Chinese Poetry Generation with Recurrent Neural Networks

Chinese Poetry Generation with Recurrent Neural Networks

Xingxing Zhang and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
x.zhang@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

We propose a model for Chinese poem generation based on recurrent neural networks which we argue is ideally suited to capturing poetic content and form. Our generator *jointly* performs content selection ("what to say") and surface realization ("how to say") by learning representations of individual characters, and their combinations into one or more lines as well as how these mutually reinforce and constrain each other. Poem lines are generated incrementally by taking into account the entire history of what has been generated so far rather than the limited horizon imposed by the previous line or lexical *n*-grams. Experimental results show that our model outperforms competitive Chinese poetry generation systems using both automatic and manual evaluation methods.

相 思
Missing You
红 豆 生 南 国， (* Z P P Z)
Red berries born in the warm southland.
春 来 发 几 **枝**？ (P P Z Z P)
How many branches flush in the spring?
愿 君 多 采 撷， (* P P Z Z)
Take home an armful, for my sake,
此 物 最 相 **思**。 (* Z Z P P)
As a symbol of our love.

Table 1: An example of a 5-char *quatrain* exhibiting one of the most popular tonal patterns. The tone of each character is shown at the end of each line (within parentheses); P and Z are shorthands for *Ping* and *Ze* tones, respectively; * indicates that the tone is not fixed and can be either. Rhyming characters are shown in boldface.

1 Introduction

Classical poems are a significant part of China's cultural heritage. Their popularity manifests itself in many aspects of everyday life, e.g., as a means of expressing personal emotion, political views, or communicating messages at festive occasions as well as funerals. Amongst the many different types of classical Chinese poetry, *quatrain* and *regulated verse* are perhaps the best-known ones. Both types of poem must meet a set of structural, phonological, and semantic requirements, rendering their composition a formidable task left to the very best scholars.

An example of a quatrain is shown in Table 1. Quatrains have four lines, each five or seven characters long. Characters in turn follow specific phonological patterns, within each line and across lines. For instance, the final characters in the second, fourth and (optionally) first line must rhyme, whereas there are no rhyming constraints for the third line. Moreover, poems must follow a prescribed tonal pattern. In traditional Chinese, every character has one tone, *Ping* (level tone) or *Ze* (downward tone). The poem in Table 1 exemplifies one of the most popular tonal patterns (Wang, 2002). Besides adhering to the above formal criteria, poems must exhibit concise and accurate use of language, engage the reader/hearer, stimulate their imagination, and bring out their feelings.

In this paper we are concerned with generating traditional Chinese poems automatically. Although computers are no substitute for poetic creativity, they can analyze very large online text repositories of poems, extract statistical patterns, maintain them in memory and use them to generate many possible variants. Furthermore, while amateur poets may struggle to remember and apply formal tonal and structural constraints, it is relatively straightforward for the machine to check

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 670–680,
October 25-29, 2014, Doha, Qatar.

whether a candidate poem conforms to these requirements. Poetry generation has received a fair amount of attention over the past years (see the discussion in Section 2), with dozens of computational systems written to produce poems of varying sophistication. Beyond the long-term goal of building an autonomous intelligent system capable of creating meaningful poems, there are potential short-term applications for computer generated poetry in the ever growing industry of electronic entertainment and interactive fiction as well as in education. An assistive environment for poem composition could allow teachers and students to create poems subject to their requirements, and enhance their writing experience.

We propose a model for Chinese poem generation based on recurrent neural networks. Our generator *jointly* performs content selection ("what to say") and surface realization ("how to say"). Given a large collection of poems, we learn representations of individual characters, and their combinations into one or more lines as well as how these mutually reinforce and constrain each other. Our model generates lines in a poem probabilistically: it estimates the probability of the current line given the probability of all previously generated lines. We use a recurrent neural network to learn the representations of the lines generated so far which in turn serve as input to a recurrent language model (Mikolov et al., 2010; Mikolov et al., 2011b; Mikolov et al., 2011a) which generates the current line. In contrast to previous approaches (Greene et al., 2010; Jiang and Zhou, 2008), our generator makes no Markov assumptions about the dependencies of the words within a line and across lines.

We evaluate our approach on the task of quatrain generation (see Table 1 for a human-written example). Experimental results show that our model outperforms competitive Chinese poetry generation systems using both automatic and manual evaluation methods.

2 Related Work

Automated poetry generation has been a popular research topic over the past decades (see Colton et al. (2012) and the references therein). Most approaches employ templates to construct poems according to a set of constraints (e.g., rhyme, meter, stress, word frequency) in combination with corpus-based and lexicographic resources. For example, the Haiku poem generator presented in Wu et al. (2009) and Tosa et al. (2008) produces poems by expanding user queries with rules extracted from a corpus and additional lexical resources. Netzer et al. (2009) generate Haiku with Word Association Norms, Agirrezabal et al. (2013) compose Basque poems using patterns based on parts of speech and WordNet (Fellbaum, 1998), and Oliveira (2012) presents a generation algorithm for Portuguese which leverages semantic and grammar templates.

