Learning to be Poetic: Automatic Generation of Chinese Song Ci Using RNN

Nan Du
Michigan State University
East Lansing, MI 48823, USA
dunan@msu.edu

Wei Wang
Michigan State University
East Lansing, MI 48823, USA
wangwe90@msu.edu

Zhuangdi Zhu
Michigan State University
East Lansing, MI 48823, USA
zhuzhuan@msu.edu

ABSTRACT

Song Ci is a precious cultural heritage of China, which is various in styles and sophisticated in syntactic rules. In this project, we develop a system to automatically generate Chinese Song Ci using Recurrent Neural Network (RNN). We use a vector space model to convert each Chinese character as a vector which still reserves the semantic relevance among different characters. Then we use the vector presentation as input to train the RNN model. We also implement Long Short Term Memory units (LSTMs) into the RNN model to learn the semantic meaning in long-distance. We will compare the performance of this model with the traditional genetic approaches used to generate poems. We hope that our system can learn the complete rule from training corpus without any given constraints, and can generate elegant Song Ci poems that follow syntax rules.

Keywords

Song Ci (poetry); Recurrent Neural Network; Long Short Term Memory

1. PROBLEM DESCRIPTION

1.1 Motivation

In this project, we propose and evaluate different approaches to automatically generate Chinese poems. Ci are one of the most important genres of Chinese classical poetry. As a precious cultural heritage, not many of them have been passed down onto the current generation. Therefore, the study of automatic generation of Ci is meaningful, not only because it supplements entertainment and education resources to modern society, but also because it demonstrates the feasibility of applying artificial intelligence in Art generation.

1.2 Background

Song Ci is a precious cultural heritage in China, which refers to Classical Chinese poetry typical of the Song dynasty. It arose with the so-called banquet music in Tang dynasty and reached its peak one hundred years later, as a major alternative to Shi poetry [3] .

Derived from the structure used in Tang poetry, Ci follows more complex and strict rules. There are more than 800 genres for Ci, which is called Cipai [19]. Each Cipai determine the number of characters for different lines, the arrangement of rhyme, and even the location of tones. To create a Song Ci in a specified Cipai, authors need to fill in the words according to the rule matrix associated with that Cipai. The uneven lines in Ci follow more continuous syntax consistency than traditional Chinese Tang poetry [3].

These complex rules for Song Ci make it difficult for AI systems to generate Song Ci with good property on structure or meaning consistency. Besides, compared with Tang poetry, the number of available Song Ci in a specified Cipai is relatively small [], which means we have limited numbers of training data to build any model.

1.3 Proposed Approaches

We propose one traditional approach, and two deep-learning approaches to generate Song Ci, respectively. For traditional approach, we use Genetic Algorithms (GA). We implement genetic algorithm based on the method proposed by Zhou et al. [23]. This approach implement a fitness function that is designed for evaluate the performance of Song Ci by considering the level and oblique tones-based coding method and the syntactic and semantic correctness. Thus additional information is needed, for example tone pattern and rhythm of words, syntactic pattern of sentences with different length and format of different Cipai. We use Cipai title and keywords as input to generate Song Ci that is strictly following the format and tone pattern of that Cipai. For deep learning approaches, our first model is to use a Recurrent Neural Network with Long Short Term Memory (LSTM) units. We train a RNN model by feeding sentences of Song Ci as the input, and ask the RNN model to generate the probability distribution of the next character in the sentence, given the sequence of previous characters. Especially, to capture the long term semantic dependencies between characters in Song Ci, we apply Long Short Term Memory units in the RNN model. Another deep learning approach is to use

Generative Adversarial Networks (GANs). [Nan: More content here]. We will compare the results generated by different approaches with respect to the structure, rhythmic and semantic consistency.

1.4 Technical Challenges and Proposed Solutions

The first challenge to build a general model for all types of Song Ci. Different from Shi poetry whose structure is strict, Song Ci has more than 800 set of Cipai, and different Cipai follows different structural or rhythmic patterns. Therefore, it is difficult to generalize templates or rules for all the Song Ci from limited training dataset. Our solution is to create a model based on Recurrent Neural Network. For every line generated in the Song Ci, its probability is based on the probability of all previously lines. So that the grammatical and rhythmic rules can be automatically captured.

