et al. (2019) implements a postprocess module to filter poems with unexpected format.

Large Language Models (LLMs) (OpenAI, 2022, 2023; the Qwen team, 2023) have demonstrated the power of the Transformer architecture (Vaswani et al., 2017) when trained with a large corpus. LLMs are capable of generating high-quality and diversified poetry following unrestricted prompt. However, they suffer from problems with format accuracy due to their token-based nature.

More in line with our research, Hu and Sun (2020); Belouadi and Eger (2023) build poetry generation systems based on **token-free language models**. However, those systems are trained from scratch so they do not inherit the great power from pretrained LLMs. They still accept keywords as user inputs and cannot understand complex instructions.
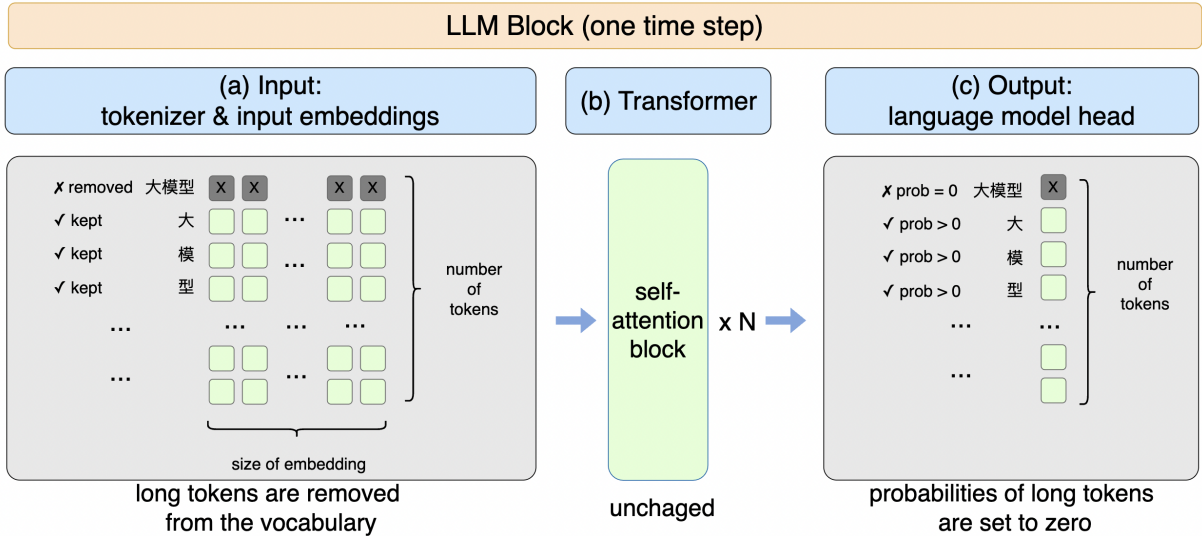
Figure 3: Prune a token-based model into a token-free one. (a) For *Input*, long tokens will be removed from the vocabulary. Text would only be tokenized into character-level or byte-level tokens; The embeddings of long tokens will never be accessed. (b) Transformer structure is left unchanged. (c) For *Output*, the logits of long tokens will be set to a large negative number and the probabilities of long tokens will be zero. The language model head would never produce long tokens.

3 Architecture

3.1 Pruning

The core of our system is a token-free LLM. Instead of being trained from scratch as in previous work (Belouadi and Eger, 2023), our token-free LLM is pruned from an existing token-based LLM to inherit the pretrained knowledge and capabilities. Our token-free model accepts unrestricted prompts as input and returns poems that excel in both format accuracy and content quality.

We have designed a procedure that can prune any typical token-based LLM into a token-free model. A typical LLM, such as Llama (the Llama team, 2023) and Qwen (the Qwen team, 2023), contains three components, the *Input* (the tokenizer and the input embeddings), the *Transformer* (Vaswani et al., 2017) and the *Output* (the language model head). Our pruning procedure modifies the *Input* component and the *Output* component, and leaves the *Transformer* component unchanged. The procedure is described below and illustrated in Figure 3.

