Automatic Poetry Generation from Prosaic Text

Tim Van de Cruys
Institut de Recherche en Informatique de Toulouse (IRIT)
Artificial and Natural Intelligence Toulouse Institute (ANITI)
CNRS, Toulouse
tim.vandecruys@irit.fr

Abstract

In the last few years, a number of successful approaches have emerged that are able to adequately model various aspects of natural language. In particular, language models based on neural networks have improved the state of the art with regard to predictive language modeling, while topic models are successful at capturing clear-cut, semantic dimensions. In this paper, we explore how these approaches can be adapted and combined to model the linguistic and literary aspects needed for poetry generation. The system is exclusively trained on standard, non-poetic text, and its output is constrained in order to confer a poetic character to the generated verse. The framework is applied to the generation of poems in both English and French, and is equally evaluated for both languages. Even though it only uses standard, non-poetic text as input, the system yields state of the art results for poetry generation.

1 Introduction

Automatic poetry generation is a challenging task for a computational system. For a poem to be meaningful, both linguistic and literary aspects need to be taken into account. First of all, a poetry generation system needs to properly model language phenomena, such as syntactic well-formedness and topical coherence. Furthermore, the system needs to incorporate various constraints (such as form and rhyme) that are related to a particular poetic genre. And finally, the system needs to exhibit a certain amount of literary creativity, which makes the poem interesting and worthwhile to read.

In recent years, a number of fruitful NLP approaches have emerged that are able to adequately model various aspects of natural language. In particular, neural network language models have improved the state of the art in language modeling, while topic models are successful at capturing clear-cut, semantic dimensions. In this paper, we explore how these approaches can be adapted and combined in order to model both the linguistic and literary aspects that are required for poetry generation. More specifically, we make use of recurrent neural networks in an encoder-decoder configuration. The encoder first constructs a representation of an entire sentence by sequentially incorporating each word of the sentence into a fixed-size hidden state vector. The final representation is then given to the decoder, which emits a sequence of words according to a probability distribution derived from the hidden state of the input sentence. By training the network to predict the next sentence with the current sentence as input, the network learns to generate plain text with a certain discourse coherence. By modifying the probability distribution yielded by the decoder, we enforce the incorporation of poetic constraints, such that the network can be exploited for the generation of poetic verse. It is important to note that the poetry system is not trained on poetic texts; rather, the system is trained on a corpus of standard, prosaic texts extracted from the web, and it will be the constraints applied to the network's probability distribution that confer a poetic character to the generated verse.

The rest of this article is structured as follows. In section 2, we present an overview of related work on automatic poetry generation. Section 3 describes the different components of our model. In section 4, we present an extensive human evaluation of our model, as well as a number of examples generated by the system. Section 5, then, concludes and discusses some future research directions.

2 Related work

Early computational implementations that go beyond mere mechanical creativity have often relied on rule-based or template-based methods. One of the first examples is the ASPERA system (Gervás,

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2471–2480

2001) for Spanish, which relies on a complex knowledge base, a set of rules, and case-based reasoning. Other approaches include Manurung et al. (2012), which combines rule-based generation with genetic algorithms, Gonçalo Oliveira (2012)'s PoeTryMe generation system, which relies on chart generation and various optimization strategies, and Veale (2013), which exploits metaphorical expressions using a pattern-based approach.

Whereas poetry generation with rule-based and template-based models has an inherent tendency to be somewhat rigid in structure, advances in statistical methods for language generation have opened up new avenues for a more varied and heterogeneous approach to creative language generation. Greene et al. (2010), for example, use an *n*-gram language model in combination with a rhythmic model implemented with finite-state transducers. And more recently, recurrent neural networks (RNNs) have been exploited for poetry generation; Zhang and Lapata (2014) use an encoder-decoder RNN for Chinese poetry generation, in which one RNN builds up a hidden representation of the current line in a poem, and another RNN predicts the next line word by word, based on the hidden representation of the current line. The system is trained on a corpus of Chinese poems. Yan (2016) tries to improve upon the encoder-decoder approach by incorporating a method of iterative improvement: the network constructs a candidate poem in each iteration, and the representation of the former iteration is used in the creation of the next one. And Wang et al. (2016) extend the method using an attention mechanism.

