Chinese Poetry Generation with a Salient-Clue Mechanism

Xiaoyuan Yi[1,2,3], **Ruoyu Li**[5], **Maosong Sun**[1,2,4*]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Institute for Artificial Intelligence, Tsinghua University
[3]State Key Lab on Intelligent Technology and Systems, Tsinghua University
[4]Beijing Advanced Innovation Center for Imaging Technology, Capital Normal University
[5]6ESTATES PTE LTD, Singapore
yi-xy16@mails.tsinghua.edu.cn, liruoyu@6estates.com,
sms@mail.tsinghua.edu.cn

Abstract

As a precious part of the human cultural heritage, Chinese poetry has influenced people for generations. Automatic poetry composition is a challenge for AI. In recent years, significant progress has been made in this area benefiting from the development of neural networks. However, the coherence in meaning, theme or even artistic conception for a generated poem as a whole still remains a big problem. In this paper, we propose a novel Salient-Clue mechanism for Chinese poetry generation. Different from previous work which tried to exploit all the context information, our model selects the most salient characters automatically from each so-far generated line to gradually form a salient clue, which is utilized to guide successive poem generation process so as to eliminate interruptions and improve coherence. Besides, our model can be flexibly extended to control the generated poem in different aspects, for example, poetry style, which further enhances the coherence. Experimental results show that our model is very effective, outperforming three strong baselines.

1 Introduction

As a fascinating literary form starting from the Pre-Qin Period, Chinese poetry has influenced people for generations and thus influenced Chinese culture and history in thousands of years. Poets often write poems to record interesting events and express their feelings. In fact, the ability to create high-quality poetry has become an indicator of knowledge, wisdom and elegance of a person in China.

Generally, a Chinese poem should meet two kinds of requirements. One is from the perspective of *form*: it must obey some structural and phonological rules strictly. For example (as shown in Figure 1), quatrain (*Jueju* in Chinese), one of the most popular types of Chinese poetry, contains

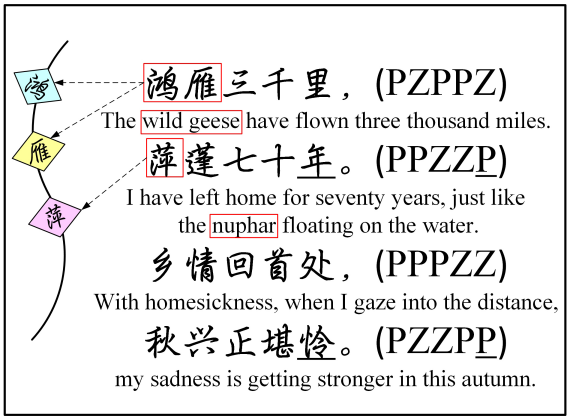

Figure 1: A *Wujue* generated by our model. The tone of each character is given in parentheses, where P and Z represent Ping (level tone) and Ze (oblique tone) respectively. Rhyming characters are underlined. The left part is an artistic illustration of the salient clue.

four lines with each consisting of five or seven characters (called *Wujue* and *Qijue* respectively); characters with particular tone must be in particular positions to make the poem cadenced and full of rhythmic beauty; and, the last character of the first (optional), second and fourth lines must rhyme. The other one is from the perspective of *content*, concerning: (1) if each line of the poem is adequate syntactically and semantically; (2) if the association between two adjacent lines is reasonable; and (3) if the poem as a whole is coherent in meaning, theme or even in artistic conception. Obviously, the second requirement is much more complicated and difficult than the first one.

In this paper, we investigate on automatic Chinese poetry generation, with emphasis on quatrains. We believe the *form* requirement is comparatively easy for a computer to deal with by some constraint checking. For the *content* requirement, point (1) and (2) can be also handled well owing to the use of powerful sequence-to-sequence neural networks (Sutskever et al., 2014), which are capable of producing well-formed tar-

*Corresponding author: sms@mail.tsinghua.edu.cn.

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 241–250
Brussels, Belgium, October 31 - November 1, 2018. ©2018 Association for Computational Linguistics

get sentence given a source sentence. A challenging problem which remains unresolved for researchers is the point (3), where inter-lines associations are 'global' throughout a poem, rather than 'local' in point (2). The relevant experience tells us this is a major reason for the distinct gap between computer-generated poems and those written by poets. In fact, most previous models don't tackle this problem well and will produce incoherences and inconsistencies in generated poems.

Inter-lines coherence is the main concern of this paper. Intuitively, there should be a clear clue to keep the theme of a poem consistent. However, setting a fixed pattern of the clue in advance, e.g, pre-determining keywords for each line, may lose the flexibility and imagination, which are essential for poetry. When writing a poem, human poets will focus on some salient parts of the context to ignore distractions and create relevant content. During this process, poets gradually build a salient clue (or framework) of the poem (Zhang, 2015), allowing not only coherence but also some flexibility.