The second challenge is to extract useful features from our training corpus. Inappropriate feature extraction approach may lose the semantic meaning of each character which leads to meaningless outputs from the system. Our solution is to build a vector space model to pre-process the poetry corpus. We first tokenize each Chinese character in the corpus, and then represent each character as a vector which retains the semantic relations among different characters. Therefore, characters with similar meanings have smaller distance in the vector space, while characters that are irrelevant in meanings have larger distance.

The third challenge is to maintain consistent and poetic meanings throughout the generated Song Ci. Compared with Shi poetry, Song Ci are much longer in length and therefore more complicated in context. It is difficult to keep long-distance memory using conventional RNN. Our solution is to use a Long Short Term Memory (LSTM) model that can track the long-distance semantic information automatically.

2. RELATED WORK

Approaches to poetry automatic generation can be divided into the following categories.

Using rules and templates. This approach adopts templates to generate poems that comply with a set of rules and constraints. These rules may be derived from user queries and additional lexical resources [17, 20], parts of speech and WordNet patterns [13], or from semantic and grammar templates [14].

Using genetic algorithms. This approach is mainly based on natural selection. It generates all possible candidates, and use search and evaluation algorithms to select the optimal one [10, 11].

Using Statistical Machine Translation (SMT) methods. This approach first receives keywords and extract most relevant constituents to theses keywords. Next, it generates poems by iteratively selecting among these constituents based on phonological, structural, and poetic requirements [9].

Using neural network. Recently, approach using neural network have achieved great success in poem generation. In general, this kind of approach will generate new sentence based on previously generated content, so that the generated sentence can capture the semantic consistency automatically [1, 18]. For example, Zhang *et al.* built a quatrain generation model using a Recurrent Neural Network model. Their model generates the first line of the poetry from the given keywords, and then generate subsequent lines by backtracking the status of the lines previously generated.

3. METHODS

To automatically generate Song Ci, First, we preprocess the Song Ci corpus and tokenize each character. Then we use a vector space model to convert each Chinese character in the corpus to be a vector presentation in the vector space so that characters with similar semantic meanings have small distance in the vector space. Using the vector space as training data, we build a Recurrent Neural Network (RNN) that can generate Song Ci with coherent and poetic meanings. We add Long short-term memory (LSTM) units in our RNN model to capture long-term semantic dependencies in Song Ci.

3.1 Genetic Algorithm

Genetic algorithm(GA) belongs to the class of evolutionary algorithms(EA). It is inspired by the process of natural evolution. Operations in biological evolution process such as mutation, crossover and selection are used in genetic algorithm [?]. Genetic algorithm employ operations close to random search for locating the globally optimal solutions. Since the first time genetic algorithm was proposed by Holland [?] in 1975, it has been studied and developed for nearly 40 years. It is widely used in many areas, especially for optimization and search problems.

In genetic algorithm, the initial population which is a group of candidate solutions are randomly generated based on the nature of the problem. Each individual has a set of properties which can be mutated and altered. Evolution is a following iterative process, population in each iteration is called a generation. In each generation, the fitness value of each individual is evaluated. Fitness value is calculated by the objective function in the optimization problem that is used to evaluate the performance of the solutions. The most fit individuals are selected under some strategy to become parents. The properties of those selected individuals will be modified to form new individuals. Modification of individuals include recombination and possibly randomly mutation.

The new generation will enter the next iteration of the algorithm. When the population reached a certain stop criteria, for example a satisfactory fitness level has been reached or a maximum number of generations has been produced.

In 2010, Zhou et al. [23] proposed an approach to automatically generate Song Ci by genetic algorithm. This approach implement a fitness function that is designed for evaluate the performance of Song Ci by considering the level and oblique tones-based coding method and the syntactic and semantic correctness.

