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 is 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 [**?**] .

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 [**?**]. 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 [**?**].

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 named Genetic Algorithms, and two deep-learning approaches named Recurrent Neural Networks and Generative Adversarial Networks, respectively.

- Genetic Algorithms (GA). We implement genetic algorithm based on the method proposed by Zhou et al. [**?**]. 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.
- Recurrent Neural Network (RNN). 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 se-

mantic dependencies between characters in Song Ci, we apply Long Short Term Memory units in the RNN model.

- Generative Adversarial Networks (GAN). [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. We explore with three approaches to address this challenge. For the GA approaches, we organize a set of rules, such as the tone patterns and rhythms of words, the syntactic patterns of sentences with different length, and the structure constraints of different Cipai, so that only candidate poems that satisfy these rules will be chosen as the output. For the RNN model and the GAN approaches, we do not set any pre-defined constraints, but feed our model with carefully organized training data, so that the grammatical and rhythmic rules can be automatically captured during the training.

The second challenge is to extract features that retain both rhythmic and semantic relevance among characters in our training corpus. Our solution is to use a word-embedding matrix to map each character in the corpus into a vector representation Therefore, characters with similar meanings or same rhymes will have smaller distance in the vector space.

The third challenge is to maintain consistent and poetic meanings throughout the generated Song Ci. For GA approach, we implement a fitness function to evaluate the performance of Song Ci. Each candidate Song Ci generated by this model will be evaluated on the fitness level , oblique tones-based coding method, and the syntactic and semantic correctness. For the RNN approach, since Song Ci are much longer in length and therefore more complicated in context compared with Shi poetry, it is difficult to keep long-distance memory using conventional RNN. Therefore, we 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 [**?**, **?**], parts of speech and WordNet patterns [**?**], or from semantic and grammar templates [**?**]. Some other approaches consider the poetry generation as an optimization problem based on a summarization framework with several constraints [**?**].

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 [**?**, **?**]. For example, Zhou et al. [**?**] proposed an approach in 2016 to automatically generate Song Ci by genetic algorithm. They implemented 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.

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 [**?**].

Using neural network. Recently, approach using neural network have achieved great success in poem generation, which considers the poetry generation as a sequence-to-sequence generation problem. 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 [**?**, **?**]. 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.

Using GAN [Nan: Softy kitty warm kitty little ball of fur ...]

3. METHODS

To automatically generate Song Ci, first, we preprocess the Song Ci corpus by mapping each character to a vector representation using a word-embedding matrix, which was previously trained using all corpus in our dataset. Then we use each character's vector representation as input to train our GA model, RNN model, and GAN model, respectively.

3.1 Preprocessing

We use word embedding to map Chinese characters from the poem corpus to vectors of real numbers, which consists of two steps: tokenization and word embedding.

3.1.1 Tokenization

We associated each Chinese character with a unique value as its ID, so that the most frequent word in the corpus will get a highest value, while the less frequent words have smaller ones. During our project, we carefully tune the size of the vocabulary set $|V|$ so that the most $|V|$ frequent words will be kept in our vocabulary, while the other unfrequent words will be mapped to the same *UNKOWN* token to avoid overfitting. Then we use the ID as the feature for each word to train our word embedding matrix.

3.1.2 Word Embedding Using Skip-Gram Model

We will create a word embedding matrix by training the skip-gram mode which is a single-layer neural network. The input of the model is a single word w_I, and the output is the words in its context $\{w_{O,1}, w_{O,2}, \ldots, w_{O,C}\}$ defined by a word window of size C. As shown in Figure 1, x represents the one-hot encoded vector corresponding to the input word in the training text, and $\{y_1, y_2, \ldots, y_C\}$ are the one-hot encoded vectors corresponding to the output words in the training text. The $V \times N$ matrix W is the weight matrix between the input layer and hidden layer whose i^{th} row represents the weights corresponding to the i^{th} word in the vocabulary. This weight matrix is what we call the word-embedding matrix because it contains the vector encodings of all of the words in our vocabulary.

