Refining Metrical Constraints in LLM-Generated Poetry with Feedback

Manex Agirrezabal
Centre for Language Technology (CST)
University of Copenhagen, Denmark
manex.aguirrezabal@hum.ku.dk

Hugo Gonçalo Oliveira
University of Coimbra, CISUC/LASI
Centre for Informatics and Systems of the University of Coimbra
Department of Informatics Engineering, Portugal
hroliv@dei.uc.pt

Abstract

Among many tasks, general purpose pretrained Large Language Models (LLMs) can be prompted for generating poetry in a zero-shot scenario, with relative success, even if, in many occasions, they fail to meet all the specified constraints. We prompt LLMs for producing poetry with a specific number of lines and syllables, and evaluate the result with another LLM or with a rule-based tool for poetry analysis. If specified constraints are not met, LLMs are prompted again with feedback and a request for generating a new poem that satisfies the target constraints. This can go through several iterations. With this setup, we not only analyse the ability of LLMs to produce metrical poetry, but we test to what extent they can: (i) improve their generations with feedback; and (ii) analyse metrical constraints. We conclude that feedback is effective in improving the metrical constrains, but this can be done at the cost of less rhyme. We also observe that feedback by the rule-based system is preferable than feedback by an LLM.

Introduction

Due to their flexibility and ease of use, general purpose instruction-tuned Large Language Models (LLMs) are being used in a broad range of text-based tasks, including the generation of linguistically-creative artefacts. For many of such artefacts, the first impression is that these models excel at their task. Specifically, LLMs can generate poetry from natural language instructions that may cover a range of features, including content and form. These will be reflected in the produced text, often further embellished with typical poetic features like metaphor and rhyme. Figure 1 serves as an example of a single prompt that produces a poem.

Despite this success, it has been noted that LLMs still have limitations regarding diversity in the generation of several poems (Zhang and Eger 2024; Chen et al. 2024), and that they do not always meet all target formal requirements, such as the number of characters per line (Yu et al. 2024), or number of lines and syllables (Agirrezabal and Gonçalo Oliveira 2024). At the same time, current instruction-tuned LLMs can keep a conversation, and will consider past context in their responses. So, their outputs can be refined through feedback (Pan et al. 2024), if necessary, for a number of interactions.

```
> user:  Please create a poem.  The poem should
contain 8 lines about nature.  Return only the poem,
nothing else.

> assistant:  Sunlight dapples through leaves of
green,
A gentle breeze whispers secrets unseen.
Flowers bloom in vibrant array,
As birdsong fills the air all day.
A babbling brook, a crystal stream,
Reflects the sky's ethereal gleam.
Nature's canvas, vast and wide,
A masterpiece where wonders reside.

> user:  Thank you!

> assistant:  You're welcome!  I'm glad you liked the
poem.  :-)
```

Figure 1: Example interaction where the LLM Gemma2 is prompted to generate a poem with 8 lines about *nature*. The value for temperature was 0.7.

In this work, we explore automatic feedback for guiding the generation of metrical poetry with LLMs. For generation, we test two recent LLMs, Llama3.1-8B (Grattafiori et al. 2024) and Gemma 2-9B (Gemma Team et al. 2024), which are prompted for generating poetry in a set of topics, with variable constraints regarding the number of lines and syllables per line. Generated poems are evaluated, either by the same models, in a different session, or by an external rule-based tool for poetry analysis, Erato (Agirrezabal, Gonçalo Oliveira, and Ormazabal 2023). If target constraints are not met, the generators are prompted with feedback together with a request for generating a new poem.

We observed that, regardless of the source, providing feedback results in poems that follow target metrical constraints more closely. Moreover, the feedback based on the rule-based tool is more effective than the feedback by the LLMs. However, we also observed that updating the poems to satisfy the syllable count constraints is sometimes done by adding words to the end of lines, breaking a previously compliant rhyme. On the other hand, the diversity of poems generated for the same constraints is not affected, suggesting that LLMs do not update poems using fixed lexical items or expressions.

Our contributions are summarised as follows:

- We explore feedback for improving metrical constraints in LLM-generated poetry;
- We show that, regardless of the evaluation by another LLM or by a rule-based tool for poetry analysis, feedback is always effective;
- We analyse the impact that meeting metrical constraints has on other features like rhyme and diversity;
- We create a dataset of over 3,200 sessions and their LLM-generated poems, which may be used for further research on this domain.

The paper is structured as follows. First, we provide some background on poetry generation, with a focus on LLM-generated poetry, and approaches for LLM feedback. After that, we present the adopted methodology for poetry generation with feedback. This is followed by a detailed description of the experimentation setup, covering the tested models, parameters and prompts used. Before concluding, we present and discuss the main results of our experimentation.

Background

From rule and knowledge-based to evolutionary algorithms, a broad range of techniques has been explored for generating poetry computationally (Gonçalo Oliveira 2017; Lamb, Brown, and Clarke 2017). Yet, as it happens to other text-related tasks, nowadays, transformer-based models, mostly LLMs, are commonly used for this purpose (Elzohbi and Zhao 2023). These can be fine-tuned for generating poetry in target forms or in the style of a specific author (Sawicki et al. 2023), but, with the size of LLMs increasing, they are also getting better at performing tasks for which they were not specifically trained for. And, since prompting an LLM is much more accessible than fine-tuning it, the latter is becoming the most popular approach.