Ghazvininejad et al. (2016) combine RNNs (for syntactic fluency) with distributional similarity (for the modeling of semantic coherence) and finite state automata (for imposing literary constraints such as meter and rhyme). Their system, *Hafez*, is able to produce well-formed poems with a reasonable degree of semantic coherence, based on a user-defined topic. Hopkins and Kiela (2017) focus on rhythmic verse; they combine an RNN, trained on a phonetic representation of poems, with a cascade of weighted finite state transducers. Lau et al. (2018) present a joint neural network model for the generation of sonnets, called *Deep-speare*, that incorporates the training of rhyme and rhythm into the neural network; the network learns iambic stress patterns from data, while rhyming word pairs are separated from non-rhyming ones using a margin-based loss. And a number of recent papers extend neural poetry generation for Chinese with various improvements, such as unsupervised style disentanglement (Yang et al., 2018), reinforcement learning (Yi et al., 2018), and rhetorical control (Liu et al., 2019).

Note that all existing statistical models are trained on or otherwise make use of a corpus of poetry; to our knowledge, our system is the first to generate poetry with a model that is exclusively trained on a generic corpus, which means the poetic character is endowed by the model itself. Secondly, we make use of a latent semantic model in order to model topical coherence, which is equally novel.

3 Model

3.1 Neural architecture

The core of the poetry system is a neural network architecture, trained to predict the next sentence S_{i+1} given the current sentence S_i. The architecture is made up of gated recurrent units (GRUs; Cho et al., 2014) that are linked together in an encoder-decoder setup. The encoder sequentially reads in each word $w^i_{1,\ldots,N}$ of sentence S_i (represented by its word embedding $\mathbf{x}$) such that, at each time step t_i, a hidden state $\hat{\mathbf{h}}_t$ is computed based on the current word's embedding $\mathbf{x}_t$ and the previous time step's hidden state $\hat{\mathbf{h}}_{t-1}$. For each time step, the hidden state $\hat{\mathbf{h}}_t$ is computed according to the following equations:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\hat{\mathbf{h}}_{t-1}) \quad (1)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\hat{\mathbf{h}}_{t-1}) \quad (2)$$

$$\bar{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \hat{\mathbf{h}}_{t-1})) \quad (3)$$

$$\hat{\mathbf{h}}_t = (1 - \mathbf{z}_t) \odot \hat{\mathbf{h}}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t \quad (4)$$

where $\mathbf{r}_t$ represents the GRU's reset gate, $\mathbf{z}_t$ represents the update gate, $\bar{\mathbf{h}}_t$ represents the candidate update state, and $\odot$ represents pointwise multiplication.

$\hat{\mathbf{h}}_t$ can be interpreted as a representation of the sequence $w_1, \ldots, w_t$, and the final hidden state $\hat{\mathbf{h}}_N$ will therefore be a representation of the entire sentence. This final hidden encoder state is transferred to the decoder. The decoder then sequentially predicts the next sentence word by word, conditioned on the encoder's final hidden representation; at each time step t_{i+1}, the decoder equally computes a hidden state $\mathbf{h}_t$ based on the current word's embedding $\mathbf{x}_t$ (which was predicted by the decoder

