# Dueling Chinese Couplets using Sequence to Sequence Model

Zou Xiaoyan

Human Computer Interaction

School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—Chinese couplet is a special type of Chinese poems aiming to be both concise and meaningful. There are lots of constraints for couplets, such as couplet length, Chinese tone pattern, lexical category. Due to these strict rules, dueling Chinese couplets is considered as a difficult pastime. In this paper, we aim to encode a machine that duels well Chinese couplets using Sequence to Sequence (Seq2Seq) model. More precisely, our model used bidirectional Gated Recurrent Units (GRU) for both encoder and decoder along with the attention mechanism. Divers constraints are incorporated into the model during either the training process or the evaluation process. Our model finally achieved the BLEU score 14.15.**[1]

## I. Introduction

Chinese couplet consists of two lines of Chinese poetry where the classical Chinese is used. In spite of its shortness, a pair of couplet should be concise and meaningful. Moreover, couplet rules make dueling couplets harder and harder. Ancient Chinese had often dueling couplet game to show their rich knowledge and their proficiency in Chinese, whereas nowadays people feel much more pain to duel couplets. This project aims to encode a machine that outputs automatically the second line of the couplet when the first line is provided.

Sequence to sequence networks are chosen to do such a project. Consisting of an encoder and a decoder, sequence to sequence model is built using Recurrent Neural Network (RNN). It is about training models to convert sequences from one domain to sequences of another domain. After introducing gates into the network, the model is able to keep tracking the past inputs, so call long term dependencies. Except simply dialogue generation[1], the most popular application for sequence to sequence model is to perform machine translation. The results can be quite satisfactory.

However, unlike translation, dueling couplets is not as simple as a word mapping problem because of its various rules which are the following:

- Two lines of couplets must have the same length.
- Two lines of couplets must have coherent and related meanings.
- The lexical category of corresponding characters should be the same.
- The tone patterns (level tone and oblique tone) of corresponding characters should be opposite.
- The first line of couplet should end with an oblique tone, which implies that the second line should end with a level tone.

The main difficulty of this project is that we need to integrate these exigent rules into the model and make couplet answers human-like at the same time.

对联上联：水榭

机器下联：山亭

对联上联：美德如珠灿

机器下联：廉明似水清

对联上联：凤翔万水声威远

机器下联：凰舞千山气势雄

对联上联：春风吹染千花艳

机器下联：喜雨润泽万物新

对联上联：三农为首，神州壮丽金瓯固

机器下联：万众同心，大地欢腾国运昌

对联上联：四海寻根，溯血脉传承，皆缘后土

机器下联：千秋仰止，看江山如画，尽是春天

Fig. 1. Several couplet results of our model. The first line of each couplet is the input of the model. The second line is the couplet answer of our model.

## II. Related Work

As aforementioned, sequence to sequence is widely applied in machine language translation domain. Sutskever et al.[2] used

[1]Master semester project done in Human Computer Interaction Lab in EPFL, Switzerland. Codes and model archives can be found at: https://github.com/zxyzz/Project

two LSTMs to accomplish an English to French translation task. The first LSTM was used to extract information from the source sentence and to form a fixed length vector representing the sentence. The latter was decoded by the second LSTM in order to obtain the translation. They improved their models by reversing the word order of the source sentence. The word order of target sentence remained unchanged. Apparently this action introduced many short term dependencies since very first words are closer to their target words. Bahdanau et al.[3] have also proposed a neural machine translation model. Instead of encoding the source sentence into a fixed length vector, which might deteriorate the performance of the model when source sentence was long, only a subset of vectors that were relevant to targets would be chosen and used in the decoder. Such vectors were determined by the score of a soft alignment model who tells how much the input around position $i$ is relevant to output around position $j$. Besides, they have also used bidirectional RNNs so that the annotation of words contained not only the information of preceding words but also those of the following words.

Furthermore, sequence to sequence is used to generate Chinese classical poems: quatrains (four-line poems). Similar to couplets, poems demand various requirements such as semantic correspondence, tone patterns and rhyme constraints. Yi et al.[4] have proposed a RNN Encoder Decoder model with gated units, aiming to generate a quatrain with a set of keywords as model input indicating the content or the emotion that the poem should cover. Keyword were used to generate the first line of the poem. Then the obtained result was taken into account to generate following lines of the poem and so on and so forth. Their model jointly learned semantic meaning and relevance among lines along with attention mechanism to capture words associations proposed by Bahdanau et al.[3]. Tone and rhyme controllers were used to fulfill the poem rules. To further improve the model, another neural language model was added into their system to make poems more meaningful. Different from Yi et al.[4], Zhang et al.[5] considered the whole history produced so far by the model as the context information to generate the next line of the poem.