Inspired by this, we propose a novel Salient-Clue Mechanism for poetry generation. Different from previous models which tried to exploit all the context, our model chooses a few salient characters out of each previously generated line, forming a vital clue for generating succeeding lines, so as to maintain the coherence of the whole poem to the maximum extent. In addition, owing to the flexible structure of our model, extra useful information (e.g., the user intent and poetry style) can be incorporated with the salient clue to control the generation process, further enhancing coherence.

Contributions of this work are as follows:

- To the best of our knowledge, we first propose to utilize the salient partial context to guide the poetry generation process.
- We extend our model to combine user intent and control the style of generated poems, which further enhance coherence.
- Experimental results show that our model outperforms three strong baselines.

2 Related Work

The research on automatic poetry generation has lasted for decades. The early approaches are based on rules and templates, such as the ASPERA system (Gervás, 2001) and Haiku system (Wu et al., 2009). Genetic algorithms are exploited to improve the quality of generated poems (Manurung, 2003; Levy, 2001). Other approaches are also tried, for instance, Yan et al. (2013) adopt the automatic summarization method. Following the work that successfully applies the Statistical Machine Translation approach (SMT) to the task of Chinese classical couplets generation (Jiang and Zhou, 2008), He et al. (2012) further extend SMT to Chinese classical poetry generation.

In recent years, a big change in research paradigm occurred in this field, that is, the adoption of neural network-based approaches, which have shown great advantages in both English poetry (Hopkins and Kiela, 2017; Ghazvininejad et al., 2017) and Chinese poetry generation, as well as other generation tasks. Context coherence is essential for text generation. In some related tasks, researchers have taken a step towards this goal, for example, the discourse Neural Machine Translation (NMT) (Tiedemann and Scherrer, 2017; Maruf and Haffari, 2017; Jean et al., 2017). For poetry generation, some neural models have also recognized the importance of poem coherence. The fundamental issue here is how to define and use the context of a poem properly.

Zhang and Lapata (2014) first propose to generate Chinese poems incrementally with Recurrent Neural Network (RNN), which packs the full context into a single vector by a Convolutional Sentence Model. To enhance coherence, their model needs to be interpolated with two SMT features, as the authors state. Yan (2016) generates Chinese quatrains using two RNNs with an iterative polishing schema, which tries to refine the poem generated in one pass for several times. Yi et al. (2017) utilize neural Encoder-Decoder with attention mechanism (Bahdanau et al., 2015) and trains different models to generate lines in different positions of a poem. Wang et al. (2016) propose a two-stage Chinese classical poetry generation method which at first plans the sub-keywords of the poem, then generates each line sequentially with the allocated sub-keyword. However, the beforehand extracted planning patterns bring some explicit constraints, which may take a risk of losing some degree of flexibility as discussed in Section 1.

These neural network-based approaches are very promising, but there is still large room for exploration. For instance, whether packing the full context into a single vector really represents the

'full' context as well as expected? Can we do better to represent the inter-lines context more properly in pursuing better coherence of the entire generated poem? Our work tries to respond to these questions.

3 Model Design

We begin by formalizing our problem. Suppose a poem P consists of n lines, $P = L_1 L_2 \dots L_n$. Given previous i-1 lines $L_{1:i-1}$, we need to generate the i-th line which is coherent with the context in theme and meaning. Since our model and most baselines are based on a powerful framework first proposed in NMT, that is, the Bidirectional LSTM (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997) Encoder-Decoder with attention mechanism (Bahdanau et al., 2015), we first denote X a line in Encoder, $X = (x_1 x_2 \dots x_T)$, and Y a generated line in Decoder, $Y = (y_1 y_2 \dots y_T)$. T is the length of a line. h_t and $h_t^{'}$ represent the Encoder and Decoder hidden states respectively. $emb(y_{t-1})$ is the word embedding of y_{t-1}. The probability distribution of each character to be generated in the i-th line is calculated by:[1]

$$h_t^{'} = LSTM(h_{t-1}^{'}, emb(y_{t-1}), c_t), \quad (1)$$

$$p(y_t|y_{1:t-1}, L_{1:i-1}) = g(h_t^{'}, emb(y_{t-1}), c_t, v), \quad (2)$$

where g is a normalization function, softmax with a maxout layer (Goodfellow et al., 2013) in this paper. $y_{1:t-1}$ means $y_1, \dots, y_{t-1}$ (similar to $L_{1:i-1}$). c_t is the local context vector in attention mechanism. v is a global context vector. To avoid confusion, in the remainder of this paper when it comes to the word 'context', we all mean the global context, that is, so-far generated lines $L_{1:i-1}$.

