Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 3519–3529
Marseille, 20-25 June 2022

Constrained Language Models for Interactive Poem Generation

Andrei Popescu-Belis,[1,2] **Àlex R. Atrio,**[1,2] **Valentin Minder,**[1]
Aris Xanthos,[3] **Gabriel Luthier,**[1] **Simon Mattei,**[1] **Antonio Rodriguez**[3]

[1] HEIG-VD / HES-SO
CH-1401 Yverdon-les-Bains
Switzerland

[2]EPFL
CH-1015 Lausanne
Switzerland

[3] Université de Lausanne
CH-1015 Lausanne
Switzerland

firstname.lastname@heig-vd.ch

firstname.lastname@unil.ch

Abstract

This paper describes a system for interactive poem generation, which combines neural language models (LMs) for poem generation with explicit constraints that can be set by users on form, topic, emotion, and rhyming scheme. LMs cannot learn such constraints from the data, which is scarce with respect to their needs even for a well-resourced language such as French. We propose a method to generate verses and stanzas by combining LMs with rule-based algorithms, and compare several approaches for adjusting the words of a poem to a desired combination of topics or emotions. An approach to automatic rhyme setting using a phonetic dictionary is proposed as well. Our system has been demonstrated at public events, and log analysis shows that users found it engaging.

Keywords: language models, text generation, poem generation, interactive systems, digital humanities.

1. Introduction

Neural language models (LMs), which are probability distributions over sequences of words or characters, have recently enabled the generation of fluent sentences and texts. However, controlling such models in order to generate specific text structures remains difficult. We propose solutions for constrained language modeling with external features such as form, topics, emotions and rhymes, and integrate them into a system for interactive poem generation, which enables the joint writing of a poem by a human and a computer. Our CR-PO system[1] leverages neural LMs to generate the initial draft of a poem in a form selected by the user, and then enters a cycle of joint human-computer co-editing, in which the user can set various parameters, according to which the computer modifies the current creation. Manual editing is also possible at any stage. The CR-PO system has been demonstrated at a public exhibition and other events at our institutions.

The goal of this paper is to present the poem generation system and our approaches to constrain neural LMs according to different dimensions that characterize poetry. We first present a functional view of CR-PO in Section 2, along with the setting and hardware. We introduce the third party software and the data used to build LMs in Section 3. Our solutions for constrained text generation are presented in Section 4, regarding the form of the poem (stanzas and lines), the adaptation of the poem to given topics or emotions, and the rhyming scheme. We present in Section 5 several actions we took towards the evaluation of CR-PO, and discuss related work in Section 6.

[1]CR-PO stands for *Création Poétique Assistée par Ordinateur*, i.e. computer assisted poetical creation in French.

2. Overview of the System

2.1. Functional Description

CR-PO addresses the following research questions regarding text generation using LMs:

1. While powerful LMs are now available for text completion tasks in well-resourced languages, how can a LM be trained to generate a specific text genre if resources are scarce?
2. How can constraints on form (lines and stanzas) be applied to the unstructured output of autoregressive LMs? For instance, given the relative scarcity of sonnets, hence the difficulty of learning such a form from data, how can it be imposed on LM-generated text?
3. As poems convey topics and emotions, how can a human steer a LM to include one or more topics into a poem, without writing explicitly the beginning of the poem?
4. How to design a system that can function autonomously for a long time in a public exhibition?

The CR-PO system attempts to answer these questions, mostly through its poem generator presented in Section 4 below. To understand the entire system, a functional description is shown in Figure 1. The stages of the creation of a poem are the following ones:

1. The user selects the intended form of the poem, among four predefined options: a quatrain (four lines), a sonnet, a haiku, or free form (3–5 lines of 32–52 characters). Although the generator adapts to any number of stanzas and lines, we found it simpler for the user to choose among a small number of fixed options.

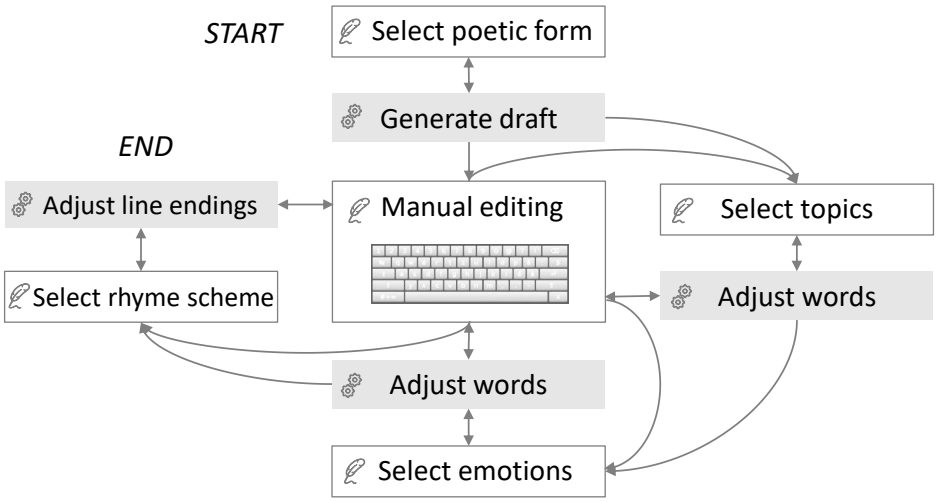

Figure 1: Functional schema of the CR-PO system. The creation of a poem proceeds clockwise. The quills in the white boxes indicate actions of the human user, and the cogwheels in the gray boxes indicate actions of the system's back-end. Manual editing is possible at any stage.

