Chinese Poetry Generation with a Working Memory Model

Xiaoyuan Yi[1], **Maosong Sun**[1*], **Ruoyu Li**[2], **Zonghan Yang**[1]
[1] State Key Lab on Intelligent Technology and Systems,
Beijing National Research Center for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2] 6ESTATES PTE LTD, Singapore

Abstract

As an exquisite and concise literary form, poetry is a gem of human culture. Automatic poetry generation is an essential step towards computer creativity. In recent years, several neural models have been designed for this task. However, among lines of a whole poem, the coherence in meaning and topics still remains a big challenge. In this paper, inspired by the theoretical concept in cognitive psychology, we propose a novel Working Memory model for poetry generation. Different from previous methods, our model explicitly maintains topics and informative limited history in a neural memory. During the generation process, our model reads the most relevant parts from memory slots to generate the current line. After each line is generated, it writes the most salient parts of the previous line into memory slots. By dynamic manipulation of the memory, our model keeps a coherent information flow and learns to express each topic flexibly and naturally. We experiment on three different genres of Chinese poetry: quatrain, iambic and chinoiserie lyric. Both automatic and human evaluation results show that our model outperforms current state-of-the-art methods.

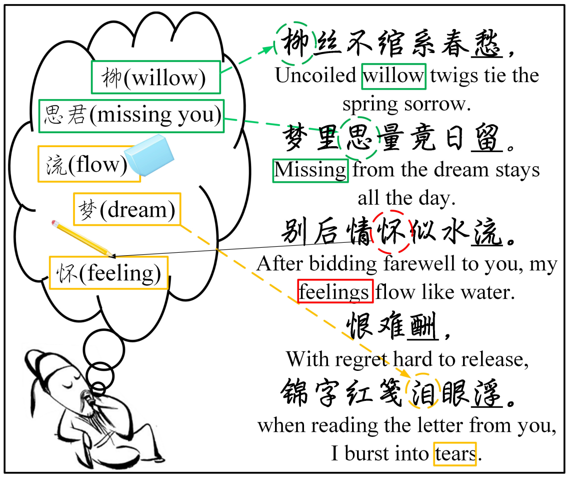
柳(willow)
思君(missing you)
流(flow)
梦(dream)
怀(feeling)
柳丝不绾系春愁，
Uncoiled willow twigs tie the spring sorrow.
梦里思量竟日留。
Missing from the dream stays all the day.
别后情怀似水流。
After bidding farewell to you, my feelings flow like water.
恨难酬，
With regret hard to release,
锦字红笺泪眼淫。
when reading the letter from you, I burst into tears.

Figure 1: An iambic generated by our model with the tune *Remember the Prince*, taking *liu* (willow) and *si jun* (missing you) as input topic words. Rhyming characters are underlined. The left part is an artistic illustration of our model, where solid and dotted arrows represent memory writing and reading respectively.

1 Introduction

Poetry is a literary form with concise language, exquisite expression and rich content, as well as some structural and phonological requirements. During the thousands of years of human history, poetry is always fascinating and popular, influencing the development of different countries, nationalities and cultures.

In Chinese, there are different genres of poetry. In this work, we mainly focus on three of them: quatrain (*Jueju*), iambic (*Ci*) and chinoiserie lyric. Both for quatrain and iambic, there are various tunes (sub-genres) and each tune defines the length of each line, the tone of each character and the number of lines (for iambic). With more than eight hundred tunes, iambic is a quite complex genre (as shown in Figure 1). By contrast, chinoiserie lyric is relatively free except for the requirement on rhyme, which gets popular in recent twenty years, driven by some famous singers [Fung, 2007].

We concentrate on automatic poetry generation. Besides the requirements on form, to create a high-quality poem, how to achieve better coherence is a key problem across different genres. Generally, two factors must be taken into account. For one thing, the topic needs to be expressed in a poem flexibly. For multiple topics, natural transition among different topics can improve coherence. For another, lines in a poem should be coherent in meaning, theme and artistic conception.

Recently, several neural models have been designed for different aspects of this task, such as poetry style transfer [Zhang *et al.*, 2017] and rhythmic constraints [Ghazvininejad *et al.*, 2016]. Nevertheless, this fundamental problem, coherence, hasn't been handled well, which is a major reason for the gap between computer-generated poems and the human-authored ones. The key point lies in that when generating a poem line, existing models assume user inputs (topics) and the history (preceding generated lines in the poem) can be packed into a *single* small vector [Yan, 2016], or assume the model is able to focus on the most important part of an *unlimited* history

[*] Corresponding author: M. Sun (sms@mail.tsinghua.edu.cn)

[Wang *et al.*, 2016a], which are implausible and against a human writing manner.