(a) Input Pruning. We prune the tokenizer's vocabulary by removing all long tokens, leaving only character-level or byte-level fragments. *Long tokens* refer to two types: tokens with more than one Chinese character and tokens consisting of a single Chinese character combined with non-Chinese characters. Once these tokens are removed, the tokenizer will only produce character-level or byte-level fragments. Subsequently, the input embeddings for these removed tokens will never be accessed or updated.

We retain non-Chinese tokens as they are. This approach ensures that the keywords commonly used in LLM chat settings like "user" and "assistant", remain intact to preserve the standard tokenization of chat templates.

(b) Transformer kept unchanged. The structure of the Transformer is left unchanged, while the parameters will still be updated during finetuning.

(c) Output Pruning. For outputs, we set the probabilities of all long tokens to zero. This is achieved by incorporating an indicator function into the original softmax transformation:

$$\text{Prob}(t_i) = \frac{(1 - \mathbb{1}_L(i)) \exp(logit_i)}{\sum_j (1 - \mathbb{1}_L(j)) \exp(logit_j)}$$

where $logit_i$ denotes the neural network's output value of the ith token prior to the softmax transformation, and the indicator function determines if the ith token is a member of the long token set L.

$$\mathbb{1}_L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}$$

In practice, we implement this by adding a large negative number to the logits of long tokens, instead of modifying the softmax function directly.
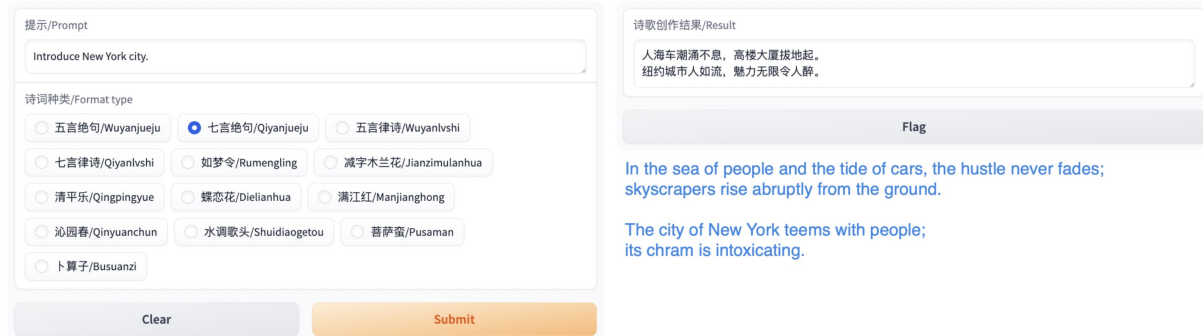
Figure 4: The user interface and generated poetry sample of CharPoet.

With the above procedure, any typical LLM could be pruned to a token-free model. In contrast to typical token-based LLMs, the pruned token-free model outputs text in a character-by-character manner and is expected to perform better on character-sensitive tasks such as poetry generation.

In this paper, we use Qwen-7B-Chat (the Qwen team, 2023) as the base model. An interesting observation is that even without further finetuning, the pruned token-free LLM is already capable of answering simple questions. We provide some examples in Appendix B. Nevertheless, we suggest further finetuning on the target dataset for better performance.

3.2 Training

Training involves two stages: general-purpose training and poetry-field training.

3.2.1 General Purpose Training

We need general-purpose training because our model is directly pruned from an existing token-based LLM and not familiar with natural language presented at character level. Here we use BELLE dataset (Ji et al., 2023), which is a high-quality general-purpose instruction-following dataset.

3.2.2 Poetry-field Training

In the second stage, we train with our in-house poetry dataset. The dataset contains 20000 human written poems, and each poem is created based on an input prompt. The prompts cover a broad range of topics, including specific scenes, emotions, and both concrete and abstract themes.