2. Using a general-domain LM trained on French poems, the system generates a first draft respecting the selected form (see Section 4.1).
3. The user can edit the poem using a keyboard, for instance to correct mistakes, improve readability, or express their own creativity. Manual editing is possible after each of the system's contributions, as shown in the central box of Figure 1. The editor is shown in Figure 2 in the Appendix.
4. The user can select one or more topics using sliders, as shown in Figure 3 of the Appendix, from a list of five topics labeled as love, art, nature, spirituality or life-and-death. The system then modifies the poem by adjusting some words to fit the desired topics (see Section 4.2).
5. Similarly, the user can select one or more emotions among happiness, sadness, or aversion. The system then modifies the poem accordingly.
6. Finally, the user selects a rhyming scheme among three possibilities offered in the interface (e.g. AABB, ABBA, or ABAB for a quatrain), and the system changes line endings accordingly (see Section 4.3). This is the last stage, and the delivery of the final poem depends on the technical setup which we now present.

2.2. Implementation and Public Displays

As the focus of this paper is the poem generator and the constraints on LMs, we list here only briefly the main implementation decisions that make the demonstrator operational. Currently, the generator and the GUI are in French, and are being ported to English.

1. CR-PO is implemented in Python, and the GUI uses the Kivy framework.[2]
2. The system runs autonomously on a small-form-factor PC with modest computational requirements.[3] Training of LMs is done separately on a workstation with two GPUs.[4]
3. Users interact with CR-PO through a 32" touchscreen, which is fixed on a stand together with the computer. The system cannot be stopped as long as the physical keyboard and mouse are hidden.
4. The stages of each poem are logged into a JSON file for further analyses, but we do not record all interactions with the interface.
5. For users to keep a memory of their poems, solutions vary according to the presentation setting, but all the following solutions preserve the privacy of the users:
   - The poems can be automatically uploaded to a website, and when users conclude their creation, its URL and an access code for their specific poem are displayed.
   - In the exhibition mentioned below, each completed poem was printed on a large plotter, displayed as a work of art in the center of the room.[5] Users could select parameters of the fonts used for printing.

[2] `https://kivy.org/`

[3] A Dell Optiplex 7060 with an Intel Core i5 3 GHz processor, 8 GB RAM and 128 GB SSD.

[4] nVidia GeForce RTX 2080 Ti 11 GB.

[5] Designed by Nicolas Baldran and David Héritier from the Center for Future Publishing in Geneva, `https://www.centerforfuturepublishing.org`.

- The poems can be printed on a regular printer, if available.
- As suggested on the final screen, users can take a picture of their poem using their smartphones.

The system has been shown at the following events:

- The Digital Lyric exhibition in Morges, Switzerland, in spring 2020, which showcased art works and devices demonstrating novel relations between poetry and technology.[6] Poems created by visitors were made available online.
- Open doors of the HEIG-VD in November 2021, where CR-PO was presented at two workshops to 10–13 years old visitors.
- Internally, CR-PO is available to visitors of the Institute for ICT at HEIG-VD in its showroom.

3. Language Models and Data

3.1. Neural Language Models

LMs are the key technology that enables the generation of raw text, which our system transforms into poems based on the user's constraints. For the initial generation, we selected character-level LMs in order to increase the variability of generated text, which potentially includes non-existing but plausible French words, but also to ensure better grammatical agreement in sentences, as not all word forms are seen during training.

The LM toolkit we used for autoregressive (left-to-right) generation of the first draft of the poem is TextGenRNN,[7] an open-source implementation by Max Woolf of a character-based LM using recurrent neural networks (RNNs) with attention. Written in Python using TensorFlow, following an approach proposed by Andrej Karpathy,[8] TextGenRNN uses LSTM layers according to the early principles of sequence modeling with RNNs (Sutskever et al., 2011; Graves, 2013).

To generate text in a poetic style, we trained a LM on our entire collection of poems (14.45 MB of text, presented below) without additional data from prose. Indeed, on the one hand, we observed that the text generated by this general model is nearly always grammatical without the need for more data, and on the other hand, it is not clear what prose genres are suitable to train a model that generates poetry. We also trained topic-specific and emotion-specific models based on smaller datasets presented below. The advantage of character-based RNNs over more recent Transformer-based decoders such as GPT-2 (Radford et al., 2019) is the lower amount of training data needed to reach acceptable quality.

For adjusting words to topics and emotions, the replacement of non-matching words is better achieved by taking into account the left and right contexts of these words, and not by left-to-right generation (as shown by the results in Section 5). We experimented with a general LM for French, namely CamemBERT (Martin et al., 2019),[9] which is an encoder model trained on a masked language modeling task with 138 GB of French text using the RoBERTa architecture (Liu et al., 2019). We adapted CamemBERT for 20 epochs to the topic-specific datasets presented below, following the documentation from HuggingFace.

3.2. Data Collection

We obtained about 7.72 MB of French poems from `https://www.poesie-francaise.fr/`, with poems in the public domain, only used for training our LMs. We gathered more than 5,000 poems, mostly from the 19$^{\text{th}}$ century and before. We obtained a similar amount of French poems (7.73 MB of text) from Project Gutenberg[10] by downloading text-only versions of works by a set of authors who have mostly written poetry, again in the public domain (about 50 authors and 76 books). The overlap between the two corpora is smaller than 4%. In order to train TextGenRNN with both datasets, we cleaned the texts by removing characters outside the ISO-8859-1 set, any material not part of the poems (such as Project Gutenberg metadata, titles, author names, etc.), and any blank lines, as we do not expect the LM to learn forms from the data.