To tackle this problem, we refer to the concept in cognitive psychology, where the working memory is a system with a *limited* capacity that is responsible for holding information available for reasoning, decision-making and behaviour [Priti and Miyake, 1999]. Previous work has demonstrated the importance of working memory in writing [McCutchen, 2000]. From the perspective of psycholinguistics, coherence is achieved if the reader can connect the incoming sentence to the content in working memory and to the major messages and points of the text [Sanders *et al.*, 2001].

Inspired by this, we propose a novel Working Memory model[1] for poetry generation. Rather than merges user topic words as one vector as previous work [Yan, 2016], our model maintains them in the memory explicitly and independently, which play the role of 'major messages'. When generating each line, our model learns to read most relevant information (topics or history) from the memory to guide current generation, according to what has been generated and which topics have been expressed so far. For each generated line, our model selects the most salient parts, which are informative for succeeding generation, and writes them into the memory. Instead of full history, our model keeps informative partial history in *multiple* but *limited* memory slots. This dynamical reading-and-writing way endows the model with the ability to focus on relevant information and to ignore distractions during the generation process, and therefore improves coherence to a significant extent. Besides, we design a special genre embedding to control the tonal category of each character and the length of each line, which makes our model structure-free and able to generate various genres of poetry.

In summary, the contributions of this paper are as follows:

- To the best of our knowledge, for poetry generation, we first propose to exploit history with a dynamically reading-and-writing memory.
- We utilize a special genre embedding to flexibly control the structural and phonological patterns, which enables our model to generate various genres of poetry.
- On quatrain, iambic and chinoiserie lyric, our model outperforms several strong baselines and achieves new state-of-the-art performance.

2 Related Work

As a long-standing concern of AI, the research on automatic poetry generation can be traced back decades. The first step of this area is based on rules and templates [Gervás, 2001]. Since the 1990s, statistical machine learning methods are adopted to generate poetry, such as genetic algorithms [Manurung, 2003] and statistical machine translation (SMT) approach [He *et al.*, 2012].

Stepping into the era of neural networks, different models have been proposed to generate poetry and shown great advantages. In general, previous neural models fall under three methodologies in terms of how the history (preceding generated lines in the poem) is exploited.

The first methodology is to pack all history into a *single* history vector. Zhang and Lapata first [2014] propose to generate Chinese quatrains with Recurrent Neural Network (RNN). Each generated line is vectorized by a Convolutional Sentence Model and then packed into the history vector. To enhance coherence, their model needs to be interpolated with two extra SMT features, as the authors state. Yan [2016] generates Chinese quatrains using two RNNs. The last hidden state in the first RNN is used as the line vector, which is packed into a history vector by the second RNN. In his model, the poem generated in one pass will be refined for several times with an iterative polishing schema to improve quality.

The second one is to concatenate full history as a long sequence, which is exploited by a sequence-to-sequence model with attention mechanism [Bahdanau *et al.*, 2015]. [Wang *et al.*, 2016b] proposes a two-stage Chinese quatrains generation method which plans sub-keywords of the poem in advance by a language model, then generates each line with the aid of the planned sub-keyword. However, such planning of keywords takes a risk of losing flexibility in topic expression.

The last one is to take the whole poem as a long sequence and generate it word by word, where history propagates implicitly along the RNN. This methodology is used to generate both English poetry [Hopkins and Kiela, 2017; Ghazvininejad *et al.*, 2017] and Chinese poetry [Zhang *et al.*, 2017; Wang *et al.*, 2016a]

These neural network-based approaches are promising, but there is still a lot of room for improvement. A single vector doesn't have enough capacity to maintain the full history. Moreover, informative words and noises (e.g., stop words) are mixed, which hinders the exploitation of history. When the input or the output sequence is too long, the performance of sequence-to-sequence model will still degrade, even with an attention mechanism, which has been observed in related tasks, e.g., Neural Machine Translation [Shen *et al.*, 2016]. Consequently, we propose our Working Memory model with *multiple* but *limited* memory slots.