A second line of research uses genetic algorithms for poem generation (Manurung, 2003; Manurung et al., 2012; Zhou et al., 2010). Manurung et al. (2012) argue that at a basic level all (machine-generated) poems must satisfy the constraints of grammaticality (i.e., a poem must syntactically well-formed), meaningfulness (i.e., a poem must convey a message that is meaningful under some interpretation) and poeticness (i.e., a poem must exhibit features that distinguishes it from non-poetic text, e.g., metre). Their model generates several candidate poems and then uses stochastic search to find those which are grammatical, meaningful, and poetic.

A third line of research draws inspiration from statistical machine translation (SMT) and related text-generation applications such as summarization. Greene et al. (2010) infer meters (stressed/unstressed syllable sequences) from a corpus of poetic texts which they subsequently use for generation together with a cascade of weighted finite-state transducers interpolated with IBM Model 1. Jiang and Zhou (2008) generate Chinese couplets (two line poems) using a phrase-based SMT approach which translates the first line to the second line. He et al. (2012) extend this algorithm to generate four-line quatrains by sequentially translating the current line from the previous one. Yan et al. (2013) generate Chinese quatrains based on a query-focused summarization framework. Their system takes a few keywords as input and retrieves the most relevant poems from a corpus collection. The retrieved poems are segmented into their constituent terms which are then grouped into clusters. Poems are generated by iteratively selecting terms from clusters subject to phonological, structural, and coherence constraints.

Our approach departs from previous work in two important respects. Firstly, we model the tasks of surface realization and content selection jointly

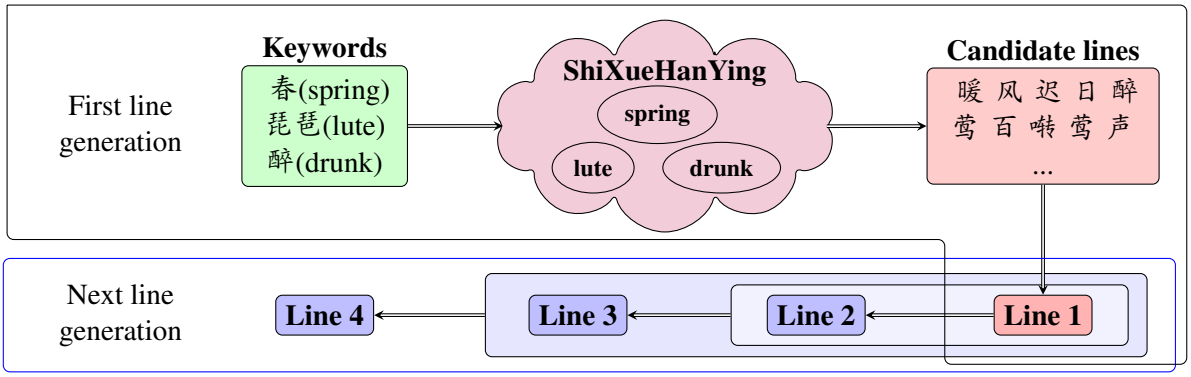

Figure 1: Poem generation with keywords *spring*, *lute*, and *drunk*. The keywords are expanded into phrases using a poetic taxonomy. Phrases are then used to generate the first line. Following lines are generated by taking into account the representations of all previously generated lines.

using recurrent neural networks. Structural, semantic, and coherence constraints are captured naturally in our framework, through learning the representations of individual characters and their combinations. Secondly, generation proceeds by taking into account multi-sentential context rather than the immediately preceding sentence. Our work joins others in using continuous representations to express the meaning of words and phrases (Socher et al., 2012; Mikolov et al., 2013) and how these may be combined in a language modeling context (Mikolov and Zweig, 2012). More recently, continuous translation models based on recurrent neural networks have been proposed as a means to map a sentence from the source language to sentences in the target language (Auli et al., 2013; Kalchbrenner and Blunsom, 2013). These models are evaluated on the task of rescoring n-best lists of translations. We use neural networks more directly to perform the actual poem generation task.

3 The Poem Generator

As common in previous work (Yan et al., 2013; He et al., 2012) we assume that our generator operates in an interactive context. Specifically, the user supplies keywords (e.g., *spring*, *lute*, *drunk*) highlighting the main concepts around which the poem will revolve. As illustrated in Figure 1, our generator expands these keywords into a set of related phrases. We assume the keywords are restricted to those attested in the *ShiXueHanYing* poetic phrase taxonomy (He et al., 2012; Yan et al., 2013). The latter contains 1,016 manual clusters of phrases (Liu, 1735); each cluster is labeled with a keyword id describing general poem-worthy topics. The generator creates the first line of the poem based on these keywords. Subsequent lines are generated based on all previously generated lines, subject to phonological (e.g., admissible tonal patterns) and structural constraints (e.g., whether the quatrain is five or seven characters long).