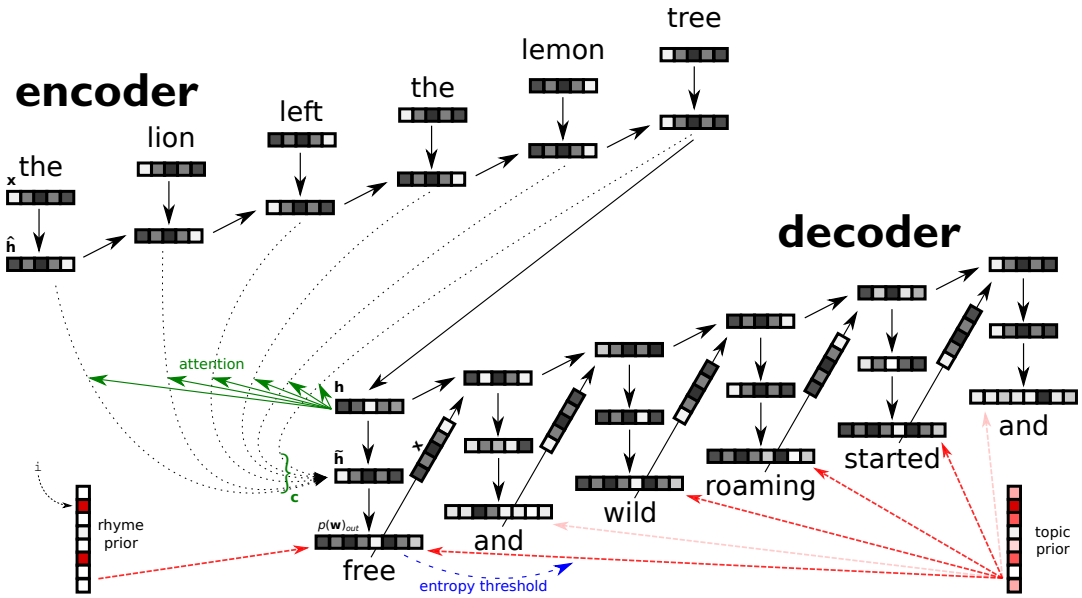

Figure 1: Graphical representation of the poetry generation model. The encoder encodes the current verse, and the final representation is given to the decoder, which predicts the next verse word by word in reverse. The attention mechanism is represented for the first time step. The rhyme prior is applied to the first time step, and the topic prior is optionally applied to all time steps, mediated by the entropy threshold of the network's output distribution.

in the previous time step) and the previous time step's hidden state $\mathbf{h}_{t-1}$ (the first hidden state of the decoder is initialized by $\hat{\mathbf{h}}_N$ and the first word is a symbolic start token). The computations for each time step $\mathbf{h}_t$ of the decoder are equal to the ones used in the encoder (equations 1 to 4).

In order to fully exploit the entire sequence of representations yielded by the encoder, we augment the base architecture with an attention mechanism, known as *general* attention (Luong et al., 2015). The attention mechanism allows the decoder to consult the entire set of hidden states computed by the encoder; at each time-step—for the generation of each word in sentence S_{i+1}—the decoder determines which words in sentence S_i are relevant, and accordingly selects a linear combination of the entire set of hidden states. In order to do so, we first compute an attention vector $\mathbf{a}_t$, which attributes a weight to each hidden state $\hat{\mathbf{h}}_i$ yielded by the encoder (based on the decoder's current hidden state $\mathbf{h}_t$). according to equation 5:

$$\mathbf{a}_t(i) = \frac{\exp(\text{score}(\mathbf{h}_t, \hat{\mathbf{h}}_i))}{\sum_{i'} \exp(\text{score}(\mathbf{h}_t, \hat{\mathbf{h}}_{i'}))} \quad (5)$$

where

$$\text{score}(\mathbf{h}_t, \hat{\mathbf{h}}_i) = \mathbf{h}_t^T \mathbf{W}_\mathbf{a} \hat{\mathbf{h}}_i \quad (6)$$

The next step is to compute a global context vector $\mathbf{c}_t$, which is a weighted average (based on attention vector $\mathbf{a}_t$) of all of the encoder's hidden states. The resulting context vector is then combined with the original decoder hidden state in order to compute a new, attention-enhanced hidden state $\tilde{\mathbf{h}}_t$.

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_\mathbf{c}[\mathbf{c}_t; \mathbf{h}_t]) \quad (7)$$

where $[\cdot;\cdot]$ represents vector concatenation. Finally, this resulting hidden state $\tilde{\mathbf{h}}_t$ is transformed into a probability distribution $p(\mathbf{w}^t|w^{<t}, S_i)$ over the entire vocabulary using a softmax layer.

$$p(\mathbf{w}^t|w^{<t}, S_i) = \text{softmax}(\mathbf{W}_\mathbf{s}\tilde{\mathbf{h}}_t) \quad (8)$$

As an objective function, the sum of the log-probabilities of the next sentence is optimized, conditioned on the hidden state representation of the current sentence.

$$J_t = \sum_{(S_i, S_{i+1}) \in C} -\log p(S_i|S_{i+1}) \quad (9)$$

At inference time, for the generation of a verse, each word is then sampled randomly according to the output probability distribution. Crucially, the decoder is trained to predict the next sentence in reverse, such that the last word of the verse is the first one that is generated. This reverse operation is important for an effective incorporation of rhyme, as will be explained in the next section. A graphical representation of the architecture, which includes the constraints discussed below, is given in Figure 1.