To improve format accuracy, we provide the model with a masked version of the expected poem as a format hint. In this *masked poem*, all Chinese characters are replaced with a mask sign **[M]** while punctuation and line breaks are kept as is. An example of a *masked poem* in the form *Rumengling* is provided in below.

> [M][M][M][M][M][M]，
> [M][M][M][M][M][M]。
> [M][M][M][M][M]，
> [M][M][M][M][M][M]。
> [M][M]，
> [M][M]，
> [M][M][M][M][M][M]。

The *masked poem* is provided together with the original user prompt. The final prompt-response format is designed as follows, where **[SOP]** denotes *start of piece*, **[EOP]** denotes *end of piece*, **{ORIGINAL USER PROMPT}** denotes the original prompt from the user, **{MASKED POEM}** is the format hint, and **{POEM}** denotes the generated poem.

> [SOP]user
> Fill in all the masks [M].
> {ORIGINAL USER PROMPT}
> Output: {MASKED POEM}
> [EOP]
> [SOP]assistant
> {POEM}
> [EOP]

In this way, the poem-generating task is transformed into a mask-filling task. With the token-free architecture, our model fills in all the masks in a

318

Format Type	#Chars	GPT-4 keyword / instruction	Jiuge-GPT-2 keyword / instruction	Qwen (Finetuned) keyword / instruction	CharPoet (Ours) keyword / instruction
WuyanJueju (SHI)	20	0.49 / 0.73	**1.00** / -	0.94 / **1.00**	0.98 / 0.99
WuyanLvshi (SHI)	40	0.29 / 0.36	**1.00** / -	0.97 / 0.98	0.97 / **0.99**
QiyanJueju (SHI)	28	0.88 / 0.78	**1.00** / -	0.99 / **1.00**	**1.00 / 1.00**
QiyanLvshi (SHI)	56	0.81 / 0.68	**1.00** / -	0.98 / 0.96	0.97 / **0.98**
Rumengling (CI)	33	0.13 / 0.09	0.90 / -	0.95 / 0.97	**1.00 / 0.99**
Jianzimulanhua (CI)	44	0.81 / 0.79	0.96 / -	0.99 / 0.97	**1.00 / 0.99**
Busuanzi (CI)	44	0.28 / 0.24	- / -	0.92 / 0.96	**0.93 / 0.98**
Pusaman (CI)	44	0.26 / 0.17	- / -	0.96 / 0.92	**0.98 / 0.97**
Qingpingyue (CI)	46	0.13 / 0.18	0.96 / -	**0.98** / 0.97	0.95 / **0.99**
Dielianhua (CI)	60	0.21 / 0.12	0.91 / -	0.94 / **0.98**	**0.99** / 0.98
Manjianghong (CI)	93	0.07 / 0.04	0.83 / -	0.88 / 0.90	**0.95 / 0.95**
Shuidiaogetou (CI)	95	0.04 / 0.00	- / -	0.89 / 0.87	**0.95 / 0.91**
Qinyuanchun (CI)	114	0.00 / 0.01	0.55 / -	0.64 / 0.75	**0.82 / 0.86**
Avg (of 10)	53.4	0.382 / 0.378	0.911 / -	0.926 / 0.948	**0.963 / 0.972**

Table 1: Evaluation on Format Accuracy. CharPoet outperforms other systems on average in both the keyword and instruction settings. CharPoet performs significantly better than other systems with longer poems, such as *Manjianghong*, *Shuidiaogetou* and *Qinyuanchun*. For comparability with previous studies, the average accuracy is calculated based on the overlapping 10 types of poetry, rather than all 13 types.

character-by-character manner. The mask-filling design ensures that the model can strictly follow the format constraints of the requested poetry type.

4 Demonstration

The user interface of our poetry generation system is shown in Figure 4. In contrast to previous systems where users need to summarize the theme of the poetry they want in one or several keywords, our system allows users to describe desired content with natural language in the prompt box. After that, the user selects a poetry form and clicks the "Submit" button. A few seconds later, the system returns a poem following the user's instruction.