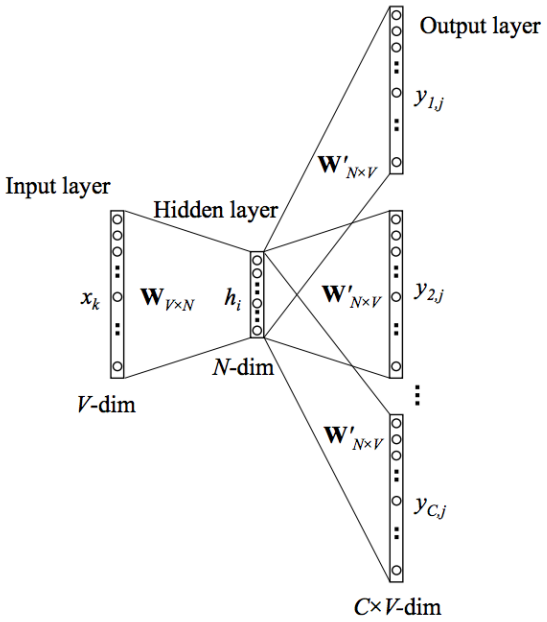

Figure 1: A skip gram neural network

3.2 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.

3.3 Recurrent Neural Network

RNNs are the family of the deep learning structures to process sequential data [**?**]. It is specialized for processing a sequence of valuesx $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$. 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 [**?**]. To solve this challenge, gated RNNs is proposed and becomes one of the most effective practical models that used for sequential data.

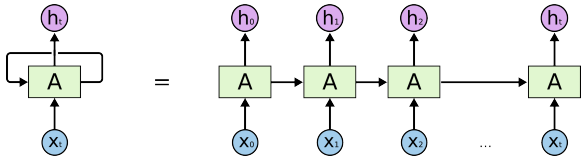

Figure 2: A Recurrent Neural Network

3.3.1 Long Short Term Memory

Long short-term memory (LSTM) model [**?**] 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. As shown in Figure 3, 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.

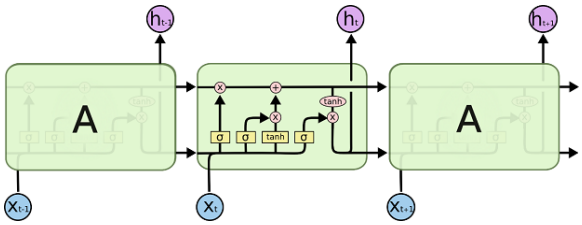

Figure 3: A Long Short Term Memory network

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 [**?**]. 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 $\mathbf{x}$ [**?**].

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

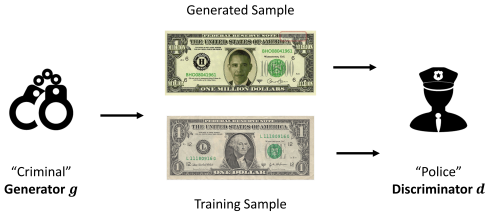

Figure 4: "Criminal" and "Police" example of GAN-model

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 [**?**] use recurrent neural network based GAN to generate continue music data.

However, to apply GANs on sequential data with discrete token is not-trivial [**?**]. For discrete sequence, the

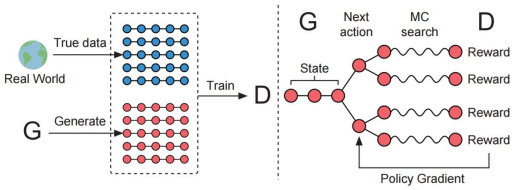

Figure 5: The scheme of SeqGAN model

sampling process cannot be described by a differentiable process we discussed before. So an alternative approach is propose for SeqGAN Model [**?**]. 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. EVALUTAION

For each of the three approaches, we use the same preprocessing steps to get the same features. Then we compare among the Song Ci generated from the three approaches based on three metrics : structure, rhyme, and semantics.

4.1 Data Collection

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 poem. And then we can move forward to Song Ci.

- **Tang Poetry Corpus.** We use Quan Tangshi as our Tang Poetry corpus. [**?**]. 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 [**?**]. The dataset is well organized and is ready to use as the input of our experiments.

- **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.

- **Hybrid Corpus.** Since the number of preserved Song Ci for each Ci Pai is very limited, which makes it difficult to train any neural network, for each Ci Pai, we generate around 1000 Song Ci with the specified Ci Pai using the Genetic Algorithm, and inject them into our dataset for further training the RNN model and GAN model.

4.2 Ci Pai Analysis

We analyze the statistics for different Ci Pai in our corpus. As shown shown in Figure **??**, the most popular Pai is Silk-Washing Stream, followed by Prelude To Water Melody, Partridge Sky, Pusaman and River of Red.