3.2 Poetic constraints as *a priori* distributions

As the neural architecture described above is trained on generic text, its output will in no way resemble poetic verse. In order to endow the generated output with a certain poetic character, we modify the neural network's output probability distribution through the application of a prior probability distribution, that constrains the standard output probability distribution, and boosts the probability of words that are a good fit within the defined constraints. We will consider two kinds of constraints: a rhyme constraint and a topical constraint.

3.2.1 Rhyme constraint

In order to adequately model the rhyme constraint, we make use of a phonetic representation of words, extracted from the online dictionary *Wiktionary*.[1] For each word of the vocabulary, we determine its rhyme sound (i.e. the final group of vowels, optionally followed by a group of consonants), as well as the group of consonants that precedes the group of vowels. A sample of rhymes that are thus extracted is represented in Table 1.

word	rhyme
embrace	(mbɹ, eɪs)
suitcase	(tk, eɪs)
sacrifice	(f, aɪs)
paradise	(d, aɪs)
reproduit	(dɥ, i)
thérapie	(p, i)
examen	(m, ɛ̃)
canadien	(dj, ɛ̃)

Table 1: A number of rhyme examples extracted from *Wiktionary*, for both English and French.

The next step then consists in creating a probability distribution for a particular rhyme sound that we want the verse to adhere to:

$$p_{rhyme}(\mathbf{w}) = \frac{1}{Z}\mathbf{x} \text{ with } \begin{cases} x_i = 1 & \text{if } i \in R \\ x_i = \epsilon & \text{otherwise} \end{cases} \quad (10)$$

where R is the set of words that contain the required rhyme sound, ϵ is a small value close to zero, used for numerical stability, and Z is a normalization factor in order to ensure a probability distribution. We can now use $p_{rhyme}(\mathbf{w})$ as a prior probability distribution in order to reweight the neural network's standard output probability distribution—according to Equation 11—each time the rhyme scheme demands it:

$$p_{out}(\mathbf{w}) = \frac{1}{Z}(p(\mathbf{w}^t|w^{<t}, S_i) \odot p_{rhyme}(\mathbf{w})) \quad (11)$$

where $\odot$ represents pointwise multiplication.[2] As we noted before, each verse is generated in reverse; the reweighting of rhyme words is applied at the first step of the decoding process, and the rhyme word is generated first. This prevents the generation of an ill-chosen rhyme word that does not fit well with the rest of the verse.

3.2.2 Topical constraint

For the modeling of topical coherence, we make use of a latent semantic model based on non-negative matrix factorization (NMF; Lee & Seung, 2001). Previous research has shown that non-negative factorization methods are able to induce clear-cut, interpretable topical dimensions (Murphy et al., 2012). As input to the method, we construct a frequency matrix $\mathbf{A}$, which captures co-occurrence frequencies of vocabulary words and context words.[3] This matrix is then factorized into two non-negative matrices $\mathbf{W}$ and $\mathbf{H}$,

$$\mathbf{A}_{i\times j} \approx \mathbf{W}_{i\times k}\mathbf{H}_{k\times j} \quad (12)$$

where k is much smaller than i, j so that both instances and features are expressed in terms of a few components. Non-negative matrix factorization enforces the constraint that all three matrices must be non-negative, so all elements must be greater than or equal to zero. Using the minimization of the Kullback-Leibler divergence as an objective function, we want to find the matrices $\mathbf{W}$ and $\mathbf{H}$ for which the divergence between $\mathbf{A}$ and $\mathbf{WH}$ (the multiplication of $\mathbf{W}$ and $\mathbf{H}$) is the smallest. The factorization is carried out through the iterative application of update rules. Matrices $\mathbf{W}$ and $\mathbf{H}$ are randomly initialized, and the rules in 13 and 14 are iteratively applied—alternating between them. In each iteration, each vector is adequately normalized, so that all dimension values sum to 1.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu}\frac{\sum_i \mathbf{W}_{ia}\frac{\mathbf{A}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_k \mathbf{W}_{ka}} \quad (13)$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia}\frac{\sum_\mu \mathbf{H}_{a\mu}\frac{\mathbf{A}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_v \mathbf{H}_{av}} \quad (14)$$

[1] www.wiktionary.org

[2] Such a multiplicative combination of probability distributions is also known as a Product of Experts (Hinton, 2002).

[3] The raw frequencies are weighted using *pointwise mutual information* (Turney and Pantel, 2010).

