From Shakespeare to Li-Bai: Adapting a Sonnet Model to Chinese Poetry

Zhuohan Xie **Jey Han Lau** **Trevor Cohn**
The University of Melbourne
zhuohanx@student.unimelb.edu.au jeyhan.lau@gmail.com t.cohn@unimelb.edu.au

Abstract

In this paper, we adapt Deep-speare, a joint neural network model for English sonnets, to Chinese poetry. We illustrate the characteristics of Chinese quatrain and explain our architecture as well as training and generation procedure, which differs from Shakespeare sonnets in several aspects. We analyse the generated poetry and find that the adapted model works well for Chinese poetry, as it can: (1) generate coherent 4-line quatrains of different topics; and (2) capture rhyme automatically to a certain extent.

1 Introduction

Classical poetry is an important Chinese cultural heritage and holds great value. The most popular poetry type is quatrain. Table 1 shows a seven-character quatrain written by Li Bai, one of the most famous poets in Chinese history, which we will use to explain characteristics of Chinese quatrain.

First, a Chinese quatrain must contain four sentences where each sentence has the same number (five or seven) of characters. Usually, these sentences follow the "introduction, follow-up, transition and conclusion" mode, which means the first sentence starts a topic, the second continues it, then the third sentence extends it to a new realm and the last provides a summarisation (Wang, 2002). The rhythmic scheme in Chinese quatrain is simple: the last characters of the second and fourth sentences should rhyme. Optionally, the last character of the first sentence can also rhyme with them, as the quatrain shows in Table 1. As for tone, each character contains one single syllable and there are two tones for middle Chinese (Pulleyblank, 2011): "Ping" (level tone) or "Ze" (downward tone). Generally, tones at the same positions of two adjacent sentences are opposite and four common tonal patterns are illustrated in Wang (2002). Character tone is indicated in Table 1 where 'P' indicates "Ping", 'Z' refers to "Ze" and '*' means this character can be either tone.

望庐山瀑布 【唐】李白	
日照香炉生紫烟， 遥看瀑布挂前川。 飞流直下三千尺， 疑是银河落九天。	*Z P P Z Z P *P*Z Z P P *P*Z P P Z *Z P P Z Z P
Waterfall on Mount Lu Li Bai The Sunlit Censer peak exhales a wreath of cloud. Like an unpended stream the waterfall sounds loud. Its torrent dashes down three thousand feet from high. As if the Silver River fell from azure sky.	

Table 1: A 7-character Tang Quatrain by Li Bai.

Following all these constraints, a well-written Chinese quatrain are full of rhythm while expressing abundant emotions.

1.1 Motivation

Inspired by Deep-speare (Lau et al., 2018), a joint neural network model that is trained on English sonnets to generate quatrains in iambic pentameter (Halle and Keyser, 1971), we seek to adapt this model to poetry in other languages. The abundance of Chinese quatrain and its important cultural status makes it a natural choice.

We analyse the similarity between characteristics of English sonnets and Chinese quatrains and found it is possible to adapt the model to Chinese as the language model can be modified to handle Chinese characters.

Lau et al. (2018) found that a vanilla language

model can learn meter automatically at human level performance after it was trained on 2K sonnets. Given this, we have the expectation that the adapted language model should learn to generate quatrains with reasonable rhythm after we train it with sufficient data.

1.2 Contribution

The main contributions of the paper are:

1. We adapt Deep-speare to generate Chinese quatrains; source code of our adapted model is available at: ANONYMISED.

2. We find that our model is able to learn rhyme patterns automatically and can generate coherent quatrains of various topics; native Chinese speakers are unable to distinguish the best machine generated quatrains from human-written ones.

3. Our model is able to maintain originality in that BLEU scores and longest matching substring demonstrate the quatrains are not too similar to the training corpus.

2 Related Work

Poetry generation is a challenging task in Natural Language Processing. Many approaches were proposed to solve such problem, among which, the earliest methods were based on templates and rules. It requires users to provide keywords or titles and a rhyming template so the system can expand words and fill them into the user-selected template. This method is used in several successful poetry generation services, for instance, the haiku generation system of Wu et al. (2009) extracts rules from corpus and expand keywords from users to generate haiku sentences based on rules.