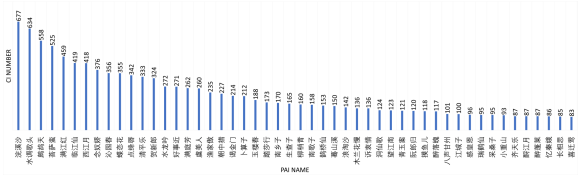

Figure 6: Poem Number of Top 50 Popular Pai

4.3 Word Frequency Analysis

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 7. 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 in Figure 8, we found that the moon, east wind, mortal

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

Figure 7: Statistical Data of Season Related Word Frequency in Dataset

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.

Figure 8: Wordcould of Frequently Used Words in Dataset

4.4 Implementation of Genetic Algorithm

Initially, we collect additional information about syntactic pattern of sentences with different length, format of tone pattern and rhythm and format of different Cipai from past research work on Song Ci. The syntactic pattern for traditional Chinese poem are shown in Figure 9. In this figure, sentence length means how many characters are in this sentence. '*' in syntactic pattern represent a character and '/' used to split sentences into several parts. Words in these sentence shell not cross the split. Otherwise, this sentence may not be that easy to read and understand. What we can see is that one several different syntactic patterns are allowed for the same sentence length, such as sentence with five characters, the sentence can either have two characters at the beginning then three characters or vice versa. This actually gives traditional poems much freedom on expression.

Then we defined several properties for characters and words in Song Ci. Tone pattern which is called pingze in

Sentence Length	Common Syntactic Pattern			
3	**/*	*/**		
4	**/**			
5	**/***	***/**	*/****	
6	**/****	****/**	***/***	
7	***/****	****/***	*/******	
8	***/*****	****/****	*/*******	**/******
9	***/******	****/*****	*****/****	******/***

Figure 9: Syntactic Patterns for Sentences with Different Length.

Chinese, we use 1 to represent Ping, 2 to represent Ze. There are twenty four rhythm in Chinese, in traditional Chinese poem, usually the last characters of every sentence are required to be the same rhythm. We get pingze and rhythm information from existing datasets. The correlation between words we use python package word2vec(https://code.google.com/archive/p/word2vec/) to quantify the relationship between words, which will be later used in generating Song Ci that is closely related with keywords.

We also extract format for different Cipai from existing dataset, which contains sentence number, length, tone pattern and the delimiter of each sentence. For example, Huanxisha is a very popular Cipai in Song Ci. The sentence format of Huanxisha is ['0201021', '0102211', '0102211', '0201122', '0102211', '0102211']. In this format, there are 6 sentences, each sentence contains 7 characters. 0,1 and 2 represent the requirement of Pingze, 0 means both ping and ze will work in this position. Each sentence will apply the syntactic pattern previous defined.

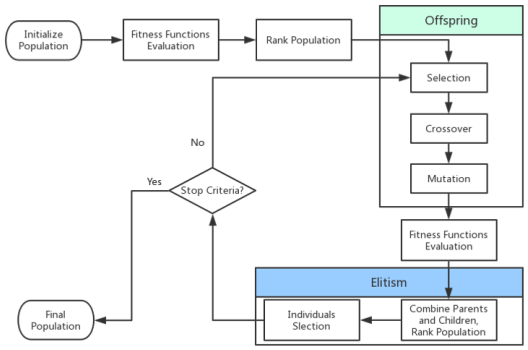

Figure 10: Process of Generating Song Ci by Genetic Algorithm

Based on the rule of compose Song Ci, we found that Song Ci is a combination of words. Thus, traditional method with pre-defined rules can easily generate Song Ci with correct format. For genetic algorithm, we use a Cipai title and several keywords as input. Based

on the sentence format and tone requirement, words that are highly correlated with keywords are selected as candidate words for initial population. Based on randomly chosen rhythm, pre-defined sentence form, syntactic patterns and tone patterns, program randomly put words that satisfied all the requirement to each position and generate the first generation of candidate poems. The initial population is strictly follows the format and rules of Song Ci.

Then all candidate poems go through the fitness evaluation process. Based on score of syntactic and semantic correctness, correlation with keywords, correlation between sentences and tone, rhythm pattern, a fitness value is generated for each candidate poem. Best several individuals are selected as parents to generate offspring with mutation and crossover. Mutation means characters within this poem will change randomly. Crossover represents two poems randomly exchange part of their sentences and form two new poems. All offspring and parents together forms the next generation and need to be evaluate on fitness value. Then this iteration will continue until satisfactory fitness level has been reached or a maximum number of generations has been produced.

