# i, Poet: Automatic Poetry Composition through Recurrent Neural Networks with Iterative Polishing Schema

**Rui Yan**[1,2,3]
[1]Department of Computer Science, Peking University
[2]Natural Language Processing Department, Baidu Research, Baidu Inc.
[3]School of Computer, Central China Normal University
yanrui02@baidu.com

## Abstract

Part of the long lasting cultural heritage of humanity is the art of classical poems, which are created by fitting words into certain formats and representations. Automatic poetry composition by computers is considered as a challenging problem which requires high Artificial Intelligence assistance. This study attracts more and more attention in the research community. In this paper, we formulate the poetry composition task as a natural language generation problem using recurrent neural networks. Given user specified writing intents, the system generates a poem via sequential language modeling. Unlike the traditional one-pass generation for previous neural network models, poetry composition needs polishing to satisfy certain requirements. Hence, we propose a new generative model with a polishing schema, and output a refined poem composition. In this way, the poem is generated incrementally and iteratively by refining each line. We run experiments based on large datasets of 61,960 classic poems in Chinese. A comprehensive evaluation, using perplexity and BLEU measurements as well as human judgments, has demonstrated the effectiveness of our proposed approach.

## 1 Introduction

Poetry is a special and important cultural heritage with more than thousands of years in humanity history. 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 opposed to free language, poems have unique elegance, e.g., aestheticism and conciseness etc. Composing classic poems is considered as a challenging task with a set of structural, phonological, and semantic requirements, hence only few best scholars are able to master the skill to manipulate or to organize terms.

With the fast development of Artificial Intelligence (A.I.), we realize that computers might play an important role in helping humans to create poems: 1) it is rather convenient for computers to sort out appropriate term combinations from a large corpus, and 2) computer programs can take great advantage to recognize, to learn, and even to remember patterns or rules given the corpus. The above observations motivate automatic poetry generation using computational intelligence.

For people to better inherit this classic art, we introduce a meaningful task of automatic poetry composition, aiming to endow the computer with artificial intelligence to mimic the generation process of human poetry so that it would be a tool that aids people to master proficiency in poem composition. We name the system as iPoet inspired from Yan *et al.* [2013], which indicates our goal is that everyone could announce proudly: "I, a poet".

To design the automatic poetry composition schema, we first need to empirically study the generation criteria. We discuss some of the general generation standards here. Unlike narratives which follow less strict rules and restrictions, a classical poem has certain generation standards. For example, classic poems generally have rigid formats with fixed length. Also, semantic coherence is a critical feature in poems. A well-written poem is supposed to be semantically coherent among all lines.

In this paper we are concerned with generating poems automatically. Although computers are no substitute for poetic creativity, they can analyze very large online text repositories of poems. Computer can extract statistical patterns, maintain them in memory and use them to generate many possible variants. Furthermore, it is relatively straightforward for the machine to check whether a candidate poem conforms to those requirements. Beyond the long-term goal of building an autonomous intelligent system capable of creating meaningful poems eventually, there are potential short-term applications for A.I. augmented human expertise/experience to possibly enable everyone to be a poet due to entertainment or educational purpose.

We propose the iPoet system based on recurrent neural networks for language generation [Zhang and Lapata, 2014; Li *et al.*, 2015; Mou *et al.*, 2015]. Given a large collection of poems, we learn representations of individual characters, and their combinations into one or more lines as well as how they mutually reinforce and constrain each other. Given the user specified writing intents, the system could generate a poem via sequential language modeling. Unlike the traditional single-pass generation in previous neural networks, our proposed system will be able to polish the generated poem for one or more iterations to refine the wording and to be more poetic, which is quite like a real human writing process. In

this way, the poem is generated incrementally and iteratively by refining each line one-by-one. The hidden representations of the generated lines will be fed into the recurrent language model to *polish* the next version of lines in the poem. In contrast to previous approaches, our generator makes utilizations of word dependencies within a line and across lines through an iterative polishing schema, which is novel. To sum up, our contributions are as follows:

• For the first time, we propose a recurrent neural network-based poetry generation model with iterative polishing schema, which enables more coherent written poems conformed to poetic requirements. The generation model is more like a real human poetry composing experience with rethinking and re-wording enabled.

• We have formulated a new system framework to take in human writing intents and to output the composed poems. The writing intents are encoded, and then decoded via recurrent neural networks with hierarchical structure, i.e., representations of "characters" and "lines" in two hierarchies.

