

SI630 Final Project Report

Version 1.0

Xiaoxue Xin

Abstract

Poetry generation is an interesting problem in the field of natural language processing. Most research approaches employ templates to construct poems according to a set of constraints. Genetic algorithms also used for the poetry generation. We propose a generator model based on recurrent neural network that learned from Chinese poems corpus. In our model, we can generate both five-character and seven-character Chinese poems with rhyme. The evaluation with human judgement shows that our auto-generated poems outperform the baseline.

1 Introduction

Poetry is the abstract of human language. People use poetry to depict natural landscape and express their emotions. Some of them express optimistic viewpoints toward struggling situations, and some of them show their depressive feelings in tough conditions. All of them either inspire other people or unleash common rage. Audience get similar feelings, such as happiness, optimism, or depression, from reading poems. Nowadays, more and more people use poems to show their feelings in daily life. Among a variety of types of poetry, quatrain is one of the most common styles. An example of a quatrain is shown in Figure 1.

Every language has its own rules about writing poetry. In our model, we focus on traditional Chinese poetry generating with rhyme at the end of each line. Since the rhyme is very important for poetry, we also consider the rhyme problem in our model. We try to generate a reasonable poem, which means that we can use our four lines poem to depict a picture or express a certain feeling or tell a story.

Some researchers used templates to construct poems according to a set of constraints. The Haiku

静夜思

床前明月光, (P P Z Z P)

疑是地上霜。 (* Z Z P P)

举头望明月, (* Z P P Z)

低头思故乡。(P P Z Z P)

Thinking in silent night

Bai Li

There is moon light in front of bed,

I doubt it is frost on the ground.

I raise my head and look at the moon,

I bow my head and miss my hometown.

Figure 1: This is an example of quatrain. It is a famous poem written by Tang dynasty poet Li, Bai. P represents the level-tone, and Z represents the downward-tone.

presented in previous work produces poems by expanding users queries. And some of the other authors used genetic algorithm to complete poetry generation. They generated a few poems and then used stochastic search to find the meaningful ones. Some scholars used statistical machine translation to generate poetry. Neural networks also have been used to realized poetry generation.

In our model, there are two improvement compared with previous works. The first aspect is that we employ recurrent neural network to accomplish the poetry generation problem. The recurrent neural network can learn the structure of the poems by itself. The second aspect is that there are labels representing the start and end of lines of poems. Learning the last character in each line is helpful in solving the rhyme problem. From the poems generated by our model, we can get a landscape or a sensible story that express a certain feeling. People could also use these poems to show their

'START', '南', '海', '烟', '霞', '尽', '属', '君', 'END',
'START', 'UNK', '怅', '暮', '春', '好', '时', '节', 'END',
'START', '白', '沙', '红', '树', '草', '连', '云', 'END',
'START', 'UNK', '日', '扶', '桑', 'UNK', '绝', 'UNK', 'END',
'START', '花', '繁', '官', '阁', '静', '无', '尘', 'END',
'START', '海', '天', '对', '月', '闲', '吟', '际', 'END',
'START', '好', '报', '平', '安', '慰', '故', '人', 'END',
'START', '寒', '烟', '碧', '草', '暗', '离', '披', 'END',
'START', '隐', '隐', '高', '原', '见', '古', 'UNK', 'END',
'START', 'UNK', '说', '从', '人', '皆', 'UNK', '妇', 'END',
'START', '应', '夸', '死', '义', '是', '男', '儿', 'END',
'START', 'UNK', '未', '解', '王', 'UNK', 'UNK', 'END',
'START', '千', '载', '犹', '闻', '鬼', '子', '悲', 'END',
'START', '异', '域', '天', '荒', '开', '世', 'UNK', 'END',

Figure 2: This is an example of training data. The START means the beginning of poem sentence, and the END means the end of poem sentence. The UNK in the sentence means the low-frequent characters.

emotions.