To create the first line, we select all phrases corresponding to the user's keywords and generate all possible combinations satisfying the tonal pattern constraints. We use a language model to rank the generated candidates and select the best-ranked one as the first line in the poem. In implementation, we employ a character-based recurrent neural network language model (Mikolov et al., 2010) interpolated with a Kneser-Ney trigram and find the n-best candidates with a stack decoder (see Section 3.5 for details). We then generate the second line based on the first one, the third line based on the first two lines, and so on. Our generation model computes the probability of line $S_{i+1} = w_1, w_2, \ldots, w_m$, given all previously generated lines $S_{1:i} (i \geq 1)$ as:

$$P(S_{i+1}|S_{1:i}) = \prod_{j=1}^{m-1} P(w_{j+1}|w_{1:j}, S_{1:i}) \qquad (1)$$

Equation (1), decomposes $P(S_{i+1}|S_{1:i})$ as the product of the probability of each character w_j in the current line given all previously generated characters $w_{1:j-1}$ and lines $S_{1:i}$. This means that $P(S_{i+1}|S_{1:i})$ is sensitive to previously generated content and currently generated characters.

The estimation of the term $P(w_{j+1}|w_{1:j}, S_{1:i})$ lies at the heart of our model. We learn representations for $S_{1:i}$, the context generated so far, using a recurrent neural network whose output

serves as input to a second recurrent neural network used to estimate $P(w_{j+1}|w_{1:j}, S_{1:i})$. Figure 2 illustrates the generation process for the $(j+1)$th character w_{j+1} in the $(i+1)$th line S_{i+1}. First, lines $S_{1:i}$ are converted into vectors $v_{1:i}$ with a *convolutional sentence model* (CSM; described in Section 3.1). Next, a *recurrent context model* (RCM; see Section 3.2) takes $v_{1:i}$ as input and outputs u_i^j, the representation needed for generating $w_{j+1} \in S_{i+1}$. Finally, $u_i^1, u_i^2, \ldots, u_i^j$ and the first j characters $w_{1:j}$ in line S_{i+1} serve as input to a *recurrent generation model* (RGM) which estimates $P(w_{j+1} = k|w_{1:j}, S_{1:i})$ with $k \in V$, the probability distribution of the $(j+1)$th character over all words in the vocabulary V. More formally, to estimate $P(w_{j+1}|w_{1:j}, S_{1:i})$ in Equation (1), we apply the following procedure:

$$v_i = \text{CSM}(S_i) \tag{2a}$$

$$u_i^j = \text{RCM}(v_{1:i}, j) \tag{2b}$$

$$P(w_{j+1}|w_{1:j}, S_{1:i}) = \text{RGM}(w_{1:j+1}, u_i^{1:j}) \tag{2c}$$

We obtain the probability of the $(i+1)$th sentence $P(S_{i+1}|S_{1:i})$, by running the RGM in (2c) above $m-1$ times (see also Equation (1)). In the following, we describe how the different components of our model are obtained.

3.1 Convolutional Sentence Model (CSM)

The CSM converts a poem line into a vector. In principle, any model that produces vector-based representations of phrases or sentences could be used (Mitchell and Lapata, 2010; Socher et al., 2012). We opted for the convolutional sentence model proposed in Kalchbrenner and Blunsom (2013) as it is n-gram based and does not make use of any parsing, POS-tagging or segmentation tools which are not available for Chinese poems. Their model computes a continuous representation for a sentence by sequentially merging neighboring vectors (see Figure 3).

Let V denote the character vocabulary in our corpus; $L \in \mathbb{R}^{q \times |V|}$ denotes a character embedding matrix whose columns correspond to character vectors (q represents the hidden unit size). Such vectors can be initialized randomly or obtained via a training procedure (Mikolov et al., 2013). Let w denote a character with index k; $e(w) \in \mathbb{R}^{|V| \times 1}$ is a vector with zero in all positions except $e(w)_k = 1$; $T^l \in \mathbb{R}^{q \times N^l}$ is the sentence representation in the lth layer, where N^l is the number of columns in the lth layer ($N^l = 1$ in the top layer); $C^{l,n} \in \mathbb{R}^{q \times n}$ is an array of weight matrices which compress neighboring n columns in the lth layer to one column in the $(l+1)$th layer. Given a sentence $S = w_1, w_2, \ldots, w_m$, the first layer is represented as:

$$\begin{aligned} T^1 &= [L \cdot e(w_1), L \cdot e(w_2), \ldots, L \cdot e(w_m)] \\ N^1 &= m \end{aligned} \tag{3}$$

The $(l+1)$th layer is then computed as follows:

$$\begin{aligned} T^{l+1}_{:,j} &= \sigma\Big(\sum_{i=1}^{n} T^l_{:,j+i-1} \odot C^{l,n}_{:,i}\Big) \\ N^{l+1} &= N^l - n + 1 \\ 1 &\le j \le N^{l+1} \end{aligned} \tag{4}$$

where T^l is the representation of the previous layer l, $C^{l,n}$ a weight matrix, $\odot$ element-wise vector product, and σ a non-linear function. We compress two neighboring vectors in the first two layers and three neighboring vectors in the remaining layers. Specifically, for quatrains with seven characters, we use $C^{1,2}$, $C^{2,2}$, $C^{3,3}$, $C^{4,3}$ to merge vectors in each layer (see Figure 3); and for quatrains with five characters we use $C^{1,2}$, $C^{2,2}$, $C^{3,3}$.