Tables 2 and 3 present a number of example dimensions induced by the model, for both English and French.

dim 13	dim 22	dim 28
sorrow	railway	planets
longing	trains	planet
admiration	rail	cosmic
earnest	station	universe

Table 2: Three example dimensions from the NMF model for English (4 words with highest probability)

dim 1	dim 20	dim 25
tendresse	gare	hypocrisie
joie	bus	mensonge
bonheur	métro	accuser
sourires	rer	hypocrite

Table 3: Three example dimensions from the NMF model for French (4 words with highest probability)

The factorization that comes out of the NMF model can be interpreted probabilistically (Gaussier and Goutte, 2005; Ding et al., 2008): matrix $\mathbf{W}$ can be considered as $p(\mathbf{w}|k)$, i.e. the probability of a word given a latent dimension k. In order to constrain the network's output to a certain topic, it would be straightforward to simply use $p(\mathbf{w}|k)$ as another prior probability distribution applied to each output. Initial experiments, however, indicated that such a blind modification of the output probability distribution for every word of the output sequence is detrimental to syntactic fluency. In order to combine syntactic fluency with topical consistency, we therefore condition the weighting of the output probability distribution on the entropy of that distribution: when the output distribution's entropy is low, the neural network is certain of its choice for the next word in order to generate a well-formed sentence, so we will not change it. On the other hand, when the entropy is high, we will modify the distribution by using the topical distribution $p(\mathbf{w}|k)$ for a particular latent dimension as prior probability distribution—analogous to Equation 11—in order to inject the desired topic. The entropy threshold, above which the modified distribution is used, is set experimentally.

Note that the rhyme constraint and the topical constraint can straightforwardly be combined in order to generate a topical rhyme word, through pairwise multiplication of the three relevant distributions, and subsequent normalization in order to ensure a probability distribution.

3.3 A global optimization framework

The generation of a verse is embedded within a global optimization framework. There are two reasons to integrate the generation of a verse within an optimization procedure. First of all, the generation of a verse is a sampling process, which is subject to chance. The optimization framework allows us to choose the best sample according to the constraints presented above. Secondly, the optimization allows us to define a number of additional criteria, that assist in the selection of the best verse. For each final verse, the model generates a considerable number of candidates; each candidate verse is then scored according to the following criteria:

- the log-probability score of the generated verse, according to the encoder-decoder architecture (section 3.1);
- compliance with the rhyme constraint (section 3.2.1); additionally, the extraction of the preceding group of consonants (cf. Table 1) allows us to give a higher score to rhyme words with disparate preceding consonant groups, which elicits more interesting rhymes;
- compliance with the topical constraint (section 3.2.2); the score is modeled as the sum of the probabilities of all words for the defined dimension;
- the optimal number of syllables, modeled as a Gaussian distribution with mean μ and standard deviation σ;[4]
- the log-probability score of a standard n-gram model.

The score for each criterion is normalized to the interval $[0, 1]$ using *min-max* normalization, and the harmonic mean of all scores is taken as the final score for each candidate.[5] After generation of a predefined number of candidates, we keep the candidate with the highest score, and append it to the poem.

[4] We equally experimented with rhythmic constraints based on meter and stress, but initial experiments indicated that the system had a tendency to output very rigid verse. Simple syllable counting tends to yield more interesting variation.

[5] The harmonic mean is computed as $\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$; we choose this measure in order to balance the different scores.

4 Results and evaluation

4.1 Implementational details

We train two different models for the generation of poetry in both English and French. The neural architecture is trained on a large corpus of generic web texts, constructed on the basis of the CommonCrawl corpus.[6] In order to filter out noise and retain clean, orderly training data, we apply the following filtering steps:

- we only keep sentences written in the relevant language;
- we only keep sentences of up to 20 words;
- we only keep sentences that contain at least one function word from a predefined list—the idea again is to filter out noisy sentences, and only keep well-formed, grammatical ones; we create a list of about 10 highly frequent function words, specific to each language;
- of all the sentences that remain after these filtering steps, we only keep the ones that appear successively within a document.

Using the filtering steps laid out above, we construct a training corpus of 500 million words for each language. We employ a vocabulary of 15K words (those with highest frequency throughout the corpus); less frequent words are replaced by an `<unk>` token, the probability of which is set to zero during generation.