In addition, to train topic-specific and emotion-specific LMs, we gathered annotated corpora of poems. We used topic and emotions labels from `https://www.poesie-francaise.fr/` – assigned by the creators of the collection – to obtain small datasets for five topics and three emotions, using a correspondence between these eight coarse-grained categories and the observed labels.[11] We augmented these datasets with a second collection of 19$^{\text{th}}$ century French poems with labels, from the University of Lausanne.[12] The data sizes for each topic and emotion are the following ones:

- Topics: *amour* (love, 136,557 words, 776 KB of text), *art* (97,522 words, 561 KB), *nature* (130,923 words, 775 KB), *spiritualité* (spirituality

[6] The exhibition was held at the Château de Morges, from February 14 to May 10, 2020. Called *Code/Poésie* in French, it was curated by Antonio Rodriguez (University of Lausanne) and Sarah Kenderdine (EPFL). For more information, see `https://lyricalvalley.org/digital-lyric-exposition/`.

[7] `https://github.com/minimaxir/textgenrnn`

[8] `https://github.com/karpathy/char-rnn`

[9] CamemBERT-Base model from `https://huggingface.co/camembert-base/`.

[10] `https://www.gutenberg.org/`

[11] Annotating emotions in poems requires in reality a broader range of categories (Haider et al., 2020) but we were constrained here by the available labels.

[12] A 1,001 poem collection put together by Mélina Marchetti under the supervision of Antonio Rodriguez for the Digital Lyric exhibition.

and religion, 95,169 words, 542 KB), *vie et mort* (life and death, 153,554 words, 886 KB).

- Emotions: *joie* (joy, 41,126 words, 229 KB), *tristesse* (sadness, 62,248 words, 352 KB), *aversion* (hate, 44,004 words, 253 KB).

These datasets are too small for training or adapting topic- or emotion-specific LMs. To augment them, we used them as training sets for classifiers, which we then applied to our entire collection of poems (15.45 MB). The classifiers are word-based Naive Bayes models: 5-way for topics and 3-way for emotions. Although they are quite imperfect, the benefits of larger data sets thus obtained exceed their drawbacks. The augmented datasets have the following sizes in MB:

- Topics: *amour* (3.3), *art* (1.0), *nature* (2.0), *spiritualité* (5.3), *vie et mort* (5.8).
- Emotions: *joie* (4.3), *tristesse* (8.2), *aversion* (4.9).

The quality of topic-specific and emotion-specific corpus augmentation has been evaluated by holding out a small portion (10%) of the training data (labeled poems) and testing classifiers on it. We experimented with three different options: the type of model (naive Bayes, decision tree, or logistic regression), the representation of lexical features (Bernoulli, i.e. 'present' vs. 'absent', or multinomial, i.e. 'number of occurrences'), and the chunks used for classification (fixed number of lines, whole stanzas, or whole poems). Since the emotion-specific sub-corpora are not divided in stanzas, we consider chunks of at least seven lines stopping at the first punctuation mark. Overall, our experiments led us to select a naive Bayes classifier (confusion matrix presented in Table 1), with lexical features represented using the Bernoulli model, and splitting the data into stanzas.

	Love	Art	Nature	Spirituality	Life & death
Love	16.8	2.9	4.5	1.0	1.7
Art	8.5	4.6	7.1	1.3	2.2
Nature	8.2	2.8	8.6	0.8	1.2
Spirituality	6.6	1.0	2.4	2.4	1.5
Life & death	7.4	1.4	2.3	0.8	1.9

Table 1: Confusion matrix of the topic classifier (in percentages; horizontally: reference topics; vertically: predicted topics).

4. Constrained Autoregressive Generation of Poems

4.1. Setting the Poetic Form

In the first stage, the character-based TextGenRNN LM generates a poem with the form chosen by the user. Any form can be generated, with any number of stanzas, lines per stanza, and lengths of lines, although in the current interface only four fixed possibilities are offered. The following parameters have been set empirically.

4.1.1. Sampling Probability of the LM

The general LM is made of an input layer of 40 units, an embedding layer of 100 units, and two concatenated bidirectional LSTM layers of 256 units each (128 in each direction). Attention is applied to the concatenation of the embedding layer and the two recurrent layers, and the softmax dense output layer has 89 units, which is the size of our character set V. This covers largely the French character set, with common punctuation signs.

Once trained, the general LM provides a probability distribution over the character vocabulary V, conditioned on context, i.e. on the $N = 40$ previous characters noted c_{n-N+1}^{n-1}, with c_n being the character to generate. To sample character c_n from V using this distribution, we use the temperature parameter t available in TextGenRNN, which we set to $t = 0.4$. Such a value augments the highest probability values in the distribution, and thus makes the model more "prudent" when sampling. Formally, we sample from V with the distribution:

$$P'(c_n) = \frac{P(c_n|c_{n-N+1}^{n-1})^{1/t}}{\sum_{c \in V} P(c|c_{n-N+1}^{n-1})^{1/t}} \quad (1)$$

where $P(c|c_{n-N+1}^{n-1})$ are the probabilities of the LM. Furthermore, in our implementation, we consider a set F of forbidden characters from V that we disallow our system to generate at a given stage. The use of F will be explained below.

Due to the lack of context, the initial parts of most generated sequences are often ungrammatical, contain numerous repeated words, and lack variety – a phenomenon known as "cold start". To address this, we generate four lines that are not shown to the user, and only start displaying the generated poem after them, i.e. after feeding the LM these four lines as initial context.

4.1.2. Adjusting the Length of Lines

The next goal is to control the length of the generated lines, in characters. This can be set at any value, but in our implementation, we approximate the number of characters per syllable as 4 for French, and, for instance, for a 12-syllable line, we aim for 48 characters per line on average.