Our system is fully open source, available at https://modelscope.cn/models/CharPoet/CharPoet. We have included a Jupyter notebook in the project. Using this notebook, anyone can launch the application and try our system. We also provide some example poems in Appendix C.

5 Evaluation

5.1 Test settings

We evaluate performance on two aspects: format accuracy and content quality.

For comparability with previous studies, we assess performance on four types of SHI and six types

of CI, as in Hu and Sun (2020) exactly. To better study the relationship between format accuracy and the length of poetry, we have additionally included three popular types of form in the evaluation set, which are *Busanzi*, *Pusaman* and *Shuidiaogetou*.

We conduct tests under two user input settings: the first is the conventional keyword setting, where the user input consists of a single keyword; the second is the instruction setting, where the user input is a natural language instruction, such as "Write me a poem for my mother's birthday.". In both settings, one specific format is selected as the expected format.

We conduct 100 tests for each type of form and each setting. In the keyword setting, we use a collection of 100 frequently used Chinese idioms sourced from the internet. Chinese idioms convey rich meanings in simple expressions, and are thus more challenging than regular words. In the instruction setting, we ask GPT-4 to generate 100 prompts as user inputs. We have double-checked the GPT-generated prompts; they cover a broad range of topics, including specific scenes, emotions, and both concrete and abstract themes.

We do not use human-written prompts because human researchers could potentially manipulate the prompt set to alter research conclusions. For example, human researchers may remove the prompts
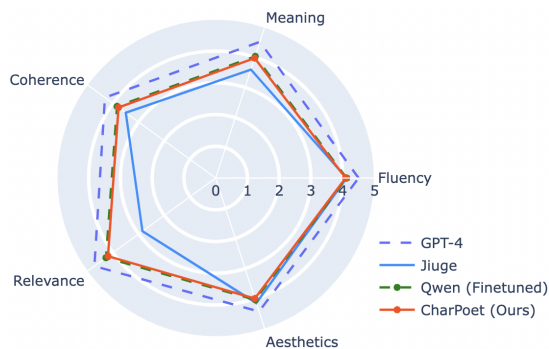
Figure 5: Evaluation on Content Quality by GPT-4 under the Keyword Setting.
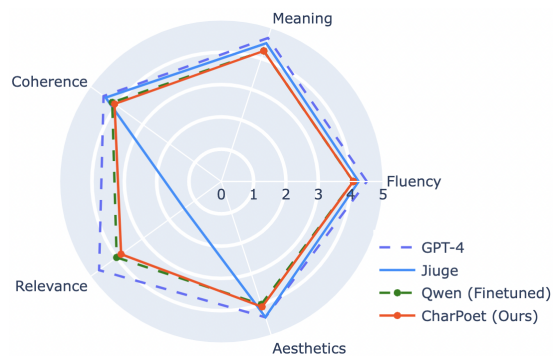

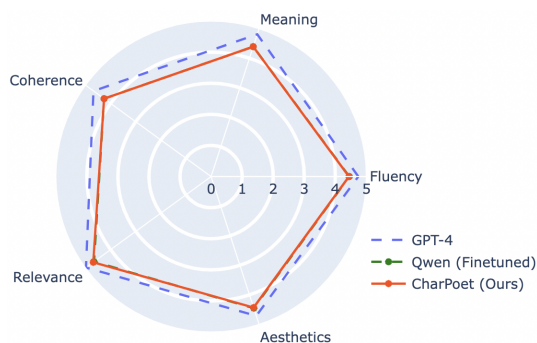Figure 7: Evaluation on Content Quality by Human under the Keyword Setting.