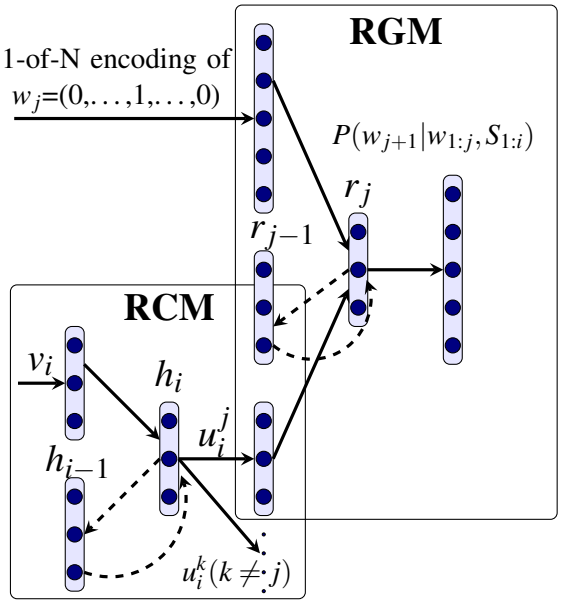

Figure 2: Generation of the $(j+1)$th character w_{j+1} in the $(i+1)$th line S_{i+1}. The recurrent context model (RCM) takes i lines as input (represented by vectors $v_1, \ldots, v_i$) and creates context vectors for the recurrent generation model (RGM). The RGM estimates the probability $P(w_{j+1}|w_{1:j}, S_{1:i})$.

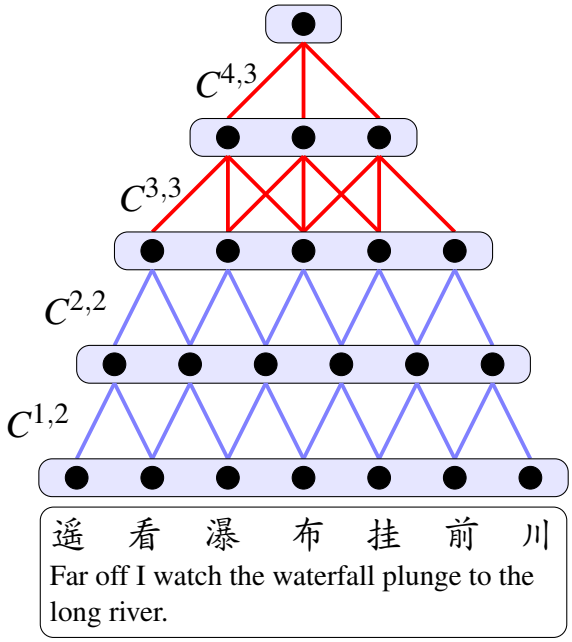

Figure 3: Convolutional sentence model for 7-char quatrain. The first layer has seven vectors, one for each character. Two neighboring vectors are merged to one vector in the second layer with weight matrix $C^{1,2}$. In other layers, either two or three neighboring vectors are merged.

3.2 Recurrent Context Model (RCM)

The RCM takes as input the vectors representing the i lines generated so far and reduces them to a single context vector which is then used to generate the next character (see Figure 2). We compress the i previous lines to one vector (the hidden layer) and then decode the compressed vector to different character positions in the current line. The output layer consists thus of several vectors (one for each position) connected together. This way, different aspects of the context modulate the generation of different characters.

Let $v_1, \ldots, v_i \quad (v_i \in \mathbb{R}^{q\times 1})$ denote the vectors of the previous i lines; $h_i \in \mathbb{R}^{q\times 1}$ is their compressed representation (hidden layer) which is obtained with matrix $M \in \mathbb{R}^{q\times 2q}$; matrix U_j decodes h_i to $u_i^j \in \mathbb{R}^{q\times 1}$ in the $(i+1)$th line. The computation of the RCM proceeds as follows:

$$\begin{aligned} h_0 &= \mathbf{0} \\ h_i &= \sigma(M \cdot \begin{bmatrix} v_i \\ h_{i-1} \end{bmatrix}) \\ u_i^j &= \sigma(U_j \cdot h_i) \qquad 1 \le j \le m-1 \end{aligned} \tag{5}$$

where σ is a non-linear function such as sigmoid and m the line length. Advantageously, lines in classical Chinese poems have a fixed length of five or seven characters. Therefore, the output layer of the recurrent context model only needs two weight matrices (one for each length) and the number of parameters still remains tractable.