Both encoder and decoder are made up of two GRU layers with a hidden state of size 2048, and the word embeddings are of size 512. Encoder, decoder, and output embeddings are all shared (Press and Wolf, 2017). Model parameters are optimized using stochastic gradient descent with an initial learning rate of 0.2, which is divided by 4 when the loss does no longer improve on a held-out validation set. We use a batch size of 64, and we apply gradient clipping. The neural architecture has been implemented using PyTorch (Paszke et al., 2017), with substantial reliance on the OpenNMT module (Klein et al., 2017). For the application of the topical constraint, we use an entropy threshold of 2.70.

The n-gram model is a standard Kneser-Ney smoothed trigram model implemented using *KenLM* (Heafield, 2011), and the NMF model is factorized to 100 dimensions. Both the n-gram model and the NMF model are trained on a large, 10 billion word corpus, equally constructed from web texts without any filtering steps except for language identification. For syllable length, we use $\mu = 12, \sigma = 2$.

We generate about 2000 candidates for each verse, according to a fixed rhyme scheme (`ABAB CDCD`). Note that no human selection whatsoever has been applied to the poems used in the evaluation; all poems have been generated in a single run, without *cherry picking* the best examples. Four representative examples of poems generated by the system are given in Figure 2.

4.2 Evaluation procedure

Quantitatively evaluating creativity is far from straightforward, and this is no less true for creative artefacts that are automatically generated. Automatic evaluation measures that compute the overlap of system output with gold reference texts (such as BLEU or ROUGE), and which might be used for the evaluation of standard generation tasks, are of little use when it comes to creative language generation. The majority of research into creative language generation therefore makes use of some form of human evaluation, even though one needs to keep in mind that the evaluation of textual creativity is an inherently subjective task, especially with regard to poetic value. For a discussion of the subject, see Gonçalo Oliveira (2017).

We adopt the evaluation framework by Zhang and Lapata (2014), in which human annotators are asked to evaluate poems on a five point scale with regard to a number of characteristics, viz.

- *fluency*: is the poem grammatical and syntactically well-formed?
- *coherence*: is the poem thematically structured?
- *meaningfulness*: does the poem convey a meaningful message to the reader?
- *poeticness*: does the text display the features of a poem?

Additionally, we ask annotators to judge if the poem is written by a human or a computer.

In total, we evaluate four different sets of poems, yielded by different model instantiations. The different sets of poems considered during evaluation are:

[6] `commoncrawl.org`

At the moment it seems almost impossible
Yet life is neither good nor evil
The divine mind and soul is immortal
In other words, the soul is never ill

So far, it has barely lost its youthful look
But no man is ever too young for the rest
He thought deeply, and yet his heart shook
At that moment he seemed utterly possessed

~

The moon represents unity and brotherhood
The earth stands in awe and disbelief
Other planets orbit the earth as they should
The universe is infinite and brief

The sky has been so bright and beautiful so far
See the moon shining through the cosmic flame
See the stars in the depths of the earth you are
The planet the planet we can all see the same

Malgré mon enthousiasme, le chagrin s'allonge
Le bonheur est toujours superbe
Toi, tu es un merveilleux songe
Je te vois rêver de bonheur dans l'herbe

Tu trouveras le bonheur de tes rêves
Je t'aime comme tout le monde
Je t'aime mon amour, je me lève
Je ressens pour toi une joie profonde

~

Rien ne prouve qu'il s'indigne
Dans le cas contraire, ce n'est pas grave
Si la vérité est fausse, c'est très mauvais signe
Il est vrai que les gens le savent

Et cela est faux, mais qu'importe
En fait, le mensonge, c'est l'effroi
La négation de l'homme en quelque sorte
Le tort n'est pas de penser cela, il est magistrat

Figure 2: Four representative examples of poems generated by the system; the left-hand poems, in English, are respectively generated using dimensions 13 and 28 (cf. Table 2); the right-hand poems, in French, are generated using dimensions 1 and 25 (cf. Table 3).