We build iPoet on the poem dataset to verify its effectiveness compared with several baselines using automatic and manual evaluation metrics. We start by reviewing previous works. In Sections 3 & 4 we formulate a generative system framework via recurrent neural network generation model with iterative polishing schema. We describe experiments in Section 5, and draw conclusions in Section 6.

## 2 Related Work

As poetry is one of the most significant literature heritage of various cultures all over the world, there are some formal researches into the area of computer-assisted poetry generation. Scientists from different countries have studied the automatic poem composition in their own languages through different ways: 1) Genetic Algorithms. Manurung *et al.* [2004; 2011] propose to create poetic texts in English based on state search; 2) Statistical Machine Translation (SMT). Greene *et al.* [2010] propose a translation model to generation cross-lingual poetry, from Italian to English; 3) Rule-based Templates. Oliveira [2009; 2012] has proposed a system of poem generation platform based on semantic and grammar templates in Spanish. An interactive system has been proposed to reproduce the traditional Japanese poem named *Haiku* based on rule-based phrase search related to user queries [Tosa *et al.*, 2008; Wu *et al.*, 2009]. Netzer *et al.* [2009] propose another way of Haiku generation using word association rules.

Besides studies in English, Japanese, Spanish and Italian poetry composition, there is continuing research on Chinese poetry. Poetry generation is theoretically similar with different adaption for different languages. Since we mainly illustrate Chinese poem generation in this paper, we introduce more Chinese poetry generation systems here.

There are now several Chinese poetry generators available, usually template based. Zhou *et al.* [2010] use a genetic algorithm for Chinese poetry generation by tonal codings and state search. In a study of Chinese couplet generation, which could be narrowed down as a minimal poem form of 2 lines only, a SMT model is proposed to generate the 2nd sentence given the 1st sentence of a couplet [Jiang and Zhou, 2008]. He *et al.* [2012] extend the SMT framework to generate a 4-line poem by giving previous sentences sequentially, considering structural templates. Yan *et al.* [2013] proposed a generative manner to compose poems, based on the summarization framework [Yan *et al.*, 2011c; 2011b; 2012; 2011a]. Along with the prosperity of neural networks, a recurrent neural network based language generation is proposed [Zhang and Lapata, 2014]: the generation is more or less a translation process. Given the previous line, the system generates the next line and it is a single-pass generation process.

To the best of our knowledge, we are the first to apply the recurrent neural network with polishing schema for the language generation problem in poetry. We also design a hierarchical structure for different modelings of lines and characters. The proposed neural networks with iterative polishing schema look more like the real poetry process of humans.

## 3 Overview

One plausible procedure for a poet to create a poem is to first outline the main writing intents, which could be represented by a set of keywords. It is an iterative process since the author can always change part of terms to polish the idea till the entire poem is finished. iPoet tries to imitate such a process.

**Problem formulation.** We define the problem as follows:

• *Input.* Given the keywords of $\kappa$=$\{k_1, k_2, \ldots, k_{|\kappa|}\}$ from an author as the writing intent (i.e., topics, subjects, scenarios, or themes for the poem to generate), where $k_i$ is a keyword term. Each keyword consists of one or more characters, i.e., $k_i = \{c_1, c_2, \ldots\}$. We generate a poem from the keywords.

• *Output.* We generate a poem $\mathcal{P} = \{c_{1,1}, \cdots, c_{1,n}; \cdots; c_{m,1}, \cdots, c_{m,n}\}$, $c_{i,j} \in \mathcal{V}$, where $\mathcal{V}$ is the vocabulary. $n$ is the number of characters within a line of the poem; $m$ is the number of lines. For classic Chinese poetry, i.e., quatrains and regulated verses [Yan *et al.*, 2013], $n$ (either 5 or 7) and $m$ (either 4 or 8) are fixed numbers.

**System Framework.** Our system works in an encoding-decoding fashion, which represents the user intention as a single vector, and then decodes the vector to a whole poem. Figure 1 shows the architecture of our iPoet system, which comprises mainly three parts:

*Intention representation.* The system accepts a set of user-specified keywords $\kappa$ as the input. We use either a convolutional neural network (CNN) or recurrent neural network (RNN) over characters to capture the meaning of a particular keyword term; then the information of different terms is integrated by a pooling layer. Thus we obtain a single vector representation of the user intent.