Figure 6: Evaluation on Content Quality by GPT-4 under the Instruction Setting.
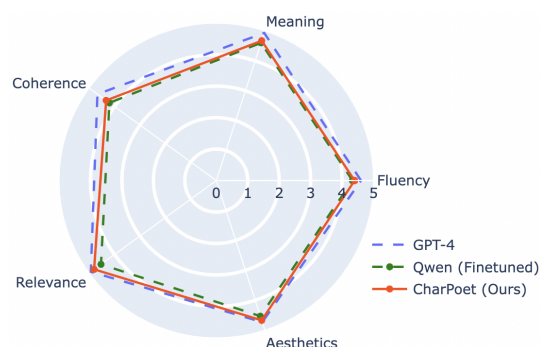

Figure 8: Evaluation on Content Quality by Human under the Instruction Setting.

where the model performs poorly, making the evaluation scores appear better. By directly using GPT-generated prompts without any modifications, we can effectively avoid biases caused by such manipulation issues.

5.2 Models for Comparison

We compare our system CharPoet with two categories of public available systems. One category is general-purpose LLMs, with GPT-4 (OpenAI, 2023) being the top performer. The other category is systems exclusively designed for automatic poetry generation, with Jiuge (Guo et al., 2019) being the most representative.

GPT-4. To exploit GPT-4's potential in format accuracy, we have carefully designed the prompt. We find that GPT performs better if provided with an example poem of the required form. The prompt template is provided in Appendix D.

Jiuge. Jiuge (Guo et al., 2019) is a comprehensive system with a postprocessing module to ensure format accuracy; therefore, when evaluating format accuracy, we compare instead with Jiuge-GPT-2 (Hu and Sun, 2020), the most recent work in the Jiuge series, which is more comparable since it is transformer-based and end-to-end.

Ablation study. To verify the effectiveness of our token-free architecture, we also conducted an ablation study, where we compared our system to its token-based equivalent. The token-based equivalent is identical to CharPoet in every aspect including model size, prompt design and training dataset, except that it is built on the original token-based Qwen-Chat (the Qwen team, 2023) instead of our pruned token-free version. The token-based equivalent is marked as *Qwen (Finetuned)* in corresponding tables and figures.

5.3 Evaluation on Format Accuracy

Format accuracy results are shown in Table 1. A poem is counted as accurate only if the number of characters for every line is correct (perfect match). The figures for Jiuge-GPT-2 are directly collected from the original paper, while the figures for other models are obtained from our testing procedure. The figures for Jiuge-GPT-2 under the instruction setting are not available since Jiuge-GPT-2 does not support user instructions. For comparability, the average accuracy is calculated based on the overlapping 10 types of poetry, rather than all 13 types.

CharPoet performs better in format accuracy than all competing models, achieving an overall accuracy above 0.96 under both settings. Our ablation study comparing CharPoet with its token-based equivalent Qwen (Finetuned) confirms that the token-free architecture is effective, bringing a 3% gain in format accuracy.

Consistent with Hu and Sun (2020), our results show that SHI is simple and all models listed here achieve decent accuracy. As for CI, which is more complex and challenging, our system beats previous systems by a large margin. For example, in terms of *Qinyuanchun*, the longest type of poem in our test set, our system achieves 0.84 accuracy, compared to 0.55 of Jiuge-GPT-2 and nearly zero of GPT-4. Regression analysis also indicates that CharPoet is less sensitive to poem length (See Appendix E for details).

5.4 Evaluation on Content Quality

Following Yi et al. (2018), we evaluate content quality with five criteria; each criterion needs to be scored on a 5-point scale:

Fluency. Does the poem obey the grammatical, structural and phonological rules?

Meaning. Does the poem convey some certain messages?

Coherence. Is the poem as a whole coherent in meaning and theme?

Relevance. Does the poem express user topics well?

Aesthetics. Does the poem have some poetic and artistic beauties?

We first ask GPT-4 to conduct the scoring process. Though we have seen in the previous section that GPT-4 performs poorly in a poetry format, it remains top-notch in terms of content quality, making it a qualified evaluator for content assessment. The GPT-4 results under the two settings are shown in Figure 5 and Figure 6. The performance of CharPoet is basically the same as that of Qwen (Finetuned) and not far from GPT-4, while it significantly surpasses Jiuge, especially in terms of *Relevance*. The gain in content relevance indicates that pretrained LLMs can provide significantly better control over content compared to traditional models.