For each line, we first generate 85% of it, disallowing end-of-sentence punctuation to avoid sentence splits in the middle of the line. Then, we set the minimum length of the remaining part of the line to 2 characters, and the maximum length to 1.8 times the remaining length. For instance, for a line with 48 characters, we first generate 41 characters (85% of 48), and then allow ending sequences between 2 and 13 characters ($1.8 \times 15\%$ of 48).

The algorithm makes several attempts to generate a series of characters *ending with a punctuation mark* within the desired character limit. The attempts are made in the following order, and when one succeeds, the algorithm stops and appends the result to the initial part of the line.

1. Generate a sequence until the maximal ending length is reached.
2. Retry, disallowing the first character of the previous try to encourage diversity.
3. Retry, relaxing the constraint on minimal length.
4. Retry, accepting whitespace as punctuation.
5. Retry, relaxing the constraint on maximal length.

A newline character is appended to the finished line, and the line is appended to the context of the LM. The generation of the next line thus also takes into account the newline character, which drives generation towards a sequence resembling a line start as learned during training. Therefore, our lines are genuinely distinct poem lines, and not just sequences divided manually into lines. The result is post-processed as follows:

1. Remove duplicate whitespaces.
2. Fix whitespaces before and after punctuation.
3. Uppercase line starts.
4. Delete 25% of all punctuation marks at line ends, to avoid too many lines ending with punctuation.
5. Ensure stanzas end with hard punctuation.

Although the creativity of a character-based LM sometimes leads to interesting new words, we decided that for a public exhibition it was preferable to spell check the output before display. We use a French dictionary of 142,541 words[13] (New et al., 2004) and replace unknown words with their closest correct match using Python's SequenceMatcher from the Difflib library.

4.2. Adjusting Poems to Topics and Emotions

The next two stages enable the user to adjust the words of the poem towards one or more desired topics, and then emotions. The principles and interfaces for topic and emotion adjustment are similar, and we present them together. As stated in Section 2, when designing the CR-PO system, we settled on five topics (love, art, nature, spirituality, life-and-death) and three emotions (happiness, sadness, aversion) that appear frequently in poems. The user selects the desired proportion of each topic in the poem (and then emotion) using the sliders shown in Figure 3 of the Appendix. The values for the m topics (or emotions) are coded as a vector $\mathbf{w} = (w_1, \ldots, w_m)$ of m weights between 0 and 1.

Poem adjustment requires two operations: select the words to be replaced (4.2.1), i.e. those that do not match well with the desired topics or emotions, and replace them with words that match better (4.2.2). For each operation, topic- or emotion-specific LMs (Section 3.2) provide an obvious solution: words that have higher perplexities for these LMs than for the general LM should be replaced with words generated using these specific LMs. However, we propose for each operation an alternative solution, which human evaluators have found to perform better (as presented in Section 5).

4.2.1. Word Selection

Using topic- or emotion-specific LMs, the baseline criterion for word selection is the likelihood of each word according to these specific models: words with the lowest values are good candidates for replacement. In our implementation, we compute the difference between the weighted average (by $\mathbf{w}$) of the likelihoods given by the specific LMs and the likelihood of the general LM, then rank all words by decreasing values, and select about 8% of the words that are at least 3 characters long.[14]

The second method for word selection uses *independence quotients (IQs)* following the approach of Egloff and Bavaud (2018). The IQ value Q_{ik} for word i and category k is the ratio of the observed count of word i in poems belonging to category k to its expected count assuming independence of words and categories. Formally:

$$Q_{ik} = \frac{C_{ik} \cdot C_{\bullet\bullet}}{C_{i\bullet} \cdot C_{\bullet k}} \tag{2}$$

where C_{ik} is the count of word i in the poems of category k and the '$\bullet$' sign denotes summation over the corresponding index. The IQ values are non-negative, smaller than 1 if word i is under-represented in category k, greater than 1 if i is over-represented in k, and 1 if the count of i in k equals its expected count assuming independence. The IQ matrix $\mathbf{Q} = (Q_{ik})_{1 \le i \le n, 1 \le k \le m}$ for n words and m categories can be pre-computed.

The dot product $\mathbf{Q} \cdot \mathbf{w^T}$ represents the fitness values of all words given a choice of topics (or emotions), which shows how closely the profile of IQs of each word i matches the used-defined weights $\mathbf{w}$ of the categories. A fraction of the words with the lowest fitness are then replaced.

We built the IQ matrices using labeled poems from `https://www.poesie-francaise.fr/` with a mapping of their labels into our five topics or three emotions. We only considered nouns, verbs and adjectives as found by TreeTagger[15] (Schmid, 1994) and we excluded stopwords[16] and words appearing fewer than

[13] `http://www.lexique.org/`

[14] As TextGenRNN is a character-based LM, the likelihood of a word is the average of the character probabilities.

[15] `https://www.cis.lmu.de/~schmid/tools/TreeTagger/`

[16] From the list available at `http://members.unine.ch/jacques.savoy/clef/frenchST.txt`.

5 times. This resulted in 8,146 word types for topics and 2,621 for emotions in the respective IQ matrices.

4.2.2. Word Replacement

We have tested two methods for generating word replacements: either with the character-level LM from TextGenRNN, trained from scratch with topic or emotion specific data, or with a word-level CamemBERT model fine-tuned using the topic or emotion specific data. When several topics (or emotions) have non-zero weights in $\mathbf{w}$, we decide randomly which model to use to generate each character or word, based on probabilities derived from $\mathbf{w}$.

As the first method uses a left-to-right RNN, only the left context of the word to be replaced can be considered. The replacement is generated character by character, forbidding punctuation for the first three characters, then allowing it (including whitespace) and stopping when a punctuation mark is generated. For the second method, once the specific CamemBERT model is drawn, we give it the left and right contexts of the word to replace and a mask token in the respective position, and we select randomly among the five words receiving the highest probabilities.

