Knowledge-Based Systems 232 (2021) 107409

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A new context-aware approach for automatic Chinese poetry generation

Tian Gao [a], Shanliang Zhu [a,*], Jing Liu [b], Jun Shen [c], Jialie Shen [d], Shuguo Yang [a], Pengcheng Xiong [a]

[a] Research Institute for Mathematics and Interdisciplinary Sciences, School of Mathematics and Physics, Qingdao University of Science and Technology, Qingdao, 266061, China
[b] Weifang University of Science and Technology, Weifang, 262700, China
[c] School of Computing and Information Technology, University of Wollongong, 2522 Wollongong, NSW, Australia
[d] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, UK

ARTICLE INFO

Article history:
Received 21 November 2020
Received in revised form 8 July 2021
Accepted 16 August 2021
Available online 15 September 2021

Keywords:
Poetry generation
Neural networks
Seq2seq
Keyword team

ABSTRACT

Chinese poetry has been a favorite literary genre for thousands of years. Chinese ancient poetry is still being read and practiced, and many famous ancient Chinese poets are honored and adorned. Recently, deep learning has been widely adopted for poetry generation. In this paper, we present a new context-aware Chinese poetry generation method based on sequence-to-sequence framework. We generate a new concept called keyword team, which is a combination of all the keywords to capture the context of the Chinese poetry. Then we use the keyword, the keyword team and the previously generated lines to generate the present line in the poetry. We find that, by including keyword teams into the generation of the poetry, it can additionally perceive the keywords of preceding and succeeding lines to generate the present line, which can effectively improve the adhesion among the overall lines. The comprehensive evaluation results show that our proposed model outperforms many of the state-of-the-art poetry generation models.

1. Introduction

Chinese ancient poetry plays an important role in Chinese classical literature and has a far-reaching influence. Among the classical Chinese poetry, jue ju is a very important part, which is also known as quatrain. A quatrain comprises four lines and each line has five or seven characters, which all need to follow the structure of ancient poetry, tonal and semantic requirements. Each line should meet certain phonological patterns. For example, the last characters of the 2nd and 4th lines must be rhyming, and there is no restriction on the 3rd line. In addition, the poetry must follow the prescribed tone pattern: Ping (the level tone) or Ze (the downward tone). Classical Chinese poetry has always high requirements on semantics. Every ancient poem conveyed a poet's feelings. Thus readers can feel the poet's state of mind and inner feelings. Table 1 shows one of the most popular quatrain in China.

Automatic poetry generation has been studied for decades. Early studies are based on rules and templates, such as [1–4]. Later on, Statistical Machine Translation (SMT) methods were used [5]. In recent years, deep learning has become the mainstream approach and it considers the poetry generation as a sequence-to-sequence (seq2seq) [6] generation problem [7–12]. Some studies use the Recurrent Neural Network Language Models (RNNLM) [13] and the Recurrent Neural Network Poetry Generation (RNNPG) models [7] to generate ancient poems. These methods mainly suffer from theme drift and the generated poetry are neither fluent nor coherent. To overcome such a problem, many methods are used to improve those works. For instance, methods based on planning [11], polishing schema [9] and conditional variational auto-encoder [14] were investigated. There are also some poetry generation systems [15,16] integrating multiple methods together to generate ancient poetry.

One of the most prominent methods is based on Planning Poetry Generation (PPG) model [11]. This model uses the same number of keywords as the number of the lines of poem (e.g., four keywords for four lines of a quatrain) to generate a poem. More specifically, the 1st keyword is used to generate the 1st line of the poem, and then the subsequent keywords and the previously

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: Research Institute for Mathematics and Interdisciplinary Sciences, School of Mathematics and Physics, Qingdao University of Science and Technology, Qingdao, 266061, China.
E-mail address: zhushanliang@qust.edu.cn (S. Zhu).

https://doi.org/10.1016/j.knosys.2021.107409
0950-7051/

Table 1
Take a Tang poetry as an example. Its tone is shown at the end of each line. P indicates the level tone, Z indicates the downward tone. * represents any one of them and the rhyming characters are boldface.

静夜思	Thoughts in a Still Night
床前明月**光**, (P P Z Z P)	The luminous moonshine before my bed,
疑是地上**霜**. (* Z Z P P)	Is thought to be the frost fallen on the ground.
举头望明月, (* Z P P Z)	I lift my head to gaze at the cliff moon,
低头思故**乡**. (P P Z Z P)	And then bow down to muse on my hometown.

generated lines are used to generate subsequent lines. Taking a quatrain in Table 1 as an example, suppose that the 1st to 4th keywords are "明月光" ("luminous moonshine"), "地上" ("on the ground"), "举头" ("lift head") and "故乡" ("distant home") as shown in Fig. 1, respectively. The model first uses the 1st keyword "明月光" ("luminous moonshine") to generate the 1st line "床前明月光" ("The luminous moonshine before my bed".). Then the model uses the 2nd keyword "地上" ("on the ground") and the preceding line "床前明月光" ("The luminous moonshine before my bed".) to generate the 2nd line "疑是地上霜" ("Is thought to be the on the ground fallen on the ground".).

Although a poet will refer to the keyword and the preceding lines when the poet is generating the current line, the poet often habitually refers to the context of the whole poetry, i.e., not only the keyword of this line, but also the keywords of other lines. Inspired by this, we extend the PPG model [11] by including more context. We invent a new concept called keyword team, which is composed of all the four keywords. The keywords in a keyword team is sorted according to their distance to the line that we want to generate. Based on keyword team, we propose a new poetry generation model called Keyword-team Poetry Generation (KPG) model.

For example, suppose that we have four keywords "明月光" ("luminous moonshine"), "地上" ("on the ground"), "举头" ("lift head") and "故乡" ("distant home") as shown in Fig. 1 for four lines of the poetry that we want to generate. For the 1st keyword team "明月光, 地上, 举头, 故乡" ("luminous moonshine, on the ground, lift head, distant home") for the 1st line, we place the 1st keyword "明月光" ("luminous moonshine") in the first place, followed by the 2nd keyword "地上" ("on the ground"), the 3rd keyword "举头" ("lift head") and the 4th keyword "故乡" ("distant home") according to their distance to the 1st line.

We use the 1st keyword team "明月光, 地上, 举头, 故乡" ("luminous moonshine, on the ground, lift head, distant home") and the 1st keyword "明月光" to generate the 1st line "窗前明月光" ("The luminous moonshine before my bed".). We use the 2nd keyword team "地上, 明月光, 举头, 故乡" ("on the ground, luminous moonshine, lift head, distant home"), the 2nd keyword "地上" ("on the ground") and the preceding line "床前明月光" ("The luminous moonshine before my bed".) to generate the 2nd line "疑是地上霜" ("Is thought to be the on the ground fallen on the ground".). By including a keyword team which is composed of all the keywords, we always refer to the context of the whole poetry, which mimics a poet's writing habit and pattern.

The contributions of this paper are as follows:

- Based on a poet's writing habit (all keywords will be considered in the generation of each line in a poem), we extend the sequence-to-sequence (seq2seq) model based on attention [17] and creatively introduce keyword teams into the generation of ancient poetry.
- We have conducted extensive experiments to prove that our model is better than the state-of-the-art one [11] by using a loss function as well as human evaluation based on expert evaluators. The experiments show that our model is superior from coherency and meaningfulness perspectives.

Table 2
Summary of symbols and definitions.