Memory Network (MN) [Weston *et al.*, 2015] and Neural Turing Machine (NTM) have shown great power in some tasks, e.g., Question Answering (QA). The most relevant work to our model is [Zhang *et al.*, 2017], which saves hundreds of human-authored poems in a *static* external memory to improve the innovation of generated quatrains and achieve style transfer. In fact, these MN and NTM models just learn to write external texts (poems or articles) into memory. By contrast, our model writes the generated history and hence adopts a dynamic utilization of memory, which is closer to a human manner as discussed in Section 1.

3 Model Description

3.1 Overview

Before presenting the proposed model, we first formalize our task. The inputs are user topics specified by K_1 keywords, $\{w_k\}_{k=1}^{K_1}$. The output is a poem consisting of n lines, $\{L_i\}_{i=1}^{n}$. Since we take the sequence-to-sequence framework

[1] In fact, part of our model can be also considered as a kind of Neural Turing Machine [Graves *et al.*, 2014]. We take the perspective of working memory here to emphasize the influence of this structure on human writing.

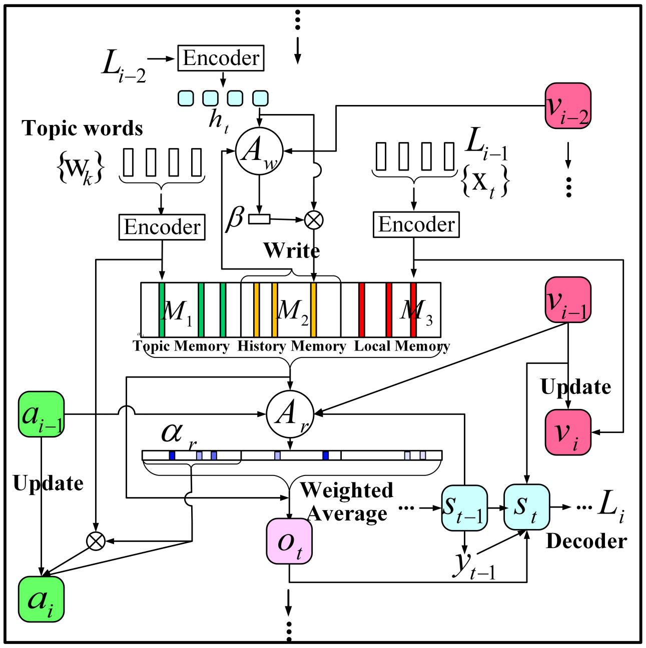

Figure 2: A graphical illustration of the Working Memory model, which consists of an encoder, a decoder and the working memory. The top half of this figure shows memory writing before generating L_i, and the bottom half shows the generation of L_i.

and generate a poem line by line, the task can be converted to the generation of an i-th line which is coherent in meaning and related to the topics, given previous i-1 lines $L_{1:i-1}$ and the topic words $w_{1:K_1}$.

As illustrated in Figure 2, the working memory is comprised of three modules: topic memory $M_1 \in \mathbb{R}^{K_1 * d_h}$, history memory $M_2 \in \mathbb{R}^{K_2 * d_h}$ and local memory $M_3 \in \mathbb{R}^{K_3 * d_h}$, where each row of the matrices is a memory slot and d_h is slot size. K_2 and K_3 are the numbers of slots. Therefore the whole working memory $M = [M_1; M_2; M_3]$, $M \in \mathbb{R}^{K * d_h}$ where $[;]$ means concatenation and K=K_1+K_2+K_3.

Each topic word w_k is written into the topic memory in advance, which acts as the 'major message' and remains unchanged during the generation process of a poem. Before generating the i-th line L_i, each character of L_{i-1} is written into the local memory. There are often strong semantic associations between two adjacent lines in Chinese poetry, therefore we feed L_{i-1} into this local memory to provide full short-distance history. Different from other two modules, the model selects some salient characters of L_{i-2} to write into the history memory. In this way, the history memory maintains informative partial long-distance history. These three modules are read jointly.

Following this procedure, we detail our model.