Now the key point lies in how to represent and utilize the context for the sake of better coherence. Before presenting the proposed method, we first introduce two basic formalisms of utilizing full context.

3.1 Basic Models

nLto1L

We call the first formalism **nLto1L**, where poetry generation is regarded as a process similar to machine translation. The difference is that the pair in the parallel corpus for NMT models, is changed to the pair of <preceding lines in a poem, a line in a poem> here, which is semantically related rather than semantically equivalent.

The 'n' in nLto1L means at most n preceding lines are concatenated as a long sequence and used simultaneously in Encoder, corresponding to the preceding-lines-in-poem part in the pair, to generate a line in Decoder. In this case, the context is captured by c_t without extra v. (Wang et al., 2016) and (Yi et al., 2017) both belong to this formalism.

The nLto1L formalism is effective, but it has two drawbacks. For one thing, as 'n' increases in nLto1L, more global context can be exploited explicitly by attention, but the number of training pairs decreases, which hurts the generalization performance. For instance, from each quatrain, only one 3Lto1L pair (but two 2Lto1L pairs) can be extracted. For another, when the input sequence is too long, the performance of NMT models will still degrade, even with an attention mechanism (Shen et al., 2016). We find this problem more prominent in poetry generation, since attention may fail to capture all important parts of the context. Our preliminary experiment shows that, regarding generating the fourth line, both for Yi's model and Wang's model, more than 70% of the top three attention values fall into just the third line area and thus neglect other lines, which validates our assumption.

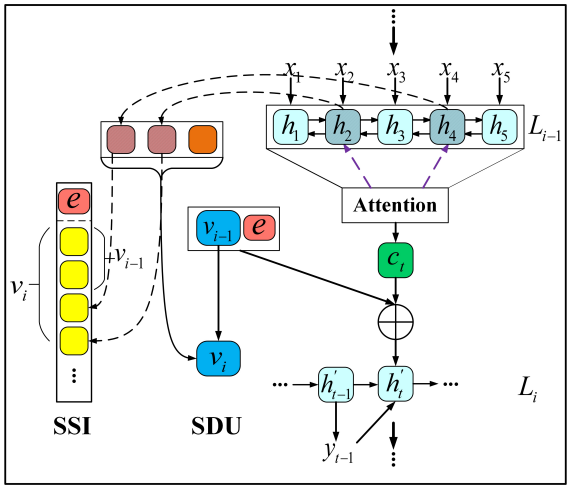

Figure 2: A graphical illustration of the proposed Salient-Clue mechanism. v_i is the salient-clue vector and e is the extension vector. We design two strategies for updating the salient clue. SDU: v_i is kept at the same size; SSI: the size of v_i increases during the generation process.

[1]For brevity, we omit biases and use h_t to represent the combined state of bidirectional LSTM Encoder.

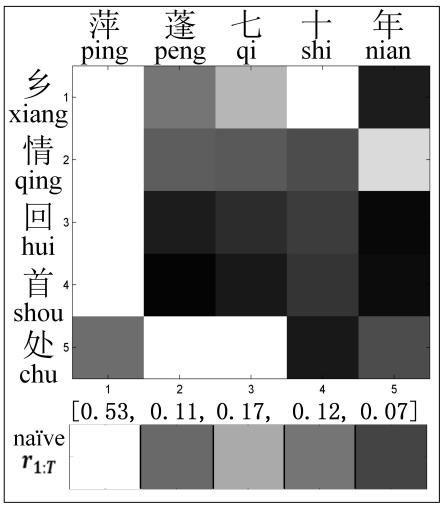

Figure 3: An example of calculating the saliency score of each character (in the x-axis) from the attention matrix (0:black, 1:white), in the naive Salient-Clue. The scores are normalized to interval [0,1] here.

Algorithm 1 Saliency Selection Algorithm

Inputs: The saliency scores of characters in the preceding line, $r_{1:T}$; K;

Outputs: The number of finally selected salient characters, N; The indices of selected characters in the preceding line, $m_{1:N}$;

Calculate the mean value of $r_{1:T}$, avg;
Calculate the standard deviation of $r_{1:T}$, std;
Get sorted indices $i_{1:T}$ in descending order of $r_{1:T}$;
$k = 1$; $val = avg + 0.5 * std$;
while $(r_{i_k} \geq val) \quad and \quad (k \leq K)$ **do**
    $m_k = i_k$; $val = val * 0.618$ (the golden ratio); $k = k + 1$;
end while
$N = k - 1$;
return $N, m_{1:N}$;

Packing Full Context

Another formalism is to pack the full context into a single vector v, which is used to generate successive lines (Zhang and Lapata, 2014; Yan, 2016). Usually, v is updated by the vector of each generated line in a poem. This formalism is not as powerful as we expected. There is still much room for improvement. A single vector doesn't have enough capacity to store all context. Moreover, meaningful words and noises (e.g., stop words) are mixed in one vector, which results in the implicit and indiscriminate utilization of the context.

3.2 The Proposed Salient-Clue Mechanism

As discussed, using the full context directly cannot necessarily lead to the best performance. It becomes clear that we still need to develop a new mechanism to exploit the context in a proper way. Our design philosophy is ignoring the uninformative parts (e.g., stop words) and using some salient characters in context to represent the full context and form a salient clue, which is used to guide the generation process. Following this idea, we propose our Salient-Clue Model.

Naive Salient-Clue

As illustrated in Figure 2, to generate L_i, our model uses standard attention mechanism to exploit L_{i-1} so as to capture short-distance relevance. And it utilizes a salient-clue vector, v_i, to exploit long-distance context. After generating L_i, our model selects up to K (K is 2 for *Wujue* and 3 for *Qijue* in this work) most salient characters from L_{i-1} according to the attention values, and uses their corresponding Encoder hidden states to update the salient clue vector v_i. Thanks to the bidirectional LSTM, even if we only focus on part of the context characters, the information of those unselected won't be lost completely.

Concretely, let A denote the attention alignment matrix in the attention mechanism (Bahdanau et al., 2015) between the preceding line L_{i-1} and the current generated line L_i. We calculate the **saliency score** of j-th character in L_i, r_j, by:

$$r_j = \frac{\sum_{i=1}^{T} A_{ij}}{\sum_{j'=1}^{T} \sum_{i=1}^{T} A_{ij'}}, \tag{3}$$

where A_{ij} is the element in i-th row and j-th column of A. Figure 3 depicts an example. The most salient character is *"ping"* (nuphar, a kind of plant, a symbol of loneliness) and the second one is *"qi"* (seven), according to their saliency scores r(*ping*) = 0.53 and r(*qi*) = 0.17.

The character *"ping"* is very informative for the generated poem but *"qi"* isn't, as signaled by the sharp distinction between their saliency scores. So we design the **Saliency Selection Algorithm 1** to further filter out characters with quite low saliency scores, like *"qi"* here. We define this algorithm as a function $SSal(r_{1:T}, K)$, which takes the saliency scores and the maximum number of selected characters as inputs and outputs the number of finally selected characters and their indices.

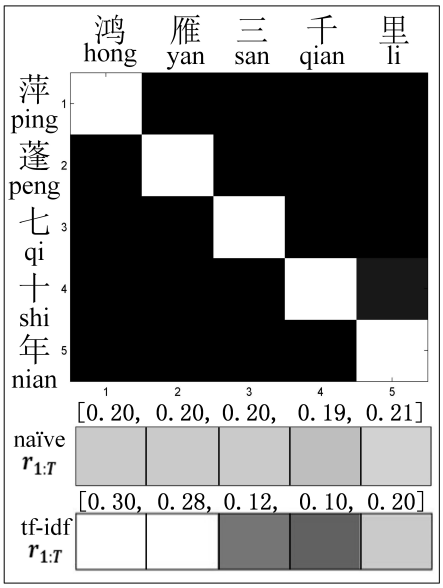

Figure 4: The comparison of saliency scores obtained by the naive model and the tf-idf weighting improved model: an example.

Then we update the salient-clue vector v_i as follows:

$$N, m_{1:N} = SSal(r_{1:T}, K), \tag{4}$$

$$s = \frac{\sum_{k=1}^{N} r_{m_k} * h_{m_k}}{\sum_{k'}^{N} r_{m_{k'}}}, \tag{5}$$

$$v_i = \sigma(v_{i-1}, s), v_0 = \vec{0}, \tag{6}$$

where σ is a non-linear layer. v_{i-1} is used to predict next character to be generated by formula (2). Please note that in formula (4), N may be smaller than K since we want to further ignore relatively less salient characters even though they are already in the list of the K most salient ones.

By the generation process presented above, each generated line is guided by the salient clue and therefore is coherent with it. Meanwhile, informative parts of each generated line are selected and maintained in the salient clue. As a result, the salient clue vector always keeps a coherent information flow, playing a role of a dynamically and incrementally built framework (skeleton) for the generated poem.

TF-IDF Weighted Attention

Observe an example in Figure 4. The scores $r_{1:T}$ given by the Naive Salient-Clue are very close to each other, not distinguishable in saliency. To cope with this, we further take into account the importance of characters both in the preceding line and the current generated line, by the traditional tf-idf scheme in Information Retrieval:

$$r_j = [(\mathbf{w}_{out} * A) \odot \mathbf{w}_{in}]_j, \tag{7}$$

where $\odot$ is element-wise multiplication and $[\cdot]_j$ is the j-th element in a vector. $\mathbf{w}_{in} \in \mathbb{R}^{1*T}$ is the tf-idf vector of preceding (input) line and the i-th element of it is the tf-idf value of i-th character. Similarly, $\mathbf{w}_{out}$ is the tf-idf vector of the current generated (output) line. Elements in $\mathbf{w}_{in}$ and $\mathbf{w}_{out}$ are normalized to [0,1].

As shown in Figure 4, by tf-idf weighting, two informative characters *"hong yan"* (wild goose, a symbol of autumn) are selected correctly, which leads to the generation of word *"qiu xing"* (sadness in autumn) in the fourth line in Figure 1.

Two Strategies for Salient-Clue

As shown in Figure 2, we use two strategies to form and utilize the salient-clue vector v_i. The first is called **Saliency Dynamic Update (SDU)** by formula (5) and (6), which means that hidden states of selected salient characters are packed into v_i. Thus v_i is kept at the same size and is updated dynamically after each line is generated.

The second one is the concatenation of these hidden states:

$$v_i = [v_{i-1}; h_{m_1}; ...; h_{m_N}], \tag{8}$$

where $[;]$ means vector concatenation. The size of v_i will increase in the generation process. We call this **Saliency Sensitive Identity (SSI)**, because the identity of each hidden state is kept independent, without being merged as one.

3.3 Extensions of Salient-Clue

Above we design different methods to select the salient partial context. Since the proposed model is quite flexible, aside from the selected characters, other information can be also utilized to form the salient clue, so as to further improve coherence. In this paper, we tried and evaluated two extensions: user intent and poetry style. This extra information is vectorized as an extension vector e and then concatenated to the salient clue vector:

$$p(y_t|y_{1:t-1}, L_{1:i-1}) = g(h'_t, emb(y_{t-1}), c_t, [v_{i-1}; e]). \tag{9}$$

Intent Salient-Clue. The poem is generated with a user intent keyword. Taking user intent into

	Models	*Wujue*	*Qijue*
Different Models	Planning	0.460	0.554
	iPoet	0.502	0.591
	seq2seqPG	0.466	0.620
	SC	**0.532**	**0.669**
Different Strategies of SC	naive-TopK-SDU	0.442	0.608
	naive-SSal-SDU	0.471	0.610
	tfidf-SSal-SDU	**0.533**	0.648
	tfidf-SSal-SSI	0.530	0.667
	tfidf-SSal-SSI-intent	0.532	**0.669**

Table 1: BLEU evaluation results. The scores are calculated by the multi-bleu.perl script.

account can prevent later generated lines diverging from earlier generated ones in a poem. In detail, we feed the keyword into Encoder, then vector e is calculated by a non-linear transformation of the average of their hidden states.

Style Salient-Clue. The style of generated poems can also benefit coherence. Here we simply use a style embedding as the vector e, which provides a high-level indicator of style and is learned during the training process. It is noteworthy that Zhang et al. (2017) achieve style transfer with the help of an external memory, which stores hundreds of poems (thus thousands of hidden states). By contrast, our Style extension is simpler but still achieves comparable performance.

4 Experiments and Evaluations

4.1 Data and Setups

Our corpus contains 165,800 poems (half *Wujue* and half *Qijue*). We use 4,000 of them for validation, 4,000 for testing and other ones for training. From each poem, a keyword is extracted by tf-idf.

The sizes of word embedding, hidden state, saline-clue vector, intent vector and style embedding are set to 256, 512, 512, 128 and 64 respectively. For SSI, to reduce the model size, we map each hidden state to a 100-d vector by a non-linear transformation. Encoder and Decoder share the same word embedding. All different strategies of Salient-Clue are used both in training and generation. The optimization objective is standard cross entropy errors of the predicted character distribution and the actual one. Adam (Kingma and Ba, 2015) with shuffled mini-batches (batch size 64) is used. Then we use beam search (beam size 20) to generate each poem, with a keyword as input. For fairness, all baselines use the same configuration.

For Style Salient-Clue model, we first use LDA (Blei et al., 2003) to train the whole corpus (with 15 topics). We select three main styles in Chinese poetry: Pastoral, Battlefield and Romantic and find the corresponding topics manually. Then all poems are labeled by LDA inference. We select 5,000 poems for each style, together with other 5,000 non-style poems (20,000 in total), to fine-tune a pre-trained normal Salient-Clue model.

4.2 Models for Comparisons

We compare: **iPoet** (Yan, 2016), **seq2seqPG** (Yi et al., 2017), **Planning** (Wang et al., 2016), **SC** (our model, tfidf-SSal-SSI-intent, which is the best configure under BLEU evaluation), **Style-SC** (the style extension of our model) and **Human** (poems created by human poets). We choose these three previous models as our 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), and prove that their models outperform baselines. Besides, the three models can be classified into the two formalisms in Section 3.1.

4.3 Evaluation Design

To demonstrate the effectiveness of our model, we conduct four evaluations:

BLEU Evaluation. Following (He et al., 2012; Zhang and Lapata, 2014; Yan, 2016), we use BLEU (Papineni et al., 2002) to evaluate our model. BLEU isn't a perfect metric for generated poems, but in the scenario of pursuing better coherence, it still makes sense to some extent. Because higher BLEU scores indicate that the model can generate more n-grams of ground-truth, which certainly have better coherence.

Human Evaluation. Following (Manurung, 2003; Zhang and Lapata, 2014), we design five criteria: **Fluency** (are the lines fluent and well-formed?), **Coherence** (is the theme of the whole quatrain consistent?), **Meaningfulness** (does the poem convey some certain messages?), **Poeticness** (does the poem have some poetic features?), **Entirety** (the reader's general impression on the poem). Each criterion needs to be scored in a 5-point scale ranging from 1 to 5.

We select 20 typical keywords and generate two quatrains (one *Wujue* and one *Qijue*) for each keyword using these models. For Human, we select quatrains containing the given keywords. Therefore, we obtain 240 quatrains (20*6*2) in total.

Models	Fluency		Coherence		Meaningfulness		Poeticness		Entirety	
	Wujue	*Qijue*	*Wujue*	*Qijue*	*Wujue*	*Qijue*	*Wujue*	*Qijue*	*Wujue*	*Qijue*
Planning	2.56	2.84	2.50	2.64	2.49	2.64	2.59	2.88	2.39	2.66
iPoet	3.13	3.45	2.89	2.91	2.60	2.80	2.79	3.05	2.54	2.85
seq2seqPG	3.54	3.65	3.31	3.16	3.15	3.01	3.26	3.29	3.06	3.08
SC	4.01^{**}	4.04^{**}	3.85^{**}	3.86^{**}	3.55^{**}	3.63^{**}	$\mathbf{3.74}^{**}$	$\mathbf{3.69}^{*}$	3.63^{**}	$\mathbf{3.70}^{**}$
Style-SC	$\mathbf{4.03}^{**}$	$\mathbf{4.16}^{**}$	$\mathbf{3.90}^{**}$	$\mathbf{4.01}^{**}$	$\mathbf{3.68}^{**}$	$\mathbf{3.75}^{**}$	3.61^{*}	3.68^{*}	$\mathbf{3.65}^{**}$	$\mathbf{3.70}^{**}$
Human	4.09	4.43	3.90	4.33^{+}	3.94	4.35^{++}	3.83	4.24^{++}	3.81	4.24^{++}

Table 2: Human evaluation results. Diacritics * ($p < 0.05$) and ** ($p < 0.01$) indicates SC models significantly outperform the three baselines; + ($p < 0.05$) and ++ ($p < 0.01$) indicates Human is significantly better than all the five models. The Intraclass Correlation Coefficient of the four groups of scores is 0.596, which indicates an acceptable inter-annotator agreement.

We invite 12 experts on Chinese poetry to evaluate these quatrains, who are Chinese literature students or members of a poetry association. Experts are divided into four groups and required to focus on the quality as objectively as possible, even if they recognize the human-authored ones. Each group completes the evaluation of all poems and we use the average scores.

Style-SC is not suitable for BLEU, because we can't expect LDA to predict a correct style label by a short keyword. Thus Style-SC is only tested under Human Evaluation. We label each keyword with an appropriate style manually, which is used to guide the generation.

Style Control Evaluation. Poetry style is usually coupled with content. Not all keywords are compatible with every style. Therefore we select ten normal keywords without obvious style (e.g., moon and wind). We use SC to generate one poem and use Style-SC to generate three poems with the three specified styles. The experts are asked to identify the style of each poem from four options (Unknown, Battlefield, Romantic and Pastoral).

Saliency Selection Evaluation. The main idea of our method is to select the salient partial context to guide successive generation. To evaluate the reasonableness of selected characters, we randomly select 20 *Wujue*s and 20 *Qijue*s from the test set. Then three experts are asked to select up to K salient characters from each line. When experts have different opinions, they stop and discuss until reaching an agreement. **Jaccard similarity** is used to measure the overlap of human-selected characters and the model-selected ones.

4.4 Evaluation Results and Discussion

As shown in Table 1, our SC outperforms other models under BLEU evaluation. We also compare different strategies of SC. As we expected, tfidf-SC models outperform the naive ones, because tf-idf values lower the weights of uninformative characters. We also compare our SSal 1 with TopK (just select top K salient characters) and SSal gets better results. Please note that, from naive-TopK-SDU to tfidf-SSal-SDU, BLEU scores are getting higher without any increase of model size (Table 4). SSI is better on *Qijue*, but performs slightly worse than SDU on *Wujue*. We use SSI for Human evaluation here, but SDU is more suitable for longer poetry, e.g., Chinese Song Iambics. Besides, the intent extension makes a little bit of improvement, not as prominent as we expected. We think the reason may lie in that the keyword selected by tf-idf can't accurately represent the user intent. Generally, the results show both for packing and concatenation formalisms, the proper utilization of partial salient context (SDU and SSI) can be better than the improper utilization of full context (Packing Full Context and nLto1L).

Table 2 gives human evaluation results. SC and Style-SC achieve better results than other models and get close to Human, though there is still a gap. Especially on Coherence, our Style-SC gets the same score as Human for *Wujue*. Moreover, Style-SC makes a considerable improvement on Coherence compared to SC (+0.05 for *Wujue* and +0.15 for *Qijue*), which demonstrates that consistent style can actually enhance the coherence, though it's not easy to predict an appropriate style automatically. Interestingly, Style-SC outperforms SC on most criteria, except for Poeticness. We believe this is mainly because that style control forces the model to always generate some style-related words, which limits the imagination and thus hurts the Poeticness.

Besides, as we can see, seq2seqPG outperforms other two baselines, but at the expense that it is three times the model size of iPoet (Table 4). Surprisingly, Planning gets the worst re-

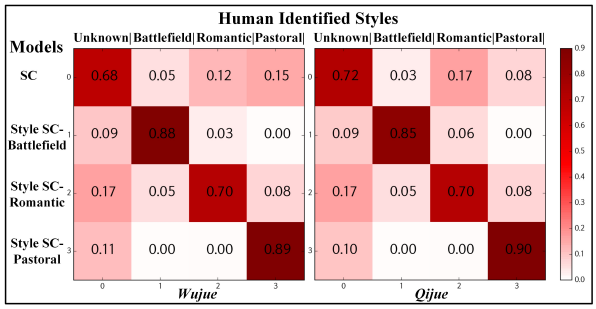

Figure 5: Style control evaluation results. The values are ratios that generated poems are identified as different styles by human experts.

Models	*Wujue*	*Qijue*
Random	0.271	0.247
tf-idf	0.417	0.378
naive-TopK SC	0.347	0.415
naive-SSal SC	0.431	0.441
tfidf-SSal SC	**0.525**	**0.461**

Table 3: Saliency selection results. Random: randomly select K characters for three times and use the average Jaccard values. tf-idf: directly select K characters in terms of tf-idf, without SC.

sults. This is because that Planning uses a neural language model to generate the planned sub-keywords, which performs poorly on our corpus and thus hurts fluency and coherence.

Figure 5 gives style control evaluation results. Incorporating one style vector with the salient-clue, our Style-SC achieves comparable performance with (Zhang et al., 2017). The good results partially lie in that we only use those non-style keywords, such as moon, wind, water and so on, for this experiment. Empirically, transferring from a keyword with obvious style to arbitrary style is intractable, e.g., from the word 'army' to Pastoral style, which may need more complicated model design. Even so, our results still show that rough style control is not as difficult as we thought.

Table 3 shows the effectiveness of our salient character selection methods. tfidf-SSal achieves about 50% overlap of human-selected salient characters, which is notably higher than Random and tf-idf. With only attention values, naive-TopK performs worse than tf-idf on *Wujue*. Combining tf-idf and SSal makes notable improvement.

Models	Innovation	Param	Speed
Planning	**0.039**	15.1	2.49
iPoet	0.044	**11.0**	1.76
seq2seqPG	0.047	37.1	1.85
SC Model	0.041	13.6	**1.57**
naive-TopK-SDU	0.058	13.7	1.49
naive-SSal-SDU	0.056	13.7	1.51
tfidf-SSal-SDU	0.042	13.7	1.52
tfidf-SSal-SSI	0.043	**13.1**	**1.48**
tfidf-SSal-SSI-intent	**0.041**	13.6	1.57

Table 4: Extra comparisons of different models. Innovation, Param (million parameters) and Speed (seconds per poem) are compared. The generation speed is tested on an Intel CORE i5 4-core CPU.

4.5 Extra Comparisons and Case Study

Besides the metrics above, we also compare innovation, model size and generation speed of different models. The innovation is measured by Jaccard similarity of generated poems (3,000 for each model). Intuitively, the basic requirement for innovation is that poems generated with different keywords should be different with each other.

As shown in Table 4, SC makes a good balance on quality, innovation, generation speed and model size. The iPoet has the smallest size, but the generation is slow, since it may polish the poem for several times, costing more time than other one-pass-generation models. For SC, the use of tf-idf significantly improves innovation. Due to planned sub-keywords, Planning achieves the best innovation but the worst quality, which shows pursuing innovation takes the risk of abruptness and incoherence.

Figure 6 shows two *Wujue*s generated by seq2seqPG and SC respectively, with the same input "Yangzhou City". A word "moon" is generated by seq2seqPG in the first line, which determines the time (at night) of the whole poem. However, in the fourth line, seq2seqPG still generates an inconsistent word "sunset". For the poem generated by SC, the word "autumn" in the second line is selected for successive generation. As a result, a word "fallen leaves" is generated in the fourth line. Furthermore, in the second line, except for "autumn", other four uninformative characters, which have quite low saliency scores, are filtered out by **SSal** 1 as shown at the bottom of Figure 6.

5 Conclusion and Future Work

In this paper, we address the problem of the context coherence in poetry generation. How to prop-

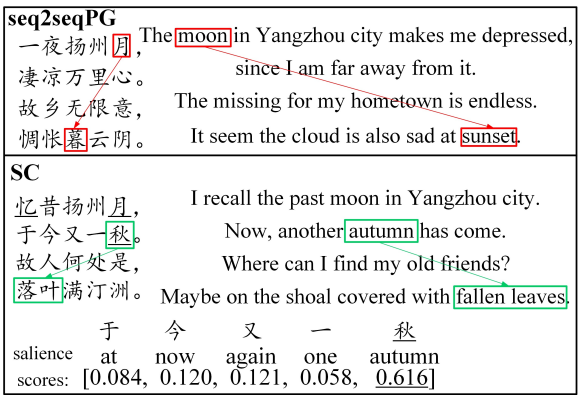

Figure 6: Two *Wujue*s generated with the same input. Green boxes and arrows show consistencies, and the red ones show inconsistencies. Automatically selected charcters by SC are underlined.

erly treat the global context is a key factor to consider. We propose a Salient-Clue mechanism[2]. Our model selects highly salient characters in preceding generated lines to form a representation of the so-far context, which can be considered as a dynamically and incrementally built framework, then uses it to guide the successive generation. Both automatic and human evaluations demonstrate that our model can effectively improve the global coherence of meaning, theme and artistic conception of generated poems. This implies the proper treatment of a partial context could be better than the improper treatment of the full context.

Furthermore, our model can be flexibly combined with different auxiliary information and we show the utilization of style and user intent can further enhance coherence.

There still exists a gap between our model and human poets, which indicates that there are lots to do in the future. Though we experimented on Chinese corpus, the proposed model is genre-free. We also plan to extend our model to generate other types of poetry, such as Chinese Regulated Verse and English sonnet. Besides, we also want to design some explicit supervisory signals or utilize external knowledge to improve the saliency selection algorithm.

Acknowledgments

We would like to thank Cheng Yang, Jiannan Liang, Zhipeng Guo, Huimin Chen, Wenhao Li and anonymous reviewers for their insightful comments. This research is funded by Major Project of the National Social Science Foundation of China (No. 13&ZD190). It is also partially supported by the NExT++ project, the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative.

[2]Based on this work, we build an online poetry generation system, **Jiuge**: https://jiuge.thunlp.cn.

References

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

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *machine Learning research*, (3):993–1022.

Pablo Gervás. 2001. *An Expert System for the Composition of Formal Spanish Poetry*. Springer London.

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

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1319–1327, Atlanta, USA.

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.

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

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

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*.

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.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization.

Robert P. Levy. 2001. A computational model of poetic creativity with neural network as measure of adaptive fitness. In *Proceedings of the ICCBR-01 Workshop on Creative Systems*.

Hisar Maruli Manurung. 2003. *An evolutionary algorithm approach to poetry generation*. Ph.D. thesis, University of Edinburgh.

Sameen Maruf and Gholamreza Haffari. 2017. Document context neural machine translation with memory networks. *arXiv preprint arXiv:1711.03688*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *In Advances in Neural Information Processing Systems 2014*, Montreal, Canada.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark.

Zhe Wang, Wei He, Hua Wu nad Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. 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.

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. 2016. 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.

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.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2017. Generating chinese plassical poems with rnn encoder-decoder. In *Proceedings of the Sixteenth Chinese Computational Linguistics*, pages 211–223, Nanjing, China.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1364–1373. Association for Computational Linguistics.

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

Yingzhong Zhang. 2015. *How to Create Chinese Classical Poetry*. The Commercial Press.