However, the simplicity of LLM-based poetry generation does not come at no cost, as authors have noted several limitations, including the lack of diversity (Zhang and Eger 2024; Chen et al. 2024), or a difficulty to meet formal constraints (Yu et al. 2024; Agirrezabal and Gonçalo Oliveira 2024). Some (Walsh, Preus, and Gronski 2024) further discuss that a model like ChatGPT has its own poetic style, more constrained, with higher trends to use specific words, more rhymes and predefined forms. In theory, increasing the temperature would lead to more diverse outputs. However, as shown for narrative generation (Peeperkorn et al. 2024), temperature is not more than weakly correlated with novelty, and it is moderately correlated with incoherence.

Indeed, LLMs are known to be inconsistent (Jang, Kwon, and Lukasiewicz 2021). A possible way of minimising this, as well as other reliability issues, is to interact with the LLM while providing feedback. Several methods have been proposed for correcting the output of LLMs through feedback (Pan et al. 2024), which may be provided by the LLM itself, also known as self-feedback (Madaan et al. 2024); or externally, by a human, another LLM (Yang et al. 2022), specific tools (Zhang et al. 2023; Jiang, Wang, and Wang 2023), structured knowledge (Peng, Zhang, and Shang 2024), or a combination of different sources (Nathani et al. 2023).

Feedback can be provided during or after generation (post-hoc), but the latter is more flexible, as it does not require additional training, and allows feedback to be provided in natural language, through simple prompting, if necessary, across several iterations. Among others, post-hoc feedback provided by an external tool was applied to code generation (Zhang et al. 2023; Jiang, Wang, and Wang 2023) and question answering (Nathani et al. 2023; Chern et al. 2023). In the domain of creativity, post-hoc feedback was applied to story generation (Yang et al. 2022), by a model trained for reranking produced passages.

Yet, to our knowledge, feedback mechanisms have not been applied to poetry generation with an LLM. In this task, feedback could be provided by a human or by computational tools for poetry scansion and analysis in general (De Sisto et al. 2024). In this domain, we highlight two frameworks developed in recent years, namely CARMEN (Possi et al. 2023), focused on a single metric for evaluating poems considering features like meter and rhyme; and Erato (Agirrezabal, Gonçalo Oliveira, and Ormazabal 2023), which provides a set of tools for analysing poetic features in detail, also covering novelty and fluency. Moreover, LLMs are being studied in the detection of poetry form (Walsh, Antoniak, and Preus 2024). They have shown to successfully identify both common and uncommon fixed poetic forms, but performance varies significantly when it comes to unfixed forms.

Methodology

In order to refine the metrical constraints of poems generated by prompting an LLM, we adopt iterative post-hoc feedback. Figure 2 summarises the proposed approach.

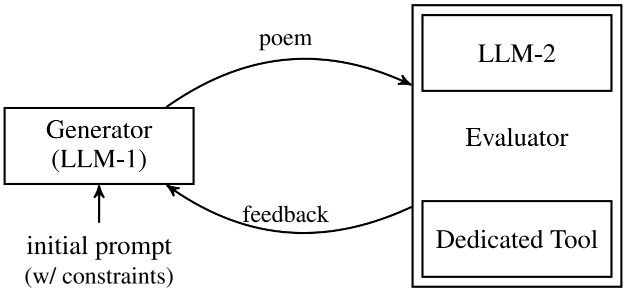

Figure 2: LLM-based poetry generation with Iterative Post-Hoc Feedback. The Generator is prompted for producing a poem with specific constraints; the poem is analysed by the Evaluator; if constraints are not met, feedback is provided to the Generator, with a request for producing another poem.

Briefly, an LLM is prompted for generating a poem with specific constraints. This poem is then evaluated, either by another LLM or by a tool for poetry analysis. If any of the target constraints is not met, feedback is provided to the generator LLM, as a new natural language prompt. This is performed until the constraints are met or a predefined number of iterations is reached.

If feedback is effective, the poem produced in the final iteration will be better-aligned with the target constraints. However, satisfying such constraints might have an impact in other poetry-relevant features. In this paper, we focus on two metrical constraints, number of lines and number of syllables, but also analyse the impact that meeting such constraints has on the produced rhymes and on diversity.

Experimentation Setup

This section describes how the proposed methodology was implemented, thus enabling experimentation towards useful conclusions. It describes the models used for generation and evaluation, the metrical constraints and topics considered, as well as the prompts used for interacting with the models.

Models

We experimented with two different open LLMs of similar size, Llama3.1-8B (Grattafiori et al. 2024) and Gemma2-9B (Gemma Team et al. 2024), both as generators and as evaluators. They were accessed in a quantised version, through the Ollama tool[1]. Additionally, we used Erato as an evaluator, thus resulting in six (2×3) different generator–evaluator combinations.

Target Constraints

The selected LLMs were prompted to produce poems with a diverse number of lines, namely 4, 7, 8 and 14 (four different possibilities). Experimentation with syllable counts was performed only for the 4-line poems. For these poems, the values tested for the number of syllables in each line were 8, 10 and 13 (three different possibilities).

Target Topics

Since poems are typically written with a goal or a topic in mind, poems were generated for a target topic. The tested topics were selected before experimentation, with the help of Llama3.1, using the following prompts:

```
Give me three single word topics that are
very common in poetry
```

```
Give me frequent words that are not very
common in poetry
```

This resulted in the following six topics: *love*, *death*, *nature* (from the first prompt), *router*, *briefcase*, *garbage* (from the second prompt). For each generator–evaluator combination, the generator LLMs were prompted for creating poems in each of the six topics. Additionally, we tested the generation without a target topic, which resulted in seven different possibilities for this option.