Another influential approach for Chinese quatrain generation is statistical machine translation (SMT). It is first utilised to generate Chinese couplets, which can be regarded as a special quatrain comprised of two sentences (Jiang and Zhou, 2008). They take the first sentence as input from users and generate an N-best list of second sentence candidates via a phrase-based SMT decoder. This method is extended by He et al. (2012) to generate Chinese quatrains where users are asked to provide the first sentence as input and then the system "translates" three following sentences based on the previous sentences.

Recently, neural networks are the predominant technique in the literature. Zhang and Lapata (2014) propose using recurrent neural networks (RNN) to generate the poem sentence by sentence. However, the model they provide is rather complex as it has one CNN and two RNNs, and it still observes theme drift when generating long sequences. To solve these problems, Wang et al. (2016) propose a simpler neural model which treats the whole poem as a character sequence. This approach can be easily extended to generate other genres such as Song Iambics or Haiku and they utilise the attention mechanism (Bahdanau et al., 2014) to avoid theme drift.

The closest work to our study is Deep-speare, which is a joint neural model designed for English sonnets. Deep-speare has 3 components: a language model, a pentameter model and a rhyme model. The language model is an LSTM encoder-decoder with attention that generates word by word and the pentameter model is trained to learn the iambic pentameter, an alternating stress pattern that sonnets exhibit. The rhyme model is optimised to separate rhyming word pairs automatically from non-rhyming ones in quatrains, which can help to enforce rhyme when generating the quatrains. All models are trained together (as a multi-task model), and the authors found that the model can generate sonnet quatrains with good stress and rhyme patterns.

3 Dataset

We source our poem dataset from Zhang and Lapata (2014), and use a modern Chinese pinyin dictionary[1].

3.1 Overview

We choose to use Tang quatrains as the Tang dynasty is one of the most famous dynasties for Chinese quatrains. We filter out non-quatrains from the Tang poetry collection and keep $2,349$ five-character and $7,219$ seven-character quatrains for our experiments. We sample 80% of the quatrains for training, 10% for development and 10% for test. Dataset statistics for each partition is presented in Table 2. There are $4,484$ unique characters in the full quatrain dataset ($9,568$ quatrains in total) and our training partition has a coverage of $4,283$ characters.

[1] https://github.com/DevinZ1993/Chinese-Poetry-Generation

Partition	#Qs	#Chs	#UniChs
Train (5-ch)	1879	38k	2785
Train (7-ch)	5775	162k	4091
Dev (5-ch)	235	5k	1279
Dev (7-ch)	722	20k	2385
Test (5-ch)	235	5k	1277
Test (7-ch)	722	20k	2325
Total	9568	249k	4484

Table 2: Dataset statistics. "Qs" denotes quatrains, "Chs" characters and "UniChs" unique characters.

To pre-train the word embeddings, we use approximately 300k quatrains from the full collection (quatrains in the development and test partition are excluded). The word embedding is trained per character, where each line in the quatrain is treated as a sentence and each character is treated as a word. We also tried *jieba* (Sun, 2012), a Chinese segmentation tool to tokenise phrases according to *ShiXueHanYing* (Liu, 1735), a poetic phrase taxonomy. We found that in preliminary experiments that the approach did not work well, and we hypothesise that this may be because classical poets compose their poems in a succinct manner by using minimum characters, and so each character is meaningful and can be treated as a word.

3.2 Rhyme Analysis

In total, $4,484$ unique characters exist in the whole quatrain corpus, and only some of them are rhyming characters that appear at the end of sentences. We analyse the number of rhyme combinations (i.e. rhyme tuples and triples) for each partition in Table 3. $2,450$ unique rhyme tuples and $2,882$ unique rhyme triples exist for seven-character quatrains and $1,034$ unique rhyme tuples and 244 unique rhyme triples appear in the five-character quatrains. We take into account of the order of rhyme combinations, for instance, ('尘', '春', '人') and ('春', '尘', '人') are considered different.

One interesting observation is that although the proportion of rhyme tuples in both five-character and seven-character quatrains is about the same (80%), the proportion of triples in seven-character quatrains is nearly 60%, which is much higher than that of five-character quatrains (13%).