3.2 Working Memory Model

Based on the sequence-to-sequence framework, we use GRU [Cho *et al.*, 2014] for decoder and bidirectional encoder. Denote X a line in encoder (L_{i-1}), $X = (x_1 x_2 \dots x_{T_{enc}})$, and Y a generated line in decoder (L_i), $Y = (y_1 y_2 \dots y_{T_{dec}})$. h_t and s_t represent the encoder and decoder hidden states respectively. $e(y_t)$ is the word embedding of y_t. The probability distribution of each character to be generated in L_i is calculated by[2]:

$$s_t = GRU(s_{t-1}, [e(y_{t-1}); o_t; g_t; v_{i-1}]), \quad (1)$$

$$p(y_t|y_{1:t-1}, L_{1:i-1}, w_{1:K_1}) = softmax(Ws_t), \quad (2)$$

where o_t is the memory output and W is the projection parameter. v_{i-1} is a global trace vector, which records what has been generated so far and provides implicit global information for the model. Once L_i is generated, it is updated by a simple vanilla RNN:

$$v_i = \sigma(v_{i-1}, \frac{1}{T_{enc}} \sum_{t=1}^{T_{enc}} h_t), v_0 = \mathbf{0}. \quad (3)$$

σ defines a non-linear layer and $\mathbf{0}$ is a vector with all 0-s.

Genre embedding. g_t in Eq. (1) is a special genre embedding. Since poetry must obey structural and phonological rules, we use this genre embedding to control the genre of a generated poem over each character. g_t is the concatenation of a phonology embedding and a length embedding, which are learned during training. We define 36 phonology categories in terms of [Ge, 2009]. The phonology embedding indicates the required category of y_t. The length embedding indicates the number of characters to be generated after y_t in L_i and hence controls the length of L_i.

Memory Reading. We begin by defining an *Addressing Function*, $\alpha = A(\tilde{M}, q)$, which calculates the probabilities that each slot of the memory is to be selected and operated. Concretely, we have:

$$z_k = b^T \sigma(\tilde{M}[k], q), \quad (4)$$

$$\alpha[k] = softmax(z_k), \quad (5)$$

where q is the query vector, b is the parameter, $\tilde{M}$ is the memory to be addressed, $\tilde{M}[k]$ is the k-th slot (row) of $\tilde{M}$ and $\alpha[k]$ is the k-th element in vector α.

Then, the working memory is read as:

$$\alpha_r = A_r(M, [s_{t-1}; v_{i-1}]), \quad (6)$$

$$o_t = \sum_{k=1}^{K} \alpha_r[k] * M[k], \quad (7)$$

where α_r is the reading probability vector and the trace vector v_{i-1} is used to help the Addressing Function avoid reading redundant content. Joint reading from the three memory modules enables the model to flexibly decide to express a topic or to continue the history content.

Memory Writing. Here we use hidden states as vector representations of characters. For topic memory, we feed characters of each topic word w_k into the encoder, then get a topic vector by a non-linear transformation of the corresponding hidden states. Then each topic vector is directly filled into a slot. Before generating L_i, the encoder hidden states of characters in L_{i-1} are filled into local memory slots.

[2] For brevity, we omit biases and use h_t to represent the combined state of bidirectional encoder.

	# of Poems	# of Lines	# of Characters
Quatrains	72,000	288,000	1,728,000
Iambics	33,499	418,896	2,099,732
Lyrics	1,079	37,237	263,022

Table 1: Details of our corpus.

After L_i is generated and before the generation of L_{i+1}, for each encoder state h_t of L_{i-1}, the model select a history memory slot by writing addressing function and fill h_t into it. Formally, we have:

$$\alpha_w = A_w(\tilde{M_2}, [h_t; v_{i-1}]), \tag{8}$$

$$\beta[k] = I(k = \arg\max_j \alpha_w[j]), \tag{9}$$

$$\tilde{M_2}[k] \leftarrow (1 - \beta[k]) * \tilde{M_2}[k] + \beta[k] * h_t, \tag{10}$$

where I is an indicator function and α_w is the writing probabilities vector. $\tilde{M_2}$ is the concatenation of history memory M_2 and a null slot. If there is no need to write h_t into history memory, model learns to write it into the null slot, which is ignored when reading memory by Eq. (6).

Since Eq. (9) is non-differentiable, it is only used for testing. For training, we simply approximate β as:

$$\beta = tanh(\gamma * (\alpha_w - \mathbf{1} * max(\alpha_w))) + \mathbf{1}, \tag{11}$$

where $\mathbf{1}$ is a vector with all 1-s and γ is a large positive number. Eq. (11) is a rough approximation but it's differentiable, by which the model learns to focus on one slot with a higher writing probability. We expect h_t to be written into only one slot, because we want to keep the representations of salient characters independent as discussed in Section 2.