Except these works on Chinese poems, several works were done for dueling Chinese couplets. In addition to a basic attention based sequence to sequence model, Yan et al.[6] proposed an extra polishing schema based on Convolutional Neural Network (CNN) which can extract patterns for successive characters. The draft output was reused to polish every character of the final output in order to refine semantic coherence by one or more iterations. Zhang et al.[7] have added two innovations to the usual attention based sequence to sequence model. One was the additional local attention within a limited window computed in the same way as the global attention. The other innovation was the separate treatment of entities such as person names and addresses since usually the model could not deal well with special entities.

In this project, we likewise produced an attention based sequence to sequence model using bidirectional GRU for the encoder and the decoder to generate automatically Chinese couplets. Couplets constraints were added into training or evaluation processes depending on constraints.

## III. Model

### A. Sequence to sequence background

Sequence to sequence model consists of an encoder and a decoder as shown in Fig. 2. The encoder is represented by red cells and the decoder is represented by green cells. Cells are constructed using RNN.

Suppose we need to translate the source sentence *ABC* into *A'B'C'*, characters of the source sentence are entered one at a time into the RNN cell of the encoder. Each cell produces a hidden state served as the next input for the next cell. This action is represented by horizontal arrows. In that way, all hidden states are connected within the network. That gives to the model the ability of capturing the long term dependencies. At the end, the encoder will output a final encoder output and an final hidden state. The latter is a fixed length representation of the source sentence and is served as the initial hidden state of the decoder. The decoder is in charge of mapping the source sentence to the target sentence. *SOS* token indicating *Start Of Sequence* is inserted to the beginning of each decoder input. *EOS* token indicating *End Of Sequence* is appended to each decoder output. Each cell of the decoder produces a temporary output and a temporary hidden state. They are both fitted into the next cell. If the cell outputs an *EOS* token, then the whole process is terminated. Because the model judges that it has achieved the end of the sentence, the translation process is finished. The final target sentence is obtained by concatenating all temporary outputs.

The whole mechanism of mapping is relied on the conditional probability to figure out the best output matching the source sentence. Namely, we want to find the target sentence $y$, which maximizes the conditional probability of $y = (y_1, ..., y_{T_y})$ given the source sentence $x = (x_1, ..., x_{T_x})$:

$$\hat{y} = \arg\max_{y} Pr(y|x) \tag{1}$$

This probability can be computed according to the chain rule. Thus, we rewrite $Pr(y|x)$ as the following:

$$Pr(y_1, ..., y_{T_y}|x_1, ..., x_{T_x}) = \prod_{i=1}^{T_y} Pr(y_i|x_1, ..., x_{T_x}, y_1, ..., y_{i-1}) \tag{2}$$

where $T_x$ is the length of the source sentence and $T_y$ is the length of the target sentence. They are not necessarily equal in language translation. In the sequence to sequence network, the source sentence $x$ is not directly used to compute such probability but $v$, the representation of the whole sentence obtained in the encoder. Each conditional probability $Pr(y_i|x_1, ..., x_{T_x}, y_1, ..., y_{i-1})$ is computed by the following equation:

$$Pr(y_i|x_1, ..., x_{T_x}, y_1, ..., y_{i-1}) = g(v, s_{i-1}, y_{i-1}) \tag{3}$$

where $y_1, ..., y_{i-1}$ are previously predicted words. $s_{i-1}$ is the previous hidden state of the cell of the decoder, and $g$ is non-linear function that outputs the probability.

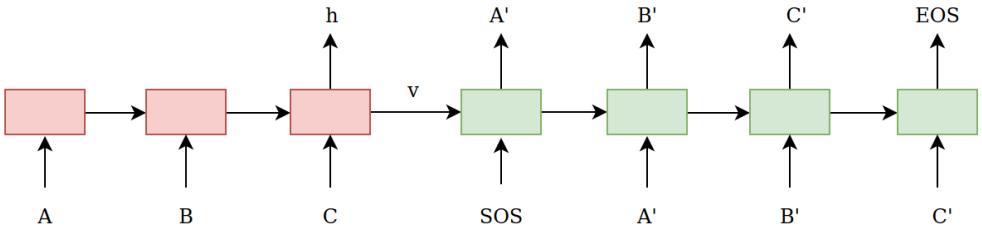


Fig. 2. This is a diagram of a basic sequence to sequence model. The left part with red cells represents the encoder and the right part with green cells represents the decoder. This diagram maps the source sentence *ABC* to *A'B'C'*. $h$ is the encoder output representing all information of source sentence *ABC*. $v$ is the final hidden state of the encoder. It is fitted into the decoder as one of its inputs.