*Sequential generation.* Conditioned on the vector representation of user intention, we use an RNN to compose a poem in a character by character-wise generation. Note that poems contain multiple lines, and that each line further contains multiple characters, we use a hierarchical architecture for poem generation. Concretely, we have an RNN representing *global* information for each line: the global information vector impacts on all character generations in the line. Based on the global RNN, we also have another RNN representing *local* information, which guides the generation of a single character within the line. The details is shown in Figure 1.

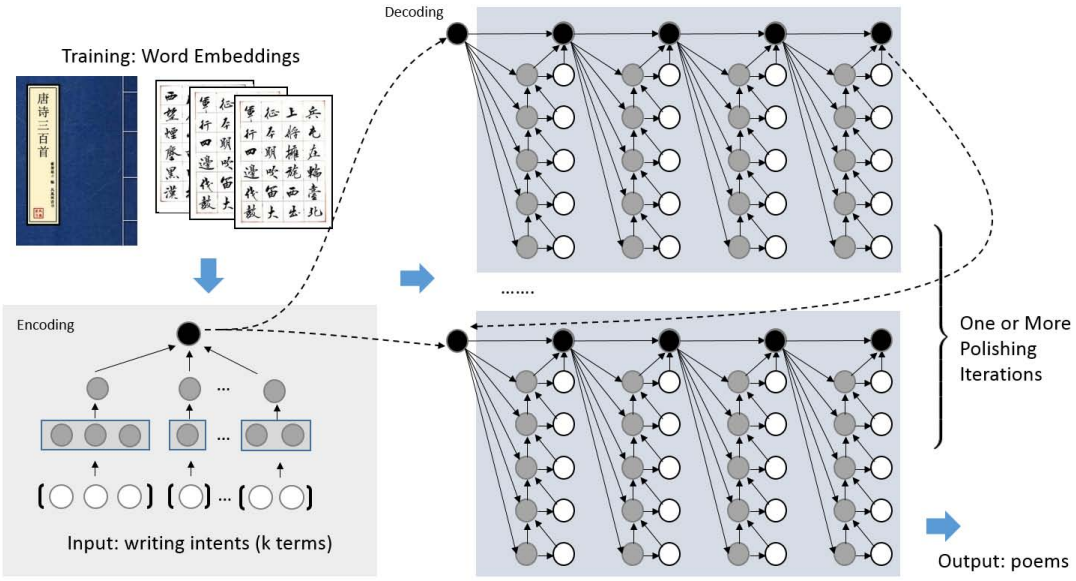



Figure 1: The illustration of the iPoet system frame including *encoding* and *decoding* neural networks. The system takes the users' writing intents ($k$ terms, $k \geq 1$) as queries, and encodes the intents as a hidden vector. We have two strategies for intent encoding (in Figure 2). With the hidden vector as a triggering state and the learned embeddings as well as the poetic language model, we "compose" a poem in a sequential decoding process through recurrent neural networks. The model is based on an iterative polishing generation schema. The white circles denote generated characters, which are observable. Shaded circles (grey and black) indicate hidden vectors in *local* and *global* hierarchies, which are hidden states to generate characters.

*Iterative polishing.* To mimic a human poet, who may re-compose his/her works for multiple times, we develop an iterative polishing schema to refine the obtained poem after one-pass generation. The process is essentially the same as sequential generation except that the information representation of the previous draft is utilized as input, serving as additional information of user intention, as well as facilitating the overall semantic coherence for the whole poem.

To sum up, the system encodes writing intents, and generates the poem in accordance with such intents through a decoding process. The generation is basically a *line*-by-*line* process, with a hierarchical concept incorporated. We polish the poem to extend the single-pass generation to a multi-pass generation, which is a novel insight. In the following section, we further delve into these steps.

# 4 The iPoet Neural Model

To be self-contained, we firstly briefly overview word embeddings, which are the foundation of our proposed neural networks. Traditional models usually treat a word as a discrete token; thus, the internal relation between similar words would be lost. Word embeddings [Mikolov *et al.*, 2013] are a standard apparatus in neural network-based text processing. A word is mapped to a low dimensional, real-valued vector. This process, known as vectorization, captures some underlying meanings. Given enough data, usage, and context, word embeddings can make highly accurate guesses about the meaning of a particular word. Embeddings can equivalently be viewed that a word is first represented as a one-hot vector and multiplied by a look-up table [Mikolov *et al.*, 2013]. In our model, we first vectorize all words using their embeddings. Word embeddings are initialized randomly, and then tuned during training based on the poem collections.