Before the generation, all memory slots are initialized with $\mathbf{0}$. For empty slots, a random bias is added to z_k in Eq. (5) to prevent multiple slots getting the same probability.

3.3 Topic Trace Mechanism

Though we use a global trace vector v_i to save all generated content, it seems not enough to help the model remember whether each topic has been used or not. Therefore we design a **Topic Trace (TT)** mechanism, which is a modified coverage model [Tu *et al.*, 2016], to record the usage of topics in a more explicit way:

$$c_i = \sigma(c_{i-1}, \frac{1}{K_1}\sum_{k=1}^{K_1} M[k] * \alpha_r[k]), c_0 = \mathbf{0}, \tag{12}$$

$$u_i = u_{i-1} + \alpha_r[1:K_1], u_i \in \mathbb{R}^{K_1 * 1}, u_0 = \mathbf{0}, \tag{13}$$

$$a_i = [c_i; u_i]. \tag{14}$$

c_i maintains the content of used topics and u_i explicitly records the times of reading each topic. a_i is the topic trace vector. Then we rewrite Eq. (6) as:

$$\alpha_r = A_r(M, [s_{t-1}; v_{i-1}; a_{i-1}]). \tag{15}$$

We will show that this Topic Trace mechanism can further improve the performance of our model in Section 4.

	Models	BLEU	PP
Quatrains	iPoet	0.425	138
	WM	**1.315**	**86**
Iambics	iambicGen	0.320	262
	WM	**0.699**	**72**
Lyrics	lyricGen	0.312	302
	WM	**0.568**	**138**

Table 2: Automatic evaluation results. BLEU scores are calculated by the multi-bleu.perl script. PP means perplexity.

Strategies	Quatrains		Iambics		Lyrics	
	BLEU	PP	BLE	PP	BLEU	PP
WM$_0$	1.019	127	0.561	152	0.530	249
WM$_0$+GE	1.267	87	0.672	74	0.542	144
WM$_0$+GE+TT	**1.315**	**86**	**0.699**	**72**	**0.568**	**138**

Table 3: Comparison of different strategies. GE: genre embedding. TT: Topic Trace mechanism. WM$_0$ is the model without GE or TT.

4 Experiments

4.1 Data and Setups

Table 1 shows details of our corpus. We use 1,000 quatrains, 843 iambics and 100 lyrics for validation; 1,000 quatrains, 900 iambics and 100 lyrics for testing. The rest are used for training.

Since our model and most previous models take topic words as input, we run TextRank [Mihalcea and Tarau, 2004] on the whole corpus and extract four words from each poem. In training, we build four <keyword(s), poem> pairs for each poem using 1 to 4 keywords respectively, so as to improve the model's ability to cope with different numbers of keywords. In testing, we randomly select one pair from the four and use the phonological and structural pattern of ground truth.

We set K_1 = 4 and K_2 = 4. The sizes of word embedding, phonology embedding, length embedding, hidden state, global trace vector, topic trace vector are set to 256, 64, 32, 512, 512, 24 (20+4) respectively. Since we directly feed hidden states of bidirectional encoder into memory, the slot size d_h is 1024. The word embedding is initialized with word2vec vectors pre-trained on the whole corpus. Different memory modules share the same encoder. We use two different addressing functions for reading and writing respectively. For all non-linear layers, tanh is used as the activation function.

Adam with shuffled mini-batches (batch size 64) is used for optimization. To avoid overfitting, 25% dropout and ℓ_2 regularization are used. Optimization objective is standard cross entropy errors of the predicted character distribution and the actual one. Given several topic words as input, all models generate each poem with beam search (beam size 20). For fairness, all baselines share the same configuration.

4.2 Models for Comparisons

Besides **WM**[3] (our Working Memory model) and **Human** (human-authored poems), on quatrains we compare **iPoet** [Yan, 2016], **Planning** [Wang *et al.*, 2016b] and **FCPG**

[3] https://github.com/xiaoyuanYi/WMPoetry.