To ensure the reliability of GPT's evaluation, we have also engaged human evaluators for double-checking. We ask human labelers to score a subset of the evaluation set independently (without referring to GPT-4). The human results under the two settings are shown in Figure 7 and Figure 8. The results are in general consistent with GPT-4. We have also calculated the correlations between human and GPT-4 judgments using Pearson, Spearman, and Kendall-Tau. All correlations are greater than 0.5 with p-values less than 0.01, indicating that GPT-4 is a qualified evaluator in our settings.

6 Conclusion

In this paper, we address the problem of achieving effective control over both format and content in the field of automatic Chinese classical poetry generation. We propose a token-free system Char-Poet, which generates in a character-by-character manner, enabling precise control over the number of characters. Moreover, CharPoet allows for human instructions in natural language, in contrast to traditional models that only accept keywords.

CharPoet achieves format accuracy above 0.96 without any postprocessing, higher than Jiuge-GPT-2 (0.91) and GPT-4 (0.38). Our ablation study comparing CharPoet with its token-based equivalent shows that the token-free architecture brings a 3% gain in format accuracy. In addition, our system's performance in content quality surpasses traditional systems, and is comparable to existing LLMs.

7 Limitations

Rhyme. In this paper, we propose the token-free method to enhance format accuracy. As a side effect, the token-free method may also enhance *rhyme*, which is also a highly character-sensitive task. Rhyme is an important aspect in Chinese classical poetry and deserves further study. Though rhyme is somehow covered in our evaluation process as part of the *phonological rule* in the fluency criterion, it deserves direct research with specially designed criteria and detailed indicators, and this is left to future work.

Other character-sensitive tasks & general abilities. We have proposed a simple method to convert a pretrained token-based language model to a token-free one. It may be interesting to further investigate how the converted model performs in other character-sensitive tasks, such as named entity recognition and spelling correction. It may also be interesting to investigate how much general knowledge and abilities are retained during the conversion. These topics are also left to future work.

References

Jonas Belouadi and Steffen Eger. 2023. Bygpt5: End-to-end style-conditioned poetry generation with token-free language models. In *61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Zhipeng Guo, Xiaoyuan Yi, Maosong Sun, Wenhao Li, Cheng Yang, Jiannan Liang, Huimin Chen, Yuhui Zhang, and Ruoyu Li. 2019. Jiuge: A human-machine collaborative chinese classical poetry generation system. In *Proceedings of the 57th annual meeting of the association for computational linguistics: system demonstrations*, pages 25–30.

Jinyi Hu and Maosong Sun. 2020. Generating major types of chinese classical poetry in a uniformed framework. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4658–4663.

Itay Itzhak and Omer Levy. 2022. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5061–5068.

Yunjie Ji, Yan Gong, Yong Deng, Yiping Peng, Qiang Niu, Baochang Ma, and Xiangang Li. 2023. Towards better instruction following language models for chinese: Investigating the impact of training data and evaluation.

Ayush Kaushal and Kyle Mahowald. 2022. What do tokens know about their characters and how do they know it? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2487–2507.

OpenAI. 2022. Introducing ChatGPT.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725. Association for Computational Linguistics (ACL).

the Llama team. 2023. Llama 2: Open foundation and fine-tuned chat models.

the Qwen team. 2023. Qwen technical report.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Daisy Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1051–1060.

Rui Yan. 2016. i, poet: automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2238–2244.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2017. Generating chinese classical poems with rnn encoder-decoder. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data: 16th China National Conference, CCL 2017, and 5th International Symposium, NLP-NABD 2017, Nanjing, China, October 13-15, 2017, Proceedings 16*, pages 211–223. Springer.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Zonghan Yang. 2018. Chinese poetry generation with a working memory model. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4553–4559.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1364–1373.