We show the 5 most common rhyme tuples and triples in Table 4. We find that seven-character quatrains largely limit their rhyming patterns to the same words such as '尘' (chen), '春' (chun), '人' (ren), '新' (xin), '身' (shen).[2] By contrast, five-character quatrains have more variations for rhyming patterns, e.g. '道' (dao), '老' (lao), '草' (cao) and '卮' (zi), '辞' (ci), '离' (li).

4 Architecture

The original Deep-speare contains language model, rhyme model and pentameter model (Lau et al., 2018), here we only extend vanilla language model to work on Chinese quatrains where we treat each Chinese character as an individual word.

4.1 Language Model

The language model is a variant of an LSTM encoder-decoder where the encoder encodes previous sentences and the decoder predicts the next character in the current sentence, while attending to the encodings of previous sentences. Figure 1 shows the process of predicting the second sentence from the example in Table 1 where the encoder encodes the first sentence (bottom characters of Figure 1) while the decoder predicts the second sentence per character and each character (top-right of Figure 1) is generated based on its previous one (top-left of Figure 1) beginning from <s>, the sentence starting symbol. The encoder encodes first two sentences when decoder predicts the third sentence.

The encoder first uses a shared character embedding matrix W_{char} to embed the first sentence S_i to character vectors X_i. Then X_i is fed into a single layer bidirectional LSTM (Gers et al., 1999) to yield a sequence of hidden states $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. Then selective encoding (Zhou et al., 2017) is applied to each h_i to filter out less useful information. We use the last forward hidden state concatenated with the first backward hidden state to represent the whole sentence $\overline{h} = [\overrightarrow{h_{-1}}; \overleftarrow{h_1}]$ (h_{-1} here means the last hidden state). The selective encoding filters out information from each hidden state h_i using $\overline{h}$ by the following equation:

$$h'_i = h_i \odot \sigma(W_a h_i + U_a \overline{h} + b_a)$$

where $\odot$ indicates element-wise multiplication and W, U and b refer to model parameters.

[2]The pinyin of the characters largely follow '人辰韵' pattern ('en', 'in', 'ün').

Partition	#Quatrains	#Tuples	%Tuples	#Triples	%Triples
Train (5-ch)	1879	1491	79.35	258	13.73
Train (7-ch)	5775	4769	82.58	3440	59.57
Dev (5-ch)	235	186	79.15	24	10.21
Dev (7-ch)	722	597	82.67	427	59.14
Test (5-ch)	235	184	78.30	30	12.77
Test (7-ch)	722	604	83.66	438	60.66
Total	9568	7831	81.85	4617	48.25

Table 3: Rhyme proportion in human-written quatrains.

R	Tuple	F	Triple	F
1	(‘春’, ‘人’)	31	(‘人’, ‘氛’, ‘云’)	2
2	(‘深’, ‘心’)	18	(‘道’, ‘老’, ‘草’)	2
3	(‘开’, ‘来’)	15	(‘草’, ‘道’, ‘老’)	2
4	(‘人’, ‘春’)	12	(‘半’, ‘远’, ‘转’)	2
5	(‘来’, ‘开’)	9	(‘卮’, ‘辞’, ‘离’)	2

(a) 5-char

R	Tuple	F	Triple	F
1	(‘春’, ‘人’)	95	(‘尘’, ‘春’, ‘人’)	22
2	(‘身’, ‘人’)	57	(‘春’, ‘尘’, ‘人’)	16
3	(‘尘’, ‘人’)	49	(‘春’, ‘新’, ‘人’)	13
4	(‘开’, ‘来’)	42	(‘春’, ‘身’, ‘人’)	11
5	(‘家’, ‘花’)	37	(‘新’, ‘春’, ‘人’)	11

(b) 7-char

Table 4: Top-5 rhyme combinations in human-written quatrains. “R” denotes rank and “F” denotes frequency.

The decoder embeds the current sentence S_t with the same shared character embedding matrix W_{char} to generate character vectors X_t and feed them into a unidirectional LSTM to produce the decoding state s_t:

$$s_t = \text{LSTM}(X_i, s_{t-1})$$

To attend to the encodings of previous sentences (h'_i), we compute a weighted sum (h^*_t) as follows:

$$e^t_i = v^T_b \tanh(W_b h'_i + U_b s_t + b_b)$$
$$a^t = \text{softmax}(e^t)$$
$$h^*_t = \sum_i a^t_i h'_i$$

Finally, s_t and h^*_t are combined by a gating unit (Chung et al., 2014) to generate state s'_t and fed into the output layer where softmax activation is used to yield a probability distribution over the character vocabulary.

$$s'_t = \text{GRU}(s_t, h^*_t)$$
$$p = \text{softmax}(W_{out} s'_t + b_{out})$$

We use standard cross-entropy loss to optimise the model and dropout as regularisation.