## 4.1 Intention Representation

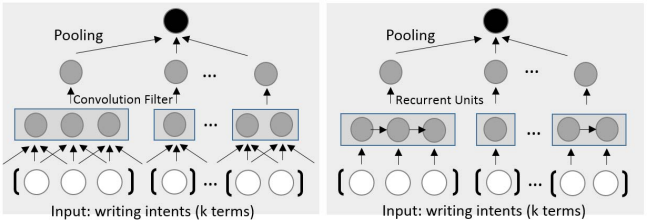
In our system, the user intention is specified as $\kappa$ keyword terms, each comprising one or more characters. We first use a convolutional neural network or recurrent neural network to capture the meaning of a keyword term; then a pooling layer can integrate representations over different terms, serving as a way of semantic compositionality [Hu *et al.*, 2014].

Without loss of generality, we let a term $k$ have $|k|$ characters, $c_1, \cdots c_{|k|}$. A convolutional neural network (CNN, Figure 2.a) applies a fixed-size window to extract local (neighboring) patterns of successive characters. Suppose the window is of size $t$, the detected features at a certain position $x_i, \cdots, x_{i+t-1}$ is given by

$$\boldsymbol{y}_i = f\left(W[\boldsymbol{x}_i; \cdots; \boldsymbol{x}_{i+t-1}] + \boldsymbol{b}\right) \quad (1)$$

where $\boldsymbol{x}$ is the vector representation (i.e., *embedding*) of the character. $W$ and $\boldsymbol{b}$ are parameters for convolution. Semicolons refer to column vector concatenation. $f(\cdot)$ is the nonlinear activation function and we use ReLU [Nair and Hinton, 2010] in the experiment. Note that we pad zero at the end of the term if a character does not have enough subsequent characters to fill the slots in the convolution window. In this way, we obtain a set of detected features $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n$. Then a max pooling layer aggregates information over different characters into a fixed-size vector, i.e.,

$$\boldsymbol{y}[j] = \max\{\boldsymbol{y}_1[i], \cdots, \boldsymbol{y}_n[i]\} \quad (2)$$




(a). CNN-based structure. (b). RNN-based structure.

Figure 2: User intention representation structures.

where $[\cdot]$ indexes a particular dimension in a vector.

Alternatively, we can use a recurrent neural network (RNN, Figure 2.b) to iteratively pick up information over the character sequence $x_i, \cdots, x_{i+t-1}$. For each character, the RNN allocates a hidden state $\boldsymbol{h}_i$, which is dependent on the current character's embedding $\boldsymbol{x}_i$ and the previous state $\boldsymbol{h}_{i-1}$. Since a term typically comprises 2–3 characters, which is not actually a very long sequence, it is sufficient that we use a vanilla RNN with basic interaction, i.e.,

$$\boldsymbol{h}_i = f(W_h \boldsymbol{h}_{i-1} + W_x \boldsymbol{x}_i + b) \tag{3}$$

We also use a max pooling layer over all hidden states as in CNNs; this is generally more effective than using the last hidden state as the sequence's vector representation.

RNNs can deal with sentence boundary smoothly, as opposed to CNNs where 0-padding is needed; thus we shall reasonably expect RNNs are more robust in this scenario [Xu *et al.*, 2016]. We will verify the performance of both structures in the experimental section. For unification, we use $\boldsymbol{h}_{in}$ to denote the encoding of the writing intents given by either CNN or RNN.

### 4.2 Sequential Generation

Having represented user intention as a fixed-size vector, we feed it to a hierarchical natural language generator (similar to Li *et al.* [2015]) for poem synthesis.