3.3 Recurrent Generation Model (RGM)

As shown in Figure 2, the RGM estimates the probability distribution of the next character (over the entire vocabulary) by taking into account the context vector provided by the RCM and the 1-of-N encoding of the previous character. The RGM is essentially a recurrent neural network language model (Mikolov et al., 2010) with an auxiliary input layer, i.e., the context vector from the RCM. Similar strategies for encoding additional information have been adopted in related language modeling and machine translation work (Mikolov and Zweig, 2012; Kalchbrenner and Blunsom, 2013; Auli et al., 2013).

Let $S_{i+1} = w_1, w_2, \ldots, w_m$ denote the line to be generated. The RGM must estimate $P(w_{j+1}|w_{1:j}, S_{1:i})$, however, since the first i lines have been encoded in the context vector u_i^j, we compute $P(w_{j+1}|w_{1:j}, u_i^j)$ instead. Therefore, the probability $P(S_{i+1}|S_{1:i})$ becomes:

$$P(S_{i+1}|S_{1:i}) = \prod_{j=1}^{m-1} P(w_{j+1}|w_{1:j}, u_i^j) \tag{6}$$

Let $|V|$ denote the size of the character vocabulary. The RGM is specified by a number of matrices. Matrix $H \in \mathbb{R}^{q\times q}$ (where q represents the hidden unit size) transforms the context vector to a hidden representation; matrix $X \in \mathbb{R}^{q\times|V|}$ transforms a character to a hidden representation, matrix $R \in \mathbb{R}^{q\times q}$ implements the recurrent transformation and matrix $Y \in \mathbb{R}^{|V|\times q}$ decodes the hidden representation to weights for all words in the vocabulary. Let w denote a character with index k in V; $e(w) \in \mathbb{R}^{|V|\times 1}$ represents a vector with zero in all positions except $e(w)_k = 1$, r_j is the hidden layer of the RGM at step j, and y_{j+1} the output of the RGM, again at step j. The RGM proceeds as follows:

$$r_0 = \mathbf{0} \tag{7a}$$

$$r_j = \sigma(R \cdot r_{j-1} + X \cdot e(w_j) + H \cdot u_i^j) \tag{7b}$$

$$y_{j+1} = Y \cdot r_j \tag{7c}$$

where σ is a nonlinear function (e.g., sigmoid).

The probability of the $(j+1)$th word given the previous j words and the previous i lines is estimated by a *softmax* function:

$$P(w_{j+1} = k | w_{1:j}, u_i^j) = \frac{\exp(y_{j+1,k})}{\sum_{k=1}^{|V|} \exp(y_{j+1,k})} \quad (8)$$

We obtain $P(S_{i+1}|S_{1:i})$ by multiplying all the terms in the right hand-side of Equation (6).

3.4 Training

The objective for training is the cross entropy errors of the predicted character distribution and the actual character distribution in our corpus. An l_2 regularization term is also added to the objective. The model is trained with back propagation through time (Rumelhart et al., 1988) with sentence length being the time step. The objective is minimized by stochastic gradient descent. During training, the cross entropy error in the output layer of the RGM is back-propagated to its hidden and input layers, then to the RCM and finally to the CSM. The same number of hidden units ($q = 200$) is used throughout (i.e., in the RGM, RCM, and CSM). In our experiments all parameters were initialized randomly, with the exception of the word embedding matrix in the CSM which was initialized with *word2vec* embeddings (Mikolov et al., 2013) obtained from our poem corpus (see Section 4 for details on the data we used).

To speed up training, we employed word-classing (Mikolov et al., 2011b). To compute the probability of a character, we estimate the probability of its class and then multiply it by the probability of the character conditioned on the class. In our experiments we used 82 (square root of $|V|$) classes which we obtained by applying hierarchical clustering on character embeddings. This strategy outperformed better known frequency-based classing methods (Zweig and Makarychev, 2013) on our task.

Our poem generator models content selection and lexical choice and their interaction, but does not have a strong notion of local coherence, as manifested in poetically felicitous line-to-line transitions. In contrast, machine translation models (Jiang and Zhou, 2008) have been particularly successful at generating adjacent lines (couplets). To enhance coherence, we thus interpolate our model with two machine translation features (i.e., inverted phrase translation model feature and inverted lexical weight feature). Also note, that in our model surface generation depends on the last observed character and the state of the hidden layer before this observation. This way, there is no explicitly defined context, and history is captured implicitly by the recurrent nature of the model. This can be problematic for our texts which must obey certain stylistic conventions and sound poetic. In default of a better way of incorporating poeticness into our model, we further interpolate it with a language model feature (i.e., a Kneser-Ney trigram model).

Throughout our experiments, we use the RNNLM toolkit to train the character-based recurrent neural network language model (Mikolov et al., 2010). Kneser-Ney n-grams were trained with KenLM (Heafield, 2011).

3.5 Decoding

Our decoder is a stack decoder similar to Koehn et al. (2003). In addition, it implements the tonal pattern and rhyming constraints necessary for generating well-formed Chinese quatrains. Once the first line in a poem is generated, its tonal pattern is determined. During decoding, phrases violating this pattern are ignored. As discussed in Section 1, the final characters of the second and the fourth lines must rhyme. We thus remove during decoding fourth lines whose final characters do not rhyme with the second line. Finally, we use MERT training (Och, 2003) to learn feature weights for the decoder.