To reduce the number of parameters, the output matrix W_{out} is obtained from W_{char} via a projection matrix W_{proj}:

$$W_{out} = \tanh(W_{char} W_{proj})$$

4.2 Training Procedure

We combine five-character and seven-character quatrains together for training. In preliminary experiments we found that the model works better if we train quatrains from right to left, i.e. instead of generating starting with <s> to predict the first character and so on, we generate from <$/s$>, the ending symbol, to predict the last character and generate backwards terminating with <s>. The context from the encoder is modified accordingly, and it does not encode the first sentence as shown in Figure 1. It instead encodes the last two sentences in reverse order for the second sentence. We think this is better way because: (1) the rhyme characters appear at the end of sentences, and it would be better if the model produces them first; and (2) for noun compounds (e.g. adjective+noun),

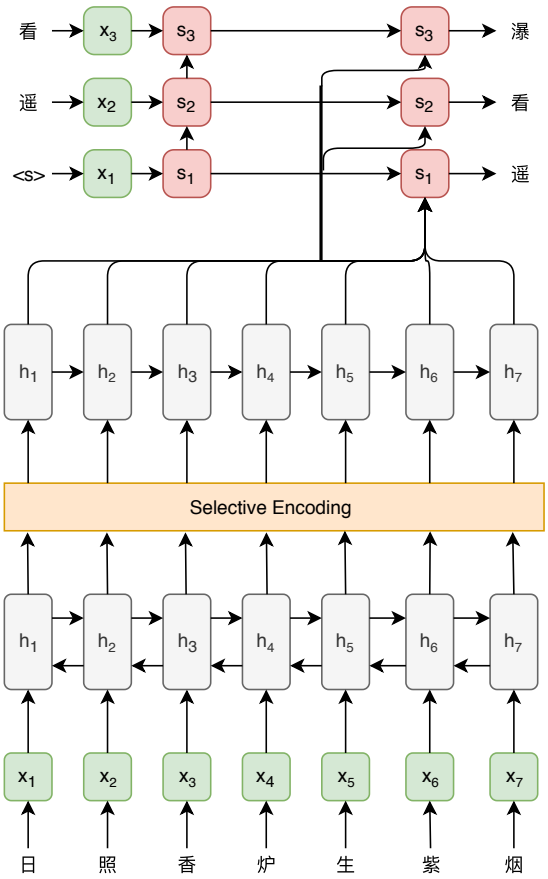


Figure 1: Architecture of the language Model

generating the last character first is a more sensible approach, as it limits the choices of adjective. For instance, 青山 (green mountain), 青龙 (green dragon), 青草 (green grass) are all frequent phrases in Chinese quatrains. The adjective would be 青 (green) if 草 (grass) is generated first. On the other hand, if 青 (green) is generated first, then there are many words that can appear after it, complicating the prediction.

We tune hyper-parameters of the model based on its performance on the development partition. We train the model for 30 epochs and use the Adagrad optimiser (Duchi et al., 2011). We pre-train CBOW (Mikolov et al., 2013a,b) embeddings on the full poetry collection, as described Section 3.1. The embeddings are updated during training. We set the learning rate to 0.2 and dropout probability to 0.4.

4.3 Generation Procedure

We use the same model to generate five-character and seven-character quatrains, by specifying the total length of quatrains (20 for five-character and 28 for seven-character). We generate one sentence at a time (4 in total), and we sample characters one at a time and stop when we have a sufficient number of characters (5 or 7) for a sentence. We did not apply beam search to generate the quatrain since we found the topics of machine generated quatrains can be more diverse if we use sampling.