RNN has various forms of networks. In this project, we used GRU, a simplified version of Long Short Term Memory (LSTM). Because GRU is faster than LSTM and it can have more or less the same performance as LSTM.

### B. Encoder

The input of encoder is the source sentence represented by a sequence of vectors $x = (x_1, ..., x_{T_x})$. Each $x_i$ will go through an embedding layer to form its embedded vector $x_i^e$, $i = 1, ..., T_x$.

$$x_i^e = E(x_i) \tag{4}$$

where $E$ is an embedding function which is nonlinear. Note that if $x_i$ corresponds to a *PAD* token, then it will be embedded into an all-zero vector. Along with hidden states, the embedded vectors are fitted into GRU afterwards, as described in Eq.5

$$h_i, s_i = GRU(x_i^e, s_{i-1}) \tag{5}$$

This produces the temporary hidden state $s_i$ and the temporary encoder output $h_i$ for the cell of $i$th iteration. Since a bidirectional GRU is used in the network, all $s_i$s and $h_i$s are of the form forward pass value concatenating the backward pass value. Namely,

$$s_i = \begin{bmatrix} \overrightarrow{s_i} \\ \overleftarrow{s_i} \end{bmatrix}$$

and

$$h_i = \begin{bmatrix} \overrightarrow{H_i} \\ \overleftarrow{H_i} \end{bmatrix}$$

where, left arrows indicate the forward pass and right arrows stand for the backward pass. The final encoder output $h$ is obtained by concatenating horizontally all $h_i$s. Final hidden state value is simply the last hidden state of both forward pass and backward pass. Note that all padding tokens are ignored by GRU.

$$h = \begin{bmatrix} \overrightarrow{H_1} \cdots \overrightarrow{H_{T_x}} \\ \overleftarrow{H_1} \cdots \overleftarrow{H_{T_x}} \end{bmatrix}$$

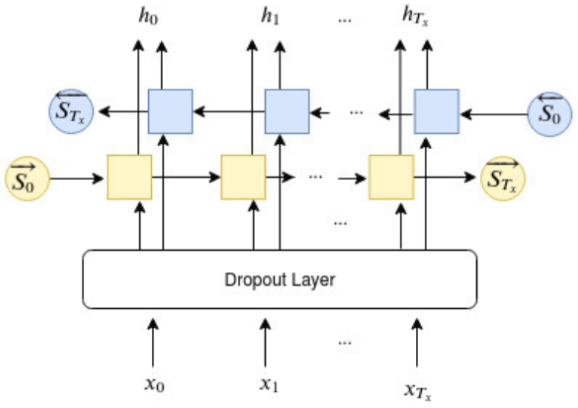


Fig. 3. The encoder of sequence to sequence model. Yellow cells are GRU for the forward pass while blue cells are GRU for the backward pass. Inputs are embedded into high dimensional vectors and fitted into GRU.

### C. Decoder

For the training process, the input of the decoder is the desired answer of the second line of the couplet, leading by a *SOS* token. Different from the encoder, the $i$th output $y_i$ of the decoder has to be one of the next inputs of itself in order to compute the next output $y_{i+1}$, as one can observe in the right part of Fig.2. That is how the decoder figures out the whole prediction, based on what has been produced so far according to Eq.3. Since the final output is obtained by concatenating temporary outputs together, we should model the decoder in the way that it produces temporary outputs as similar as desired answers. The model stops evaluating the next output when *EOS* token is generated.

During the training process, same as the encoder, the decoder begins with an embedding layer which projects input characters $y_i$ into a higher dimension space to form an embedded vector $y_i^e$. Followed by a dropout layer, through which each entry of embedded vector has some chance being set to zero in order to avoid data over-fitting. The output of the dropout layer is combined later with the attention-applied encoder output, called also the context vector $c_i$. Thanks to attention scores $\alpha_i$, each character $y_i$ has a different context vector with different focus on the source sentence.