The 2×2 options for word selection (TextGenRNN vs. IQs) and replacement (TextGenRNN vs. CamemBERT) were evaluated by human judges, and results are summarized in Section 5 below.

4.3. Setting the Rhyming Scheme

In the final stage, the user can select a rhyming scheme to apply, represented using letters, e.g., 'AABB', 'ABAB' or 'ABBA' for a quatrain. The system selects which line endings must be changed, retrieves a list of candidate words from a dictionary, scores them with the general LM and selects the highest-scoring one with the same POS tag as a replacement. We use the same dictionary as in Section 4.1, as it includes the phonetic representation and POS tag of each word.

The main challenge is to exploit the phonetic forms of words to identify the rhymes, i.e. the ending sounds which must match across words. We formulate the following stages using regular expressions: (1) identify the final vowel of the word; (2) extract either the following consonant, if any, or the immediately previous ones, if any; (3) otherwise, extract the immediately previous vowels plus the previous consonant. All the rhymes are two or three sounds long, as exemplified in Table 2.

Although these rules produce imperfect rhymes, stricter ones were also tested, but they either did not find rhyming words, or found words that were too similar, e.g., singular vs. plural of the same words – perhaps due to the reduced dictionary size (150k words).

To apply a rhyming scheme to a poem, we keep the final word of a line if it is the first one in the rhyming scheme: e.g., with 'ABBA', the first two lines are not changed. We also store their endings in order not to repeat them below. Then, for each line ending to change, we search for at most five candidate words that have the same rhymes, with priority to words that have the same POS tag as the word to be changed. We score the candidates in context with the general LM and choose the candidate with the highest score. If the user edits the poem, the same process is run based on the edited version.

Word	Rhyme	Word	Rhyme
endormant	m@	raisonnassent	as
chipez	pe	perturbent	yRb
plaisantions	tj§	béatement	m@
guet-apens	p@	soumettions	tj§
brouillaient	ujE	misait	zE
vallonnés	ne	observées	ve
fumant	m@	pugilistes	ist
envole	Ol	hydrophiles	il

Table 2: Examples of rhymes extracted from the phonetic representations of some words in the dictionary.

5. Evaluation

We performed several experiments with human users of the CR-PO system. First, we summarize an experiment comparing the 2×2 adjustment methods from Section 4.2. Then, we present statistics from the two public events where CR-PO was used.

5.1. Results from A/B Testing

We tested all four combinations of word selection and replacement methods presented in Section 4.2 with 15 automatically-generated poems presented to 13 users. Each user saw the initial poem and had to compare two outputs of topic or emotion adaptation, differing on one of the two stages, either word selection or word replacement. Users were asked to choose the best of the two outputs, with ties allowed, on the dimensions of topic relevance and fluency, by answering the following questions:

- Which output is closer to the indicated topic?
- Which is the most understandable output, i.e. the one that makes the most sense in French?

We present the answers in Table 3. The use of IQ values for word selection leads to poems that are clearly perceived as more topically-relevant than when using the LM probability difference. These poems are also perceived as more fluent, although the difference is lower. The smaller effect on fluency can be explained by the fact that the IQ score does not consider the context of words, which may lead to replace words that are particularly important for fluency. For word replacement, CamemBERT clearly outperforms TextGenRNN. This can be explained by the larger amount of training data in comparison with the RNN character-level models, which were trained from scratch, and by the bidirectional nature of CamemBERT. As a result, we chose IQ

as a criterion for word selection, and CamemBERT for word replacement.

	Topic	Fluency
Word selection		
TextGenRNN is better	17.3	24.4
IQ values are better	**67.7**	**41.7**
The two are similar	14.9	33.9
Word replacement		
TextGenRNN is better	14.3	3.1
CamemBERT is better	**68.3**	**79.2**
The two are similar	17.5	17.7

Table 3: Answers of human judges (%) for each method of word selection and replacement.

	AVG	STD
1. Generation of 1st draft	1.62	1.10
Manual editing	0.31	0.47
2. Topic adjustment	1.05	1.23
Manual editing	0.36	0.48
3. Emotion adjustment	0.95	0.91
Manual editing	0.26	0.45
4. Rhyming scheme	1.26	1.67
Manual editing	0.38	0.49

Table 4: Average number of interactions with the CR-PO system, for each stage, at the 2021 Open Doors of HEIG-VD (25 visitors, 42 poems).

5.2. Results from Use by the General Public

As stated at the end of Section 2, CR-PO was presented to the general public on two occasions, and is also available in the ICT Showroom at HEIG-VD. Upon the first occasion, at the Digital Lyric exhibition (see footnote 6), we collected about 100 poems in 13 days, before the exhibition was closed due to the Covid-19 pandemic. The interactions with CR-PO were logged in a central database, which will be analyzed in the future, when more poems are collected.

CR-PO was presented at a workshop for young visitors, aged 10–13, at the HEIG-VD Open Doors in November 2021. After a discussion about artificial creativity and an overview of CR-PO, each visitor could experience the co-creation of a poem. We gathered 42 poems from 25 visitors, who all felt quite engaged by CR-PO and tried all its functions. Table 4 shows the average number of interactions and calls to the editing window for each stage. An average of 1.62 interactions at the first stage means that some users started over and asked for a new first draft, while 0.31 manual editing at this stage means that on average, 1 user out of 3 modified the first draft using the editor. The topic and emotion adjustments were each tried once per poem, on average, with manual edits in 1/4–1/3 of the times. Despite coming at the end, the automatic generation of rhyming schemes also raised interest from users. Examples of generated poems are given in the Appendix.