- *rnn*: poems generated by the neural architecture defined in section 3.1, without any added constraints;
- *rhyme*: poems generated by the neural architecture, augmented with the rhyme constraint;
- nmf_{rand}: poems generated by the neural architecture, augmented with both the rhyme constraint and the topical constraint, where one of the automatically induced NMF dimensions is selected randomly;
- nmf_{spec}: poems generated by the neural architecture, augmented with both the rhyme constraint and the topical constraint, where one of the automatically induced NMF dimensions is specified manually.[7]

For a proper comparison of our system, we equally include:

- *random*: poems yielded by a baseline model where, for each verse, we select a random sentence (that contains between 7 and 15 words) from a large corpus; the idea is that the lines selected by the baseline model should be fairly fluent (as they come from an actual corpus), but lacking in coherence (due to their random selection);
- *human*: poems written by human poets; the scores on this set of poems function as an upper bound;
- *Hafez* and *Deep-speare*: poems generated by two state of the art poetry generation systems for English, respectively by Ghazvininejad et al. (2016) and Lau et al. (2018); we use the code made available by the respective authors.[8] Note that we only compare to other poetry generation systems for English, as no other readily available systems exist for French.

4.3 Results for English

For English, 22 annotators evaluated 40 poems in total (5 poems for each of the different sets considered in the evaluation; each poem was evaluated by at least 4 annotators). The annotators consist of native speakers of English, as well as master students in English linguistics and literature. For the *human* set, we select five poems by well-established English poets that follow the same rhyme scheme as the generated ones.[9] For nmf_{spec}, we select dimension 13 of Table 2. The results of the evaluation for English are presented in the upper part of Table 4.

First of all, note that all our model instantiations score better than the *random* baseline model,

[7] This can be regarded as manually defining the theme of the generated poem. The specified dimension is selected for its poetic character.

[8] *Hafez* needs to be initialized with user-defined topics; for a fair comparison, we seed the system with the top words of the NMF dimension used for our best performing model.

[9] The selected poets are W.H. Auden, E.E. Cummings, Philip Larkin, Sarojini Naidu, and Sylvia Plath.

English					
model	fluency	coherence	meaningfulness	poeticness	written by human (%)
rnn	2.95	2.50	2.45	2.55	0.18
rhyme	3.41	2.77	2.82	2.95	**0.59**
nmf_{rand}	3.32	3.09	2.86	2.95	0.32
nmf_{spec}	**3.64**	**3.41**	**3.27**	**3.86**	0.55
random	2.68	2.09	1.91	2.41	0.14
Deep-speare	2.11	2.00	2.00	3.00	0.22
Hafez	3.44	3.11	3.11	3.50	0.53
human	3.73	3.73	3.68	4.00	0.73
French					
model	fluency	coherence	meaningfulness	poeticness	written by human (%)
rnn	3.45	2.73	2.59	2.55	0.27
rhyme	**3.82**	2.55	2.18	3.23	0.14
nmf_{rand}	3.64	3.32	3.09	2.86	0.27
nmf_{spec}	**3.82**	**3.82**	**3.55**	**3.95**	**0.45**
random	2.95	1.86	1.68	2.18	0.00
human	4.59	4.59	4.50	4.81	0.95

Table 4: Results of the human evaluation (mean score of all annotators) for English and French; values in **bold** indicate best performance of all generation models

even with regard to grammatical fluency. The good scores on fluency for the constrained models indicate that the applied constraints do not disrupt the grammaticality of the generated verse (*rhyme* is significantly different[10] with $p < 0.05$; nmf_{rand} and nmf_{spec} with $p < 0.01$; recall that the baseline consists of actual sentences from a corpus). Secondly, we note that the rhyme constraint seems to improve poeticness (though not significantly), while the topical constraint seems to improve both coherence ($p < 0.01$ for nmf_{spec}) and meaningfulness (not significantly). Interestingly, a large proportion of the poems produced by the *rhyme* model are labeled as human, even though the other scores are fairly low. The score for poeticness is considerably higher ($p < 0.01$) for nmf_{spec} (with a manually specified theme selected for its poeticness) than for nmf_{rand} (with a randomly selected topic, which will often be more mundane). And the best scores on all criteria are obtained with the nmf_{spec} model, for which more than half of the poems are judged to be written by a human; moreover, the difference between nmf_{spec} and human poetry is not significant. Finally, our poetry generation compares favourably to previous work: nmf_{spec} scores markedly and significantly better than *Deep-speare* (which does not differ significantly from the random baseline), and it equally attains better scores than *Hafez* on all four criteria (though not significantly so).

[10] Significance testing is carried out using a two-tailed permutation test.