Since we train our data in reverse order, we generate quatrains in reverse order as well. We feed the hidden state of the previous character to the decoder of language model to compute the character distribution for the current character and sample from it using a temperature τ (Hinton et al., 2015) between 0.1 and 0.2.

$$\widetilde{p}_i = f_\tau(p)_i = \frac{p_i^{\frac{1}{\tau}}}{\sum_j p_j^{\frac{1}{\tau}}}$$

We resample if the sampled character is: (1) a character that has appeared more than once; (2) one of the preceding three characters; (3) the last character of any sentence; or (4) UNK or PAD special tokens. UNK tokens represent characters that are not in our vocabulary[3]; we also use UNK tokens to represent the previous context for the first sentence. PAD tokens are used to pad five-character quatrains so that they can be trained with seven-character quatrains together.

We find that setting temperature in the above range limits the possibility of generating more diverse poetry because the generated quatrains would usually end in frequent characters such as '人' (person), '来' (come) or '中' (in). As the rest of the quatrain will be generated based on these characters, it limits the topic and content for the poem, and there is little diversity in the generated quatrains. In light of this, we set temperature to 0.9 when generating the last character of the quatrain. We generate 100 quatrains under two different temperatures (0.2 and 0.9) to compare the differences between them; results are presented in Figure 2. The model produces 67 unique sentence-ending characters under temperature 0.9, but only 5 unique characters when we set temperature to 0.2. Also, the cumulative proportion of the top-9 characters is only 35% when temperature is 0.9. In contrast, the most frequent character '人' (person) alone occupies over 90% cumulative proportion at 0.2 temperature.

[3]Out-of-Vocabulary (OOV) Rate is 0.13% in our model.

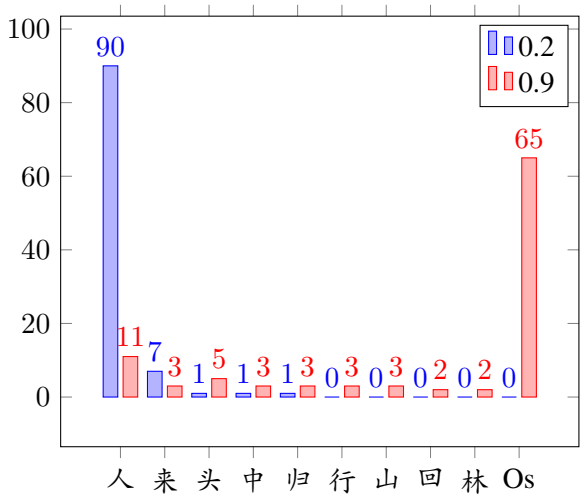


Figure 2: Relative frequency of sentence-ending characters under different temperatures. "Os" denotes Others.

桃花荷叶红， 春水不成空。 花开绿窗下， 何人入梦中。	孤舟行路不胜愁， 万山云雨峡中幽。 知君莫向荆州客， 不见青山湘水流。

Table 5: Quatrains generated by our model.

5 Evaluation

We select two representative quatrains (one five-word and one seven-word) generated by our system and present them in Table 5. We find that our system is able to: (1) capture rhyme patterns automatically; (2) generate quatrains of various topics; and (3) compose coherent poetry.

For the presented quatrains, the last characters of the first, second and last sentences rhyme. It demonstrates that the language model is able to learn rhyming pattern automatically, without requiring an additional rhyme model to enforce rhyming.

Various topics exist in Tang quatrains, such as "love" and "wanderer staying long in a foreign land" and our system is able to generate quatrains of various topics. The topic of the left quatrain is boudoir resentment, which is one of the major themes of love poems. Phrases that symbolise female appear in such quatrain, such as 桃花 (peach blossom) and 荷叶 (lotus leaf). The overall narrative depicts a love story. The topic of the right quatrain is about travelling, and it describes the scenery the poet observes when travelling alone on a boat, using phrases such as 万山 (thousands of mountains) and 云雨 (clouds and rain).

Our evaluation includes rhyme evaluation, similarity evaluation and human evaluation. We see various topics appear in the generated quatrains and they are coherent, therefore, we would like to perform diversity and coherence evaluation in the future work.

5.1 Automatic Evaluation

We generate 500 five-character quatrains and 500 seven-character using the hyper-parameter configuration discussed in Section 4.3, and analyse the number of rhyming quatrains and the variation of rhyming patterns. We also use BLEU and longest matching substring to check for similarity between our machine generated quatrains with the training corpus (so as to confirm that the model is not simply memorising and reciting quatrains from the training data).