The global-level RNN (black circles in the right-hand side of Figure 1) captures the global information representation, and leads to the generation of a certain line of the poem. The starting hidden vector of the global RNN chain is given by the user intention, and then it changes as the RNN generates new lines. Let $\boldsymbol{h}_i^{(\text{high})}$ be its hidden vector, we have

$$\boldsymbol{h}_i^{(\text{global})} = f\left(W_x \boldsymbol{h}^{(\text{local})} + W_h \boldsymbol{h}_{i-1}^{(\text{global})}\right) \tag{4}$$

Here $\boldsymbol{h}^{(\text{local})}$ is the last hidden state of the low-level (a.k.a., local) RNN (gray circles in Figure 1) in the lower hierarchy, which serves as the sentence generator. Each hidden state controls the generation of one character. Its input is the word embedding of the previous character, augmented with the global information vector (black circle in Figure 1), namely $h^{(\text{global})}$, as a static attention mechanism. The output is a softmax classifier predicting the probability of a certain character at the current step. Formally, we give the formula for computing the hidden layer as follows.

$$\boldsymbol{h}_i^{(\text{local})} = f\left(W_x \boldsymbol{x}_{i-1} + W_g \boldsymbol{h}^{(\text{global})} + W_h \boldsymbol{h}_{i-1}^{(\text{local})}\right) \tag{5}$$

### 4.3 Iterative Polishing

Inspired by the observation that a human poet shall recompose their poems for several times, we propose a polishing mechanism for poem generation. Specifically, after a one-pass generation, the RNN itself shall be aware of the generated poem. Hence, we regard the global-level RNN's hidden state (corresponding to the last line), as the "gist" of the overall semantic representation of the poem, functioning similar to the user intention. Note that we also feed the original writing intention representation for further information mixing during each iteration process. The intuition is that we try not deviate from the original writing intents as polishing goes.

$$\boldsymbol{h}_0^{(\text{global})} = \begin{cases} f\left(W_i \boldsymbol{h}_{in}\right) & (1^{\text{st}} \text{ iteration}) \\ f\left(W_i \boldsymbol{h}_{in} + W_h \boldsymbol{h}^{(\text{global})}\right) & (\text{Polishings}) \end{cases} \tag{6}$$

where $\boldsymbol{h}^{\text{global}}$ is the last global information representation in the RNN chain during the previous iteration. We mix the generated poem representation with the original writing intents as the initial global state for each polishing iteration.

We have the stopping criteria as follows.

- After each iteration process, we have the gist representation of the whole generated poem $\boldsymbol{h}^{(\text{global})}$ (i.e., the last black circle for the global RNN chain). We stop the algorithm iteration when the cosine similarity between the two $\boldsymbol{h}^{(\text{global})}$ from two successive iterations exceeds a threshold $\Delta$ ($\Delta$ = 0.5 in this study).
- It is necessary to incorporate a termination schedule when the generator polishes for many times. We stop iPoet system after a fixed number of recomposition. Here we empirically set the threshold as 10 iterations.

## 5 Experiments and Evaluations

### 5.1 Experimental Setups

**Datasets.** As mentioned, in this paper we generate classic Chinese poems for experiments. During the *Tang Dynasty* (618-907 A.D.) and *Song Dynasty* (960-1279 A.D.), Chinese literature reached its golden age. We downloaded "*Poems of Tang Dynasty*" (PTD), "*Poems of Song Dynasty*" (PSD), which amounts to 61,960 poems. More detailed statistics are listed in Table 1, which shows the number of total lines, unique characters in the corpus. There are several writing formats for Chinese poetry, while *quatrain* (consisting of 4 lines) and *regulated verse* (consisting of 8 lines) show dominative culture prominence throughout Chinese history [Yan *et al.*, 2013]. They both have 5 or 7 characters per line. More than 90% of the poems in our corpus are written in these two formats, and we learn to write such poems by iPoet (i.e., 5-character or 7-character poems). In our datasets, each poem is associated with a title and the corresponding content. Hence we regard the titles as the writing intents, which is natural. In this way, we obtain abundant samples to learn how to generate poems given the intents. We randomly choose 2,000 poems for validation and 1,000 poems for testing, other non-overlap ones for training.

**Training.** The objective for training is the cross entropy errors of the predicted character distribution and the actual

Table 1: Detailed basic information of the poem datasets.

| | #Poem | #Line | #Character |
|---|---|---|---|
| PTD | 42,974 | 463,825 | 10,205 |
| PSD | 18,986 | 268,341 | 6,996 |

Table 2: Human judgement scoring criteria.

| | |
|---|---|
| Fluency | Is the poem grammatically & syntactically formed? |
| Poeticness | Does the text display the features of a poem? |
| Coherence | Is the poem thematically coherent across lines? |
| Meaning | Does the poem convey meaningful information? |

character distribution in our corpus. An $\ell_2$ regularization term is also added to the objective. The model is trained with back propagation through time with the length being the time step. The objective is minimized by stochastic gradient descent. During training, the cross entropy error of the output is back-propagated through all hidden layers to the writing intents.