2 Problem Definition and Data

The problem we want to solve is poetry automatic generation. We feed the model with training data corpus. Given a character, the well-trained model can generate a poem by itself. There are two evaluation metrics. One is randomly generated poems, i.e. every character in the poems is randomly picked from the unique words bag. We can see whether the generated poem is more readable than the randomly picked one. The second evaluation method is comparison with human written poems. Native speakers provide their judgements based on their understanding. If most of the native speakers cannot figure out which one is written by machine and which one is written by human, then the poetry generator is successful.

The dataset we used to train our model is Chinese poem corpus that contain ancient poems in different dynasty. They are 50MB in total. There are three features: title, author and body. The training data is very clean. We just need little process, i.e. keeping the body of the poem. Next, we also need to add two labels: START and END at the ends of each line of poem. We replace the uncommon words with label UNK. The training data is shown in Figure 2.

3 Related Work

Some researchers used templates to construct poems according to a set of constraints, e.g. rhyme, meter, stress, and word frequency. The Haiku

poem presented in Wu et al. 2009 and Tosa et al. 2008 produces poems by expanding users queries. And some of the other authors used genetic algorithm to complete poetry generation, for example Manurung, 2003; Manurung et al., 2012; Zhou et al., 2010. They generated a few poems and then used stochastic search to find the meaningful ones. There are also scholars that used statistical machine translation to generate poetry. Jiang and Zhou 2008 generate two line poems using a phrase-based approach. Neural networks also have been used to realized poetry generation. Zhuang et al. 2014 is an example of using recurrent neural network.

Our approach differs from them in two aspects. The first is the structure of our training data set. We use labels, such as START, UNK, and END, to distinguish the low-frequency words and ending characters. The second difference is the structure of neural network. Since different frame of neural network will have different result, the neural network we used is a good trail of poem generation.

4 Methodology

The first step of solving this problem is preprocessing of data. We just need the body of the poems rather than title, author, and volume. Thus we read the training data line by line and keep the last component, i.e. the body of the poem, for later use. Then we split the sentence character by character. In such process, we also remove the strange symbols, such as ‘,’, ‘.’, ‘(’, ‘)’. Since what we have about the data is mixed poems corpus, we need to distinguish seven-character and five-character poems in order to learn the rhyme. We collect different types of poems in different list and add ‘START’ and ‘END’ at the beginning and ending of poems sentences. In order to generate understandable poems, we also delete the characters whose appearance times are less than frequency limitation, which is a parameter assigned at the beginning. After cleaning the data, now we have the training data like Figure 2.

The second step is building the recurrent neural network using Pytorch. In our recurrent neural network model, there are three layers: input layer, hidden layer, and output layer. For simplicity, we just use linear function as activation function. During the forward process, we use the randomly generated hidden layer to accomplish the process. In the calculating of hidden layer, the ac-

tivation function is tanh. We also randomly zeros some elements by applying dropout function to the output layer. After building the framework of the recurrent neural network by Pytorch, we can train the model in following.

The third step is training the model. First, word-embedding is used to project the character into higher dimensional vector space. Then we use the corresponding parameters to initialize the neural network. During the backward propagation, the loss is calculated by cross entropy and the optimizer is chosen as Adam. Next we convert the input character into tensor by word-embedding. The next character is the target tensor. After calculating the loss, we use the backward to optimize the values in neural network. After many times of iteration, we get well-trained model.

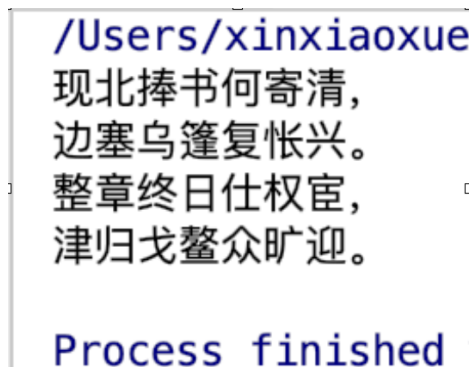
The final step is getting the poem. After fixing the parameters in the last training process, we submit a randomly chosen character. Then we get the next word. And we use the previous output to get the next result. Thus, we get a full poem. The code can be found at <https://github.com/xiaoxuexin/Natural-Language-Processing.git>