3.2 RNN

RNNs are the family of the deep learning structures to process sequential data [15]. Parameter sharing across the different parts of the model is the key idea that makes RNNs to be able to deal with the sequential data. However, a simple RNNs cannot learn long time dependency as in the optimization this term tends to vanish or explode very fast [5]. To solve this challenge, gated RNNs is proposed and becomes one of the most effective practical models that used for sequential data.

3.3 LSTM

Long short-term memory (LSTM) model [7] is one branch of such gated RNNs that is extremely successful in the application like speech recognition, machine translation, and handwriting generation. The key idea of LSTM is to introduce a self loop so that gradient can flow for long duration. The self loop (internal recurrence) is located in "LSTM cells" with outer recurrence like ordinary recurrent network. The weight of self-loop is controlled by a forget gate $f_i^{(t)}$:

$$f_i^{(t)} = \sigma(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)})$$

Where $\boldsymbol{x}^{(t)}$ is the current input vector and $\boldsymbol{h}^{(t)}$ is the current hidden layer vector, containing the outputs of all the LSTM cells. $\boldsymbol{b}^f$, $\boldsymbol{U}^f$, and $\boldsymbol{W}^f$ are biases, input weights, and recurrent weights of the forget gate, respectively. The internal state of LSTM cell is updated with the following equation:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma(b_i + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)})$$

And the external input gate unit $g_i^{(t)}$ is computed with the following equation:

$$g_i^{(t)} = \sigma(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)})$$

The output $h^{(t)}$ and the output gate $q_i^{(t)}$, are updated using sigmoid function also:

$$\begin{aligned} h_i^{(t)} &= \tanh(s_i^{(t)}) q_i^{(t)} \\ q_i^{(t)} &= \sigma(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}) \end{aligned}$$

LSTM is proven to be able to learn long-term dependencies more effectively than normal RNNs. In our project, we will use LSTM as our main method. We also plan to compare LSTM performance with other network structures.

3.4 Generative Adversarial Networks

Deep generative model is one of the most promising approached to achieve the target than analyze and understand real-word data, such like image, video, and text. The main characteristic of generative model is trying to model the distribution of input implicitly or explicitly [2]. So it possible to generate synthetic data points in the input space.

Differentiable generator network is the key idea for many different generative models. Neural network can be treated as a differentiable function $g(\boldsymbol{z}; \boldsymbol{\theta}^{(g)})$, transforming sample of latent variables $\mathbf{z}$ to samples $\mathbf{x}$ or to distributions over $\mathbf{x}$ [5].

Generative Adversarial Network (GANs) [6] is a very popular different generative models by pairing the generator network with a discriminator network.

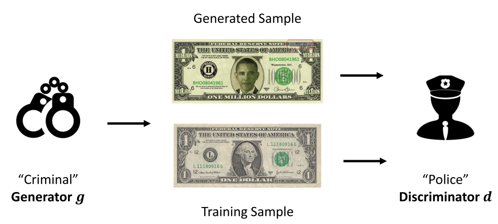

Figure 1: "Criminal" and "Police" example of GAN-model

Underlying scheme of GAN is the competition of generator network and discriminator network in a game theoretic scenario. Intuitively, we can think a scenario of money counterfeiting criminals and policemen. The generator, in this scenario the criminals, want to produce the counterfeited currency without detected by police. And the discriminator, in this case police, want to detect the fake currency. Both criminals and police will improve their methods to compete with each others in the scenario. Ideally, the competition will reach a zero-sum game, which counterfeits are indistinguishable from the genuine articles, and the discriminator output $\frac{1}{2}$ everywhere.

To formulate the learning of GANs, we can use a function $v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$. In the contrast, the generator receives $-v(\boldsymbol{\theta}^{(g)}$ as its own payoff. At the convergence g^*, we have:

$$g^* = \arg\min_g \max_d v(g, d)$$

And the payoff function v is given by:

$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x} \sim p_{data}} \log d(\boldsymbol{x}) + \mathbb{E}_{\mathbf{x} \sim p_{model}} \log(1 - d(\boldsymbol{x}))$$

This will drive generator attempt to emulate to the real sample so that it can fool the classifiers. Meantime, the discriminator will attempt to learn to correctly classify real and fake samples.