4 Experimental Design

Data We created a corpus of classical Chinese poems by collating several online resources: *Tang Poems*, *Song Poems*, *Song Ci*, *Ming Poems*, *Qing Poems*, and *Tai Poems*. The corpus consists of 284,899 poems in total. 78,859 of these are quatrains and were used for training and evaluating our model.[1] Table 2 shows the different partitions of this dataset (POEMLM) into training (QTRAIN)[2], validation (QVALID) and testing (QTEST). Half of the poems in QVALID and QTEST are 5-char quatrains and the other half are 7-char quatrains. All poems except QVALID

[1]The data used in our experiments can be downloaded from `http://homepages.inf.ed.ac.uk/mlap/index.php?page=resources`.

[2]Singleton characters in QTRAIN (6,773 in total) were replaced by <R> to reduce data sparsity.

	Poems	Lines	Characters
QTRAIN	74,809	299,236	2,004,460
QVALID	2,000	8,000	48,000
QTEST	2,050	8,200	49,200
POEMLM	280,849	2,711,034	15,624,283

Table 2: Dataset partitions of our poem corpus.

Models	Perplexity
KN5	172
RNNLM	145
RNNPG	**93**

Table 3: Perplexities for different models.

and QTEST were used for training the character-based language models (see row POEMLM in Table 2). We also trained *word2vec* embeddings on POEMLM. In our experiments, we generated quatrains following the eight most popular tonal patterns according to Wang (2002).

Perplexity Evaluation Evaluation of machine-generated poetry is a notoriously difficult task. Our evaluation studies were designed to assess Manurung et al.'s (2012) criteria of grammaticality, meaningfulness, and poeticness. As a sanity check, we first measured the perplexity of our model with respect to the goldstandard. Intuitively, a better model should assign larger probability (and therefore lower perplexity) to goldstandard poems.

BLEU-based Evaluation We also used BLEU to evaluate our model's ability to generate the second, third and fourth line given previous goldstandard lines. A problematic aspect of this evaluation is the need for human-authored references (for a partially generated poem) which we do not have. We obtain references automatically following the method proposed in He et al. (2012). The main idea is that if two lines share a similar topic, the lines following them can be each other's references. Let A and B denote two adjacent lines in a poem, with B following A. Similarly, let line B' follow line A' in another poem. If lines A and A' share some keywords in the same cluster in the *Shixuehanying* taxonomy, then B and B' can be used as references for both A and A'. We use this algorithm on the *Tang Poems* section of our corpus to build references for poems in the QVALID and QTEST data sets. Poems in QVALID (with auto-generated references) were used for MERT training and Poems in QTEST (with auto-generated references) were used for BLEU evaluation.

Human Evaluation Finally, we also evaluated the generated poems by eliciting human judgments. Specifically, we invited 30 experts[3] on Chinese poetry to assess the output of our generator (and comparison systems). These experts were asked to rate the poems using a 1–5 scale on four dimensions: *fluency* (is the poem grammatical and syntactically well-formed?), *coherence* (is the poem thematically and logically structured?), *meaningfulness* (does the poem convey a meaningful message to the reader?) and *poeticness* (does the text display the features of a poem?). We also asked our participants to evaluate system outputs by ranking the generated poems relative to each other as a way of determining overall poem quality (Callison-Burch et al., 2012).

Participants rated the output of our model and three comparison systems. These included He et al.'s (2012) SMT-based model (SMT), Yan et al.'s (2013) summarization-based system (SUM), and a random baseline which creates poems by randomly selecting phrases from the *Shixuehanying* taxonomy given some keywords as input. We also included human written poems whose content matched the input keywords. All systems were provided with the same keywords (i.e., the same cluster names in the *ShiXueHanYing* taxonomy). In order to compare all models on equal footing, we randomly sampled 30 sets of keywords (with three keywords in each set) and generated 30 quatrains for each system according to two lengths, namely 5-char and 7-char. Overall, we obtained ratings for 300 ($5\times30\times2$) poems.

5 Results

The results of our perplexity evaluation are summarized in Table 3. We compare our RNN-based poem generator (RNNPG) against Mikolov's (2010) recurrent neural network language model (RNNLM) and a 5-gram language model with Kneser-Ney smoothing (KN5). All models were trained on QTRAIN and tuned on QVALID. The perplexities were computed on QTEST. Note that

[3] 27 participants were professional or amateur poets and three were Chinese literature students who had taken at least one class on Chinese poetry composition.

Models	1 → 2		2 → 3		3 → 4		Average	
	5-char	7-char	5-char	7-char	5-char	7-char	5-char	7-char
SMT	0.0559	0.0906	0.0410	0.1837	0.0547	**0.1804**	0.0505	0.1516
RNNPG	**0.0561**	**0.1868**	**0.0515**	**0.2102**	**0.0572**	0.1800	**0.0549**	**0.1923**

Table 4: BLEU-2 scores on 5-char and 7-char quatrains. Given i goldstandard lines, BLEU-2 scores are computed for the next $(i+1)$th lines.