	Models	Fluency	Meaning	Coherence	Relevance	Aesthetics
Quatrains	Planning	2.28	2.13	2.18	2.50	2.31
	iPoet	2.54	2.28	2.27	2.13	2.45
	FCPG	2.36	2.15	2.15	2.65	2.28
	WM	**3.57****	**3.45****	**3.55****	**3.77****	**3.47****
	Human	3.62	3.52	3.59	3.78	3.58
Iambics	iambicGen	2.48	2.73	2.78	2.36	3.08
	WM	**3.39****	**3.69****	**3.77****	**3.87****	**3.87****
	Human	4.04	4.10^{++}	4.13^{++}	4.03	4.09
Lyrics	lyricGen	1.70	1.65	1.81	2.24	1.99
	WM	**2.63****	**2.49****	**2.46****	**2.53**	**2.66****
	Human	3.43^{++}	3.20^{++}	3.41^{++}	3.34^{++}	3.26^{++}

Table 4: Human evaluation results. Diacritic ** ($p < 0.01$) indicates WM significantly outperforms baselines; ++ ($p < 0.01$) indicates Human is significantly better than all models. The Intraclass Correlation Coefficient of the four groups of scores is 0.5, which indicates an acceptable inter-annotator agreement.

[Zhang *et al.*, 2017]. We choose these previous models as baselines, because they all achieve satisfactory performance and the authors have done thorough comparisons with other models, such as RNNPG [Zhang and Lapata, 2014] and SMT [He *et al.*, 2012]. Moreover, the three models just belong to the three methodologies mentioned in Section 2 respectively.

On iambics we compare **iambicGen** [Wang *et al.*, 2016a]. To the best of our knowledge, this is the only one neural model designed for Chinese iambic generation.

On chinoiserie lyrics, since there is no specially designed model in the literature, we implement a standard sequence-to-sequence model as the baseline, called **lyricGen**.

4.3 Evaluation Design

Automatic Evaluation. Referring to [Zhang and Lapata, 2014; Yan, 2016], we use BLEU and perplexity to evaluate our model. BLEU and Perplexity are not perfect metrics for generated poems, but they can still provide an aspect for evaluation and make sense to some extent in the context of pursuing better coherence. Furthermore, automatic evaluation can save much labour and help us determine the best configure.

Human Evaluation. We design five criteria: **Fluency** (does the poem obey the grammatical, structural and phonological rules?), **Meaning** (does the poem convey some certain messages?), **Coherence** (is the poem as a whole coherent in meaning and theme?), **Relevance** (does the poem express user topics well?), **Aesthetics** (does the poem have some poetic and artistic beauties?). Each criterion needs to be scored in a 5-point scale ranging from 1 to 5.

From the testing set, for quatrains, iambics and lyrics we randomly select 30, 30 and 20 sets of topic words respectively to generate poems with these models. For Human, we select poems containing the given words. Therefore, we obtain 150 quatrains (30*5), 90 iambics (30*3) and 60 lyrics (20*3). We invite 16 experts[4] on Chinese poetry to evaluate these poems, who are divided into four groups. Each group completes the evaluation of all poems and we use the average scores.

Planning and FCPG are not suitable for automatic evaluation, because FCPG is designed for innovation and Planning will plan the sub-topics by itself, which increase the perplexity. Thus we leave them for human evaluation.

4.4 Evaluation Results

As shown in Table 2, WM outperforms other models under BLEU and perplexity evaluations. On quatrains, WM gets almost three times higher BLEU score than iPoet does. This significant improvement partially lies in that more than 70% of the input topics are expressed[5] in poems generated by WM, benefiting from the topic memory. By contrast, this expression ratio is only 28% for iPoet, since iPoet merges words and history into two single vectors respectively, resulting in implicit and indiscriminate exploitation of topics and history. On iambics, WM also achieves notable performance. Because iambicGen generates the whole iambic as a long sequence by the decoder, it handles short iambics well but fails to generate high-quality longer ones. For iambics with less than 70 characters, perplexity of iambicGen is 235. For those with more characters, perplexity of iambicGen increases to 290. On chinoiserie lyrics, WM also gets better results, though the performance is not so satisfactory (both for WM and lyricGen), due to the small training set.

It is worth mentioning that the improvement partially results from the genre embedding. By incorporating structural and phonological control into the model, WM greatly reduces the uncertainty of generation. To demonstrate the effectiveness of the working memory itself, we show the performance of different strategies of WM in Table 3. As we can see, even without genre embedding, our model still outperforms baselines prominently. Besides, Topic Trace mechanism further improves performance.