6. Related Work

Poetry generation is a specific task within the field of Natural Language Generation, but also a significant issue in the field of computational creativity (McGregor et al., 2016). Before the advent of deep neural LMs, various combinations of rule-based approaches and n-gram LMs have been tried. For instance, in their broad discussion of computational creativity, McGregor et al. (2016) define a poem generation model, which uses word vectors to infer semantic relations, followed by a phonological model, an n-gram LM, and a sentiment model. Their basis for poem generation are topics from Switchboard conversations annotated with sentiment scores.

Large neural LMs have brought high expectations regarding their capacities to generate structured texts such as poems. However, controlling LMs during generation is still an open problem (Dathathri et al., 2019), despite promising models such as CTRL[17] (Keskar et al., 2019). Direct poem generation with GPT-2 has been anecdotally discussed in blog entries.[18]

Deep neural networks have been used several times for poem generation, mostly in English or in Chinese. The main challenge is to learn and use the constraints of poetic style from the data, and most studies focus on rhythm and rhymes. Most of these systems are not interactive: the human user can only, in principle, set the initial parameters. For instance, Hopkins and Kiela (2017) train a large recurrent neural LM with LSTM units directly on the phonetic encoding of poems (1.5 MB of text from `https://www.sonnets.org`), and constrain the form with a finite state machine. In their study, human judges were not able to distinguish machine-generated poems from human-authored ones. "Deep-speare" is a system for sonnet generation in Shakespearean style, which captures especially rhythm and rhyme, with a fixed form (Lau et al., 2018). The model also uses a bidirectional RNN with LSTM units. The results of evaluation by experts show that while constraints on form can be quite easily satisfied, readability and substance still hamper machine-generated poems. More recently, Wöckener et al. (2021) used conditioned RNNs in a system that is able to learn stylistic features from the data, such as length and sentiment, although not rhymes. Moreover, the authors show that models such as GPT-2 struggle with rhymes as well.

For Chinese, one of the earliest systems using RNNs was proposed by Zhang and Lapata (2014), starting from user-provided keywords and generating a quatrain line-by-line, with a convolutional sentence model and pre-defined line lengths and tonal patterns. Previously,

[17] `https://github.com/salesforce/ctrl`

[18] E.g., `https://www.gwern.net/GPT-2`.

as done e.g. by Yan et al. (2013), the numerous constraints of Chinese poetry were solved through numeric optimization algorithms. The task of poem generation can be additionally constrained: for instance, Yang et al. (2019) study the problem of generating a poem from prose, an approach that allows users to convey more precise meanings than when seeding the poem with keywords only. Using a variational encoder and adversarial training, the system designed by Li et al. (2018) generates a Chinese poem given the title as a representation of its topic. They evaluate their poems in terms of topic consistency, fluency, meaningfulness and "poeticness." Additional constraints can be imposed: for instance, to generate an acrostic poem where the first letters of the lines spell a given word, Agarwal and Kann (2020) use a left-to-right 3-layer RNN with LSTM units and a separate rhyming model.

Turning now to interactive systems, ASPERA is an expert system for prose-to-poetry conversion in Spanish (Gervás, 2001). The rule-based system gathers information from the user about the style and the content, and uses NLG techniques and an example-based approach to generate a poem, although collaborative creation does not seem to be possible.

PoeTryMe is a rule-based interactive poem generation system initially designed for Portuguese and later extended to Spanish and English (Gonçalo Oliveira, 2012; Gonçalo Oliveira et al., 2019). Developed over a long period of time, and also available through a web-based interface,[19] the system leverages its grammatical and semantic knowledge to offer capabilities for word-level or line-level modifications in terms of number of syllables, "surprise" level, or keywords. Although many default forms are possible, there are no direct ways to express topics or emotions, and the fluency in English appears to be limited by the smaller amount of knowledge available to the system.

Poem Machine (Hämäläinen, 2018; Hämäläinen and others, 2018) was developed for Finnish and was focused from the start on assisting primary school children in creating poetry as a follow up to previous experiments (Kantosalo et al., 2015). To cope with the complex morphology of Finnish, the system uses finite state transducers and a semantic network. Predefined forms and topics are proposed, and an assistance with rhythm is provided too. Similar to what we observed when children used CR-PO, Poem Machine helps to teach the constraints of poetic forms and makes experiments with poem creation more entertaining for children.

Hafez was one of the first systems to combine interaction with deep neural LMs (Ghazvininejad et al., 2016; Ghazvininejad et al., 2017). The system proceeds in the opposite order of CR-PO: it first asks the user for a number of parameters, including sample words, sentiment, and repetitiveness, then it formulates internally the constraints as a set of transducers, and uses a word-based RNN LM to generate a poem (a quatrain) satisfying these constraints. User satisfaction was shown to increase when learning the initial parameters from previous interactions. Our system makes a different use of the LM: we use a character-based LM which better captures the grammatical inflections of French regardless of the forms seen during training, and we use also LMs to replace mismatching words. Moreover, CR-PO can be run with acceptable speed (less than 5 seconds) on a small computer witout a GPU, which makes our hardware easy to set up.

[19] `https://poetryme.dei.uc.pt/` – see also `@poetartificial` on Twitter.

7. Conclusion

We presented CR-PO, a system for interactive poetry generation in French, putting forward solutions that combine neural LMs and rule-based constraints on form, topic, emotion, and rhyming scheme. Together with the hardware and the graphical interfaces, we achieved a fully functional, robust system, which was left without supervision in a public exhibition. The system also represents a platform which can be extended through future developments, such as porting it to English, improving the management of rhythm, and allowing users to provide seed words.