Notation	Definition
N	The number of lines in a poem
$\mathcal{L}_i$	The ith line of a poem
$\mathcal{L}_{1:i-1}$	The previous i-1 lines of poem
K_i	The keyword for the ith line
M_i	The keyword team for the ith line
$S(V_i)$	The TextRank score
$E(V_i)$	A set of vertices connected with V_i
w_{ij}	The weight of edge ij
V_i	A candidate word i
$\overrightarrow{\mathbf{h}_t}$	A hidden state of the forward LSTM in time t
$\overleftarrow{\mathbf{h}_t}$	A hidden state of the backward LSTM in time t
$\mathbf{h}_t$	A hidden state of the Bi-directional LSTM in time t
$\mathbf{c}_t$	The attention vector in each time t
$\mathbf{s}_t$	The decoder hidden state in each time t
$\hat{\mathbf{y}}_t$	The output character by decoder in time t
$\mathbf{s}_1$	The initial hidden state of decoder
$\mathbf{r}_k$	The keyword information in KPG model
$\mathbf{r}_m$	The keyword team information in KPG model
$\mathbf{W}$, $\mathbf{v}$	A weight matrix to be learned in the neural network

The rest of this paper is organized as follows. Section 2 describes our poetry generation model. Section 3 introduces the experimental design. Section 4 introduces the experimental results. Section 5 describes some previous work on poetry generation. Finally, Section 6 summarizes the paper and looks forward to the future work.

2. The proposed Keyword team based Poetry Generation (KPG) model

Suppose that a poem P consists of N lines with $\mathcal{L}_i$ representing the ith line of the poem. The previous i-1 lines in a poem are defined as $\mathcal{L}_{1:i-1}$ ($i = 1, 2, \ldots, N$). The keyword of the ith line is defined as K_i ($i = 1, 2, \ldots, N$). The keyword team for the ith line consisting of N keywords is defined as M_i ($i = 1, 2, \ldots, N$). Table 2 presents the major notations used in this paper.

In our Keyword team based Poetry Generation (KPG) model, the $\mathcal{L}_i$ is generated by using K_i, M_i and $\mathcal{L}_{1:i-1}$. Especially, the $\mathcal{L}_1$ is generated by using only K_1, M_1. Fig. 2 shows the process of generating the 4th line of a poem in Table 1 according to our model. We can see that our poetry generation model consists of four key components: a preceding lines encoder, a keyword encoder, a keyword team encoder and a decoder.

We put a stop token (eos) at the end of each keyword, keyword team and the preceding lines. The input of the preceding lines encoder is the previously generated lines $\mathcal{L}_{1:i-1} = \alpha_1, \alpha_2, \ldots, \alpha_{T_x}$ (e.g., "The luminous...I lift my head to gaze at the cliff moon eos"), where the α_i ($i = 1, 2, \ldots, T_x$) means the ith character in the preceding lines. The input of the keyword encoder is the keyword $K_i = \beta_1, \beta_2, \ldots, \beta_{T_k}$ (e.g., "distant home eos"), where the β_i ($i = 1, 2, \ldots, T_k$) means the ith character in the keyword. The input of keyword team encoder is the keyword team M_i for the ith line, where M_i consists of N keywords. M_i can also be expressed as $\gamma_1, \gamma_2, \ldots, \gamma_{T_m}$ (e.g., "distant home ...luminous moonshine eos"), where γ_i ($i = 1, 2, \ldots, T_m$) means the ith character in the keyword team. The output line is $\mathcal{L}_i = \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{T_n}$ (e.g., "And the bow down to muse on my distant home eos") character by character according to the decoder, where the $\hat{y}_i$ ($i = 1, 2, \ldots, T_n$) means the ith generated character.

	The start-of-the-art poetry generation model (PPG)		Our contribution
The Current Lines	Keywords	The Preceding Lines	Keyword Teams
床前明月光 The luminous moonshine before my bed.	明月光 luminous moonshine	——	明月光，地上，举头，故乡 luminous moonshine, on the ground, lift head, distant home
疑是地上霜 Is thought to be the frost fallen on the ground.	地上 on the ground	床前明月光 The luminous moonshine before my bed.	地上，明月光，举头，故乡 on the ground, luminous moonshine, lift head, distant home
举头望明月 I lift my head to gaze at the cliff moon.	举头 lift head	床前明月光；疑是地上霜 The luminous moonshine before my bed, is thought to be the frost fallen on the ground.	举头，故乡，地上，明月光 lift head, distant home, on the ground, luminous moonshine
低头思故乡 And then bow down to muse on my distant home.	故乡 distant home	床前明月光；疑是地上霜；举头望明月 The luminous moonshine before my bed, is thought to be the frost fallen on the ground. I lift my head to gaze at the cliff moon.	故乡，举头，地上，明月光， distant home, lift head, on the ground, luminous moonshine

Fig. 1. Training dataset extracted from the quatrain in Table 1.

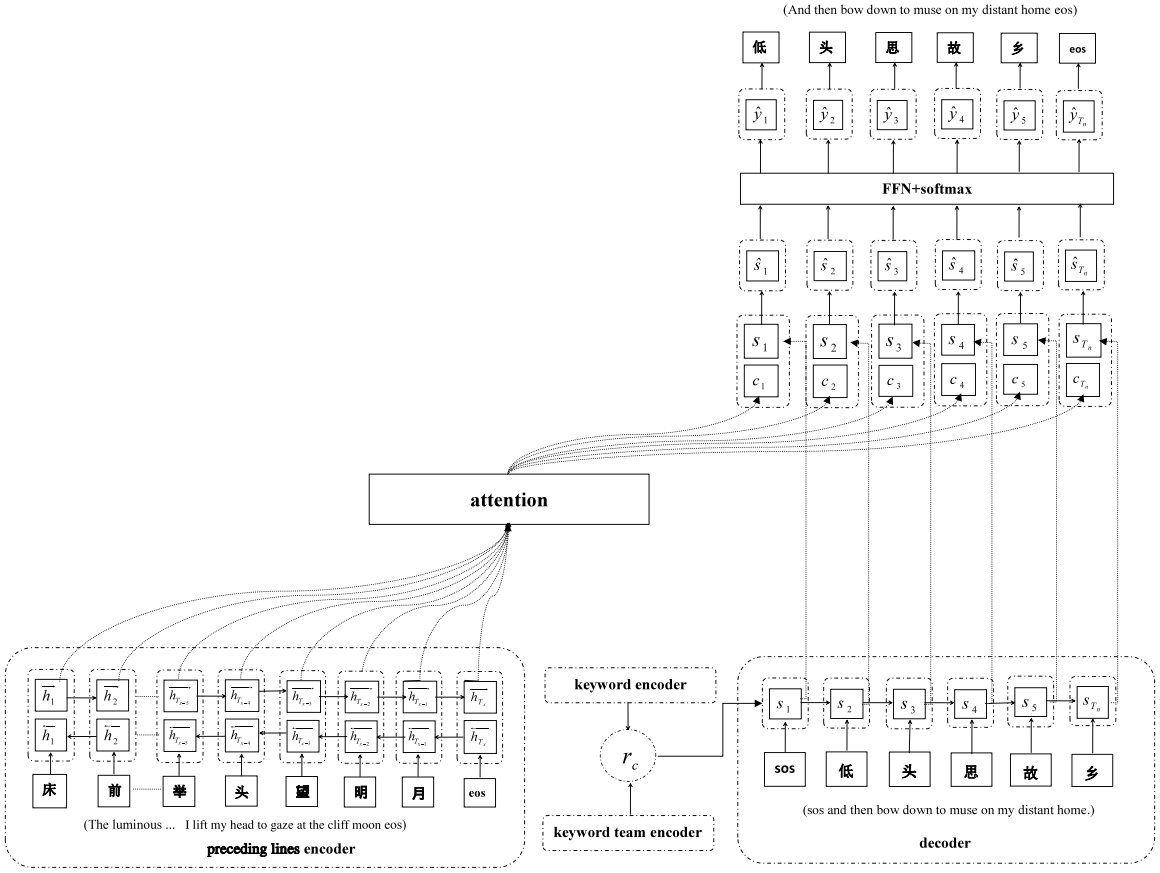

Fig. 2. Illustration of the Keyword team Poetry Generation (KPG) model.

2.1. Character embedding

We use character embedding for our training purpose. We first create a poetry vocabulary which is composed of the most frequent characters in the training dataset and two special tokens (i.e., a stop token (eos) and a start token (sos)). Then we map each character in the poetry vocabulary to a vector which is called character embedding. The dimension of the poetry vocabulary is $[d, q]$, where d represents the size of poetry vocabulary, q represents the dimension of character embedding.

The word2vec model (based on the CBOW algorithm [18]) can take context into account and has fewer dimensions, which makes it has better embedding effect and faster generation speed than other earlier embedding methods. Therefore, we choose the word2vec model to generate the character embedding.

2.2. The preceding lines encoder

The preceding lines encoder is a recurrent neural network, using the Bi-directional Long Short Team Memory (LSTM) units [19]. A Bi-directional LSTM consists of two parallel LSTMs [20], i.e., a forward LSTM and a backward LSTM. The other encoders, i.e., the keyword encoder and the keyword team encoder, are also the recurrent neural networks based on the Bi-directional Long Short Team Memory (LSTM) units.

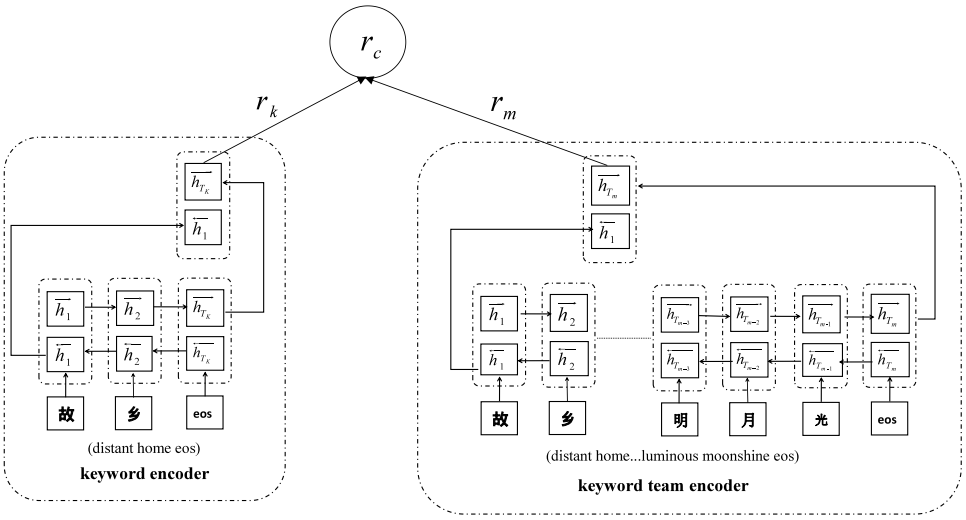

Fig. 3. Illustration of the keyword encoder and the keyword team encoder.

For the preceding lines encoder, we set the size of hidden state in every parallel LSTM as n. The forward LSTM encodes preceding lines $\alpha_1, \alpha_2, \ldots, \alpha_{T_x}$ (e.g., "The luminous...I lift head to gaze at the cliff moon eos") into a hidden state sequence $\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_{T_x}$ from left to right, where the size of $\overrightarrow{\mathbf{h}}_i$ $(i = 1, 2, \ldots, T_x)$ is $[n,1]$. The backward LSTM encodes the preceding lines $\alpha_1, \alpha_2, \ldots, \alpha_{T_x}$ (e.g., "The luminous...I lift head to gaze at the cliff moon eos") into a hidden state sequence $\overleftarrow{\mathbf{h}}_{T_x}, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_1$ from right to left, where the size of $\overleftarrow{\mathbf{h}}_i$ $(i = 1, 2, \ldots, T_x)$ is $[n,1]$.

For each time step t, we can consider $\mathbf{h}_t$ as the hidden state of the Bi-directional LSTM, where $\mathbf{h}_t$ is the concatenate of a forward hidden state $\overrightarrow{\mathbf{h}}_t$ and a backward hidden state $\overleftarrow{\mathbf{h}}_t$. The motivation is to include both the preceding and the following characters in the hidden state of a character.

$$\mathbf{h}_t = \begin{bmatrix} \overrightarrow{\mathbf{h}}_t \\ \overleftarrow{\mathbf{h}}_t \end{bmatrix} \tag{1}$$

Based on Eq. (1), we can get a hidden state sequence $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}$ by the preceding lines encoder, where the size of $\mathbf{h}_i$ $(i = 1, 2, \ldots, T_x)$ is $[m,1]$ $(m = 2n)$.

2.3. The keyword encoder

For the keyword encoder, we set the size of hidden state in every parallel LSTM as v $(v = n/2)$. The forward LSTM encodes a keyword $\beta_1, \beta_2, \ldots, \beta_{T_k}$ (e.g., "distant home") into a hidden state sequence $\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_{T_k}$ from left to right, where the size of $\overrightarrow{\mathbf{h}}_i$ $(i = 1, 2, \ldots, T_k)$ is $[v,1]$. The backward LSTM encodes the keyword $\beta_1, \beta_2, \ldots, \beta_{T_k}$ (e.g., "distant home") into a hidden state sequence $\overleftarrow{\mathbf{h}}_{T_k}, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_1$ from right to left, where the size of $\overleftarrow{\mathbf{h}}_i$ $(i = 1, 2, \ldots, T_k)$ is $[v,1]$. We can concatenate the last forward hidden state of the forward LSTM, i.e., $\overrightarrow{\mathbf{h}}_{T_k}$, and the first backward hidden state of the backward LSTM, i.e., $\overleftarrow{\mathbf{h}}_1$, to represent the keyword information. As illustrated in Fig. 3, We can get a vector $\mathbf{r}_k$ whose size is $[n,1]$, where

$$\mathbf{r}_k = \begin{bmatrix} \overrightarrow{\mathbf{h}}_{T_k} \\ \overleftarrow{\mathbf{h}}_1 \end{bmatrix} \tag{2}$$

2.4. The keyword team encoder

The keyword team encoder is used to encode a keyword team (e.g., "distant home...luminous moonshine") of a line and generate a hidden state to represent the keyword team information. For a quatrain which has four lines, we suppose that the 1st to 4th keywords for each line are K_1, K_2, K_3, K_4, respectively. According to these keywords, we can generate the keyword teams for each line into a poem. The keyword team of a line is made up of these four keywords, i.e., K_1, K_2, K_3, K_4 according to their distance to the current line as shown in Table 3 .

Table 3
The sequence of the keyword teams for a poem.

The ith line	Keywords	Keyword teams
The 1st line	K_1	K_1, K_2, K_3, K_4
The 2nd line	K_2	K_2, K_1, K_3, K_4
The 3rd line	K_3	K_3, K_4, K_2, K_1
The 4th line	K_4	K_4, K_3, K_2, K_1

For the keyword team encoder, we set the size of hidden state in every parallel LSTM as v $(v = n/2)$. The forward LSTM encodes a keyword team $\gamma_1, \gamma_2, \ldots, \gamma_{T_m}$ (e.g., "distant home ...luminous moonshine eos") into a hidden state sequence $\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_{T_m}$ from left to right, where the size of $\overrightarrow{\mathbf{h}}_i$ $(i = 1, 2, \ldots, T_m)$ is $[v,1]$. The backward LSTM encodes the keyword team $\gamma_1, \gamma_2, \ldots, \gamma_{T_m}$ (e.g., "distant home ...luminous moonshine eos") into a hidden state sequence $\overleftarrow{\mathbf{h}}_{T_m}, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_1$ from right to left, where the size of $\overleftarrow{\mathbf{h}}_i$ $(i = 1, 2, \ldots, T_m)$ is $[v,1]$. We can get a vector $\mathbf{r}_m$ whose size is $[n,1]$. The vector $\mathbf{r}_m$ represents the keyword team information, where

$$\mathbf{r}_m = \begin{bmatrix} \overrightarrow{\mathbf{h}}_{T_m} \\ \overleftarrow{\mathbf{h}}_1 \end{bmatrix} \tag{3}$$

2.5. The decoder

The decoder is a recurrent neural network, using the Long Short Team Memory (LSTM) units [20]. We set the size of hidden state in the LSTM as m, i.e., the size of s_t $(t=1, 2, \ldots, T_n)$ is $[m,1]$. In the process of decoder's decoding, the output $\hat{\mathbf{y}}_t$ of the previous step is continuously taken as the input of the next step until the output of a stop token (eos). We use a start token (sos) as the initial input for the decoder. For example, we use a start token as input to generate the character "低" (blow). Then, we use the character "低"(blow) to generate the character "头" (head) as shown in Fig. 2.

For the decoder, the output $\hat{\mathbf{y}}_t$ is generated by decoder hidden state $\mathbf{s}_t$ and attention vector $\mathbf{c}_t$ at each step t. This can be computed as follows:

$$\hat{\mathbf{y}}_t = \arg\max P(y|\mathbf{s}_t, \mathbf{c}_t) \tag{4}$$

where $P(\cdot)$ represents the probability of the character $\hat{\mathbf{y}}_t$ given the $\mathbf{s}_t$ and the $\mathbf{c}_t$. The size of $\mathbf{c}_t$ is $[m,1]$.

Our KPG model also includes an attention mechanism [17]. The attention vector $\mathbf{c}_t$ is a weighted sum of the preceding lines encoder with hidden states $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}$:

$$\mathbf{c}_t = \sum_{i=1}^{T_x} \alpha_{t,i}\mathbf{h}_i \tag{5}$$

where $\alpha_{t,i}$ is computed by decoder hidden state $\mathbf{s}_t$ and each preceding lines encoder hidden states $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}$ as below:

$$\alpha_{t,i} = align(\mathbf{s}_t, \mathbf{h}_i) \tag{6}$$

$$= \frac{\exp(score(\mathbf{s}_t, \mathbf{h}_i))}{\sum_{i'=1}^{T_x} \exp(score(\mathbf{s}_t, \mathbf{h}_{i'}))} \tag{7}$$

The score $\alpha_{t,i}$ is calculated based on how well $\mathbf{s}_t$ and $\mathbf{h}_i$ match by the alignment model [17].

$$score(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^T \tanh(\mathbf{W}_a \begin{bmatrix} \mathbf{s}_t \\ \mathbf{h}_i \end{bmatrix}) \tag{8}$$

where $\mathbf{v}_a$ and $\mathbf{W}_a$ are the weight matrices to be learned in the alignment model. The size of $\mathbf{v}_a$ and $\mathbf{W}_a$ are $[z,1]$ (z=2m) and $[z, z]$, respectively.

Next, $\mathbf{c}_t$ is concatenated with the decoder hidden state $\mathbf{s}_t$ in time t:

$$\hat{\mathbf{s}}_t = \tanh(\mathbf{W}_c \begin{bmatrix} \mathbf{c}_t \\ \mathbf{s}_t \end{bmatrix}) \tag{9}$$

where the size of $\begin{bmatrix} \mathbf{c}_t \\ \mathbf{s}_t \end{bmatrix}$ is $[z,1]$. We use a weight matrices $\mathbf{W}_c$ to attain the size of $\hat{\mathbf{s}}_t$ as $[m,1]$. The size of $\mathbf{W}_c$ is $[m, z]$.

Note that when we are generating the 1st line, the length of the preceding lines is zero, i.e., the preceding lines encoder hidden states $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}$ do not exist. Therefore, when generating the 1st line, the following equation holds :

$$\hat{\mathbf{s}}_t = \mathbf{s}_t \tag{10}$$

where the sizes of $\hat{\mathbf{s}}_t$ and $\mathbf{s}_t$ are both $[m,1]$.

We use a Feed-Forward Network (FFN) to turn $\hat{\mathbf{s}}_t$ into a vector $[d,1]$, where d represents the size of the vocabulary. We can predict probability distribution $\tilde{\mathbf{y}}_t$ through *softmax* function:

$$\tilde{\mathbf{y}}_t = softmax(\mathbf{W}_s \hat{\mathbf{s}}_t) \tag{11}$$

where $\mathbf{W}_s$ is a weight matrix to be learned in the FFN model. The size of $\mathbf{W}_s$ is $[d, m]$.

We use the argmax function and select character in the vocabulary that has the highest probability as the output character $\hat{\mathbf{y}}_t$.

$$\hat{\mathbf{y}}_t = \arg\max \tilde{\mathbf{y}}_t \tag{12}$$

Note that $\mathbf{r}_k$ and $\mathbf{r}_m$ are connected together as the initial hidden state of the decoder $\mathbf{s}_1$, where,

$$\mathbf{s}_1 = \mathbf{r}_c = \begin{bmatrix} \mathbf{r}_k \\ \mathbf{r}_m \end{bmatrix} \tag{13}$$

2.6. Loss function

We use cross-entropy loss L_{cross} as our loss function. It characterizes the difference between the real probability distribution $\mathbf{y}_t$ and the predicted probability distribution $\tilde{\mathbf{y}}_t$ Our objective is to minimize L_{cross} which can be written as:

$$min\ L_{cross} = -\sum_{i=1}^{d} \mathbf{y}_{t_i} \ln \tilde{\mathbf{y}}_{t_i} \tag{14}$$

3. Experimental design

3.1. Dataset

We build a corpus[1] of 284308 quatrains from the classical poetry books and the Chinese ancient poetry websites. The poetry of corpus ranges from the Tang Dynasty to the Qing Dynasty, including the poetry of famous poets such as Li Bai[2] and Bai Juyi,[3] as well as the poetry of less famous poets.

There are many Chinese characters in ancient times, e.g., according to the *Kangxi Dictionary*,[4] there are 46933 Chinese characters. However, our poetry vocabulary does not cover all those words. We first choose the top most common 5998 characters according to their frequency in our poetry corpus. The poetry vocabulary includes those 5998 characters and two special tokens (sos and eos), i.e., d is set to 6000. Then we filter out poetry which includes characters that are out of vocabulary. As a result, our corpus for training contains 13011 5-char quatrains and 63411 7-char quatrains, the total is 76422 quatrains.

Out of those, we randomly select 68422 poems for training, 4000 poems for verification and 4000 poems for testing.

3.2. Extract keywords

Keywords/Words are the combination of characters in a poem. For example, the word "举头" (lift head) contains the character "举" (lift) and "头" (head). We use the *shixuehanying*δ_i as the collection of words. Then we use this to extract words from our poems.

Suppose that a line in a poem consists of H characters, i.e., δ_1, δ_2, …, δ_H, where δ_i (i = 1, 2, …, H) means the ith character in the line. In Chinese language, a word may consist of one or more characters. Given a line, one-character word can be δ_1, two-character word can be (δ_1, δ_2) or (δ_2, δ_3), …, H-characters word can be (δ_1, δ_2, …, δ_H). As a result, we can get $\frac{H(H+1)}{2}$ possible word candidates from H characters. For all the candidates, we only keep those that exist in the *shixuehanying*. For example, we can extract the word "床" ("bed"), "前" ("before"), "明 月 光" ("luminous moonshine"), "明 月" ("bright moon") and "月 光" ("moonshine") from a line "床 前 明 月 光" ("The luminous moonshine before my bed".). Note that, "床" ("bed") and "前" ("before") are one-character words while "明 月 光" ("luminous moonshine") is a 3-character word.

TextRank [21] is an algorithm to measure the importance of words. TextRank algorithm is a graph-based ranking algorithm based on PageRank [22]. In TextRank, a vertex V_i represents a word i. An edge is inserted between two words when they co-occurs in a window of a specified length. The words i and j's total count of co-occurrence is set as the edge's weight w_{ij}. We can

[1] https://pan.baidu.com/s/13UraNiQU-ItOH2ZVHmMaBg code:tthg

[2] Li Bai (701–762), also known as Li Bo, courtesy name Taibai, is a Chinese poet acclaimed from his own day to the present as a genius and a romantic figure who took traditional poetic forms to new heights.

[3] Bai Juyi (772–846), is a renowned Chinese poet and Tang dynasty government official. Many of his poems concern his career or observations make about everyday life, including as governor of three different provinces

[4] The Kangxi Dictionary is the standard Chinese dictionary during the 18th and 19th centuries. The Kangxi Emperor of the Manchu Qing Dynasty ordered its compilation in 1710.

initialize the TextRank score $S(V_i)$ to 1.0 and run it iteratively until the following equation convergence:

$$S(V_i) = (1-p) + p \sum_{V_j \in E(V_i)} \frac{w_{ji}}{\sum_{V_k \in E(V_j)} w_{jk}} S(V_j) \quad (15)$$

where $E(V_i)$ is a the set of vertices connected with V_i and p represents a damping factor, which is usually set to 0.85 [22].

After running TextRank algorithm, each word will get a score that represents its importance. The higher the word's score, the more important the word is. Then we choose the word that has the highest word score as the keyword for each line in a poem.

Take the ancient poem in Table 1 as an example. For the first line, we first extract the word "床" ("bed"), "前" ("before"), "明月光" ("luminous moonshine"), "明月" ("bright moon") and "月光" ("moonshine") by the *shixuehanying* from the line "床前明月光" ("The luminous moonshine before my bed".).

According to TextRank algorithm on those words, then we can get the score for each word. The scores of the word "床" ("bed"), "前" ("before"), "明月光" ("luminous moonshine"), "明月" ("bright moon") and "月光" ("moonshine") are 1074, 403, 14, 33553 and 1313, respectively. And we select the word "明月光" ("luminous moonshine") as the keyword for 1st line. Because it is the word with the highest score in this line. Similarly, we can get the keywords "地上" ("on the ground"), "举头" ("lift head") and "故乡" ("distant home") for the 2nd to the 4th lines.

3.3. Baseline model

We choose the Planning Poetry Generation (PPG) model [11] as our baseline model. The PPG model is a poetry generation model based on a seq2seq framework. Each line of a poem is generated sequentially with the keyword and previous lines. This model has been compared with several state-of-art models [5,7,8] and shows significant advantages. We choose their model as our baseline as it is the state-of-the-art.

In order to make fair comparison, the PPG model and our KPG model use the same training and testing configuration.

3.4. Training

The code and the experimental settings in our experiment are at https://github.co *m/saberliliy/poetry-generation-model*. We implement the PPG model and KPG model with Tensorflow [23] and Adam [24] optimizer.

For the hyper-parameters, we set the initial values for all trainable parameters in the model, e.g., $\mathbf{W}_s$, $\mathbf{W}_a$ and $\mathbf{W}_c$, as random values from a uniform distribution $[-0.08, 0.08]$. We set the learning rate as 0.0002. To avoid overfitting, we set the dropout rate as 0.3. We set the dimension of a character embedding as 512, i.e., $q = 512$.

We tune the hidden units of layers in the decoder and three encoders from 256 to 1024 ([256, 512, 1024]) and test the model performance separately. We find that the larger the size of the hidden units, the smaller the loss value, but it will need more training time. To balance the training time with the loss value, the size of hidden units in keyword encoder, keyword team encoder, preceding lines encoder and decoder are set to 256, 256, 512, 1024, respectively, i.e., $v = 256$, $n = 512$, $m = 1024$.

We need to consider both the training time and the convergence rate when we choose the batch size. The larger the batch size, the faster the training time, but it will lead to slower convergence. To balance the training time and convergence rate, we set the batch size as 128.

3.5. Evaluation method overview

3.5.1. Loss function evaluation

The cross-entropy loss L_{cross} measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

3.5.2. Human evaluation

We invite 5 experts as human evaluators to participate the poetry evaluation [5,7,9]. We use four standards for human evaluation, i.e., "Poeticness", "Fluency", "Coherency" and "Meaningfulness" as listed in Table 4. To introduce the four standards more specifically, we take the famous poetry *Seeing Meng Haoran Off from Yellow Crane Tower* in Table 5 as an example. The "Poeticness" focuses on whether the poetry follow the regulation on tones and rhymes. For example, the character "楼" ("tower"), "州" ("town") and "流 " ("flow") in the poetry are rhymes. The "Fluency" focuses on the fluency of sentences and express reasonable meaning. For example, we can fluently read the first line, i.e., "故人西辞黄鹤楼" ("My friend has left the west from the Yellow Crane".), and clearly know who did what. The "Coherency" focuses on whether the poem adheres to a single theme. For example, the poetry mainly expresses the poet's fond farewell to his friend. The "Meaningfulness" focuses on whether the poem stimulates any aesthetic feeling. For example, the 3rd and 4th lines of the poetry show readers the magnificent mountains and rivers of the motherland in only a few words.

Each standard can be defined as a score ranging from 0 (worst) to 5 (best), and a higher score represents a better result. We invite 5 experts to participate in the grading of generated poetry. These experts are professionals and have deep understanding of ancient Chinese poetry. Most of them are from the Chinese literature and art departments in our university. These experts are asked to rate twenty 5-character quatrains and twenty 7-character quatrains generated by the PPG model and the KPG model independently, and as objectively as possible. The grading scores of these experts are averaged as the final score.

4. Experimental results

In this Section, we show the experimental results for the PPG model and our model. We first show the results based on loss function evaluation. Then we show the results based on human evaluation. Finally, we analyze two concrete examples.

4.1. Experimental results based on loss function

In order to make a more reasonable comparison between the generation effects of the PPG model and the proposed KPG model, we choose to stop training only after the convergence of the two models is achieved rather than choosing an identical time. Fig. 4 shows results based on loss function evaluation. The *x*-axis represents the training epoches and the *y*-axis represents the loss value. With the increase of the training epoches, the loss values of two models become smaller and smaller. After 17 epochs, the loss of the KPG model falls faster than that of the PPG model. Both of the two models' loss curves converge at around 30 epochs.

At the 30th epoch, the loss value of PPG model is 0.30 while the loss value of our KPG model is 0.165. The result show that the loss value of KPG model is 45% lower than that of the Planning Poetry Generation (PPG) model, which indicates an overall better performance.

Table 4
Evaluation standards in human judgment.

Poeticness	How does the poem follow the rhyme and tone requirements?
Fluency	How does the poem read smoothly and fluently?
Coherency	How is the poem coherent across lines?
Meaningfulness	How does the poem have a certain meaning and artistic conception?

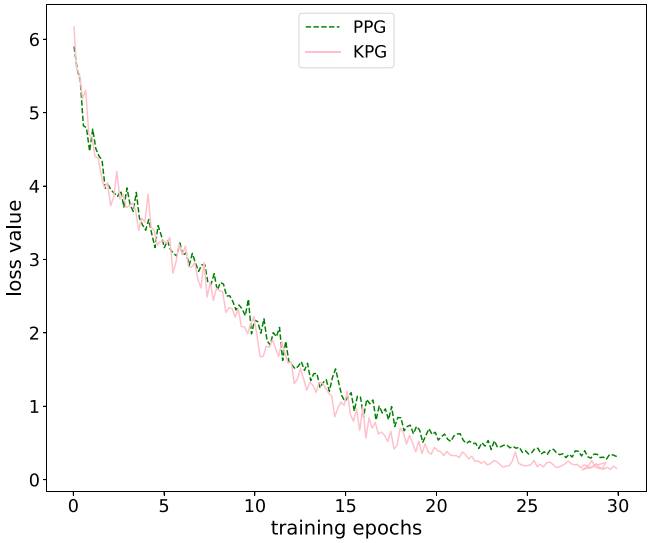

Fig. 4. Loss curves of two models.

4.2. Experimental results based on human evaluation

Fig. 5 shows results based on human evaluation. Both of the models achieve a high score (over 3), but our model achieves even better results than the PPG model, especially in the field of coherency and meaningfulness.

As we expect, the average scores of our model on 5-character quatrains and 7-character quatrains are 3.7% and 7.1% higher than PPG model, respectively. This indicates that our model outperforms the start-of-the-art model with better understanding of context, and improves the quality of the generated poetry after the introduction of the keyword teams into the model.

This is consistent with loss function evaluation result. However, the percentage of improvement in human evaluation is lower than that of loss function evaluation. This may be because human evaluation is subjective, but loss function evaluation is more objective.

4.3. Two concrete examples

In order to demonstrate the details of analysis of two models, we select two very famous real Chinese quatrains as standard examples. We extract four keywords from each quatrain and use the same keywords to generate poetry by PPG model and KPG model, respectively. The results are shown in Table 5 and Table 6, respectively.

The first famous poetry is *Seeing Meng Haoran Off from Yellow Crane Tower*, which is written by the Tang Dynasty's poet—Li Bai[5], as shown in the top half of Table 5. This is a farewell poem. There is no word about 'farewell' in the whole poem, but it is full of feeling of parting for a longish period. In the 1st and 2nd lines, the poet sent his friend to the beautiful place Yangzhou in the "Yellow Crane Tower" in March when the flower was blossoming. The scene presented in the poem is very beautiful and evocative. In the 3rd and 4th lines, the poet stood alone on the shore and watched the sailing boat out of sight, watching the waves of the Yangtze River, feeling emotional. The whole poem not only expresses the cherishing of his friend, but also praises the magnificent mountains and rivers of the motherland. We extract four keywords "辞" ("Farewell"), "三月" ("March"), "帆" ("Sail") and "流 " ("Flow") from the poetry according to TextRank algorithm.

The ancient poetry generated by the PPG model based on keywords is shown in the middle half of Table 5. The 1st and 2nd lines depict a scenario where an officer is writing a farewell letter to travel in March. The 3rd and 4th lines depict that the officer wants to visit the Prime Minister, but he did not drift fast enough. Although we can understand the meaning of each line, it is not quite coherent as "Prime Minister" is unexpected. And the meaning is not clear as well.

The ancient poetry generated by the KPG model based on keywords and keyword teams is shown in the bottom part of Table 5. In the ancient poem generated by our KPG model, the 1st and 2nd lines depict the scenario where a guest is leaving for Chuzhou[6] in March, expressing the poet's deep feelings of departing with a sunset background. The 3rd and 4th lines depict the scene of the guest sailing through the wind and waves, while appraising the magnificent rivers of the motherland. We can see that the poem generated by our model does not mention farewell but it describes a parting scenario similar to the original poem.

The second famous poetry is *The Deerpark Village*, which is written by Tang Dynasty poet—Wang Wei[7], as shown in the top half of Table 6. It depicts an empty mountain with few people and a forest with old and tall trees, and the poet wants to describe a quiet and secluded atmosphere. We extract four keywords "空山" ("lonely hills"), "响" ("ring"), "景" ("scenery") and "复" ("again") from the poetry.

The ancient poetry generated by the PPG model is shown in the middle part of Table 6. The 1st and 2nd lines depict a butterfly flying in the empty mountains and the valley echoes. The 3rd and 4th lines depict that it gets colder as dusk coming, and the mountains connect one by one. The ancient poem generated by the PPG model is relatively unclear in its meaningfulness as a butterfly is hard to make echos in valleys.

The ancient poetry generated by the KPG model is shown in the bottom part of Table 6. The 1st and 2nd lines depict a scene of cranes flying through the empty mountains while forest echoes the gloomy valley. The 3rd and 4th lines depict the Bashan[8] land after the rain. The poem also expresses a quiet and secluded atmosphere because cranes are often a symbol of fairyland comparing to butterflies.

[5] Li Bai (701–762), also known as Li Bo, courtesy name Taibai, is a Chinese poet acclaimed from his own day to the present as a genius and a romantic figure who took traditional poetic forms to new heights.

[6] Chuzhou is a prefecture-level city in eastern Anhui Province, China. It borders the provincial capital of Hefei to the south and southwest, Huainan to the west, Bengbu to the northwest, and the province of Jiangsu to the east

[7] Wang Wei (699–759), is a Chinese poet, musician, painter, and politician during the Tang dynasty. He is one of the most famous men of arts and letters of his time. Many of his poems are preserved, and twenty-nine are included in the highly influential 18th-century anthology Three Hundred Tang Poems.

[8] Bashan, also known by their Chinese name as the Daba Mountains, are a mountain range in Central China between the watersheds of the Yellow and Yangtze Rivers.

Models	Poeticness		Fluency		Coherency		Meaningfulness		Average	
	5-*char*	7-*char*	5-*char*	7-*char*	5-*char*	7-*char*	5-*char*	7-*char*	5-*char*	7-*char*
PPG	4.10	4.17	3.87	3.91	4.08	3.85	3.98	3.86	4.01	3.95
KPG	4.11	4.16	4.22	4.20	4.17	4.35	4.12	4.22	4.16	4.23

Fig. 5. Human evaluation result of two models.

Table 5
Two 7-character quatrains generated with *Seeing Meng Haoran Off from Yellow Crane Tower*.

黄鹤楼送孟浩然之广陵	Seeing Meng Haoran Off from Yellow Crane Tower
故人西辞黄鹤楼,	My friend has left the west where the Yellow Crane towers,
烟花三月下扬州.	For River Town veiled in green willows and red flowers.
孤帆远影碧空尽,	A sail's faint figure dots the blue sky's end,
唯见长江天际流.	Where seen but River rolling till its bend.
keyword:	辞 (Farewell), 三月 (March), 帆 (Sail), 流 (Flow)
PPG	
院吏忙抄诏草辞,	An officer is busy writing a farewell letter
山间三月已相随.	Before traveling to mountains in March.
纵令未远来参相,	Although the journey takes short time to visitt the Prime Minister,
不独开流不足持.	the officer is still feeling not fast enough while drifting on the river.
KPG	
倦客深辞日下楼,	A tired guest is leaving a Tower at sunset,
乾初三月谒初州.	Looking forward to arriving in Chuzhou in early March.
朝来更定帆长浪,	Sailing on the long waves from morning to evening,
扑扑朝尘处处流.	The guest is seeing the River rolling all the time.

Table 6
Two 5-character quatrains generated with *The Deerpark Village*.

鹿柴	The Deerpark Village
空山不见人,	No wight is seen in the lonely hills round here,
但闻人语响.	But whence is wafting the human voice I hear.
返景入深林,	So deep in the forest the sunset glow can cross,
复照青苔上.	That it seems to choose to linger on the moss.
keyword:	空山 (Lonely hills), 响 (Ring), 景 (Scenery), 复 (Again)
PPG	
空山化蝶飞,	A butterfly is flying in the empty mountains,
谷响口飞急.	while valley echoes make the flight faster.
冻景涧风昏,	The mountains get fozen when the sunset.
复岭连天外.	More and more mountains are connecting to the sky.
KPG	
空山鸾鹤飞,	Cranes are flying over the empty valley,
林木响幽谷.	while forest echoes the gloomy valley.
风雨巴山景,	The scenery of Ba Mountain after the wind and rain is so beautiful,
亦复行过雨.	before everything comes back to normal.

In summary, although the ancient poems we generated are much different than the poems written by poets, they still depict a scene to a certain extent and successfully expresses a poet's emotion, which is better than the ancient poems generated by the state-of-the-art one (PPG model). However, the proposed KPG model is based on keywords, so the generation of ancient poetry is not flexible enough, nor does it generate a specific style of classical poetry.

5. Related work

Automatic poetry generation is a challenging research in the field of Natural Language Generation (NLG). Table 7 shows the mainstream poetry generation models and the comparison among them. Most of the traditional models are based on rules and templates, which try to generate poetry by using some rule-based constraints. Tosa et al. [1] proposed a rule-based model that used rules extracted from corpus to translate user writing intent into poems. Statistical Machine Translation (SMT) is an another important approach to generation of poems. He, Zhou and Jiang [5] applied the SMT to the Chinese classical poems generation. The traditional models only focus on the surface forms of characters or words and have little deep understanding of the meaning of a poem or lyric.

In recent years, the adoption of neural network-based approaches has achieved great success in classical poetry generation. Graves [25] proposed a generation textual sequences model to generate poetry, which was based on the Recurrent Neural Network Language Model (RNNLM) [13]. A poem is generated character by character as a sequence. Zhang and Lapata [7] proposed a Recurrent Neural Network Poetry Generation (RNNPG)

Table 7
The mainstream generation models.

Paper	Model name	Model category	Model input	The main contribution
Tosa et al. [1]	Rule-based model	Traditional model	Keywords	This model is based on the syntactic analysis of the input keywords to expand the poem.
He, Zhou & Jiang [5]	SMT model	Traditional model	The first line of poetry	The model uses statistical machine translation (SMT) to translate the first line into the second.
Graves [25]	RNNLM model	RNN model	Keywords	RNN neural network is used to generate ancient poems.
Zhang & Lapata [7]	RNNPG model	seq2seq model	Keywords	The model uses RNNLM to generate the first line from the input keywords and the subsequent lines are generated sequentially.
Yan [9]	Polishing schema model	seq2seq model	Keywords	The model polishes and refines the generated lines for several times by using an iterative schema.
Wang et al. [8]	NMT model	seq2seq model	The first line of poetry	The model expands the traditional statistical machine translation (SMT) algorithm, using a seq2seq framework to generate poetry.
Yi, Li & Sun [26]	Poem blocks model	seq2seq model	Keywords	The model can learn the semantic relevance among the lines of poetry.
Ghazvininejad et al. [27]	Hafez model	seq2seq model	the phrases	The model translates phrases into rhyme-words and builds encoder–decoder model, to keep the poem on theme.
Yi et al. [28]	work memory model	seq2seq model	Keywords	The model guides the poetry generation according to maintains the input keywords and informative limited history in memory.
Yang et al. [14]	Conditional variational auto-encoder model	seq2seq model	Keywords, phrases and sentences	The model uses a conditional variational auto-encoder with a hybrid decoder to mine the implicit topic information contained within poems lines.
Wang et al. [11]	PPG model	seq2seq model	Keywords, phrases and sentences	The model determines a serious keyword according to the input, and each line is generated sequentially with the allocated keyword and previous lines.
Our paper	KPG model	seq2seq model	Keywords, phrases and sentences	The model creatively introduces a sequence of keywords called keyword team into the generation of each line in the poetry.
Zhang et al. [10]	Memory-augmented neural model	seq2seq model	Keywords	The model uses an external memories component to improve the novelty of generated poetry.
Yang et al. [29]	SPG model	seq2seq model	Poetry style keywords	The model can generate the various styles of poetry with unsupervised training manner.
Liu et al. [30]	ACCVAE model	seq2seq model	sentences	The model uses a rhetorically controlled encoder–decoder to generate poetry.

model. The model uses a RNNLM model to generate the first line from a given keyword and then sequentially generate subsequent lines by accumulating the status of the lines generated so far. Yan [9] proposed a polishing schema model, which used an iterative polishing schema based on two recurrent neural networks to generate a poem through refinement repeatedly. Wang et al. [8] proposed a Neural Machine Translation (NMT) model, which was based on Statistical Machine Translation (SMT). The difference between NMT and SMT is that NMT uses a seq2seq framework based on attention [17] as the machine translation system. The problem of this model is that it requires the user to provide the first line of the poem as input. Therefore, in order to generate a high-quality ancient poem, it is necessary to input a high-quality line. Yi, Li and Sun [26] proposed a poem block model to resolve this problem by using a separate neural machine translation (NMT) model. The NMT model can generate the first line of the poem using a keyword provided by the user. In addition, this model uses a poem block component to learn semantic meaning within a single line and semantic relevance among lines in a poem. The input of the model is usually a single sentence or a single word. It has large influence on the generation of the first line of the poem. However, this influence gradually diminishes on the generation of the following lines. There is a possibility of theme drift which will result in bad coherency of the poem.

Therefore, Wang et al. [11] proposed a two-stage classical Chinese poetry generation model to avoid this situation. First, a planning-based model was used to determine a keyword for each line, rather than sharing a keyword for all the lines. Then each line was generated sequentially with the corresponding keyword and the previously generated lines. Ghazvininejad et al. [27] presented a different two-stage classical Chinese poetry generation model called Hafez model. In the first stage, the given keyword was used to generate the rhyme words which were related to the keyword. In the second stage, a seq2seq model was used to generate classical poetry by these rhyme words. The above two models avoid the theme drift to a certain extent by introducing a keyword to the generation process of each line of a poem. In addition, there are some other models to solve the problem of theme drift. Yi et al. [28] proposed a working memory model, which maintained keywords and informative limited history in a memory component to guide the generation. Yang et al. [14] proposed a conditional variational auto-encoder model with augmented word2vec architecture, which could improve the conformity between the theme and generated poems.

In addition to the above works which aim at improving the coherency of the generated poetry, there are also other works which focus on improving the ability of models to generate poetry in a variety of styles. Zhang et al. [10] proposed a memory-augmented neural model that used an external memories component to improve the novelty of generated poetry. Yang et al. [29] proposed a Stylistic Poetry Generation (SPG) model to generate the

different styles of the poetry under the same poetic imagery. Liu et al. [30] proposed a Automatic Control Conditional Variational Auto-Encoder (ACCVAE) model. This model applies a continuous latent variable to capture the rhetorical patterns of poetry.

Comparing with previous poetry generation work, our KPG model uses a sequence of keywords called keyword team to participate in the generation of each line of ancient poetry. By using a keyword team rather than a single keyword for each line, the model can better capture the context of the whole poem and achieve better coherency and meaningfulness over the state-of-the-art models.

6. Conclusion and future work

In this paper, we propose a new automatic context-aware Chinese poetry generation approach. We use a sequence of keywords called keyword team to capture the context of the Chinese poetry. We then use keyword, keyword team and previous lines to generate new lines. Including keyword team in the generation of each line mimics a poet's writing habit, where a poet always refers to the context of a poem to make it coherent and meaningful. Moreover, we also include an attention layer to better focus on the key words in preceding lines while generating a new line. Both loss function evaluation and human evaluation demonstrate that our approach can effectively improve the quality of the generated poetry, outperforming the state-of-the-art model ones.

The future work includes: (1) we plan to incorporate more context into Chinese poetry's automatic generation. (2) we also plan to introduce the keyword team as a context into other literature generation, e.g., essay and novel generation. (3) finally, we plan to wrap our model as a web service for public use.

CRediT authorship contribution statement

Tian Gao: Methodology, Software, Writing – original draft, Editing. **Shanliang Zhu:** Conceptualization, Project administration, Writing – review & editing. **Jing Liu:** Data curation, Visualization, Investigation. **Jun Shen:** Supervision, Software, Writing – review & editing. **Jialie Shen:** Supervision, Project administration. **Shuguo Yang:** Data curation, Investigation, Validation. **Pengcheng Xiong:** Conceptualization, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] N. Tosa, H. Obara, M. Minoh, Hitch haiku: An interactive supporting system for composing haiku poem, in: International Conference on Entertainment Computing, Springer, 2008, pp. 209–216, http://dx.doi.org/10.1007/978-3-540-89222-9_26.

[2] E. Greene, T. Bodrumlu, K. 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, 2010, pp. 524–533, https://www.aclweb.org/anthology/D10-1051/.

[3] H.G. Oliveira, PoeTryMe: a versatile platform for poetry generation, in: Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence, 2012, https://www.mendeley.com/catalogue/0c6040c0-c0b6-3caa-ad45-a5522c4c39a4/.

[4] X. Wu, N. Tosa, R. Nakatsu, New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system, in: International Conference on Entertainment Computing, 2009, pp. 191–196, http://dx.doi.org/10.1007/978-3-642-04052-8_19.

[5] J. He, M. Zhou, L. Jiang, Generating Chinese classical poems with statistical machine translation models, in: Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012, http://dx.doi.org/10.1108/03090590610715013.

[6] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, Adv. Neural Inf. Process. Syst. (2014) 3104–3112, https://arxiv.org/abs/1409.3215.

[7] X. Zhang, M. Lapata, Chinese poetry generation with recurrent neural networks, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 670–680, http://dx.doi.org/10.3115/v1/D14-1074.

[8] Q. Wang, T. Luo, D. Wang, C. Xing, Chinese song iambics generation with neural attention-based model, in: International Joint Conference on Artificial Intelligence, IJCAI, 2016, pp. 2943–2949, https://arxiv.org/abs/1604.06274.

[9] R. Yan, i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema, in: International Joint Conference on Artificial Intelligence, IJCAI, 2016, pp. 2238–2244, https://www.ijcai.org/Proceedings/16/Papers/319.pdf.

[10] J. Zhang, Y. Feng, D. Wang, Y. Wang, A. Abel, S. Zhang, A. Zhang, Flexible and creative Chinese poetry generation using neural memory, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vol. 1, Association for Computational Linguistics, 2017, pp. 1364–1373, http://dx.doi.org/10.18653/v1/P17-1125.

[11] Z. Wang, W. He, H. Wu, H. Wu, W. Li, H. Wang, E. Chen, Chinese poetry generation with planning based neural network, in: 26th International Conference on Computational Linguistics, COLING, 2016, http://arxiv.org/pdf/1610.09889.

[12] M. Hreskova, K. Machova, Michiko: Poem models used in automated haiku poetry generation, in: International Conference on Current Trends in Theory & Practice of Informatics, 2018, pp. 469–476, http://dx.doi.org/10.1007/978-3-319-73117-9_33.

[13] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, S. Khudanpur, Recurrent neural network based language model, in: Eleventh Annual Conference of the International Speech Communication Association, 2010, http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

[14] X. Yang, X. Lin, S. Suo, M. Li, Generating thematic Chinese poetry using conditional variational autoencoders with hybrid decoders, in: International Joint Conference on Artificial Intelligence, 2018, pp. 4539–4545, http://dx.doi.org/10.24963/ijcai.2018/631.

[15] G. Zhipeng, X. Yi, M. Sun, W. Li, R. Li, Jiuge: A human-machine collaborative Chinese classical poetry generation system, in: Meeting of the Association for Computational Linguistics: System Demonstrations, 2019, pp. 25–30, http://dx.doi.org/10.18653/v1/P19-3005.

[16] H.G. Oliveira, T. Mendes, A. Boavida, A. Nakamura, M. Ackerman, CoPoeTryMe: Interactive poetry generation, Cogni. Syst. Res. 54 (5) (2019) 199–216, http://dx.doi.org/10.1016/j.cogsys.2018.11.012.

[17] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, 2015, https://arxiv.org/abs/1409.0473.

[18] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: International Conference on Learning Representations, ICLR2013, 2013, https://arxiv.org/abs/1301.3781.

[19] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681, http://dx.doi.org/10.1109/78.650093.

[20] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[21] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004, pp. 404–411, https://digital.library.unt.edu/ark:/67531/metadc30962/.

[22] S. Brin, L. Page, The anatomy of a large-scale hypertextual Web search engine, Comput. Netw. ISDN Syst. 30 (1–7) (1998) 107–117, http://dx.doi.org/10.1016/S0169-7552(98)00110-X.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., TensorFlow: a system for large-scale machine learning, in: USENIX Symposium on Operating Systems Design and Implementation, 2016, pp. 265–283, http://www.arxiv.org/abs/1605.08695.

[24] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, 2015, http://arxiv.org/abs/1412.6980.

[25] A. Graves, Generating sequences with recurrent neural networks, 2013, https://arxiv.org/abs/1308.0850,

[26] X. Yi, R. Li, M. Sun, Generating chinese classical poems with rnn encoder-decoder, in: Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, Springer, 2017, pp. 211–223, http://dx.doi.org/10.1007/978-3-319-69005-6_18.

[27] M. Ghazvininejad, X. Shi, Y. Choi, K. Knight, Generating topical poetry, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1183–1191, http://dx.doi.org/10.18653/v1/D16-1126.

[28] X. Yi, M. Sun, R. Li, Z. Yang, Chinese poetry generation with a working memory model, in: International Joint Conference on Artificial Intelligence, 2018, pp. 4553–4559, http://dx.doi.org/10.24963/ijcai.2018/633.

[29] C. Yang, M. Sun, X. Yi, W. Li, Stylistic chinese poetry generation via unsupervised style disentanglement, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 3960–3969, http://dx.doi.org/10.18653/v1/D18-1430.

[30] Z. Liu, Z. Fu, J. Cao, G. De Melo, Y. Tam, C. Niu, J. Zhou, Rhetorically controlled encoder-decoder for modern Chinese poetry generation, in: Meeting of the Association for Computational Linguistics, 2019, pp. 1992–2001, http://dx.doi.org/10.18653/v1/P19-1192.