Original GANs focusing on using convolutional neural network based model to generate image data. By constructing the generator and discriminator using RNNs replacing CNNs, we can build a GANs to generate sequential data. For example, C-RNN-GAN model [12] use recurrent neural network based GAN to generate continue music data.

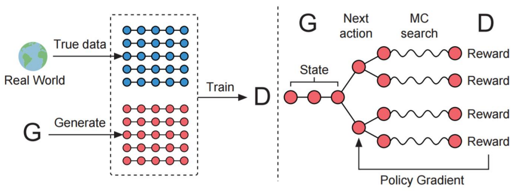

Figure 2: The scheme of SeqGAN model

However, to apply GANs on sequential data with discrete token is not-trivial [8]. For discrete sequence, the sampling process cannot be described by a differentiable process we discussed before. So an alternative approach is propose for SeqGAN Model [21]. The generator in the SeqGAN is treated as an agent of reinforcement learning. Discriminator will evaluate the sequence and the feedback of discriminator will used as guide of learning of generator. The generative model is trained by policy gradient, avoid the challenge of differential of discrete tokens in traditional GANs. SeqGAN model uses RNNs as generator and CNNs for discriminator.

4. DATA DESCRIPTION

For our experiment, we obtained dataset for both Tang Shi and Song Ci. Many research projects were conducted for automatically generating Tang Shi. So we can evaluate our experiment result by comparing with these machine-created Tang Shi. And then we can move forward to Song Ci.

4.0.1 Tang Poetry Corpus

We use Quan Tangshi as our Tang Poetry corpus. [4]. It was commissioned by Yin Cao in 1705 and published under the name of Kangxi Emperor. It contains 49,000 lyric poems (in the dataset we used it has 49,274 poems) and is believed the largest collection of Tang poetry. We obtained the dataset from the server of [22]. The dataset is well organized and is ready to use as the input of our experiments.

4.0.2 Song Ci Corpus

We download the Quan Song Ci dataset as our corpus for Song Ci. Quan Song Ci collected 21,116 Song Ci poems (Data set we used contains 18986 poems). Although there are several previous projects, unlike the parsed Tang Poetry Corpus, the dataset we can get is not well-formatted. For our analysis, preprocessing is conducted in the following steps: First, we extracted the name of all poets from the list in collections. Second, then we can distinguish those lines of names and those lines of poems. Third, we filtered the length of text smaller that certain number and without any period in it. Most titles are just the Cipai, some may contain a subtopic. We treat Cipai separately as it set the format of the poem. Fourth, we then identify the title of poems with its main text. Finally, we generate a tab separated file for further process.

5. EVALUTAION

5.1 Completed Milestones

5.1.1 Background Survey

We conducted large amount of survey on the state of the art approaches of SongCi generation. We find that this task attracts many interests both from the Natural Language Processing area and Machine Learning area. The approaches can be generalized into two kinds: We either specify the generation rules (using templates), or build a model which can learn these rules automatically (using neural network). We also implemented some of the approaches proposed in previous work.

5.1.2 Corpus Search and Analysis

The dataset we use contains 18668 Ci, which contains a total number of 1183 poets and 1170 Pai. This dataset basically covers Ci generated during the entire Song Dynasty and the beginning of Yuan Dynasty. We analyze the number of poems wrote by each poet, which shown in Figure 3. Most of the poems are created by the first poets. The one that creates the most poems is Qiji Xin, one of the most famous poet of the Southern Song Dynasty. His poems covers a wide range of styles. Among all those poems, the bold style is most will known by now. In addition, we statistic the Pai of each poem. Ci was first used as a lyric, and Pai is the name of the tune. Each Pai has a specific melody and rhythm, so Ci has a fixed format requirements, such as the number of sentences, the number of words per sentence, pronunciation of those words, rhyme and so on. The statistical result is shown in Figure **??**. The most popular Pai is Silk-Washing Stream, followed by Prelude To Water Melody, Partridge Sky, Pusaman and River of Red. And there is good reason to believe that the songs that corresponding to these Pai are beautiful in melody, lively in rhythm and easy to sung, which caused them to be so popular in ancient China.