4.4 Results for French

The setup of the French evaluation is analogous to the English one: an equal number of 22 annotators have evaluated a total of 30 poems (5 poems for each of the six sets considered in the evaluation; each poem was evaluated by at least 4 annotators). The annotators are all native speakers of French. For the human poems, we select five poems with the same rhyme scheme as the generated ones, among the highest ranked ones on `short-edition.com`—a website with submissions by amateur poets. For nmf_{spec}, we select dimension 1 of Table 3. The results for French are presented in the lower part of Table 4.

Generally speaking, we see that the results for French confirm those for English. First of all, all model instantiations obtain better scores than the *random* baseline model, even with regard to fluency ($p < 0.01$), again confirming that the application of the rhyme constraint and topical constraint are not detrimental to the grammaticality of the verse. Secondly, the rhyme constraint significantly improves the score for poeticness ($p < 0.05$ compared to *rnn*), while the topical constraint improves both coherence ($p < 0.05$) and meaningfulness ($p < 0.01$). Contrary to the English results, only a small proportion of poems from the *rhyme* model are thought to be human. We do again see that the score for poeticness is considerably higher ($p < 0.01$) for

nmf_{spec} than for nmf_{rand}, which seems to indicate that the topic of a poem is an important factor in people's judgements on poeticness. Finally, we again see that the best scores on all criteria are obtained with nmf_{spec}, for which almost half of the poems are judged to be written by a human.

5 Conclusion

We presented a system for automatic poetry generation that is trained exclusively on standard, non-poetic text. The system uses a recurrent neural encoder-decoder architecture in order to generate candidate verses, incorporating poetic and topical constraints by modifying the output probability distribution of the neural network. The best verse is then selected for inclusion in the poem, using a global optimization framework. We trained the system on both English and French, and equally carried out a human evaluation for both languages. The results indicate that the system is able to generate credible poetry, that scores well with regard to fluency and coherence, as well as meaningfulness and poeticness. Compared to previous systems, our model achieves state of the art performance, even though it is trained on standard, non-poetic text. In our best setup, about half of the generated poems are judged to be written by a human.

We conclude with a number of future research avenues. First of all, we would like to experiment with different neural network architectures. Specifically, we believe hierarchical approaches (Serban et al., 2017) as well as the Transformer network (Vaswani et al., 2017) would be particularly suitable to poetry generation. Secondly, we would like to incorporate further poetic devices, especially those based on meaning. Gripping poetry often relies on figurative language use, such as symbolism and metaphor. A successful incorporation of such devices would mean a significant step towards truly inspired poetry generation. And finally, we would like to adapt the model for automatic poetry translation—as we feel that the constraint-based approach lends itself perfectly to a poetry translation model that is able to adhere to an original poem in both form and meaning.

In order to facilitate reproduction of the results and encourage further research, the poetry generation system is made available as open source software. The current version can be downloaded at `https://github.com/timvdc/poetry`.

Acknowledgements

This work is supported by a grant overseen by the French National Research Agency ANR (project QUANTUM – ANR-19-CE23-0025); it has equally benefited from a GPU donated by NVIDIA Corporation.

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Chris Ding, Tao Li, and Wei Peng. 2008. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927.

Eric Gaussier and Cyril Goutte. 2005. Relation between plsa and nmf and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602. ACM.

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

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

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Hugo Gonçalo Oliveira. 2017. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 524–533. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the*

EMNLP 2011 Sixth Workshop on Statistical Machine Translation, pages 187–197, Edinburgh, Scotland, United Kingdom.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958. Association for Computational Linguistics.

Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562.

Zhiqiang Liu, Zuohui Fu, Jie Cao, Gerard de Melo, Yik-Cheung Tam, Cheng Niu, and Jie Zhou. 2019. Rhetorically controlled encoder-decoder for modern Chinese poetry generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1992–2001, Florence, Italy. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Ruli Manurung, Graeme Ritchie, and Henry Thompson. 2012. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950. The COLING 2012 Organizing Committee.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Tony Veale. 2013. Less rhyme, more reason: Knowledge-based poetry generation with feeling, insight and wit. In *Proceedings of the international conference on computational creativity*, pages 152–159.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016. Chinese song iambics generation with neural attention-based model. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2943–2949.

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

Cheng Yang, Maosong Sun, Xiaoyuan Yi, and Wenhao Li. 2018. Stylistic Chinese poetry generation via unsupervised style disentanglement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3960–3969, Brussels, Belgium. Association for Computational Linguistics.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic poetry generation with mutual reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153, Brussels, Belgium. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680. Association for Computational Linguistics.