**Hyperparameters and Setups.** In this paper, we used 128-dimensional word embeddings through vectorization, and they were initialized randomly and learned during training. We use the ReLU function as the activation function in neural networks. As our dataset is in Chinese, we performed standard Chinese segmentation into characters. We set the width of convolution filters as 3. To train the network we use stochastic gradient descent with shuffled mini-batches (with a mini-batch size of 100) for optimization. Gradient is computed by standard back-propagation. Initial learning rate was set to 0.8, and a multiplicative learning rate decay was applied. The above parameters were chosen empirically. We used the validation set for early stopping. In practice, the training converges after a few epochs.

## 5.2 Evaluation Metrics

It is generally difficult to judge the effect of poem generated by computers. We propose to evaluate the experimental results from three instinctively different evaluation metrics.

**Perplexity (PPL).** For most of the language generation research, language perplexity is a sanity check. Our first set of experiments involved intrinsic evaluation of the "perplexity" evaluation for the generated poems. Perplexity is actually an entropy based evaluation. In this sense, the lower perplexity for the poems generated, the better performance in purity for the generations, and the poems are likely to be good ones.

**BLEU.** The Bilingual Evaluation Understudy (BLEU) score-based evaluation is generally used for machine translation: given the reference translation(s), the algorithm evaluates the quality of text which has been machine-translated from the reference translation as groundtruth. We adapt the BLEU evaluation under the poetry generation scenario. Take a poem from the dataset, we generate the A.I. authored poem given the title, and compare it with the original poem written by the human poet. There is a concern for such an evaluation metric is that BLEU score can only reflect the partial capability of the models; there is (for most cases) only one ground truth for the generated poems but actually there are more than one appropriate ways to generate a good poem. The merit of BLEU evaluation is to examine how likely to approximate the computer generated poems towards human authored ones.

**Human Evaluation.** Evaluators are requested to express an opinion over the automatically composed poems. A clear criterion is necessary for human evaluation. We adopt the evaluation standards discussed in [Wang, 2002; He *et al.*, 2012; Yan *et al.*, 2013; Zhang and Lapata, 2014]: "Fluency", "Poeticness", "Coherence", and "Meaning". We clearly illustrate the criteria in Table 2, so that human evaluators can easily follow. They only need to assign 0-1 scores according to the four criteria ('0'-no, '1'- yes). After that, the total score of the poem is calculated by summing up the four individual scores, in a 5-point scale ranging from 0 to 4. The evaluation process is conducted as a blind-review.

## 5.3 Algorithms for Comparisons

We implemented several poetry generation methods as baselines. For fairness, we conduct the same pre-generation process to all algorithms.

*Random.* The method randomly chooses characters as a poem. It is a lower bound for computer-generated poems.

*SMT.* A Chinese poetry generation method is proposed based on statistical machine translation [He *et al.*, 2012]. The process is that given one generated line, the system generates the next line by translating the previous sentence as a pair of "couplet" one by one, which is a single-pass generation.

*SUM.* Given the writing intents, the *Sum* method first retrieves relevant poems from the corpus, and then summarizes the retrieved poems into a single one based on a generative summarization framework [Yan *et al.*, 2013].

*RNNPG.* The RNN-based poem generator (RNNPG) is proposed to generate a poem: the first line is generated by a standard recurrent neural network language model [Mikolov *et al.*, 2010] and then generate all other lines using previously generated lines as contexts. The generation process is literally a single-pass manner [Zhang and Lapata, 2014].

*LSTM-RNN.* LSTM-RNN is basically a recurrent neural network using the Long Short Term Memory (LSTM) architecture [Hochreiter and Schmidhuber, 1997]. The RNN with LSTM units consists of memory cells in order to store information for extended periods of time. We first use an LSTM-RNN to encode the writing inputs to a vector space, and then use another LSTM-RNN to decode the vector into a generated poem, which is literally a sequence-to-sequence process [Sutskever *et al.*, 2014].

*iPoet.* Here we propose the recurrent neural network-based poetry generation model with iterative polishing schema. There are two prominent advantages for the iPoet system: 1) the polishing schema enables better coherence and 2) two RNNs for *characters* and *lines* characterizes hierarchical modelings. Our proposed system is a multi-pass generation.