Rhyme Evaluation We present the proportion of rhyming quatrains in Table 6. Our system works better with respect to rhyme when generating short sentences: 25% of the five-character quatrains rhyme, but only 8.4% of the seven-character quatrains rhyme. We hypothesise that this may be because shorter sentences are easier for the system to learn the rhyming patterns.

We also examine the number of unique rhyme combinations generated by our system for the 1,000 quatrains. 52 unique rhyme tuples and 22 triples appear in the five-character quatrains and 26 unique rhyme tuples and 8 triples appear in the seven-character quatrains. This again implies that it is more difficult for the system to learn the rhyming patterns for longer sentences. We show the top-5 rhyme combinations for the generated quatrains in Table 6. Comparing these to Table 4, we find that frequent combinations are learnt by the model.

The results show our system is able to capture rhyme pattern to some extent, but the generated quatrains do not rhyme as much as human written ones and rhyming combination is limited as well. This might suggest that we need a rhyming model to generate quatrains with better rhyme schemes.

Similarity Check We treat each generated quatrain as the hypothesis, and compute BLEU (Chen and Cherry, 2014) score against the training corpus (reference). The average BLEU score is 0.46 and 0.44 for five-character and seven-character quatrains respectively. Seven-character quatrains gen-

Partition	#Quatrains	#Tuples	%Tuples	#Triples	%Triples
5-ch	500	125	25	44	8.8
7-ch	500	42	8.4	11	2.2
Total	1000	167	16.7	55	5.5

Table 6: Rhyme proportion in machine generated quatrains.

R	Tuple	F	Triple	F
1	(‘空’, ‘中’)	12	(‘尘’, ‘春’, ‘人’)	6
2	(‘开’, ‘来’)	12	(‘身’, ‘春’, ‘人’)	6
3	(‘春’, ‘人’)	12	(‘开’, ‘苔’, ‘来’)	5
4	(‘愁’, ‘头’)	10	(‘台’, ‘开’, ‘来’)	4
5	(‘家’, ‘花’)	6	(‘愁’, ‘秋’, ‘头’)	2

(a) 5-char

R	Tuple	F	Triple	F
1	(‘归’, ‘飞’)	6	(‘身’, ‘春’, ‘人’)	3
2	(‘春’, ‘人’)	5	(‘尘’, ‘春’, ‘人’)	2
3	(‘里’, ‘西’)	3	(‘尘’, ‘尽’, ‘春’)	1
4	(‘尽’, ‘春’)	2	(‘台’, ‘开’, ‘来’)	1
5	(‘开’, ‘来’)	2	(‘头’, ‘首’, ‘州’)	1

(b) 7-char

Table 7: Top-5 rhyme combinations in machine generated quatrains. “R” denotes rank and “F” denotes frequency.

erally have lower BLEU scores, and they also have a bigger spread.

We also check the longest substring that can be matched in the training corpus for each of the generated quatrain. The average length of the longest substring is 3.70 for five-character quatrains (20 characters in total) and 4.05 for seven-character quatrains (28 characters in total). The length of the longest matching substring is 6 and 7 for five-character and seven-character quatrains respectively; for most quatrains this length ranges from 3 to 4, which is less than a quarter of the length of quatrain.

Taking the BLEU scores and longest matching substring results together, we see that our system can generate original poetry, i.e. it is not just memorising the quatrains from the training data and reciting them during generation.