Overall, the main observation made when combining LMs and explicit constraints is that the poems generated by such an approach lack a high-level meaning conveyed by several lines or even stanzas, for instance as complex visual scenes or short narratives. This is a shortcoming of most poem generation systems based on deep neural LMs, and only extremely large LMs such as GPT-3 seem to be able to overcome it for plain text. Therefore, finding solutions that increase the coherence of texts generated by smaller LMs is a promising research question.

8. Acknowledgments

The design and implementation of the system were directly supported by the SNSF Agora project "Digital Lyric" (n. 184330), for which Antonio Rodriguez received the Agora Optimus prize. We also acknowledge the support of HES-SO through the PhD support fund (AGP n. 99864), of SNSF through the DOMAT project (n. 175693), and of the Institute for ICT at HEIG-VD.

9. Bibliographical References

Agarwal, R. and Kann, K. (2020). Acrostic poem generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1230–1240, Online. Association for Computational Linguistics.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. (2019). Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Egloff, M. and Bavaud, F. (2018). Taking into account semantic similarities in correspondence analysis. In *Proceedings of the Workshop on Computational Methods in the Humanities 2018 (COMHUM 2018)*, volume 2314, pages 45–51. CEUR Workshop Proceedings.

Gervás, P. (2001). An expert system for the composition of formal spanish poetry. In *Applications and Innovations in Intelligent Systems VIII*, pages 19–32, London. Springer.

Ghazvininejad, M., Shi, X., Choi, Y., and Knight, K. (2016). Generating topical poetry. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

Ghazvininejad, M., Shi, X., Priyadarshi, J., and Knight, K. (2017). Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics.

Gonçalo Oliveira, H., Mendes, T., Boavida, A., Nakamura, A., and Ackerman, M. (2019). CoPoeTryMe: interactive poetry generation. *Cognitive Systems Research*, 54:199–216.

Gonçalo Oliveira, H. (2012). PoeTryMe: a versatile platform for poetry generation. In *Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence (C3GI)*, Montpellier, France.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Haider, T., Eger, S., Kim, E., Klinger, R., and Menninghaus, W. (2020). PO-EMO: Conceptualization, annotation, and modeling of aesthetic emotions in German and English poetry. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*, pages 1652–1663, Marseille, France. European Language Resources Association.

Hämäläinen, M. et al. (2018). Harnessing nlg to create finnish poetry automatically. In *Proceedings of the 9th International Conference on Computational Creativity*. Association for Computational Creativity.

Hämäläinen, M. (2018). Poem Machine - a co-creative NLG web application for poem writing. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 195–196, Tilburg University, The Netherlands. Association for Computational Linguistics.

Hopkins, J. and Kiela, D. (2017). Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 168–178, Vancouver, Canada. Association for Computational Linguistics.

Kantosalo, A. A., Toivanen, J. M., and Toivonen, H. T. T. (2015). Interaction evaluation for human-computer co-creativity: A case study. In *Proceedings of the 6th International Conference on Computational Creativity*. Brigham Young University.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Lau, J. H., Cohn, T., Baldwin, T., Brooke, J., and Hammond, A. (2018). 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*, pages 1948–1958, Melbourne, Australia. Association for Computational Linguistics.

Li, J., Song, Y., Zhang, H., Chen, D., Shi, S., Zhao, D., and Yan, R. (2018). Generating classical Chinese poems via conditional variational autoencoder and adversarial training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3890–3900, Brussels, Belgium. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: a robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de La Clergerie, É. V., Seddah, D., and Sagot, B. (2019). CamemBERT: a tasty French language model. *arXiv preprint arXiv:1911.03894*.

McGregor, S., Purver, M., and Wiggins, G. (2016). Process based evaluation of computer generated poetry. In *Proceedings of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation*, pages 51–60, Edinburgh, UK. Association for Computational Linguistics.

New, B., Pallier, C., Brysbaert, M., and Ferrand, L. (2004). Lexique 2 : A new French lexical database. *Behavior Research Methods, Instruments, & Computers*, 36:516–524.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA, USA.

Wöckener, J., Haider, T., Miller, T., Nguyen, T.-K., Nguyen, T. T. L., Pham, M. V., Belouadi, J., and Eger, S. (2021). End-to-end style-conditioned poetry generation: What does it take to learn from examples alone? In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics*

for Cultural Heritage, Social Sciences, Humanities and Literature, pages 57–66, Punta Cana, Dominican Republic (online). Association for Computational Linguistics.

Yan, R., Jiang, H., Lapata, M., Lin, S.-D., Lv, X., and Li, X. (2013). i, Poet: Automatic Chinese poetry composition through a generative summarization framework under constrained optimization. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2197–2203. AAAI Press.

Yang, Z., Cai, P., Feng, Y., Li, F., Feng, W., Chiu, E. S.-Y., and Yu, H. (2019). Generating classical Chinese poems from vernacular Chinese. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6155–6164, Hong Kong, China. Association for Computational Linguistics.

Zhang, X. and Lapata, M. (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, Doha, Qatar. Association for Computational Linguistics.

Appendix

Three sample outputs of our system, respectively prompted internally with the strings "*Je rêve*", "*Être distrait*", and "*Ma tête dans les nuages*" (I dream, being distracted, and my head in the clouds, in French), are provided hereafter to illustrate its results. Put together, these outputs constitute CR-PO's first participation to a poetry contest, on the same topic as the prompt strings. To improve quality, the three outputs were selected from a total of six.

Je rêve des couleurs de la forêt en silence.
La terre s'accroche aux premiers pas de l'homme.
Le soir, dans tout ce qu'il trouve dans ces tomes
Sous les champs d'or de la fleur de la naissance !