Models	Fluency		Coherence		Meaning		Poeticness		Rank	
	5-char	7-char	5-char	7-char	5-char	7-char	5-char	7-char	5-char	7-char
Random	2.52	2.18	2.22	2.16	2.02	1.93	1.77	1.71	0.31	0.26
SUM	1.97	1.91	2.08	2.33	1.84	1.98	1.66	1.73	0.25	0.22
SMT	2.81	3.01	2.47	2.76	2.33	2.73	2.08	2.36	0.43	0.53
RNNPG	$\mathbf{4.01}^{**}$	$\mathbf{3.44}^{*}$	$\mathbf{3.18}^{**}$	$\mathbf{3.12}^{*}$	$\mathbf{3.20}^{**}$	**3.02**	$\mathbf{2.80}^{**}$	$\mathbf{2.68}^{*}$	$\mathbf{0.73}^{**}$	$\mathbf{0.64}^{*}$
Human	4.31^{+}	4.19^{++}	3.81^{++}	4.00^{++}	3.61^{+}	3.91^{++}	3.29^{++}	3.49^{++}	0.79	0.84^{++}

Table 5: Mean ratings elicited by humans on 5-char and 7-char quatrains. Diacritics ** ($p < 0.01$) and * ($p < 0.05$) indicate our model (RNNPG) is significantly better than all other systems except Human. Diacritics $^{++}$ ($p < 0.01$) and $^{+}$ ($p < 0.05$) indicate Human is significantly better than all other systems.

the RNNPG estimates the probability of a poem line given at least one previous line. Therefore, the probability of a quatrain assigned by the RNNPG is the probability of the last three lines. For a fair comparison, RNNLM and KN5 only leverage the last three lines of each poem during training, validation and testing. The results in Table 3 indicate that the generation ability of the RNNPG is better than KN5 and RNNLM. Note that this perplexity-style evaluation is not possible for models which cannot produce probabilities for gold standard poems. For this reason, other related poem generators (Yan et al., 2013; He et al., 2012) are not included in the table.

The results of our evaluation using BLEU-2 are summarized in Table 4. Here, we compare our system against the SMT-based poem generation model of He et al. (2012).[4] Their system is a linear combination of two translation models (one with five features and another one with six). Our model uses three of their features, namely the inverted phrase translation model feature, the lexical weight feature, and a Kneser-Ney trigram feature. Unfortunately, it is not possible to evaluate Yan et al.'s (2013) summarization-based system with BLEU, as it creates poems as a whole and there is no obvious way to generate next lines with their algorithm. The BLEU scores in Table 4 indicate that, given the same context lines, the RNNPG is better than SMT at generating what to say next. BLEU scores should be, however, viewed with some degree of caution. Aside from being an approximation of human judgment (Callison-Burch et al., 2012), BLEU might be unnecessarily conservative for poem composition which by its very nature is a creative endeavor.

The results of our human evaluation study are shown in Table 5. Each column reports mean ratings for a different dimension (e.g., fluency, coherence). Ratings for 5-char and 7-char quatrains are shown separately. The last column reports rank scores for each system (Callison-Burch et al., 2012). In a ranked list of N items ($N = 5$ here), the score of the ith ranked item is $\frac{(N-i)}{(N-1)}$. The numerator indicates how many times a systems won in pairwise comparisons, while the denominator normalizes the score.

With respect to 5-char quatrains, RNNPG is significantly better than Random, SUM and SMT on fluency, coherence, meaningfulness, poeticness and ranking scores (using a t-test). On all dimensions, human-authored poems are rated as significantly better than machine-generated ones, with the exception of overall ranking. Here, the difference between RNNPG and Human is not significant. We obtain similar results with 7-char quatrains. In general, RNNPG seems to perform better on the shorter poems. The mean ratings

[4] Our re-implementation of their system delivered very similar scores to He et al. (2012). For example, we obtained an average BLEU-1 of 0.167 for 5-char quatrains and 0.428 for 7-char quatrains compared to their reported scores of 0.141 and 0.380, respectively.

白鹭窥鱼立, Egrets stood, peeping fishes. 青山照水开. Water was still, reflecting mountains. 夜来风不动, The wind went down by nightfall, 明月见楼台. as the moon came up by the tower.	满怀风月一枝春, Budding branches are full of romance. 未见梅花亦可人. Plum blossoms are invisible but adorable. 不为东风无此客, With the east wind comes Spring. 世间何处是前身. Where on earth do I come from?

Table 6: Example output produced by our model (RNNPG).

are higher and the improvements over other systems are larger. Also notice, that the score margins between the human- and machine-written poems become larger for 7-char quatrains. This indicates that the composition of 7-char quatrains is more difficult compared to 5-char quatrains. Table 6 shows two example poems (5-char and 7-char) produced by our model which received high scores with respect to poeticness.