Table 4 gives human evaluation results. WM achieves better results than other models. On quatrains, WM gets close to Human on Coherence and Relevance. Planning gets the worst results on Fluency and Meaning. This is mainly because planning mechanism can't guarantee the quality of planned sub-

[4] The experts are Chinese literature students or members of a poetry association. They are required to focus on the quality as objectively as possible, even if they recognize the human-authored ones.

[5] If a topic word or at least one of its relevant words is generated, we say this topic is expressed.

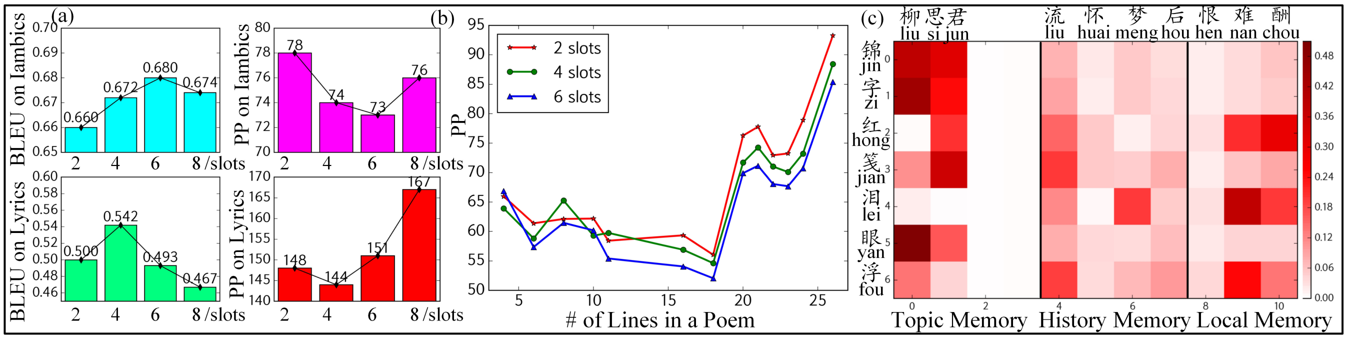

Figure 3: (a) Over different numbers of history memory slots, BLEU and perplexity on iambics and lyrics. (b) On iambics, perplexity over different numbers of lines in a poem. (c) The visualization of memory (in the x-axis) reading probabilities, α_r, when generating the last line (in the y-axis) of the iambic shown in Figure 1.

keywords and the fixed keywords order loses some freedom of topic expression, hurting fluency and meaning. iPoet gets the lowest score on Relevance, since it packs all topic words into one vector, resulting in a low topic expression ratio. By contrast, WM maintains keywords in the topic memory independently and the expression order is flexibly decided by the model in terms of the history. Benefiting from TT, an unexpressed word still has the chance to be generated in later lines. Thus WM gets a comparable score with Human on Relevance. FCPG performs worst on Coherence. As discussed in Section 2, FCPG generates the whole poem as a long sequence and the history is saved in RNN state implicitly, which therefore can't be utilized effectively. On iambics and lyrics, WM gets better results, but there is still a distinct gap with Human. Iambic is a quite complex form and the longest iambic in our testing set consists of more than 150 characters (25 lines). It's much harder for the model to generate a high-quality iambic. For lyrics, due to the limited small data, the results are not as good as we expected. We put the requirements of structure and phonology into Fluency criterion. As a result, WM gets a much higher Fluency score than baselines, benefiting from the genre embedding.

4.5 Analyses and Discussions

We test the performance of WM[6] on different numbers of slots. As shown in Figure 3 (a), both on iambics and lyrics, as the number of slots increases, BLEU gets better first and then deteriorates and so does perplexity. Some lyrics consist of more than 100 lines. More slots should have led to better results on lyrics. However, with the small lyrics corpus, the model can't be trained adequately to operate many slots. Figure 3 (b) gives perplexity over different numbers of lines on iambics. There is little difference for iambics with less than 10 lines. For longer iambics, the model with 6 slots gets better results, though perplexity still increases with more lines.

With too many slots (e.g., infinite slots), our history memory falls back to the second methodology discussed in Section 2. Without any slot, it falls back to the first methodology. The number of memory slots is an important parameter and should be balanced carefully in accordance with the conditions.