## 5.4 Performance

In Table 3 we show the overall performance of our iPoet system compared with strong competing methods as described above. We see that, for both PPL and BLEU metrics, our system outperforms all baseline models. The results are also conservative in both settings of 5-character and 7-character poem generations.

Table 3: Overall performance comparison against baselines.

| Algo. | 5-Character | | | 7-Character | | |
|---|---|---|---|---|---|---|
| | PPL | BLEU | Human | PPL | BLEU | Human |
| Random | – | 0.002 | 0.259 | – | 0.051 | 0.135 |
| SMT | 126 | 0.051 | 1.943 | 134 | 0.144 | 1.957 |
| SUM | 149 | 0.035 | 2.219 | 131 | 0.128 | 2.013 |
| RNNPG | 103 | 0.053 | 1.964 | 119 | 0.163 | 2.205 |
| LSTM-RNN | 123 | 0.048 | 1.762 | 136 | 0.159 | 1.633 |
| iPoet | **91** | **0.088** | **2.352** | **96** | **0.185** | **2.568** |

Table 4: Performance comparison from different strategies.

| Algo. | 5-Character | | 7-Character | |
|---|---|---|---|---|
| | PPL | BLEU | PPL | BLEU |
| CNN for Intents | 95 | 0.085 | 103 | 0.181 |
| RNN for Intents | 91 | 0.088 | 96 | 0.185 |
| No Polishing/Hierarchy | 121 | 0.045 | 132 | 0.142 |
| No Polishing | 105 | 0.064 | 116 | 0.168 |
| No Hierarchy | 114 | 0.049 | 123 | 0.161 |
| iPoet | 91 | 0.088 | 96 | 0.185 |

The *Random* method has the worst performance, since it is naive without considering any language characteristics. For standard translation method *SMT* and summarzation method *SUM*, both approaches manipulate characters according to the poem dataset, either by translation or summarization. In general, the *SUM* performs better due to the optimization for more poetic characteristics while *SMT* does not. For the generation models based on neural networks, *RNNPG* performs better than *LSTM-RNN*. LSTM-RNN is not really a poem-driven generation method but likely to be an advanced SMT method. The poems generated by LSTM-RNN are less meaningful to get the decent scorings, since LSTM-RNN better suits longer sequences in general. *RNNPG* is strong baseline which applies both CNN and RNN structures and models previous lines as contexts. There is a major drawback that a single-pass generation is insufficient to characterize the general process of human composition. We hence introduce the iterative polishing schema into the natural language generation process through neural networks. Such a schema enables the system to revise and refine the generated poem, which leads to better performance in experimental evaluations.

For evaluations, the *perplexity* scores and *BLEU* scores are quite consistent. We observe that the BLEU scores are quite low for almost all methods. It is not surprising that these methods are not likely to generate the exactly same poems as the ground truth, since that is not how the objective function works. BLEU can only partially calibrate the capability of poetry generation because there are many ways to create poems which do not look like the ground truth but also make sense to people. Although quite subjective, the *human evaluations* can to some extent show the potentials of all poetry generators. For the 4 standards, "coherence" and "meaning" criteria make our iPoet distinguished from other baselines most.

### 5.5 Analysis and Discussions

There are two different strategies to represent the writing intents through the deep neural networks: 1) CNN-based structure and 2) RNN-based structure for representations. We make a direct comparison between such two ways of writing intents modelings, and list the results in Table 4. As we expect, the RNN-based structure performs slightly better than CNN-based one. Thus, we deploy the RNN-based structure in iPoet system frame in Figure 1.

One of the contributions in this paper is that we propose an iterative polishing schema, which enables a multi-pass generation process. The iPoet generator can generate a line utilizing the gist information of the entire poem rather than the information from generated lines only. It is a major improvement over previous methods. Here we analyze the effect and benefits of the iterative polishing schema in Table 4. The generated poem from the first iteration is actually the approach without polishing schema. In general, the polishing process stops after several iterations.

In this paper, we also incorporate a hierarchical generation structure, from *characters* to *lines*: the hidden vectors for characters serve as "local" information to have impacts on a single character within a line, while the hidden vectors of lines and writing intents are "global" information to influence the generation of all characters of a line. We also examine the effects of such hierarchical modeling of the poem structure by removing the global hidden vector for the lines. In this way, the local hidden vector from the previous line directly links to the first local hidden vector in the next line. We maintain the iterative polishing schema in the structure without hierarchical modeling: the last local hidden vector in the last line links to the first local hidden vector in the first line within the next generation pass. For a complete comparison, we also show the results of the system with neither polishing schema or hierarchial structure, which is a plain model. The results in Table 4 show that both strategies make prominent contributions to the performance of iPoet system.