Interestingly, poems generated by SUM[5] are given ratings similar to Random. In fact SUM is slightly worse (although not significantly) than Random on all dimensions, with the exception of coherence. In the human study reported in Yan et al. (2013), SUM is slightly better than SMT. There are several reasons for this discrepancy. We used a more balanced experimental design: all systems generated poems from the same keywords which were randomly chosen. We used a larger dataset to train the SMT model compared to Yan et al. (284,899 poems vs 61,960). The Random baseline is not a straw-man; it selects phrases from a taxonomy of meaningful clusters edited by humans and closely related to the input keywords.

6 Conclusions

In this paper we have presented a model for Chinese poem generation based on recurrent neural networks. Our model jointly performs content selection and surface realization by learning representations of individual characters and their combinations within and across poem lines. Previous work on poetry generation has mostly leveraged contextual information of limited length (e.g., one sentence). In contrast, we introduced two recurrent neural networks (the recurrent context model and recurrent generation model) which naturally capture multi-sentential content. Experimental results show that our model yields high quality poems compared to the state of the art. Perhaps unsurprisingly, our human evaluation study revealed that machine-generated poems lag behind human-generated ones. It is worth bearing in mind that poetry composition is a formidable task for humans, let alone machines. And that the poems against which our output was compared have been written by some of the most famous poets in Chinese history!

Avenues for future work are many and varied. We would like to generate poems across different languages and genres (e.g., Engish sonnets or Japanese haiku). We would also like to make the model more sensitive to line-to-line transitions and stylistic conventions by changing its training objective to a combination of cross-entropy error and BLEU score. Finally, we hope that some of the work described here might be of relevance to other generation tasks such as summarization, concept-to-text generation, and machine translation.

[5]We made a good-faith effort to re-implement their poem generation system. We are grateful to Rui Yan for his help and technical advice.

Acknowledgments

We would like to thank Eva Halser for valuable discussions on the machine translation baseline. We are grateful to the 30 Chinese poetry experts for participating in our rating study. Thanks to Gujing Lu, Chu Liu, and Yibo Wang for their help with translating the poems in Table 6 and Table 1.

References

Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga, and Mans Hulden. 2013. POS-Tag Based Poetry Generation with WordNet. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 162–166, Sofia, Bulgaria.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation

Modeling with Recurrent Neural Networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the 7th Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-FACE Poetry Generation. In *Proceedings of the International Conference on Computational Creativity*, pages 95–102, Dublin, Ireland.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

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, Cambridge, MA.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating Chinese Classical Poems with Statistical Machine Translation Models. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1650–1656, Toronto, Canada.

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, July.

Long Jiang and Ming Zhou. 2008. Generating Chinese Couplets using a Statistical MT Approach. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 377–384, Manchester, UK, August.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, Edmonton, Canada.

Wenwei Liu. 1735. *ShiXueHanYing*.

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.

Ruli Manurung. 2003. *An Evolutionary Algorithm Approach to Poetry Generation*. Ph.D. thesis, University of Edinburgh.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proceedings of 2012 IEEE Workshop on Spoken Language Technology*, pages 234–239, Miami, Florida.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proceedings of INTERSPEECH*, pages 1045–1048, Makuhari, Japan.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011a. Strategies for Training Large Scale Neural Network Language Models. In *Proceedings of ASRU 2011*, pages 196–201, Hilton Waikoloa Village, Big Island, Hawaii, US.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5528–5531, Prague, Czech Republic.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, United States.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1439.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating Haiku with Word Associations Norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39, Boulder, Colorado.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Hugo Gonçalo Oliveira. 2012. PoeTryMe: a Versatile Platform for Poetry Generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1988. *Learning Representations by Back-propagating Errors*. MIT Press, Cambridge, MA, USA.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea.

Naoko Tosa, Hideto Obara, and Michihiko Minoh. 2008. Hitch Haiku: An Interactive Supporting System for Composing Haiku Poem How I Learned to Love the Bomb: Defcon and the Ethics of Computer Games. In *Proceedings of the 7th International Conference on Entertainment Computing*, pages 209–216, Pittsburgh, PA.

Li Wang. 2002. *A Summary of Rhyming Constraints of Chinese Poems (Shi Ci Ge Lv Gai Yao)*. Beijing Press, 2002.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New Hitch Haiku: An Interactive Renku Poem Composition Supporting Tool Applied for Sightseeing Navigation System. In *Proceedings of the 8th International Conference on Entertainment Computing*, pages 191–196, Paris, France.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 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*, pages 2197–2203, Beijing, China.

Cheng-Le Zhou, Wei You, and Xiaojun Ding. 2010. Genetic Algorithm and its Implementation of Automatic Generation of Chinese SongCi. *Journal of Software*, pages 427–437.

Geoffrey Zweig and Konstantin Makarychev. 2013. Speed Regularization and Optimality in Word Classing. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8237–8241, Florence, Italy.