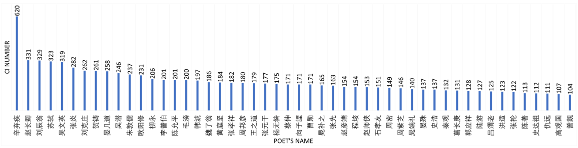

Figure 3: Poem Number Created by Top 50 Productive Poets

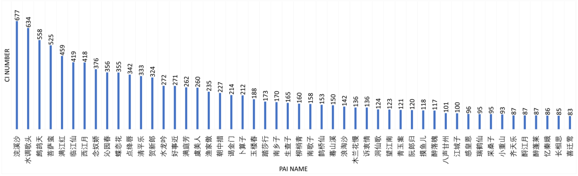

Figure 4: Poem Number of Top 50 Popular Pai

Word frequency analysis is to statistic and analyze the number of important words in the text, which is an important method of text mining. It is a traditional and useful content analysis method. The basic principle is to determine the overall style and theme of the entire article by the frequency of the words. By analyzing the word frequency in the poems, we have a general understanding of the style of poems and the process of writing those the poem, which can help us get more familiar with the grammar rules and themes of Ci. The most commonly used words can reveal common theme of Ci and the corresponding feelings. For example, we analyzed word frequency of season in our database. The result is shown in Figure 5. We found that spring related words reached 2606, these words appeared in our dataset for 9210 times. Followed by autumn, there are 1167 words associated with autumn and appeared 3992 times. The unique scene in spring and autumn can trigger people's emotions, which might be the reason that so many poems are related with these two seasons. From most frequently used words, shown

	Related Word	Total Appearance
Spring(春)	2606	9210
Summer(夏)	110	217
Autumn(秋)	1167	3992
Winter(冬)	99	213

Figure 5: Statistical Data of Season Related Word Frequency in Dataset

in Figure **??**, we found that the moon, east wind, mortal world, wine, dream, rain, flowers , sunset, old friends are the most commonly used images. Commonly used places, including Jiangnan, West Lake, Changan, Fairy Isle, Yangzhou. Commonly used verbs including laugh , come back, go back, lovesickness, look back, meet by chance. Commonly used emotions are hate, worry, hard, sigh, desolate, haggard. These words represent a very broad theme and style of Ci, including the description of leaving and missing, pride and enthusiasm, seasonal terms, chanting things, chanting nostalgia and so on.

5.1.3 Implementation of a Vector Space Model

Vector space models (VSMs) represent words in a continuous vector space where semantically similar words are mapped to nearby points. We implemented this model to find the semantic relations between each Chinese character, so that given a few of keyword characters, such as 'spring' and 'beauty', we can generate poetries with coherent meanings using characters which are close to these keywords in the vector space. We give a visualized result in Figure 8. The figure is embeded with 100 Chinese characters in a 2-D space, which are randomly chosen from the most frequent 500 Chinese characters in the Song Ci corpus. The 2-D space corresponds to the first two dimensions in the vector space. We can see that words such as 'spring', 'sunny', and 'breeze' are very close in the 2-D space, which convey similar sentimental feelings to readers.