In Figure 3 (c), we show an example of how our model focuses on different parts of the memory when generating a line. Our model ignores topic word *liu* (willow) when generating character *hong* (red), since the color of willow is green. The model focuses on topic word *si jun* (missing you) when generation character *jian* (letter), since in ancient China, people often sent their love and missing by letters. Besides, the model generates *lei* (tears) with a strong association with *meng* (dream) in history memory. The word 'dream' is often a symbol to express the pain that a girl is separated from her lover and can only meet him in the dream.

5 Conclusion and Future Work

In this paper, we address the problem of pursuing better coherence in automatic poetry generation. To this end, a generated poem as a whole should be relevant to the topics, express these topics naturally and be coherent in meaning and theme. Inspired by the concept in cognitive psychology, we propose a Working Memory model, which maintains user topics and informative limited history in memory to guide the generation. By dynamical reading and writing during the generation process, our model keeps a coherent information flow and ignores distractions. The experiment results on three different genres of Chinese poetry demonstrate that our model effectively improves the quality and coherence of generated poems to a significant extent.

Besides, combined with a genre embedding, our model is able to generate various genres of poetry. The specially designed Topic Trace mechanism helps the model remember which topics have been used in a more explicit way, further improving the performance.

There still exists a gap between our model and human poets, which indicates that there are lots to do in the future. We plan to design more effective addressing functions and incorporate external knowledge to reinforce the memory.

Acknowledgments

This work is supported by the National 973 Program (No.2014CB340501).

[6] We removed Topic Trace here to observe the influence of the number of slots itself.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*, San Diego, CA, 2015.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar, 2014.

[Fung, 2007] Anthony Y. H. Fung. The emerging (national) popular music culture in china. *InterAsia Cultural Studies*, 8(3):425–437, 2007.

[Ge, 2009] Zai Ge. *cilinzhengyun*. Shanghai Ancient Books Publishing House, 2009.

[Gervás, 2001] Pablo Gervás. *An Expert System for the Composition of Formal Spanish Poetry*, pages 181–188. Springer London, 2001.

[Ghazvininejad *et al.*, 2016] Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Texas, USA, 2016.

[Ghazvininejad *et al.*, 2017] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48. Association for Computational Linguistics, 2017.

[Graves *et al.*, 2014] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[Gulcehre *et al.*, 2017] Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic neural turing machine with soft and hard addressing schemes. *arXiv preprint arXiv:1607.00036*, 2017.

[He *et al.*, 2012] Jing He, Ming Zhou, and Long Jiang. Generating chinese classical poems with statistical machine translation models. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1650–1656, Toronto, Canada, 2012.

[Hopkins and Kiela, 2017] Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 168–178. Association for Computational Linguistics, 2017.

[Manurung, 2003] Hisar Maruli Manurung. *An evolutionary algorithm approach to poetry generation*. PhD thesis, University of Edinburgh, 2003.

[McCutchen, 2000] Deborah McCutchen. Knowledge, processing, and working memory: Implications for a theory of writing. *Educational Psychologist*, 35(1):13–23, 2000.

[Mihalcea and Tarau, 2004] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[Priti and Miyake, 1999] Shah Priti and Akira Miyake. *Models of working memory: Mechanisms of active maintenance and executive control*. Cambridge University Press, 1999.

[Sanders *et al.*, 2001] Ted Sanders, Joost Schilperoord, and Wilbert Spooren. *Text representation: Linguistic and psycholinguistic aspects.*, pages 258–267. John Benjamins Publishing, 2001.

[Shen *et al.*, 2016] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1683–1692. Association for Computational Linguistics, 2016.

[Tu *et al.*, 2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2016.

[Wang *et al.*, 2016a] Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. Chinese song iambics generation with neural attention-based model. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2943–2949, New York, USA, 2016.

[Wang *et al.*, 2016b] Zhe Wang, Wei He, Hua Wu nad Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. Chinese poetry generation with planning based neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics:Technical Papers*, pages 1051–1060, Osaka, Japan, 2016.

[Weston *et al.*, 2015] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *In International Conference on Learning Representations*, San Diego, CA, 2015.

[Yan, 2016] Rui Yan. i,poet:automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2238–2244, New York, USA, 2016.

[Zhang and Lapata, 2014] Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Doha, Qatar, 2014.

[Zhang *et al.*, 2017] Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. Flexible and creative chinese poetry generation using neural memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1364–1373. Association for Computational Linguistics, 2017.