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.

A Probing into LLM's knowledge in token-character relationship

For a token-based LLM, if it needs to control the number of characters precisely, it must know exactly how many characters are contained in each token. We have conducted a simple test, which shows that LLMs clearly lack such knowledge.

A.1 Method

Following Itzhak and Levy (2022), we use a probing procedure called "spelling bee" to investigate how much a LLM knows about the token-character relationship of its vocabulary. Specifically, we probe whether the model has the knowledge that the token "大模型" contains three characters "大", "模" and "型".

The models we investigate here are the Qwen-series (the Qwen team, 2023), including Qwen-1.7B-chat, Qwen-7B-chat and Qwen-14B-chat. The Qwen series is one of the earliest open-source LLMs with a strong ability in Chinese and is influential in the Chinese community. In the context of large language models, the probing procedure could be formulated as an instruction following task, designed as follows.

Prompt
List all the characters in the following token: <|extra_1|>大模型

Response
大<|extra_1|>模<|extra_1|>型

Here the special symbol <|extra_1|> is used to ensure that both the long token in the prompt and the single characters in the response are tokenized as they are. We randomly selected 1000 tokens from the vocabulary to serve as a test set, and the remaining tokens are used as training examples.

Our procedure is not exactly the same as previous work (Kaushal and Mahowald, 2022; Itzhak and Levy, 2022). The main differences are

1. Our experiment probes all language model parameters, while previous work (Kaushal and Mahowald, 2022; Itzhak and Levy, 2022) probes only the vocabulary embedding, which ignores the knowledge contained in later layers, and thus would underestimate the real knowledge level.

2. In the context of large language models, we are able to conduct probing experiments relatively easily with the model itself through supervised finetuning (Ouyang et al., 2022), while previous work probes with a separate model, which may be difficult to train and would also underestimate the real knowledge level.

To sum up, we believe that our procedure can better estimate how LLM knows about the token-character relationship relationship.

A.2 Results

The results of the spelling bee probing procedure are summarized with the *overall failure rate*, which is defined as the number of times that the LLM fails to correctly output the character sequence of the required token, divided by the size of the test set (which is 1000 in our experiment). We also pay attention to another version of the failure rate, where we count an output as a failure only if the number of characters in the output does not equal the length of the original token. This metric is more relevant since it directly influences the control of character numbers in poem generation.

Model Size	Token-based overall / #chars	Token-free overall / #chars
1.8B	15.1% / 2.5%	0.7% / 0.2%
7B	9.9% / 1.7%	0.5% / 0.1%
14B	4.1% / 0.9%	0.4% / 0.0%

Table 2: Failure rate of the spelling bee probing procedure. The token-free model has reduced the failure rate by more than 90% compared to the token-based model.

We have several observations:

1. Failure rates of token-based models are non-trivial. Failure rate near 1% seems small, but the influence is significant since mistakes would accumulate as the poem gets long.

2. The failure rate of token-based models seems not to vanish totally as model size increases.

3. The token-free model has reduced the failure rate by more than 90%.

B Token-free LLM without further finetuning

Without further finetuning the pruned token-free LLM is already capable of answering simple questions. Examples are shown below.

User:
简要介绍一下日心说

Agent:
日心说，也称日心地动说，是16th century末期由哥白尼提出的天体学说，它将日、月、水星、金星、木星、土星、天王星、海王星和冥王星等9大天体，及小天体如彗星、小星体等，都绕着以日为圆心的椭圆轨转。

User:
Briefly introduce the heliocentric theory.

Agent:
The heliocentric theory is a model of the solar system that places the Sun at the center, with the planets orbiting around it in椭ical paths. This theory was proposed by Nicolaus Copernicus in the 16th century, but it was not widely accepted until the work of later astronomers such as Johannes Kepler and Galileo Galilei. The heliocentric model is now the most widely accepted model of the solar system, and it is supported by a vast amount of observational evidence.