Interaction

Poems are produced in sessions, each starting by prompting the generator LLM for creating a poem with target constraints. This is followed by the evaluation of the produced poem, which might be performed by prompting another LLM. If specified constraints are not met, feedback is provided to the generator LLM as another prompt, together with a request for creating a new poem, in what constitutes a new iteration. The session ends when all the constraints are met, or after 10 iterations. Here, we present the prompts used for each subtask of this process.

Generation For poem generation, the generator LLMs were prompted according to the following templates:

```
Please create a poem.  The poem should
contain N_LINES lines TOPIC. Return only
the poem, nothing else.
```

```
Please create a poem.  The poem should
contain 4 lines with N_SYLLABLES syllables
each TOPIC. Return only the poem, nothing
else.
```

The placeholder `TOPIC` is optional, but necessary for indicating a topic. When selected, it always includes the string “about” followed by one of the topics (e.g., “about love”). Placeholders `N_LINES` and `N_SYLLABLES` are to be replaced by integer numbers representing the target number of lines and of syllables, respectively.

Evaluation Besides Erato, LLMs were also used as poetry analysers. This was achieved by prompting the evaluator LLMs according to the following templates, respectively for the number of lines and syllables per line:

```
How many lines does the following poem
have?  Think step by step and at the last
line return the number between square
brackets.  POEM
```

```
Syllable is a unit of pronunciation
having one vowel sound, with or without
surrounding consonants, forming the whole
or a part of a word.  Count the number of
syllables in the given sentence and
return the number between square
brackets:  POEM_LINE
```

The first prompt is made once for each poem. The placeholder `POEM` will have the whole text of the poem, including symbols for new lines, if present. The second prompt, for counting the number of syllables in a line, is made once for each line of the poem and has been previously used in literature (Suvarna, Khandelwal, and Peng 2024). The placeholder `POEM_LINE` will have the text of each line.

We empirically noted that line count by LLMs is improved if models are prompted to “think step by step”, in a similar scenario to zero-shot Chain-of-thought prompting (Kojima et al. 2022).

Feedback Based on the result of the evaluation, feedback is provided to the generator LLM in a natural language

---

[1] `https://ollama.ai/`

prompt that points out the issues found in the lines or syllables, followed by a request for generating a new poem.

If the poem has more or less than the target number of lines, such prompt is based on the following template, where `N_LINES` and `RESULT_N_LINES` are replaced, respectively, by the target number of lines and by the number of lines in the evaluated poem:

```
The poem doesn't have N_LINES lines, it
has RESULT_N_LINES lines.  Can you create
a poem with N_LINES lines?  Return only
the poem, nothing else.
```

When the target number of lines is met, but not all lines have the right number of syllables, the prompt will mention each incorrect line, as follows:

```
The number of syllables in the WHICH_LINE
line is RESULT_N_SYLLABLES and it should
have been N_SYLLABLES.
```

The placeholder `WHICH_LINE` specifies the line in which the number of syllables is incorrect, as an ordinal number (first, second, third, fourth). `RESULT_N_SYLLABLES` is the actual syllable count, according to the evaluator. The feedback prompt ends with a request for generating a new poem:

```
Can you create a poem with the right
number of syllables?  Return only the
poem, nothing else.
```

Produced Data & Analysis

Three different values were set for the temperature hyperparameter, namely: 0.0 (most deterministic), 0.1, 0.7 (less deterministic). For each combination of generator (2), evaluator (3), topic (7), number of lines / syllables (7), and temperature (3), five different sessions were created. The only exception was when temperature was 0.0. Due to its higher determinism, we decided that a single session would be enough. Overall, this resulted in 3,234 interaction sessions that were saved for analysis. Each session might have more than one generation–feedback iteration and the final poem might satisfy the target constraints or not. For illustrative purposes, Figure 3 shows a complete session that, after five iterations, resulted in a poem that meets the specified metrical constraints.

Much can be analysed in the produced data, such as quantifying how often feedback is required, which is an indicator of how LLMs perform in a simple zero-shot generation scenario. Yet, our main goal is to analyse how different LLMs respond to feedback, as well as the impact of using a specific generator or a specific evaluator. This analysis may look at the progression towards the target constraints, across several iterations, but may also look at how other features, like rhymes and diversity, are affected.

Results

This section presents the results of analysing our interactions with the LLMs, which leads to conclusions on both the usage of feedback and its impact on target constraints, i.e., the number of lines and the number of syllables. We then look at the indirect impact of feedback on rhyme and diversity.

Regardless of the evaluator, we use the analysis by Erato as the ground truth, i.e., for assessing the progression. The main reason for this is that, as a rule-based tool, it is more accurate than an LLM. This option is also supported by a pilot study performed for testing how accurate the two LLMs and Erato were for counting syllables. When prompted for feedback, LLMs were always called with the `temperature` parameter set to 0. We selected six short poems from *For Better For Verse*[2] (Tucker 2011) with three degrees of difficulty (easy, medium, hard), ranging between 6 and 15 lines in length. Accuracy, computed for each method, showed that, despite not being perfect, Erato (76.06%) clearly outperformed Gemma2 (61.97%) and Llama3.1 (25.35%). Besides showing a similar trend, the mean average error (0.27, 0.51, 1.07) further suggests the errors that these models make are within acceptable bounds.