Figure 6: Extracting rhythmic features

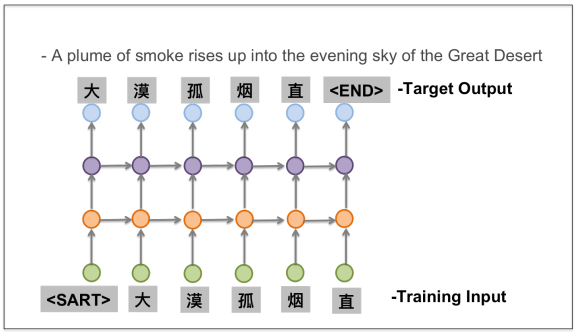

Figure 7: Workflow of the RNN model

5.1.4 Implementation of a RNN + LSTM Model

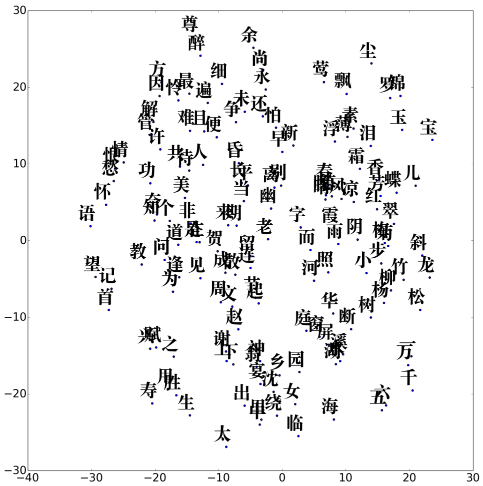

Figure 8: Projections of most frequent characters to a 2-D space using the word embedding model

We implement a preliminary version of our model. It is a RNN model with Long Short Term Memory units (LSTM), which can capture long-term dependencies. We used deep learning packages called *TensorFlow* [16] to implement our model. The code is written in Python. We present a preliminary result in Figure 10, which is a quatrain poem.

6. CONCLUSION AND FUTURE WORK

Judy: To be contniued

7. REFERENCES

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

[2] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[3] Z.-q. Cai. *How to read Chinese poetry: a guided anthology*. Columbia University Press, 2008.

[4] Y. Cao and D. Peng. Quan tangshi. *Beijing: Zhonghua shu ju*, 1960.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[8] F. Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[9] 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*, pages 377–384. Association for Computational Linguistics, 2008.

[10] H. Manurung. An evolutionary algorithm approach to poetry generation. 2004.

[11] R. Manurung, G. Ritchie, and H. Thompson. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64, 2012.

[12] O. Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[13] Y. Netzer, D. Gabay, Y. Goldberg, and M. Elhadad. Gaiku: Generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39. Association for Computational Linguistics, 2009.

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

[15] D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. Hinton. Sequential thought processes in pdp models. *Parallel distributed processing: explorations in the microstructures of cognition*, 2:3–57, 1986.

[16] TensorFlow. An open-source software library for Machine Intelligence. `https://www.tensorflow.org/`, last accessed 2017.

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

[18] Q. Wang, T. Luo, D. Wang, and C. Xing. Chinese song iambics generation with neural attention-based model. *arXiv preprint arXiv:1604.06274*, 2016.

[19] Wikipedia. Ci (poetry). `https://en.wikipedia.org/wiki/Ci_(poetry)/`, last accessed 2017.

[20] 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*, pages 191–196.

浣溪沙	— Lyrics to the Melody of Sandy Creek Washers
溪头晚月砧残梦，	The moonlight reflected from a brook lightened my withering dream.
沙巷桥边雨吹长，	The drizzle in the night blew to the small alley near a bridge.
桃花渡有故潭冰。	Ice from the old pond remains on the Peach Blossom ferry.
落叶斑斑飘雨后，	Falling leaves were so colourful after that rain.
锦衣幽锁白山流，	Brocade gowns packed up, White Mountain rivers running off.
对酒难高雁远休。	I raise my goblet to the sky with a wild goose flying away.

Figure 9: A Song Ci generated using LSTM

浣溪沙	— Lyrics to the Melody of Sandy Creek Washers
别来已是二十年，	It’s been twenty years since I left.
寒轻离别岸呜咽。	The river cried for me at that moment
阳春归路转千年。	Now it’s spring, and I don’t know when can I go back.
借问春寒梅旧否，	It’s still cold, and I asked whether plum blossom was like before.
夜来去后雨轻扇。	Rain comes without a sound during night.
佳人含笑立蹲前。	My loved one stand in front of me with smile.

Figure 10: A Song Ci generated using Genetic Algorithm

Springer, 2009.

[21] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*, 2016.

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

[23] C.-L. Zhou, W. You, and X. Ding. Genetic algorithm and its implementation of automatic generation of chinese songci. *Journal of Software*, 21(3):427–437, 2010.