5.2 Human Evaluation

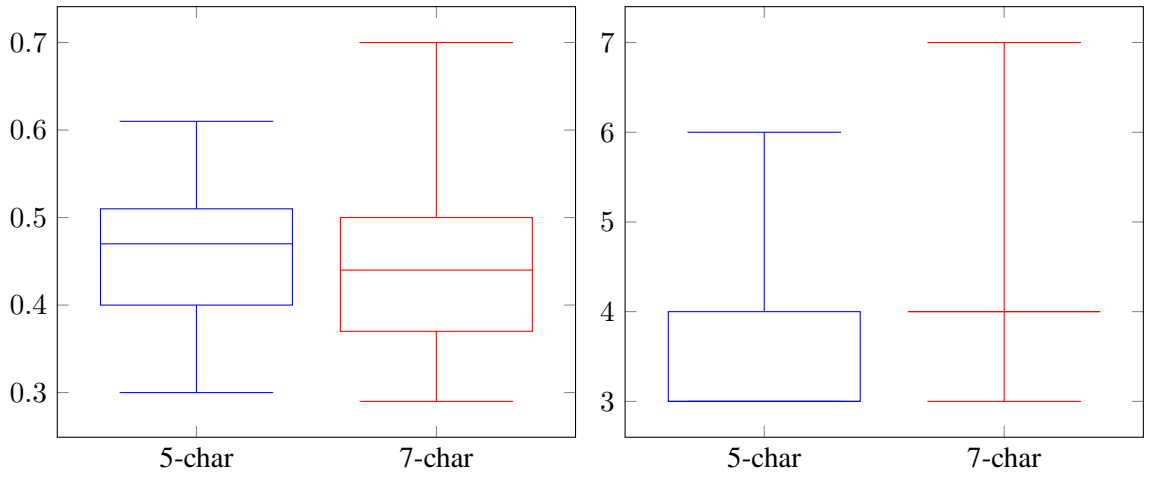
For human evaluation, we pick 5 five-character and 5 seven-character machine generated quatrains with highest quality from our perspective. 4 out of the 5 quatrains contain rhyme tuples so as to mimic real rhyming patterns (as we saw in Section 3.2). We also sample 5 five-character and 5 seven-character human-written quatrains from the training data and mix the 20 quatrains in a questionnaire and ask native Chinese speakers to judge for each quatrain whether it is computer-generated or human-written.

We receive 171 valid responses from volunteers, where they are all native mandarin speakers and most of them hold a bachelor’s degree. In general, the users found it very difficult to distinguish the machine-generated quatrains from the human-written ones. On average the accuracy is 45.15%, which is slightly worse than pure guessing at identifying the poems. The quatrain which is voted most (by 70% users) as being “human-written” is generated by our model (Table 5; right quatrain).

However, we acknowledge that our volunteers might not be the best at identifying quatrains, therefore, we would like to seek evaluations from experts in the future work.

6 Conclusion

In this paper, we extend Deep-speare, a joint neural network model for English sonnets, to generate Chinese quatrains. Chinese quatrain holds an important cultural status in China, and there is a large collection of these quatrains, making it a natural choice for adaptation. We analyse Chinese quatrains and detail how we adapt Deep-speare in terms of training and generation. We find the adapted model works well, as it can: (1) generate quatrains of different topics; (2) capture the rhyming scheme automatically; (3) compose coherent poetry. We also check the similarity of the generated quatrains with the training data, and find



(a) BLEU

(b) Longest matching substring

Figure 3: Similarity Check.

that the generated poetry is original. Human evaluation demonstrates that our best machine generated quatrains are almost indistinguishable from human-written ones for native Chinese speakers.

For future work, we believe the following are promising directions:

1. There is potential to utilise data[4] that has tone labels to better understand the tonal patterns in Chinese quatrains.

2. Adapt the rhyme model of Deep-speare to produce rhyme more consistently in Chinese quatrains.

3. Train a reverse summarisation model on Chinese quatrains to generate a title given a quatrain.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[4]https://github.com/jackeyGao/chinese-poetry

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Morris Halle and Samuel Jay Keyser. 1971. Illustration and defense of a theory of the iambic pentameter. *College English*, 33(2):154–176.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Long Jiang and Ming Zhou. 2008. 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.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1948–1958.

Wenwei Liu. 1735. Shixuehanying (诗学含英).

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Edwin G Pulleyblank. 2011. *Middle Chinese: A study in historical phonology*. UBC Press.

J Sun. 2012. ' jieba ' chinese word segmentation tool.

Li Wang. 2002. A summary of rhyming constraints of chinese poems (诗词格律概要). In *Beijing Press*.

Qixin Wang, Tianyi Luo, and Dong Wang. 2016. Can machine generate traditional chinese poetry? a feigenbaum test. In *International Conference on Brain Inspired Cognitive Systems*, pages 34–46. Springer.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *International Conference on Entertainment Computing*, pages 191–196. Springer.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv preprint arXiv:1704.07073*.