Feedback: how often?

Feedback was only provided when the generated poems did not satisfy the target constraints. Therefore, before delving into the impact of feedback, we quantify to what extent feedback was necessary. This may be seen as an indicator of the actual performance of the selected LLMs when it comes to generating poems that satisfy specific formal constraints.

Table 1 reports the proportion of poems that required feedback, organised by evaluator and generator models, and according to the temperature set. The proportion of poems requiring feedback is always between 41% and 53%.

Tables 2 and 3, however, provide additional insights on the proportion of poems requiring feedback when the number of lines or the number of syllables is specified, respectively. They reveal that, in fact, feedback is almost always necessary when targetting a specified number of syllables, and rarely otherwise. This suggests that Llama3.1 and Gemma2 can handle very well the constraint of generating poetry with a pre-specified number of lines, and is substantially different from work (Agirrezabal and Gonçalo Oliveira 2024) using the previous Llama2, where generating poems with more uncommon numbers of lines (e.g., 7, 13) was challenging. This suggests an improvement in the newer generation of Llama models.

As a supplementary analysis, we tested how well the LLMs could generate poems of 3, 4, and up to 17 lines. We found that both tested LLMs could always generate poems with 12 of fewer lines successfully. From there, both models started to decline, but Gemma2 performed much poorly, especially when line numbers were odd. Specifically, Gemma2 failed to generate poems with 13, 15 and 17, respectively 36%, 79% and 100% of the times. On the other hand, for the same numbers, Llama failed only 3%, 5% and 10% of the times.

In opposition to lower numbers of lines, when it comes to generating lines with a specific number of syllables, these models do struggle. Feedback is required for almost all poems, meaning that, in the first iteration, the majority of gen-

---

[2] `https://github.com/waynegraham/for_better_for_verse`

```
> user:  Please create a poem.  The poem should contain 4 lines with 13 syllables each about briefcase.  Return only
the poem, nothing else.

> assistant:  Leather-bound and sturdy frame
Carrying papers in its hold tight
Businessman’s trusty daily mate
Filling hands with weighty load

> user:  The number of syllables in the first line is 9 and it should have been 13.  The number of syllables in the
second line is 9 and it should have been 13.  The number of syllables in the third line is 8 and it should have been
13.  The number of syllables in the fourth line is 7 and it should have been 13.  Can you create a poem with the
right number of syllables?  Return only the poem, nothing else.

> assistant:  Leather-bound and sturdy frame structure
Carrying papers in its hold tightly now
Businessman’s trusty daily faithful mate
Filling hands with weighty heavy loads

> user:  The number of syllables in the first line is 11 and it should have been 13.  The number of syllables in the
second line is 11 and it should have been 13.  The number of syllables in the third line is 10 and it should have
been 13.  The number of syllables in the fourth line is 9 and it should have been 13.  Can you create a poem with the
right number of syllables?  Return only the poem, nothing else.

> assistant:  Leather-bound and sturdy frame structure strong
Carrying papers in its hold tightly now always
Businessman’s trusty daily faithful mate friend
Filling hands with weighty heavy loads slowly

> user:  The number of syllables in the first line is 12 and it should have been 13.  The number of syllables in the
third line is 11 and it should have been 13.  The number of syllables in the fourth line is 11 and it should have
been 13.  Can you create a poem with the right number of syllables?  Return only the poem, nothing else.

> assistant:  Leather-bound and sturdy frame structure strongly
Carrying papers in its hold tightly now always
Businessman’s trusty daily faithful mate friend true
Filling hands with weighty heavy loads slowly

> user:  The number of syllables in the third line is 12 and it should have been 13.  The number of syllables in the
fourth line is 11 and it should have been 13.  Can you create a poem with the right number of syllables?  Return only
the poem, nothing else.

> assistant:  Leather-bound and sturdy frame structure strongly
Carrying papers in its hold tightly now always
Businessman’s trusty daily faithful mate friend truly
Filling hands with weighty heavy loads slowly

> user:  The number of syllables in the fourth line is 11 and it should have been 13.  Can you create a poem with the
right number of syllables?  Return only the poem, nothing else.

> assistant:  Leather-bound and sturdy frame structure strongly
Carrying papers in its hold tightly now always
Businessman’s trusty daily faithful mate friend truly
Filling hands with weighty heavy loads slowly today

> user:  Thank you!
> assistant:  I’m glad I could finally get the poem to meet your requirements.  If you need anything else, feel free
to ask!
```

Figure 3: A session where Llama3.1 is prompted to generate a poem with 4 lines with 13 syllables about *briefcase*. Temperature was set to 0.1 and interaction stopped after five iterations, when a poem meeting the target constraints was produced.

erated poems had at least one line that did not meet the target number. This is in line with previous work (Suvarna, Khandelwal, and Peng 2024), which points out that LLMs are still far from being as accurate as humans in counting syllables.

Back to Table 1, unlike our initial expectations, the temperature has no noticeable impact on the proportion of poems requiring feedback. Moreover, despite the small differences, figures further suggest that, when Gemma2 is used as the evaluator, more poems are considered not to meet the constraints and thus requiring feedback. On the opposite direction, using the rule-based Erato leads to a lower proportion. Even if by a low margin (2-4%), when using the latter tool, more poems generated by Gemma2 require feedback.