## 6 Conclusions and Future Work

Poetry composition is a difficult task in the field of natural language generation. We propose a novel approach to model this problem based on recurrent neural network structures. Given the user writing intents, we encode the information and decode it into a poem via a sequential generation. The two innovative insights are that 1) we incorporate an iterative polishing schema and 2) a hierarchical structure with local and global information for characters and lines. The polishing schema utilizes global information of the whole poem, and enables recomposition, which is a multi-pass generation.

We compare our approach with several baselines. We apply perplexity and BLEU to evaluate the performance of poetry generation as well as human judgments of 4 criteria. Through our experiments, we show that the iPoet neural model can generate rather good poems and outperform baselines. Besides, both polishing schema and hierarchical structure contribute to the better performance the proposed approach. In the future, we plan to incorporate more poetic characteristics such as *parallelism* and *sentiments* in the generation process.

## Acknowledgments

We thank all the anonymous reviewers for their valuable comments. This paper is partially supported by National Basic

Research Program of China (No. 2014CB340505).

## References

[Greene *et al.*, 2010] Erica Greene, Tugba Bodrumlu, and Kevin Knight. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP'10, pages 524–533, 2010.

[He *et al.*, 2012] J. He, M. Zhou, and L. Jiang. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050, 2014.

[Jiang and Zhou, 2008] Long Jiang and Ming Zhou. Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 377–384, 2008.

[Li *et al.*, 2015] Jiwei Li, Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *ACL-IJCNLP*, pages 1106–1115, 2015.

[Manurung *et al.*, 2011] R. Manurung, G. Ritchie, and H. Thompson. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64, 2011.

[Manurung, 2004] H. Manurung. An evolutionary algorithm approach to poetry generation. *University of Edinburgh. College of Science and Engineering. School of Informatics.*, 2004.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3, 2010.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Mou *et al.*, 2015] Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. Backward and forward language modeling for constrained sentence generation. *arXiv preprint arXiv:1512.06612*, 2015.

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, pages 807–814, 2010.

[Netzer *et al.*, 2009] Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. Gaiku: generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 32–39, 2009.

[Oliveira, 2009] H. Oliveira. Automatic generation of poetry: an overview. *Universidade de Coimbra*, 2009.

[Oliveira, 2012] H.G. Oliveira. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21, 2012.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[Tosa *et al.*, 2008] N. Tosa, H. Obara, and M. Minoh. Hitch haiku: An interactive supporting system for composing haiku poem. *Entertainment Computing-ICEC 2008*, pages 209–216, 2008.

[Wang, 2002] Li Wang. A summary of rhyming constraints of chinese poems. Beijing Press, 2002.

[Wu *et al.*, 2009] X. Wu, N. Tosa, and R. Nakatsu. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. *Entertainment Computing–ICEC 2009*, pages 191–196, 2009.

[Xu *et al.*, 2016] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*, 2016.

[Yan *et al.*, 2011a] Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. Timeline generation through evolutionary trans-temporal summarization. In *EMNLP '11*, pages 433–443, 2011.

[Yan *et al.*, 2011b] Rui Yan, Jian-Yun Nie, and Xiaoming Li. Summarize what you are interested in: An optimization framework for interactive personalized summarization. In *EMNLP '11*, pages 1342–1351, 2011.

[Yan *et al.*, 2011c] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *SIGIR '11*, pages 745–754, 2011.

[Yan *et al.*, 2012] Rui Yan, Xiaojun Wan, Mirella Lapata, Wayne Xin Zhao, Pu-Jen Cheng, and Xiaoming Li. Visualizing timelines: Evolutionary summarization via iterative reinforcement between text and image streams. In *CIKM '12*, pages 275–284, 2012.

[Yan *et al.*, 2013] Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. i, poet: automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2197–2203, 2013.

[Zhang and Lapata, 2014] Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *EMNLP*, pages 670–680, 2014.

[Zhou *et al.*, 2010] Cheng-Le Zhou, Wei You, and Xiaojun Ding. Genetic algorithm and its implementation of automatic generation of chinese songci. *Journal of Software*, 21(3):427–437, 2010.