In the last, we'll manually choose best poem from the final generation. The whole process is shown in Figure 10. Final results are shown in following section.

4.5 Implementation of a Word Embedding Model

A word embedding model represent words in a continuous vector space where semantically similar words are mapped to nearby points. We give a visualized result in Figure 11. Each of the Chinese characters is embedded into 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.

4.6 Implementation of an RNN Model

We used deep learning Python modules called *TensorFlow* [**?**] to implement our RNN model. The core parameters are given in Table 12. The core of the model consists of an LSTM cell that processes one word at a time and computes probabilities of the possible values for the next word in the sentence. The general workflow of our model is shown in Figure 12. The memory state of the network is initialized with a vector of zeros and gets updated after reading each word.

- **Batch Input.** We want to train the RNN model in an iterative fashion, so the first step is to seperate the training set into different batches with the same size. The larger the batch, the less epoch time

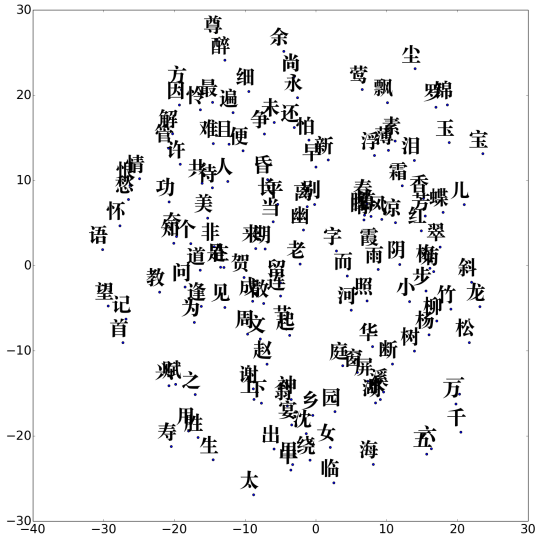

Figure 11: Projections of most frequent characters to a 2-D space using the word embedding model

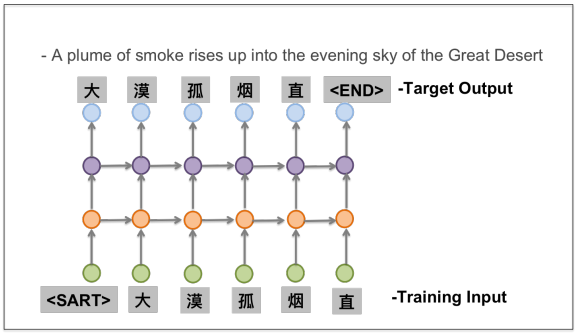

Figure 12: Workflow of the RNN model

the model needs to finish training. In our model, we tune the batch size to be 64, so each time a set of 64 Song Ci will be used as the input to our model, and the value of the model state is updated after processing each batch of texts.

- **Truncated Backpropagation** In order to make the learning process tractable, it is important to

Table 1: RNN model parameters

Parameter	Value
batch_size	64
embedding size	64
hidden layer	48
dropout rate	0.8
learning rate	0.0001
epoch time	50

create an unrolled version of the network which contains a fixed number of LSTM inputs and outputs, known as the number of steps. The model is then trained on this finite approximation of the RNN. In our model, we tune the number of steps to be the ideal character length in the Song Ci with a specified Ci Pai. For example, in order to train a Song Ci with Ci Pai "Sandy Creek Washers", we set this number to be 48, because the input Song Ci is all of length 48 at a time. Then we can perform a backward pass after each such input block.

- **Optimizing Loss Function** We aim to minimize the average negative log probability of the target words:

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln p_{target_i}$$

We implement by using *TensorFlow* 's *seq2seq* library. It will generate the weighted cross-entropy loss for a sequence of logits, based on three inputs: the logits, the targets, and the weights.

- **Stacking Multiple LSTMs** We use multiple layers of LSTMs to process the data so that the output of the current layer will become the input of its succesor. In our model, we use the libray called *MultiRNNCell* to implement three LSTM networks to strike a balance between the training time and the quality of the model output.

We present a preliminary result in Figure 14, which is a quatrain poem.

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 13: A Song Ci generated using LSTM

5. CONCLUSION AND FUTURE WORK

Judy: To be contniued

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 14: A Song Ci generated using Genetic Algorithm