Evaluated by	Generated by	Temperature 0.0	0.1	0.7
Erato	Llama3.1	22 (45%)	107 (44%)	108 (44%)
	Gemma2	20 (41%)	100 (41%)	102 (42%)
Llama3.1	Llama3.1	23 (47%)	117 (48%)	116 (47%)
	Gemma2	24 (49%)	117 (48%)	112 (46%)
Gemma2	Llama3.1	26 (53%)	128 (52%)	122 (50%)
	Gemma2	24 (49%)	126 (51%)	129 (53%)

Table 1: Poems that required feedback for different generator models and temperatures, according to different analysis methods (Erato, Llama3.1 or Gemma2).

		Generator	
No. of lines	Requested poems	Llama3.1	Gemma2
4	231	0 (0%)	0 (0%)
7	231	0 (0%)	0 (0%)
8	231	4 (2%)	1 (0.5%)
14	231	72 (31%)	80 (35%)

Table 2: Poems that required feedback for different generator models and number of lines, regardless of the analyser.

		GENERATOR	
No. of syllables	Requested poems	Llama3.1	Gemma2
8	231	231 (100%)	219 (95%)
10	231	231 (100%)	223 (97%)
13	231	231 (100%)	231 (100%)

Table 3: Poems that required feedback for different generator models and number of syllables, regardless of analyser. Note that we specified the number of syllables only when the model was asked to create 4 lines. It was, therefore, 4 lines and 8/10/13 syllables.

Impact on the Number of Syllables

The box plots in Figure 4 illustrate the progression of syllable count accuracy, i.e., the proportion of poems where all lines meet the target number of syllables. We also looked at the mean average error between the actual and target number of syllables, to confirm the same trend, so we opted for not including such plot.

For each evaluator providing feedback, the figure shows the progression for both generator LLMs, Llama3.1 and Gemma2. This is represented by the accuracy in the first iteration (BEG_), in the iteration halfway between the first and the final (MID_), and in the final iteration (END_). When there are only two iterations, nothing is considered for MID_.

We observe that, regardless of the generator LLM and of the source of the feedback, accuracy evolves positively, meaning that poems get closer to meeting the target number of syllables. Therefore, we can say that the feedback is effective. Still, it is more effective for Llama3.1 than for Gemma2. Poems generated by the former achieve, on average, 100% accuracy when the feedback is provided by Erato, whereas, for poems by Gemma2, the increase in accuracy is more modest. Also in favour of the Erato–Llama3.1 combination is the average number of necessary iterations, as shown in Table 4. Generating poems with this combination requires, on average, 6.4 iterations, whereas all the others, but one, require between 8.4 and 8.6. The combination that requires most iterations (9.5) is Llama3.1–Gemma2, where improvements in the accuracy only occur in the final iterations. Moreover, when Gemma2 is used as the evaluator, improvements are not so steady. For poems generated by Llama3.1, despite a significant improvement between the first and the middle iterations, accuracy does not keep on increasing. For poems by Gemma2, the improvement between the middle and the final iteration is also minimal.

Evaluated by	Generated by	Mean no. interactions
Erato	Llama3.1	6.39 ± 2.67
	Gemma2	8.45 ± 3.02
Llama3.1	Llama3.1	8.56 ± 2.51
	Gemma2	9.42 ± 1.64
Gemma2	Llama3.1	8.61 ± 2.26
	Gemma2	8.44 ± 2.80

Table 4: Average number of interactions when the number of syllables is specified.

Rhyme

We showed that, with feedback, we can increase the number of poems that match a target number of syllables. But we also wanted to analyse the impact of this improvement in other features, namely end rhymes. Even though this feature is not explicitly mentioned in our prompts, generated poems typically exhibit end rhymes, which are very common