We can see that the main meaning is correct, although there are some typos and a mix of Chinese and English.

C Example Poems Generated by CharPoet

In the first example, our system is asked to write a poem with the keyword *cheerful* and the required form is *Rumengling*. This is exactly how we test GPT-4 in the Introduction section. We can see that poem generated by our system precisely match the required format of *Rumengling*, in contrast to GPT-4.

笑口频开深院，更说秋风天气。
With a frequent smile in the deep courtyard, I praise the autumn weather.
心事向人知，却好兴高采烈。
I share thoughts with others, in high spirits and delight.
休觅，休觅，酒到不知醒地。
Seek no more, seek no more. Drink till you do not know where you are.

More importantly, CharPoet allows for unrestricted instructions. In the following example, the prompt is "Write me a poem for my mother's birthday." and the required form is *Rumengling*. The output follows the expected format and content.

生日恰逢今日，母爱万金难拟。
Today is your birthday. A mother's love is priceless.
恩重更情浓，岁岁同歌同醉。
Your kindness is profound. We sing and celebrate together, in each passing year.
同醉，同醉，寿星高上天际。
Drink together, drink together. You are the star today.

D Test the Performance of GPT-4 on Format Accuracy

To exploit GPT-4's potential in format accuracy, we have carefully designed the prompt. We find that GPT performs better if we provide it with an example poem of the required form. Our prompt is designed as follows.

> **Prompt**
>
> 请写一首如梦令，主题或要求
> 为"兴高采烈"。请严格按照如梦令
> 对每一句话的字数要求，下面给
> 出一个例子:
> 　常记溪亭日暮，沉醉不知归路。
> 　兴尽晚回舟，误入藕花深处。
> 　争渡，争渡，惊起一滩鸥鹭。

> **Prompt(translated into English)**
>
> Please write a poem in the form "Rumengling". The theme or instruction is "cheerful". Please strictly follow the number of character requirements for each line. Here is an example:
>
> I often recall the sun setting
> at the riverside pavilion,
> lost in the intoxication and
> unaware of the way back.
> Later on when my
> excitement wanes, I return
> on the boat, only to find
> myself unwittingly entering
> a lotus pond.
> Struggling to cross,
> struggling to cross, with
> seagulls and herons startled
> by me and flew away.

E Relationship between Format Accuracy and Poem Length.

We performed a regression analysis to investigate how format accuracy changes with poem length. Results show that in general the format accuracy decreases as the poem length increases. Results also show that CharPoet is less sensitive to poem length compared to competing models.
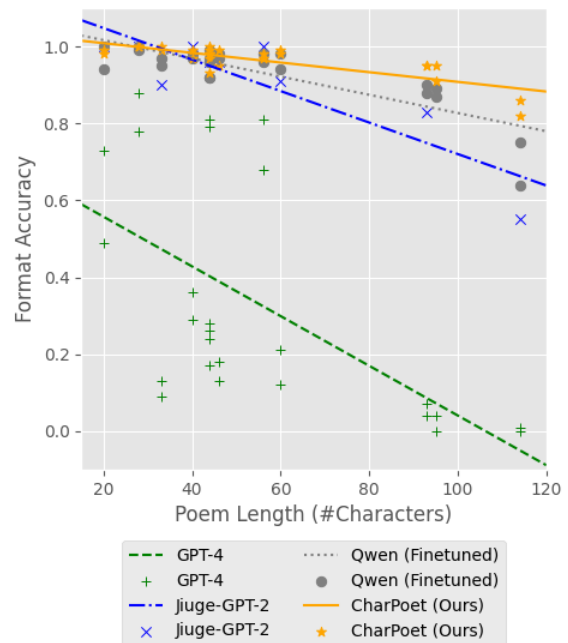

Figure 9: Relationship between Format Accuracy and Poem Length. Regression analysis indicates that the format accuracy of CharPoet is less sensitive to increase in the poem length.