The attention mechanism is further described in Section III-D. After some linear combinations and the activation function which introduces non linearity into the network, the result is fitted into GRU model, exactly as what we do in the case of the encoder described in Section III-B. The output $y_i$ of GRU is a vector of length of the vocabulary, consisting of a sequence of probabilities for each character in the vocabulary. The final predicted word $\hat{y}_i$ is the character that has the maximal probability due to Eq. 1. Till $T_y$ iterations, $y_1, ..., y_{T_y}$ can be outputted.

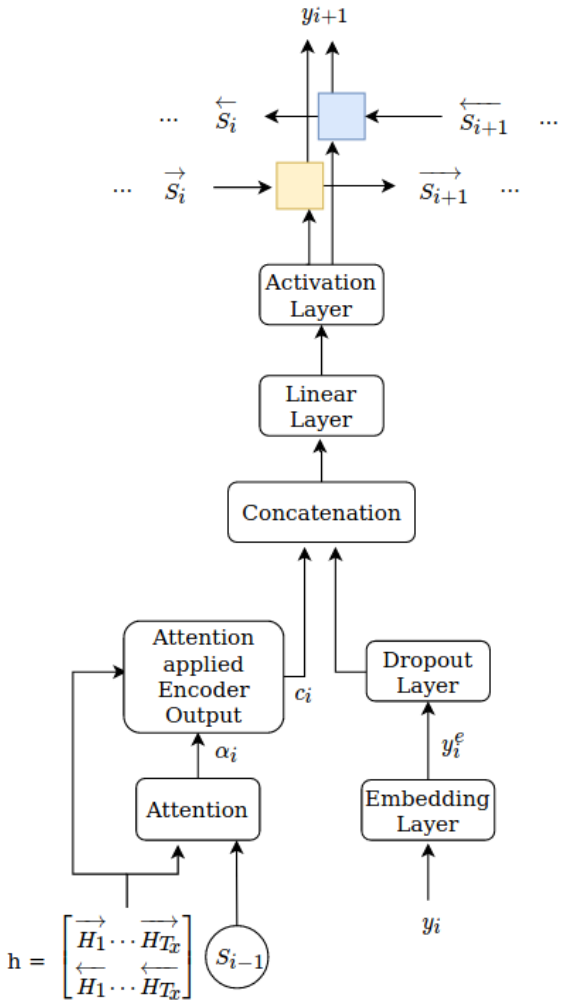


Fig. 4. The concrete diagram of the decoder for $i$th iteration. Input $y_i$ enters into the network at the right lower part of the figure. $y_i^e$ is the embedded vector of $y_i$. At the left lower part of the figure, the matrix $h$ consisting of $h_i$s is the output of the encoder, which represents the whole source sequence. $c_i$ is the context vector obtained by multiplying the encoder output and attention weights $\alpha_i$. Together with $s_i$ which represents the hidden state of the iteration $i$, the model can generate next output $y_{i+1}$.

### D. Attention

Initially proposed by Bahdanau et al.[3], the attention is an alignment model whose weight indicates how much the input around position $i$ is relevant to output around position $j$. Namely the attention aims to capture words associations. For each input character $y_i$, we will go through the whole representation of the source sentence $h = (h_1, ..., h_{T_x})$ and determine on which subset of $h_i$s the model should focus. This results in attention weights $\alpha_i$, a normalized weight vector having the length $T_x$. The greater the weight is, the more attention the model will pay to the corresponding source characters $x_i$s when decoding. There are many ways to compute the global attention[8]: *dot*, *general*, *concat* and so on.

In this project, *dot* and *general* alternatives are both prepared. But, we used the general approach of attention score because it provides more hyperparameters. The weight $\alpha_{ij}$ is computed as the following:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{6}$$

with

$$e_{ij} = s_{i-1} h_j W_e^T \tag{7}$$

where $W_e^T$ is weight matrix determined by a linear layer. After the computation, each $e_{ij}$ corresponding to *PAD* token is ignored in order to get rid of noise. Since padding tokens do not contain any helpful information for decoding. Then the remaining $e_{ij}$s are normalized by the softmax function in Eq. 6, resulted as weight coefficients. Finally, the context vector $c_i$ is given by

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{8}$$

As one can see, the value of $h_i$s which are less important are weakened by multiplying smaller weights compared to other $h_i$s associated with larger weights. This encourages the model to focus on the key parts, instead of treating equally each input character.

## IV. