Compose Like Humans: Jointly Improving the Coherence and Novelty for Modern Chinese Poetry Generation

Lei Shen[1,2], Xiaoyu Guo[3], Meng Chen[3*]
[1]*Key Laboratory of Intelligent Information Processing*,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]*University of Chinese Academy of Sciences*, Beijing, China
[3]*JD AI*, Beijing, China
shenlei17z@ict.ac.cn, guoxiaoyu404@163.com, chenmeng20@jd.com

***Abstract*—Chinese poetry is an important part of worldwide culture, and classical and modern sub-branches are quite different. The former is a unique genre and has strict constraints, while the latter is very flexible in length, optional to have rhymes, and similar to modern poetry in other languages. Thus, it requires more to control the coherence and improve the novelty. In this paper, we propose a generate-retrieve-then-refine paradigm to jointly improve the coherence and novelty. In the first stage, a draft is generated given keywords (i.e., topics) only. The second stage produces a "refining vector" from retrieval lines. At last, we take into consideration both the draft and the "refining vector" to generate a new poem. The draft provides future sentence-level information for a line to be generated. Meanwhile, the "refining vector" points out the direction of refinement based on impressive words detection mechanism which can learn good patterns from references and then create new ones via insertion operation. Experimental results on a collected large-scale modern Chinese poetry dataset show that our proposed approach can not only generate more coherent poems, but also improve the diversity and novelty.**

***Index Terms*—generate-retrieve-then-refine paradigm, automatic poetry generation, coherence and novelty**

I. Introduction

Automatic poetry generation is a sub-field of Natural Language Generation (NLG). In recent years, there have been many studies focusing on the classical Chinese poetry generation, since this kind of poetry is distinctive. Among different types of classical poems, *quatrain* (绝句) and *regulated verse* (律诗) are perhaps the best-known ones. They mainly have four requirements: 1) strict constrains in length, e.g., a *quatrain* consists of four lines, and each line contains five or seven characters; 2) tonal patterns, i.e., "*Ping*" (level tone) or "*Ze*" (downward tone); 3) rhyme schemes, e.g., for a *quatrain*, the ending character of the second and the fourth lines should have the same rhyme; 4) unified structure, e.g., a *quatrain* often follows the "beginning, continuation, transition, summary" template [1].

A classical Chinese *quatrain*: 相思 Missing You	
红豆生南国，(* Z P P Z)	Red berries born in the warm southland
春来发几枝？(P P Z Z P)	How many branches flush in the spring?
愿君多采撷，(* P P Z Z)	Take home an armful, for my sake,
此物最相思。(* Z Z P P)	As a symbol of our love.
A modern Chinese poem (part): 雨巷 A Lane in the Rain	
撑着油纸伞，独自	Alone holding an oil-paper umbrella,
彷徨在悠长、悠长	I wander along a long
又寂寥的雨巷，	Solitary lane in the rain,
我希望逢着	Hoping to encounter
一个丁香一样地	A girl like a bouquet of lilacs
结着愁怨的姑娘。	Gnawed by anxiety and resentment.

Fig. 1. A comparison between the classical and modern Chinese poetry. The upper one is a 5-char *quatrain* exhibiting one of the most popular tonal patterns, which is also used in Zhang and Lapata's paper [2]. The tone of each character is shown within parentheses, P and Z are "*Ping*" and "*Ze*", respectively. * indicates that the tone is not fixed and can be either. The lower one is part of a famous modern Chinese poem. Rhyming characters are shown with underlines.

The modern Chinese poetry has become more and more popular nowadays, and people use it to record daily life, express personal emotions, and send blessings at special occasions. It is similar to modern poetry in other languages, and does not have too many strict constraints. Meanwhile, there are some challenges for automatic modern Chinese poetry generation. Linguistic accordance (coherence) and aesthetic innovation (novelty) are two important aspects. Modern poems are more free in length, thus it is hard to control the coherence. Besides, writing poems is an artistic creation process so novelty is necessary, which means more imagination and various uses of language are needed [3]. However, recent works mainly focused on the classical Chinese poetry and could not cover both two aspects very well at the same time [4].

To improve the coherence and novelty simultaneously, we borrow thoughts from how humans compose a poem. Not like one-stage automatic poetry generation, humans tend to start with a draft, and keep polishing it. Basically, there are two

[*] Corresponding Author.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

ways to refine a draft: 1) learning from predecessors' works. They learn how to use words and organize sentences from others' masterpieces, and then apply them into their works to create new expressions; 2) deliberating one sentence based on the context. With the information from previous and following sentences, they can modify current sentence to fit in the whole poem more appropriately. There have been some works imitating above ideas individually, and Wu et al. [5] summarized them as either "retrieve-then-generate" paradigm [6]–[8] or "generate-then-refine" paradigm [5], [9]–[14]. However, we think both two ways are necessary and need to be considered together, so we bring up the "generate-retrieve-then-refine" paradigm. Besides, previous works can only utilize history sentences or word-level bidirectional context, or they simply feed all retrieval lines into model which may contain lots of noises.

To tackle the above problems, we propose a novel approach that polishes generated drafts with bidirectional sentence-level context and a "refining vector" for modern Chinese poetry generation. In the first stage, the model generates a draft which provides future sentence-level information. Second, it leverages the generated draft to get some retrieval lines, and uses the impressive words detection mechanism to get the "refining vector". At last, both bidirectional sentence-level context and the "refining vector" are applied to generate a refined poem. Since we use both the past and future information in sentence level, we can improve the coherence of the entire poem better. By using impressive words detection mechanism, we filter out noises and extract some good expression patterns to distill the "refining vector", and finally improve the novelty and diversity of language usage.

Since there is no public large-scale modern Chinese poetry dataset[1], we collect one from the internet, and will publish it in the near future. Experimental results on this dataset show that our approach outperforms other baselines significantly in terms of the coherence and novelty.

Our main contributions can be summarized as follows:

- We propose a new paradigm, generate-retrieve-then-refine, for automatic poetry generation.
- In order to jointly improve the coherence and novelty, we leverage future information from the draft and the "refining vector" produced by impressive words detection mechanism.
- We collect a large-scale modern Chinese poetry corpus, and empirically verify the effectiveness of our model in terms of fluency, coherence, novelty and diversity.

II. Related Work

Our work touches two research fields: automatic poetry generation and refinement methods.

A. Automatic Poetry Generation

Early researches in this area are based on grammatical rules [16]–[18], genetic algorithms [19]–[21] or statistical machine translation methods [1], [22], [23]. After the boom of deep learning, many new approaches have appeared. Yan et al. [24] formulate this task as an optimization problem based on a generative summarization framework. Zhang and Lapata [2] utilize Recurrent Neural Networks (RNN) to take into account the entire history of what has been generated. Wang et al. [25] propose a two-step method which first plans the sub-topics of the poem and then generates it with a modified Seq2Seq model. Yang et al. [26] employ a conditional variational autoencoder to generate thematic poems. Zhang et al. [4] leverage external memories to improve the creativity of generated poems. In order to achieve better coherence, Yi et al. [27] propose a novel Working Memory model to keep a coherent information flow and learn to express each topic flexibly and naturally. Cheng et al. [3] generate Modern Chinese poetry from images. Liu et al. [15] work on rhetorical patterns (e.g., metaphor and personification) in modern Chinese poetry.

Our work differs from the above since: 1) most of them are based on classical Chinese poetry generation; 2) the inputs are not the same, and our input is text but not images; 3) the points of interest are diverse, others focus on the theme, fluency, diversity, rhetorical patterns, etc., while we try to improve the coherence and novelty simultaneously.

B. Refinement Methods

There are two main paradigms for refinement. One is the "retrieve-then-generate" paradigm. Song et al. [9] and Pandey et al. [10] encode all retrieval candidates into vectors and feed them into a decoder for response generation. Cao et al. [11] apply this paradigm in summarization by reranking and rewriting jointly. Li et al. [12] similarly use deletion, retrieval and generation for text style transfer. Guu et al. [13] leverage latent variables to form the "edit vector" according to lexical differences (insertion and deletion words) in two sentences, while Wu et al. [5] transfer the concept of "edit vector" to response generation and explicitly utilize the lexical differences in queries. The other is the "generate-then-refine" paradigm. Yan et al. [6] generate a quatrain based on several iterations. Xia et al. [7] propose Deliberate Network that uses one decoder to generate a prototype from scratch and another decoder to revise the prototype in a joint training way. When generating a word in a sentence, this model can leverage backward and forward words. Wang et al. [8] apply the Deliberation Network to abstract generation.

The differences between the above paradigms and ours are listed below. "Generate-then-refine" paradigm only utilizes context information in word level, which means that when generating a word, it only looks backward and forward in the range of current sentence. In contrast, we use future sentences from the draft when generating a word, so we can see much wider context in sentence level. "Retrieve-then-generate" paradigm tries to edit the retrieval sentences. It cannot guarantee the content coherence as the retrieval

[1] Liu et al. [15] published a small modern Chinese poetry dataset with 60,000 sentences in total. Their poems are cut into short chunks with the size of 3 lines, while our dataset has over 9 million lines and keeps the original long poems.

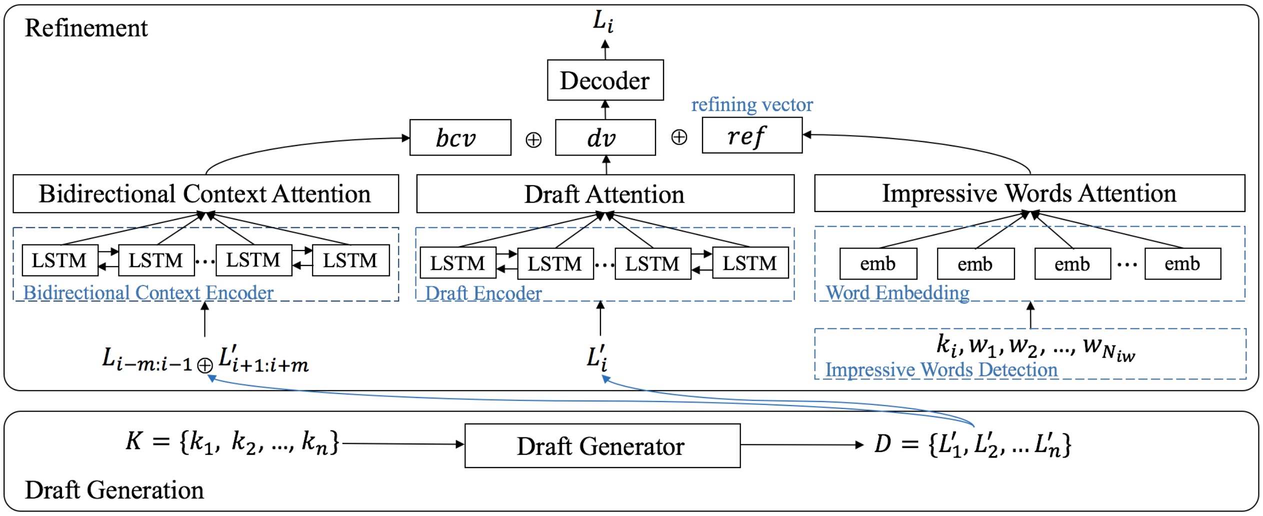
Refinement
L_i
Decoder
refining vector
bcv ⊕ dv ⊕ ref
Bidirectional Context Attention
Draft Attention
Impressive Words Attention
LSTM LSTM ··· LSTM LSTM
Bidirectional Context Encoder
LSTM LSTM ··· LSTM LSTM
Draft Encoder
emb emb emb ··· emb
Word Embedding
$L_{i-m:i-1} \oplus L'_{i+1:i+m}$
L'_i
$k_i, w_1, w_2, \ldots, w_{N_{iw}}$
Impressive Words Detection
$K = \{k_1, k_2, \ldots, k_n\}$
Draft Generator
$D = \{L'_1, L'_2, \ldots L'_n\}$
Draft Generation

Fig. 2. Overview of our GRR (generate-retrieve-then-refine) model. Lower: The draft generator generates draft D given n keywords. Upper: When generating line L_i, bidirectional context encoder is used to encode context $L_{i-m:i-1}$ and $L'_{i+1:i+m}$, draft encoder is for L'_i. Impressive word candidates consist of keyword k_i and good patterns we detected. "bcv", "dv" and "ref" are bidirectional context vector, draft vector and refining vector, respectively.

sentences may be noisy and very different in topic and style. On the contrary, we try to edit a generated draft, as it is consistent to some extent, and we tune it and make it better. Besides, we distill a "refining vector" to point out the direction for refinement. The "edit vector" in previous works [5], [13] is simply the concatenation of insertion and deletion words, while the "refining vector" represents good expression patterns in retrieval sentences, thus it contains more diverse language usage. Our approach is a combination of the above two paradigms.

III. Background

For input $X = \{x_i\}_{i=1}^{n}$, where n is the number of words, it is encoded into a sequence of hidden states $H = \{h_i\}_{i=1}^{n}$. Here we employ $\hat{e}_{x_t}$ as the embedding vector of word x_t, and the hidden state h_t is defined as:

$$h_t = f_{\mathrm{LSTM}}(h_{t-1}, \hat{e}_{x_t}), \tag{1}$$

where f_{LSTM} is the activation function of LSTM.

The decoder state s_t is updated by:

$$s_t = f_{\mathrm{LSTM}}(s_{t-1}, \hat{e}_{y_{t-1}}, c_t), \tag{2}$$

where s_{t-1} and $\hat{e}_{y_{t-1}}$ are hidden state and word embedding of decoded word at time step $t-1$ respectively. c_t is calculated by attention mechanism. Attention mechanism [28] is designed to focus on input information which is highly related to the generation of current word. The relevance between the to-be-generated word y_t and the i-th input word is computed as:

$$r_{i,t} = \mathbf{v}_{\alpha}^{T} \tanh\left(\mathbf{W}_{\alpha} s_{t-1} + \mathbf{U}_{\alpha} h_i\right). \tag{3}$$

Then, the relevance score is normalized and serves as the weight for corresponding encoder hidden state when calculating vector c_t:

$$\alpha_{i,t} = \frac{\exp\left(r_{i,t}\right)}{\sum_{i'=1}^{T} \exp\left(r_{i',t}\right)}, c_t = \sum\nolimits_{i=1}^{n} \alpha_{i,t} h_i. \tag{4}$$

IV. Approach

Coherence of a poem is mainly embodied in the relevance between several consecutive sentences, while novelty can be regarded as the way how new expression patterns are constructed. We generate a draft given some keywords at first. Then, we improve coherence and novelty jointly by leveraging bidirectional sentence-level context and a "refining vector" from the impressive words detection mechanism.

A. Writing Topic Representation

Suppose we have a dataset $\mathcal{D} = \{\hat{P}_j\}_{j=1}^{N}$, and N is the number of poems. Each poem has n lines, i.e. $\hat{P}_j = \{\hat{L}_i\}_{i=1}^{n}$. Following the works of Wang et al. [25] and Yang et al. [26] which assume that each line in the poem corresponds to a keyword (sub-topic), we use TextRank [29] to extract keyword k_i for each line. Then, we obtain $\{(\hat{L}_i, k_i)\}_{i=1}^{n}$ pairs for each poem.

TextRank is a graph-based algorithm. Each vertex stands for a candidate word and edges between two words represent their co-occurrence. Besides, the edge weight is set according to the total co-occurrence rate between these two words. The TextRank score $T(V_i)$ is initialized to a default value and computed iteratively until convergence according to the following equation:

$$T(V_i) = (1-d) + d \sum_{V_j \in E(V_i)} \frac{w_{ji}}{\sum_{V_k \in E(V_j)} w_{jk}} T(V_j), \tag{5}$$

where w_{ji} is the weight of the edge between vertex V_j and V_i, $E(V_i)$ is the set of vertices connected with V_i, and d is a damping factor. Empirically, d is set to 0.85 and the initial score of $T(V_i)$ is 1.0.

B. Draft Generation

In Draft Generation Stage, our goal is to write a draft D with n lines, i.e. $D = \{L'_i\}_{i=1}^{n}$, given keywords $\{k_i\}_{i=1}^{n}$. L'_i

is generated by taking the concatenated result of keyword k_i and previous m lines $L'_{i-m:i-1}$ as input. We use a multi-layer encoder with bidirectional Long Short-Term Memory (LSTM) [30] to encode the input by concatenating the last hidden states of the forward and backward LSTMs of the top layer, i.e. $h_i = \left[\overrightarrow{h}_i; \overleftarrow{h}_i\right]$.

Then we feed h_i to an attention-based multi-layer decoder. The parameters of the model are trained to maximize the log-likelihood on the entire training set, which is formulated as:

$$\text{argmax} \sum\nolimits_{i=1}^{M} \log P(L'_i | L'_{i-m:i-1}, k_i), \tag{6}$$

where M is the number of input-output pairs of the model.

C. Impressive Words Detection Mechanism

Algorithm 1 Impressive Words Detection

```
Input: Line $L'$ (we omit $i$) in draft $D$, candidate number $N_{iw}$
Output: Impressive words $W$
 1: Retrieve twenty human-written lines $R$ from Elasticsearch based on $L'$ and keyword $k$.
 2: Segment each line $r \in R$ into words and select one line $r'$ based on Jaccard similarity and sentence length.
 3: Label POS tags, calculate TFIDF values for each word in $L'$ and $r'$, and keep nouns ($n.$), adjectives ($a.$) and verbs ($v.$).
 4: Group words: $n.$, $a.$ for one set and $v.$ for the other, and get word lists $wL'_{na}$, $wL'_v$ and $wr'_{na}$, $wr'_v$ for $L'$ and $r'$.
 5: Sort $wL'_{na}$, $wL'_v$, $wr'_{na}$ and $wr'_v$ individually by TFIDF values in descending order.
 6: Get new word lists $wL''$, $wr''$ by concatenation, $wL'' = wL'_{na} + wL'_v$, $wr'' = wr'_{na} + wr'_v$.
 7: Let $cn = 0$, $W = []$
 8: for each $w \in wr''$ do
 9:     if $cn < N_{iw}$ then
10:         if $w \notin wL''$ then
11:             Add $w$ to $W$, $cn = cn + 1$
12:         end if
13:     else
14:         Jump out of the loop
15:     end if
16: end for
17: return $W$
```

In order to learn good patterns explicitly in poems written by humans and generate new expressions, we import the impressive words detection mechanism.

Given the entire training set, we index each line and construct a query as the combination of the draft line L'_i and its keyword k_i. We use the query to retrieve top 20 similar poem lines from the index based on a BM25 score [31]. Here, we use an open-source tool Elasticsearch[2]. Then, we pick sentences with characters more than 5 to maintain meaningful ones, perform word segmentation by Jieba[3]. Sentences that are almost identical with the draft line are not needed, since do not want to simply copy the retrieval lines. Our goal is to generate some new and impressive expressions by learning the most essential patterns from retrieval results. Therefore, we pick out the most similar retrieval lines in the range of [0.3, 0.7] (Algorithm 1 line 1 to 2) based on Jaccard similarity which is defined as:

$$J(S(L'), S(r)) = \frac{|S(L') \cap S(r)|}{|S(L') \cup S(r)|}, \tag{7}$$

where $S(L')$ and $S(r)$ are word sets of the draft line L' and retrieval line r, respectively. $|\cdot|$ denotes the size of a set.

For the obtained one retrieval line, we employ Part of Speech (POS) tagging on each word by Jieba[4] and only keep nouns($n.$), adjectives($a.$) and verbs($v.$), since they are usually semantically rich. Then we group these three kinds of words into na (nouns and adjectives) set and v (verb) set. For each set, we use the TFIDF value to sort words in descending order. Then we get two concatenated ordered word lists denoted by wL'' and wr'' (Algorithm 1 line 3 to 6). Then we select words appearing in wr'' but not in wL'' as the impressive word candidates for line L' (Algorithm 1 line 7 to 17). Finally, we have triples $\{(L'_i, k_i, \{w_{i,j}\}_{j=1}^{N_{iw}})\}_{i=1}^{n}$ for each draft, where N_{iw} is the number of impressive words candidates.

D. Refinement

In Refinement stage, we generate a new poem P with n lines, $P = \{L_i\}_{i=1}^{n}$ by taking into account both the draft and "refining vector" distilled from impressive patterns. When generating line L_i, there are three parts of the input, which are bidirectional sentence-level context, L'_i from draft and the "refining vector".

1) Construct Bidirectional Context: For NLG tasks, when generating a word in a sequence, only previously produced words can be utilized. Even with two decoders like Deliberation Network [7], the backward and forward information are limited in one sentence. In contrast, given a draft, humans tend to polish a line based on sentences before and after current one to make the poem more coherent and fluent. Inspired by this, the bidirectional sentence-level context in our model is composed of $L_{i-m:i-1}$ and $L'_{i+1:i+m}$. $L_{i-m:i-1}$ are lines we generated before L_i in refinement stage and provide information in the past, while $L'_{i+1:i+m}$ are lines in the draft version and represent information in the future. Finally, $L_{i-m:i-1}$ and $L'_{i+1:i+m}$ are concatenated and the bidirectional context is transformed to hidden vectors $\{h_j | h_j = \overrightarrow{h_j}; \overleftarrow{h_j}\}_{j=1}^{2m}$ with bidirectional LSTM.

2) Refining Vector: For each draft line L'_i, impressive words detection gives out corresponding impressive words candidates $W = \{w_i\}_{i=1}^{N_{iw}}$. Instead of feeding these words and keyword k_i directly to the model, we compute a "refining vector" by an attention mechanism defined as follows:

[2] https://www.elastic.co/products/elasticsearch

[3] https://github.com/fxsjy/jieba

[4] Jieba is for modern Chinese, which fits our task on modern Chinese poetry, and its POS tagging results are reasonable enough by human evaluation.

$$ref = \sum_{w \in W \cup \{k_i\}} \alpha_w \hat{e}_w, \tag{8}$$

where $\hat{e}_w$ is the embedding of word w. The weight α_w is computed by:

$$\alpha_w = \frac{\exp(e_w)}{\sum_{w \in W \cup \{k_i\}} \exp(e_w)}, \tag{9}$$

$$e_w = \mathbf{v}_\alpha^T \tanh(\mathbf{W}_\alpha h_{2m} + \mathbf{U}_\alpha \hat{e}_w). \tag{10}$$

Here, h_{2m} is the last hidden state of bidirectional context encoder, since we need to consider bidirectional context when we want to add impressive words in current sentence.

The decoder state s_t is updated by:

$$s_t = f_{\mathrm{LSTM}}(s_{t-1}, \hat{e}_{y_{t-1}}, r_t), \tag{11}$$

$$r_t = bcv_t \oplus dv_t \oplus ref, \tag{12}$$

where bcv_t and dv_t denotes the bidirectional context vector and draft vector at time step t, ref is time step independent. For bidirectional context and draft hidden states, we apply two attention mechanisms on them, following Equation 4. Then we get bidirectional context vector bcv and draft vector dv.

V. Experiments

We first introduce some empirical settings, including the dataset, baselines, implementation details and performance measures, then use evaluations on both automatic metrics and human judgements to prove the effectiveness of our model. Finally, we conduct case study to show the quality of generated poems.

A. Dataset

TABLE I
Statistics about our modern Chinese poetry corpus.

Number of poems in training set	210,935
Number of poems in validation set	26,367
Number of poems in test set	26,367
Lines per poem	10.25
Characters per line	12.35
Characters per poem	143.77

Since there is no public large-scale modern Chinese poetry dataset, we collect a new dataset. Our dataset is constructed with 2 parts: (1) modern Chinese poetry, collected from a online poetry website[5]; (2) modern Chinese lyrics, collected from NetEase Cloud Music[6]. Lyrics are very close to modern poems in both content and style so we can regard them as poetry. We totally collect 263,669 modern Chinese poems containing 9,209,186 sentences. Then, we tokenize each line to words by Jieba and calculate the TextRank score for each word. The word with the highest TextRank score is selected as the keyword for each line. The dataset is separated into training, validation, and test sets with the ratio 8:1:1. Table I provides descriptive statistics about our dataset.

B. Baselines

We compare our model with representative poetry generation and refinement approaches as listed below:

Plan: a Planning-based model [25] which divides poetry generation into two steps: organizing outlines (keywords) and writing poems.

DN: the Deliberation Network [7] which is firstly proposed for machine translation. When generating a word in a sentence, it looks backward and forward in the range of current sentence by jointly optimizing two decoders. It's a representative model for the "generate-then-refine" paradigm.

EED: the Exemplar Encoder-Decoder model [10] which is firstly proposed for neural conversation generation. There are two encoders: context encoder and similar-sentence encoder. These similar sentences are retrieved from training set and fed entirely into the second encoder. It's a representative model for the "retrieval-then-generate" paradigm.

Mem: a poetry generation model with neural memory [4] that contains human-written poems in a static external memory to improve the generated *quatrains*. It aims to generate creative Chinese poetry.

WM: a recent Working Memory model [27] for poetry generation that dynamically invokes a memory component by saving the writing history into memory. It focuses on generating coherent poems.

We denote our model as **GRR**, and **GRR-Refine** is the one without the "refining vector".

C. Implementation Details

We employ 54,500 words with the highest frequency as our vocabulary and define all the out-of-vocabulary words to a special token <unk>. The word embedding size is 128, and initialized with word2vec [32] pre-trained on the poetry corpus. The recurrent hidden layers of encoder and decoder contain 128 hidden units, and the number of layers is 4. The model is trained using the Adam algorithm [33], where the batch size is 512 and the learning rate is 3e-4. The dropout technique [34] is also adopted and the dropout rate is set to 0.3. The number of sentence-level context in one direction (m) is set to 1. The number of impressive word candidates (N_{iw}) is set to 2. All models are implemented with the same set of hyper-parameters. Optimization objective is standard cross entropy. For inference, beam search is utilized and the beam size is 10. We tune our hyper-parameters on validation set and measure the performance on test set. We use Tensorflow Framework[7] for our implementation.

D. Performance Measures

We use four metrics for automatic evaluation: **Perplexity (PPL)**: it measures the average fluency of generated poems. Using a 5-gram character based language model trained on our poetry corpus, we calculate the perplexity on test set. **Rouge-L**: it uses longest common sub-sequence to calculate the similarity between the generated line and its reference [35].

[5] http://www.shigeku.org

[6] http://music.163.com

[7] https://www.tensorflow.org/

TABLE II
AUTOMATIC AND HUMAN EVALUATION RESULTS. THE LAST LINE IS THE RESULTS OF HUMAN-WRITTEN POEMS IN TEST SET AND HUMAN EVALUATION SET. ALL SIGNIFICANCE TESTS ARE MEASURED BY T-TEST, AND THE RESULTS SHOW THAT THE IMPROVEMENTS OF OUR MODEL ARE SIGNIFICANT ON THE DATASET, I.E., P-VALUE < 0.01.

	Automatic Evaluation						Human Evaluation			
	PPL	**Rough-L**	**Distinct-1**	**Distinct-2**	**Novelty-2**	**Novelty-3**	**Fluency**	**Coherence**	**Impressiveness**	**Poeticness**
Plan	30.98	0.3375	0.2978	0.7025	1020	4985	3.19	2.94	2.74	2.97
DN	26.33	0.3423	0.3017	0.7203	1036	5006	3.23	3.08	2.79	3.09
EED	27.45	0.3533	0.3121	0.7655	1058	5073	3.32	3.29	3.16	3.21
Mem	27.72	0.3425	0.3335	0.7836	1077	5092	3.55	3.52	3.35	3.18
WM	26.84	0.3654	0.3226	0.7521	1043	5089	3.68	3.73	3.50	3.30
GRR-Refine	**24.06**	0.3178	0.3237	0.7636	1050	5070	3.85	3.94	3.44	3.37
GRR	28.52	**0.4138**	**0.3447**	**0.8287**	**1085**	**5102**	**3.88**	**3.98**	**3.86**	**3.40**
Human-written	25.32	/	0.3640	0.8275	1064	5072	4.06	4.20	4.35	4.42

Distinct-1/2: it reflects whether poems are diverse in content. It is defined as the ratio of unique uni/bi-grams over all uni/bi-grams in generated poems [36]. **Novelty-2/3:** it is a new metric defined in this paper, which is calculated as the number of new bi-/tri-grams that do not appear in the training set.

Human evaluation is necessary for poetry generation. In order to make our results more believable, we use four criteria for human evaluation following Yi et al. [37] and Wang et al. [25]: **Fluency:** it measures whether the poem reads smoothly and fluently. **Coherence:** it measures the relevance of adjacent lines in one poem. **Impressiveness:** one of our motivations is trying to learn some good patterns explicitly from human-written poems and then to generate new ones. We design this criterion to let annotators judge whether our model generates some impressive expressions. **Poeticness:** it represents the overall quality of a poem, such as whether a poem could convey a poetic image and artistic conception.

We randomly select 200 groups of keywords and feed them into 7 models. For each group, we shuffle these 7 poems and the corresponding human-written one, then display them in one page[8], and the annotators do not know their sources. During evaluation, annotators can also see retrieval lines to help them judge the novelty to some extent. Each criterion is assessed with a score from 1 (worst) to 5 (best) by 8 annotators, and the average score for each criterion is computed. The annotators are all postgraduate students in literature background, and they took 10 days on average to finish the evaluation. The Fleiss' kappa [38] value is 0.403.

E. Experimental Results

Now we demonstrate our experimental results on the dataset in terms of automatic evaluation and human evaluation.

1) Automatic Evaluation Results: The left part of Table II shows the automatic evaluation results on our test set. Our proposed method GRR outperforms other models almost on all metrics. GRR-Refine receives the lowest *PPL* values, which shows that bidirectional sentence-level context is beneficial to fluency. It has been proven that lower *PPL* values usually correspond to simple and general sentences, thus a higher *PPL* for GRR indicates that our model can generates more diverse and novel words. A higher *Rough-L* score shows that our generated sentences are more similar to their references, which also implies their diversity.

GRR-Refine or EED help very little on *Distinct-1/2* score, even information like future sentences or similar sentences are provided to these models. Compared to EED, Mem and GRR-Refine, the highest *Distinct-1/2* score of GRR illustrates the effectiveness of the "refining vector", that is, rather than using the entire retrieval candidates, memory-stored sentences or generated lines, the "refining vector" is more useful to reduce the noises and improve the diversity. Our approach increases both *Distinct-1/2* and *Novelty-2/3* significantly, which indicates that it generates more diverse and creative expressions. All significance tests are measured by t-test, and the results show that the improvements of our model are significant on the dataset, i.e., p-value < 0.01.

2) Human Evaluation Results: Human evaluation results are shown in the right part of Table II. GRR receives the best evaluation on all metrics. The evaluation results on *Coherence* and *Impressiveness* prove that our model can jointly improve coherence, diversity and novelty. Compared with DN and WM, GRR-Refine has higher *Coherence* score, which indicates the validity of bidirectional sentence-level context for improving the coherence. Besides, human evaluation results on *Fluency* and *Coherence* show that *PPL* is not totally reliable for sentence fluency and especially poem coherence. For human, a sentence with diverse language usage can also be fluent but its *PPL* value may be high.

Impressiveness and *Poeticness* scores almost have the same tendency. GRR outperforms other methods significantly, which means that our proposed approach can generate some good and new expression patterns successfully, and more impressive patterns also help express intents and emotions clearly, which contributes to the overall *Poeticness*. All significance tests are measured by t-test, and the results show that the improvements of our model are significant on the dataset, i.e., p-value < 0.01.

Since human-written poems are mixed with other generated ones and evaluated by annotators, we put the results of them in the last line of Table II to show the gap between generated and human-written poems in our human evaluation set.

[8] Since annotators can see 8 poems at the same time, their scores are based on comparison, which are more reliable.

keywords	彩云 colorful cloud，炊烟 chimney smoke，小花 little flower，回忆 memory，往事 past stories，排遣 dispose，深情 affection，诺言 promise
Draft	我站在彩云之南，I am standing in the south of colorful clouds. 天蓝蓝袅袅炊烟，The sky is blue and chimney smokes curl up. // 你就是我心中的小花，You are the little flower in my heart, **那是**我对你的回忆，And **that** is my memory of you. // 过去有多少往事，How many stories have been? 是年少无知难排遣，It's hard to dispose naivety and ignorance // 是多少的深情，How much affection is it? 忘了昨天的诺言。Forget the promise of yesterday.
Retrieval Lines	散布在彩云之南 Scattered in the south of colorful clouds. 炊烟升袅袅 Chimney smokes curl up. 小花开遍山野 Small flowers spread over the mountains. 回忆打上一串串烙印 Memories of a string of imprints. 往事有多少 How many past stories. 无以排遣 No way to dispose. 啊深情无限 Ah, deep affection. 昨天许下的诺言 The promise made yesterday.
GRR-Refine	彩云飘在天边，Colorful clouds floating in the sky, 故乡的炊烟，Chimney smokes of hometown, 心中的小花，Little flowers in my heart, **这是**我一生的回忆，**These** are memories of my life. // 有多少过去的往事，How many past stories 难排遣的深情，And affection that is hard to dispose? // **却**不能忘记你的深情，**But** cannot forget your affection, 前世的诺言。And the promise of past life.
GRR	彩云散布在天边，（散布）Colorful clouds scatter in the sky. (scatter) 白云升炊烟，（升）Chimney smokes rise up to clouds. (rise) 小花开遍在窗前，（开遍）Small flowers blossom in front of the window. (blossom) 一串串回忆到永远，（一串串）A string of memories last forever. (a string of) // 过去的往事就在眼前，Past stories are in sight, **却**难以排遣，**But** it's hard to dispose. // 你至今对我无限的眷恋，（无限）You have infinite affection to me so far, (infinite) **是**你对我许下的诺言 **Since** it's your promise to me.

Fig. 3. Examples generated by our method. The keywords lie in the first block. The second block is a draft and the third block is corresponding retrieval lines. Words in parentheses in the last block is the impressive words candidates extracted by impressive words detection mechanism. Words in (red) bold play a positive role on coherence, and phrases with wavy underlines are impressive expressions. In order to better understand these generated poems, we also use "//" to separate slightly different semantic chunks.

F. Case Study

We present examples generated by GRR-Refine and GRR with the same keywords in Figure 3 for case study. Comparing the draft and the poem generated by GRR-Refine, we figure out that the correct usage of conjunctions and pronouns can improve the coherence of entire poem. For the draft, "you are the little flower in my heart, and that is my memory of you", this sentence is about the memory of a person and does not have many connections with preceding and following lines. In contrast, the fourth line in GRR-Refine poem writes that "these are memories of my life", and looking back to the first three lines in it, they are all in the structure that a noun with its modifier, which enhances the relationship in the first four lines. Word "affection" with underlines in the sixth line is also the keyword in the seventh line, which shows that the generation of the sixth line is influenced by its next line. Above characteristics also occur in the poem generated by GRR, for example, "but" and "since" are well used in this example. The reason for improving coherence is that we take bidirectional context into consideration, and this can help generate closely tied sentences.

When it comes to impressive words generation, phrases with wavy underlines in GRR poem are new patterns that do not appear in neither the draft nor corresponding retrieval lines. A poem can be creative and vivid if it includes various nouns, verbs and adjectives. As we can see, "scatter", "blossom" and "a string of" are all impressive expressions that can lighten a sentence. The GRR poem shows that our generated poems can be more diverse and creative.

VI. Conclusion and Future Work

In this paper, we propose a generate-retrieve-then-refine paradigm for poetry generation by imitating humans' composition process. It enables a generative model to leverage both generated draft and retrieval results. To improve the coherence, we use bidirectional sentence-level context from previous generated lines and draft lines. Also, we introduce the "refining vector" distilled by impressive words detection mechanism to generate newer and more impressive expressions. Experimental results on a large-scale modern Chinese poetry dataset show that our model outperforms baselines in terms of coherence and novelty. In the future, we will use other datasets to demonstrate the effectiveness of our approach, and further investigate the way to fulfill impressive words detection in an End-to-End framework.

Acknowledgements

We sincerely thank the anonymous reviewers for their helpful and valuable suggestions.

References

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

[2] X. Zhang and M. Lapata, "Chinese poetry generation with recurrent neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 670–680.

[3] W.-F. Cheng, C.-C. Wu, R. Song, J. Fu, X. Xie, and J.-Y. Nie, "Image inspired poetry generation in xiaoice," *arXiv preprint arXiv:1808.03090*, 2018.

[4] J. Zhang, Y. Feng, D. Wang, Y. Wang, A. Abel, S. Zhang, and A. Zhang, "Flexible and creative chinese poetry generation using neural memory," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 1364–1373.

[5] Y. Wu, F. Wei, S. Huang, Z. Li, and M. Zhou, "Response generation by context-aware prototype editing," *arXiv preprint arXiv:1806.07042*, 2018.

[6] R. Yan, "i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema." in *IJCAI*, 2016, pp. 2238–2244.

[7] Y. Xia, F. Tian, L. Wu, J. Lin, T. Qin, N. Yu, and T.-Y. Liu, "Deliberation networks: Sequence generation beyond one-pass decoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1784–1794.

[8] Q. Wang, Z. Zhou, L. Huang, S. Whitehead, B. Zhang, H. Ji, and K. Knight, "Paper abstract writing through editing mechanism," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 260–265.

[9] Y. Song, R. Yan, X. Li, D. Zhao, and M. Zhang, "Two are better than one: An ensemble of retrieval-and generation-based dialog systems," *arXiv preprint arXiv:1610.07149*, 2016.

[10] G. Pandey, D. Contractor, V. Kumar, and S. Joshi, "Exemplar encoder-decoder for neural conversation generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 1329–1338.

[11] Z. Cao, W. Li, S. Li, and F. Wei, "Retrieve, rerank and rewrite: Soft template based neural summarization," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 152–161.

[12] J. Li, R. Jia, H. He, and P. Liang, "Delete, retrieve, generate: a simple approach to sentiment and style transfer," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 1865–1874.

[13] K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang, "Generating sentences by editing prototypes," *Transactions of the Association of Computational Linguistics*, vol. 6, pp. 437–450, 2018.

[14] L. Shen, Y. Feng, and H. Zhan, "Modeling semantic relationship in multi-turn conversations with hierarchical latent variables," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5497–5502.

[15] Z. Liu, Z. Fu, J. Cao, G. de Melo, Y.-C. Tam, C. Niu, and J. Zhou, "Rhetorically controlled encoder-decoder for modern chinese poetry generation," in *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 2019, pp. 1992–2001.

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

[17] X. Wu, N. Tosa, and R. Nakatsu, "New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system," in *International Conference on Entertainment Computing*. Springer, 2009, pp. 191–196.

[18] N. Tosa, H. Obara, and M. Minoh, "Hitch haiku: An interactive supporting system for composing haiku poem," in *International Conference on Entertainment Computing*. Springer, 2008, pp. 209–216.

[19] R. Manurung, G. Ritchie, and H. Thompson, "Using genetic algorithms to create meaningful poetic text," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 24, no. 1, pp. 43–64, 2012.

[20] C.-L. Zhou, W. You, and X. Ding, "Genetic algorithm and its implementation of automatic generation of chinese songci," *Journal of Software*, vol. 21, no. 3, pp. 427–437, 2010.

[21] H. Manurung, "An evolutionary algorithm approach to poetry generation," 2004.

[22] E. Greene, T. Bodrumlu, and 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.

[23] L. Jiang and M. Zhou, "Generating chinese couplets using a statistical mt approach," in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2008, pp. 377–384.

[24] R. Yan, H. Jiang, M. Lapata, S.-D. Lin, X. Lv, and X. Li, "i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization." in *IJCAI*, 2013, pp. 2197–2203.

[25] Z. Wang, W. He, H. Wu, H. Wu, W. Li, H. Wang, and E. Chen, "Chinese poetry generation with planning based neural network," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1051–1060.

[26] X. Yang, X. Lin, S. Suo, and M. Li, "Generating thematic chinese poetry with conditional variational autoencoder," *arXiv preprint arXiv:1711.07632*, 2017.

[27] X. Yi, M. Sun, R. Li, and Z. Yang, "Chinese poetry generation with a working memory model," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 4553–4559.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[29] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013, version 3.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014, version 1.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[36] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2016, pp. 110–119.

[37] X. Yi, R. Li, and 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.

[38] J. L. Fleiss and J. Cohen, "The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability," *Educational and psychological measurement*, vol. 33, no. 3, pp. 613–619, 1973.