5 Evaluation and Results

The seven-character we get from the model is shown in Figure 3. This poem expresses a common topic in Chinese history. The author, i.e. computer, express her situation and feeling. She is in the north border of the country to fight the invaders. Standing in military tent with book in her hand, she is wondering when her country could thrive again. All the officers in the government suck up to the powerful leader. She hopes that one day she could fight the enemy and defeat the corruption government official one day. At that day, the citizen would live a happy life and welcome her back.

The five-character poem we get from the model is shown in Figure 4. This five-word poem depicts a magnificent royal hunting scene. The wild animals are scared of hunters. The royalty catch many preys.

In order to evaluate the performance of our model, we employ two evaluation methods. The first baseline is random performance. We just randomly generate the poem word by word. The randomly generated poem is shown in Figure 5. From the poem, we can see that there is no re-



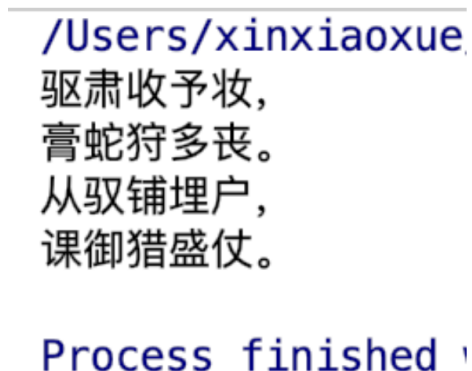
```

/Users/xinxiaoxue
现北捧书何寄清,
边塞乌篷复怅兴。
整章终日仕权宦,
津归戈鳌众旷迎。

Process finished

```

Figure 3: The seven-character poem generated from our model.



```

/Users/xinxiaoxue
驱肃收予收,
膏蛇狩多丧。
从驭铺埋户,
课御猎盛仗。

Process finished

```

Figure 4: The five-character poem generated from our model.

lation between words. And it not seems like a poem. Therefore our model outperform the randomly generated ones. The second baseline is human judgement. We mix the auto-generated poem with other four human written poems and ask several native speakers to select the poem generated by machine. We interview twenty people, and only three of them select the right answer. The probability of guessing the right answer is 0.2, but the result of survey is 0.15. Thus it is reasonable to claim that it is difficult to figure out the machine generated poem. Our model also outperform the second baseline.

6 Discussion

From the poem we generated, the seven-character poem expresses the author's certain feeling and the five-character poem depicts the magnificent spectacle of royal hunting. Both of them have rhyme at the end of sentences that satisfy the rule of poem writing. Thus, from the poems themselves, there is no critical point to blame. From the randomly generated poem, we can see that there is no con-

```
/Users/xinxiaoxue
/Users/xinxiaoxi
月秋短集面，
缠不度业量。
行痕远古悲，
到艾近消乡。

Process finished
```

Figure 5: The five-character poem is randomly generated, which is a baseline of our model.

nection between joint words. And it doesn't make sense. Thus we conclude that our auto-generated model outperform the random performance baseline. From the result of second base line, the survey implies that it is hard to distinguish the machine generated poem from human written ones. The probability of guess correctly is 0.2. However, in our survey, the probability is 0.15, i.e. three out of twenty. It is safe to say that our model generate readable poem compared with human written ones.

In the training process, there is one thing that needs special attention. That is parameter assignment. Different parameters' value may lead to totally different results. For example, if we changed the step size, or learning rate, it may converge to other optimal points. In order to converge fast, we choose a relatively large step size, i.e. 0.1. We can also change the embedding size. When the embedding size is too small, every character may cluster together. Thus the effect of learning cannot be good. If the embedding space is too large, then it will waste our time and computational resource. In our model, we set the embedding size as 100, which is large enough without wasting source. The number of iteration is also a critical point. When the number of iteration is not enough, i.e. only a small part of the training data has been learned by the model, the accuracy is low. If we run the training process for too many times, it is a waste of time and computing source. We would better stop once the algorithm is convergent. Or once the accuracy get the 0.8 we would stop our learning process. Thus we need to adjust the parameters to a suitable condition to get an efficient

and sensible result.