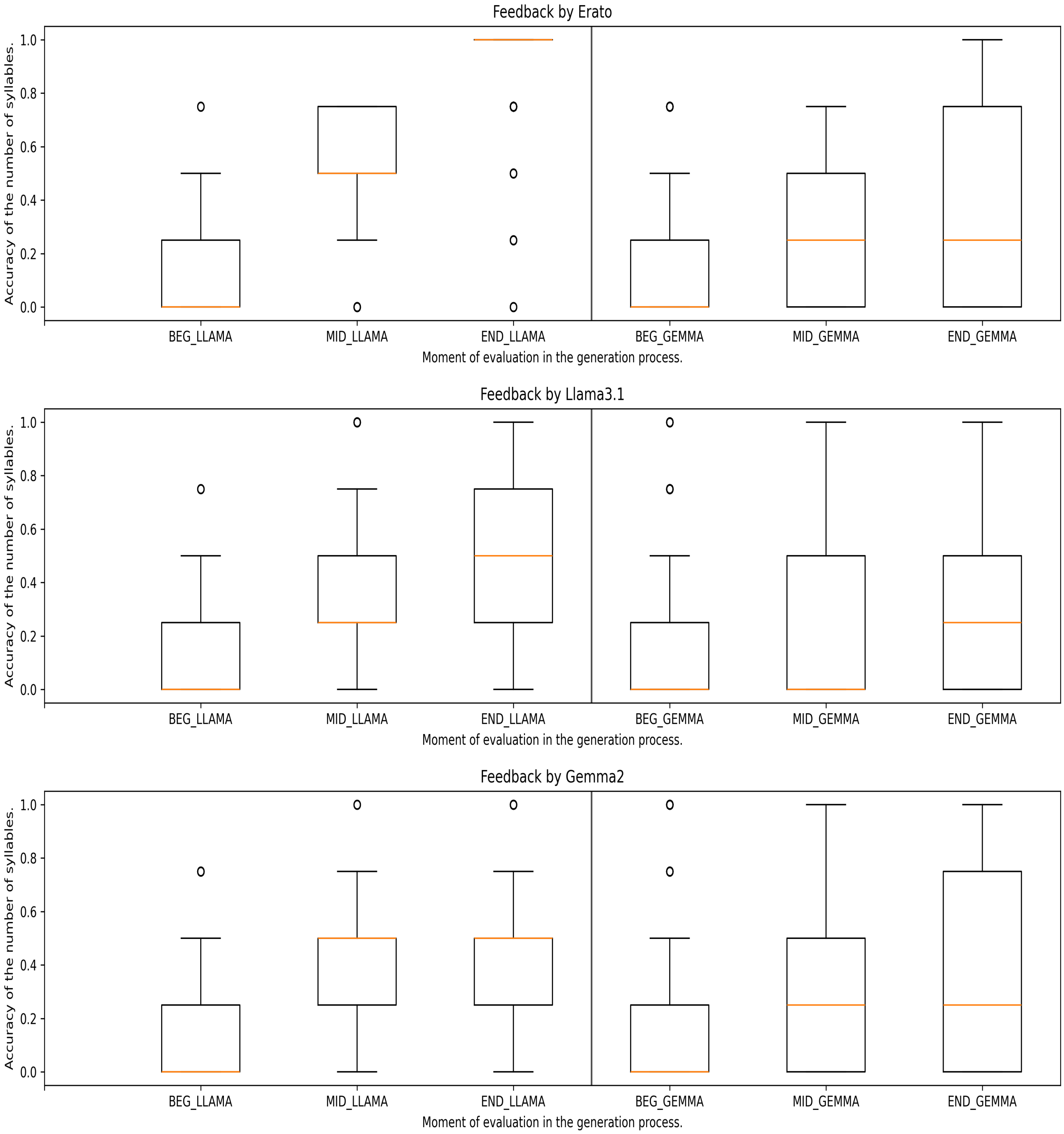

Figure 4: Progression of syllable count accuracy in poems generated by Llama3.1 and Gemma2, for each source of feedback.

in poetry. However, when the text is adjusted to meet certain formal constraints, it is possible that some rhymes are lost.

To analyse how rhymes are affected, Table 5 reports the average rhyme richness (i.e., proportion of lines that rhyme with another line of the same poem) of the poems generated by each LLM, computed in the first and in the final iteration, regardless of the source of feedback. We observe that, when compared to Llama3.1, poems by Gemma2 tend to have more rhymes in the first iteration, and that this feature is not affected by the convergence to the target number of syllables. On the other hand, despite being the model for which the feedback is more effective, the rhyme richness of the poems by Llama3.1 is lower than those of Gemma2 right from the first iteration, and it is substantially affected as the number of syllables converges to the target. After inspection, we noticed that Llama3.1 satisfies the syllable count constraint by replacing or adding words in the end of lines, which tends to break the rhyme. The session in Figure 5 illustrates this situation. On the other hand, Gemma2 changes entire pairs of lines, generally keeping the previous syntactic structure and, most importantly, keeping the rhyme. Figure 6 exemplifies this case.

Generated by	Beginning	Ending
Llama3.1	0.70±0.35	0.18±0.31
Gemma2	0.88±0.22	0.89±0.21

Table 5: Rhyme richness of the poems in the first and last interaction. We only considered those cases that had at least one feedback interaction.

```
> user:  Please create a poem.  The poem should
contain 4 lines with 8 syllables each about router.
Return only the poem, nothing else.

> assistant:
Routers humming in the night
Connecting wires to the light
Data packets flowing fast and free
Internet access for you and me

> user:  The number of syllables in the first line is
7 and it should have been 8.  The number of syllables
in the third line is 9 and it should have been 8.
The number of syllables in the fourth line is 9 and
it should have been 8.  Can you create a poem with
the right number of syllables?  Return only the poem,
nothing else.

> assistant:
Routers humming loud at night
Connecting wires to the light
Data packets flowing free now
Internet access for all me

(...)
```

Figure 5: Example of a prompt to generate a poem with 4 lines and 8 syllables. Erato was used as the evaluator and Llama3.1 as the generator, with temperature 0.7, and the topic *router*. In the second response by the LLM, the syllable count of the third line is fixed by adding the word “now” at the end, breaking the rhyme between “free” and “me”.

```
> user:  Please create a poem.  The poem should
contain 4 lines with 10 syllables each about garbage.
Return only the poem, nothing else.

(...)

> assistant:
Waste piles high, a plastic sea,
Cans and bottles, meant to be free.
Landfills groan beneath their weight,
Earth cries out, sealed by our fate.

> user:  The number of syllables in the first line is
8 and it should have been 10.  The number of
syllables in the second line is 9 and it should have
been 10.  The number of syllables in the third line
is 7 and it should have been 10.  The number of
syllables in the fourth line is 7 and it should have
been 10.  Can you create a poem with the right number
of syllables?  Return only the poem, nothing else.

> assistant:
Mountains rise of trash and waste,
Plastic bottles, cans embraced.
Landfills overflow with dread,
Planet weeps, her future bled.

(...)
```

Figure 6: Example of an interaction using the LLM Gemma2 as the generator. The LLM was asked to generate a poem with 4 lines with 10 syllables each about garbage. We can observe in this interaction that the LLM changes the poem, but the number of syllables remains the same in most lines, and the rhyme constraint is kept (with different words).

Diversity

Another feature that we analyse is diversity across the generated poems. We follow previous work (Gonçalo Oliveira et al. 2017) where ROUGE (Lin 2004) metrics were used to compute the overlap of poems generated under the same conditions. Here, we consider the groups of poems generated by the same models using the same prompts (i.e., same topic, same target features, same temperature) and compute ROUGE in the first and in the final iteration. Lower values of ROUGE mean lower overlap, and thus higher diversity. Poems generated with temperature 0.0 are excluded from this analysis.

A comparable diversity between the first and the final iteration means that the poems are updated similarly, but with varying fixes each time. An increase in diversity means less overlap between poems, indicating different update patterns for each one. On the other hand, a decrease of diversity means that the performed fixes follow a similar strategy and lead to a similar choice of words or expressions.

Figure 7 plots the progress of the average ROUGE-1 scores to show that, on average, there is no noticeable difference. When looking at other variations of ROUGE, the conclusion is similar. Since diversity of poems in the first iteration is similar to those in the last, we can say that, contrarily to rhymes in Llama3.1, diversity is not affected by the progress towards the target number of syllables.

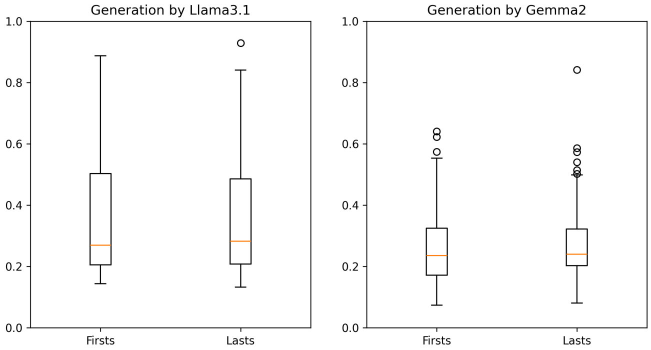

Figure 7: Difference of ROUGE-1 in poems that required feedback. These are averages across groups of poems generated for the same constraints.

Conclusions

We observed that, by simple prompting, the current generation of general-purpose pretrained LLMs is apt for producing poems with a target number of lines. This was not true for the previous generation. However, these LLMs still struggle to produce poems where lines have a specific number of syllables. In order to better satisfy the latter constraint, we explore iterative post-hoc feedback, based on the analysis of the produced poems either by another LLM or by a rule-based tool for poetry analysis.

Experimentation performed with Llama3.1 and Gemma2 and, additionally, Erato for analysis, showed that, regardless the generator or the evaluator, feedback is effective for improving the metrical constraints of LLM-generated poems. The best combination, using Llama3.1 as the generator and Erato as the evaluator, resulted mostly in poems that satisfied the target number of syllables perfectly (76% of the cases). We also found that this metrical improvement does not affect diversity, but, when Llama3.1 is used, it does have a strong negative impact on the produced rhymes. To our knowledge, this was the first time that feedback was explored for generating poetry with LLMs.

Our experimentation resulted in a dataset of 3,234 sessions, where the LLMs were prompted for generating a poem with specific metrical constraints and, whenever necessary, feedback was provided towards the generation of a new poem. Together with the software developed for creating poems using LLMs and feedback, these sessions, with all the generation and feedback prompts, as well as the generated poems, are released in a public repository.[3]

The produced dataset may be useful for future research on LLM-generated poetry, into additional insights on the progress of other poetry-relevant features. Towards broader conclusions, it should be further complemented by including similar sessions with other LLMs, or by considering different constraints and more complex combinations, such as rhyme patterns, or number of syllables for poems with more than four lines.

Acknowledgments

This work was partially supported by the Portuguese Recovery and Resilience Plan through project C645008882-00000055, Center for Responsible AI, and by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., in the framework of the Project CISUC (UIDB/00326/2025 and UIDP/00326/2025).

References

[Agirrezabal and Gonçalo Oliveira 2024] Agirrezabal, M., and Gonçalo Oliveira, H. 2024. Zero-shot Metrical Poetry Generation with Open Language Models: a Quantitative Analysis. In *Proceedings of 15th International Conference on Computational Creativity*, ICCC. Jönköping, Sweden: ACC.

[Agirrezabal, Gonçalo Oliveira, and Ormazabal 2023] Agirrezabal, M.; Gonçalo Oliveira, H.; and Ormazabal, A. 2023. Erato: Automatizing poetry evaluation. In *Proceedings of EPIA Conference on Artificial Intelligence*, 3–14. Springer.

[Chen et al. 2024] Chen, Y.; Gröner, H.; Zarrieß, S.; and Eger, S. 2024. Evaluating diversity in automatic poetry generation. *arXiv preprint arXiv:2406.15267*.

[Chern et al. 2023] Chern, I.; Chern, S.; Chen, S.; Yuan, W.; Feng, K.; Zhou, C.; He, J.; Neubig, G.; Liu, P.; et al. 2023. FacTool: Factuality detection in Generative AI–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

[De Sisto et al. 2024] De Sisto, M.; Hernández-Lorenzo, L.; De la Rosa, J.; Ros, S.; and González-Blanco, E. 2024. Understanding poetry using natural language processing tools: a survey. *Digital Scholarship in the Humanities*.

[Elzohbi and Zhao 2023] Elzohbi, M., and Zhao, R. 2023. Creative data generation: A review focusing on text and poetry. In *Proceedings of the 14th International Conference on Computational Creativity, Ontario, Canada*.

[Gemma Team et al. 2024] Gemma Team; Riviere, M.; Pathak, S.; Sessa, P. G.; Hardin, C.; Bhupatiraju, S.; Hussenot, L.; Mesnard, T.; Shahriari, B.; Ramé, A.; et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

[Gonçalo Oliveira et al. 2017] Gonçalo Oliveira, H.; Hervás, R.; Díaz, A.; and Gervás, P. 2017. Multilingual extension and evaluation of a poetry generator. *Natural Language Engineering* 23(6):929–967.

[Gonçalo Oliveira 2017] Gonçalo Oliveira, H. 2017. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of 10th International Conference on Natural Language Generation*, INLG 2017, 11–20. Santiago de Compostela, Spain: ACL.

[Grattafiori et al. 2024] Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.;

---

[3] github.com/manexagirrezabal/llmgen_poetry_feedback

Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The Llama 3 herd of models. *arXiv e-prints*.

[Jang, Kwon, and Lukasiewicz 2021] Jang, M.; Kwon, D. S.; and Lukasiewicz, T. 2021. Accurate, yet inconsistent? consistency analysis on language understanding models. *arXiv preprint arXiv:2108.06665*.

[Jiang, Wang, and Wang 2023] Jiang, S.; Wang, Y.; and Wang, Y. 2023. Selfevolve: A code evolution framework via large language models. *arXiv preprint arXiv:2306.02907*.

[Kojima et al. 2022] Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35:22199–22213.

[Lamb, Brown, and Clarke 2017] Lamb, C.; Brown, D. G.; and Clarke, C. L. 2017. A taxonomy of generative poetry techniques. *Journal of Mathematics and the Arts* 11(3):159–179.

[Lin 2004] Lin, C.-Y. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

[Madaan et al. 2024] Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems* 36.

[Nathani et al. 2023] Nathani, D.; Wang, D.; Pan, L.; and Wang, W. 2023. MAF: Multi-aspect feedback for improving reasoning in large language models. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 6591–6616. Singapore: Association for Computational Linguistics.

[Pan et al. 2024] Pan, L.; Saxon, M.; Xu, W.; Nathani, D.; Wang, X.; and Wang, W. Y. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics* 12:484–506.

[Peeperkorn et al. 2024] Peeperkorn, M.; Kouwenhoven, T.; Brown, D.; and Jordanous, A. 2024. Is temperature the creativity parameter of Large Language Models? In *Proceedings of 15th International Conference on Computational Creativity*, ICCC. Jönköping, Sweden: ACC.

[Peng, Zhang, and Shang 2024] Peng, L.; Zhang, Y.; and Shang, J. 2024. Controllable data augmentation for few-shot text mining with chain-of-thought attribute manipulation. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics: ACL 2024*, 1–16. Bangkok, Thailand: Association for Computational Linguistics.

[Possi et al. 2023] Possi, M. D. A.; Oliveira, A. D. P.; Moreira, A.; and Costa, L. M. 2023. CARMEN: A method for automatic evaluation of poems. In *Proceedings of 5th International Conference on Natural Language Processing (ICNLP)*, 244–247. IEEE.

[Sawicki et al. 2023] Sawicki, P.; Grzes, M.; Goes, F.; Brown, D.; Peeperkorn, M.; Aisha, K.; and Simona, P. 2023. On the power of special-purpose GPT models to create and evaluate new poetry in old styles. In *Proceedings of the 14th International Conference on Computational Creativity, Ontario, Canada*.

[Suvarna, Khandelwal, and Peng 2024] Suvarna, A.; Khandelwal, H.; and Peng, N. 2024. Phonologybench: Evaluating phonological skills of large language models. In *Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024)*, 1–14.

[Tucker 2011] Tucker, H. F. 2011. Poetic data and the news from poems: A for better for verse memoir. *Victorian Poetry* 49(2):267–281.

[Walsh, Antoniak, and Preus 2024] Walsh, M.; Antoniak, M.; and Preus, A. 2024. Sonnet or not, bot? poetry evaluation for large models and datasets. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 15568–15603.

[Walsh, Preus, and Gronski 2024] Walsh, M.; Preus, A.; and Gronski, E. 2024. Does ChatGPT have a poetic style? In *Proceedings of the Computational Humanities Research Conference 2023*, volume 3558. CEUR-WS.org.

[Yang et al. 2022] Yang, K.; Tian, Y.; Peng, N.; and Klein, D. 2022. Re3: Generating longer stories with recursive reprompting and revision. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 4393–4479. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

[Yu et al. 2024] Yu, C.; Zang, L.; Wang, J.; Zhuang, C.; and Gu, J. 2024. CharPoet: A Chinese classical poetry generation system based on token-free LLM. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 315–325.

[Zhang and Eger 2024] Zhang, R., and Eger, S. 2024. LLM-based multi-agent poetry generation in non-cooperative environments. *arXiv preprint arXiv:2409.03659*.

[Zhang et al. 2023] Zhang, K.; Li, Z.; Li, J.; Li, G.; and Jin, Z. 2023. Self-edit: Fault-aware code editor for code generation. *arXiv preprint arXiv:2305.04087*.