Être distrait pour la chair sourde de son âme
L'espoir d'un chant sous le cinname ;
La fille du ciel s'abat, le soir, le temps est éclatant,
Et le printemps, et le soleil se clapotant
Et ses doigts d'or s'en vont au fond du soir.

Ma tête dans les nuages, les arbres noirs, en haut,
Et les parfums pleurants s'en vont en lui rendant les cieux.
Les fleurs de la saison descend de la terre,
A l'entour des vagues du vent, les ombres de l'enfance.

In addition, we provide in the two snapshots below two more examples of CR-PO's creations.

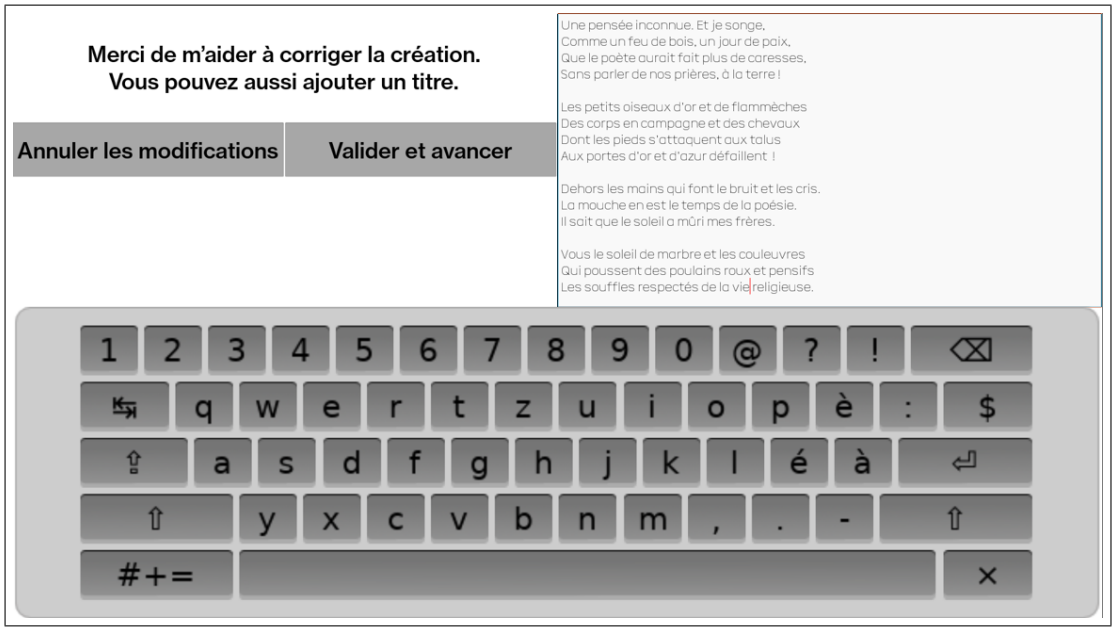

Figure 2: User interface for editing the poem, with a keyboard displayed on the touch screen. The instructions in French state: "Please help me to correct the created poem. You can also add a title." with buttons for validating or canceling edits. This snapshot is shown in reverse colors for readability.

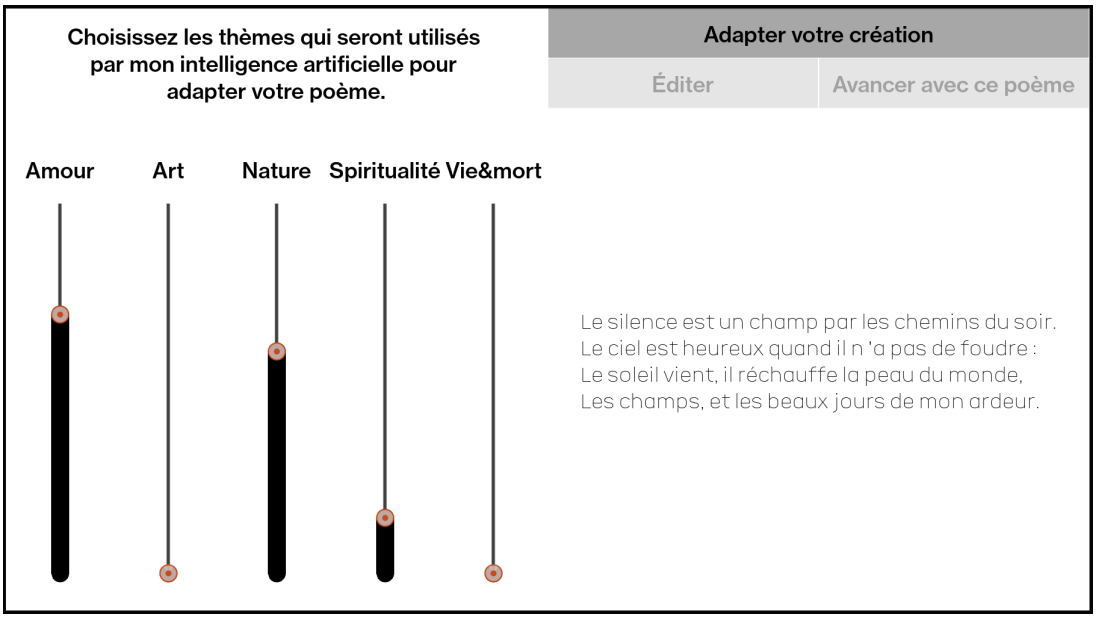

Figure 3: User interface for selecting the proportions of topics to represent in the poem. The instructions in French state: "Select the topics that my AI will use to adapt your poem." and the five topics are, in order: 'love', 'art', 'nature', 'spirituality', and 'life-and-death'. This snapshot is shown in reverse colors for readability.