7 Conclusion

In this project, we build a Chinese poem generator model with recurrent neural network. And we generate both five-character poem and seven-character poem with rhyme at the end of sentence. The poetry we generated can express certain feeling or depict landscape and scene. It is hard for native speakers to distinguish which one is generated by machine. In the further research, we may generate poetry with certain topic. By training the model with certain topic, it is not hard to achieve that goal. And the poem with certain topic would more readable than just training with all corpus. The code can be found at <https://github.com/xiaoxuexin/Natural-Language-Processing.git>

8 Other Things We Tried

We tried to use other neural network frame, for example five and eight layers with ten and sixteen nodes in each layer. We also tried to use other activation function rather than linear function. Changing the shape of the recurrent neural network may help us figure out which structure have the best learning effect and capture the poem structure fast. But there is no big difference between the poems' quality.

9 What You Would Have Done Differently or Next

If we could start the project over, we would try to use Long-short Term Memory neural network to see whether there is an improvement if the model remember more from previous information. The LSTM should be a good choice compared with RNN. In the current model, we use the Pytorch to build the learning model. Since I am not familiar with Pytorch, I spent a lot of time in matching the model. We need to figure out what is the input size and what is the output size. And after learning the poems for many times, the code keeps running out the same word. It may because the model has converged to a certain point. Or it may because the iteration time is too small to learn the pattern of poetry. Maybe using the LSTM will avoid such kind of problem.

10 Work Plan

In the proposal, my plan is first solving the data source problem. In the following month, the framework of the neural network would be built. And a raw output could be seen. In April, the final report should be finished. The project is finished following the plan approximately. I collected data in March, at the beginning of the project. Then most of the time was used in building the recurrent neural network. And after training the model, we get the auto-generated poem and finish the report in April.

Acknowledgments

I would like to acknowledge Professor Jurgens for valuable suggestion and Heeryung for helpful discussion. Thanks to their help, can I accomplish the project ahead of time.

References

- Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga, and Mans Hulden. 2013. POS-Tag Based Poetry Generation with WordNet. *In Proceedings of the 14th European Workshop on Natural Language Generation*, pages 162-166, Sofia, Bulgaria.
- Chris Callison Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. *In Proceedings of the 7th Workshop on Statistical Machine Translation*, pages 10-51, Montreal, Canada.
- Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 524-533, Cambridge, MA.
- Jing He, Ming Zhou, and Long Jiang. 2012. Generating Chinese Classical Poems with Statistical Machine Translation Models. *In Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1650-1656, Toronto, Canada.
- Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. *In Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187-197, Edinburgh, Scotland, United Kingdom, July.
- Long Jiang and Ming Zhou. 2008. Generating Chinese Couplets using a Statistical MT Approach. *In Proceedings of the 22nd International Conference on Computational Linguistics*, pages 377-384, Manchester, UK, August.
- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700-1709, Seattle, Washington.
- Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. *In Proceedings of 2012 IEEE Workshop on Spoken Language Technology*, pages 234-239, Miami, Florida.
- Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. *In Proceedings of INTERSPEECH*, pages 1045-1048, Makuhari, Japan.
- Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011a. Strategies for Training Large Scale Neural Network Language Models. *In Proceedings of ASRU 2011*, pages 196-201, Hilton Waikoloa Village, Big Island, Hawaii, US.
- Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. *In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5528-5531, Prague, Czech Republic.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *In Advances in Neural Information Processing Systems*, pages 3111-3119, Lake Tahoe, Nevada, United States.
- Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388-1439.
- Xingxing Zhang and Mirella Lapata. 2014. Chinese Poetry Generation with Recurrent Neural Networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670-680, Doha, Qatar.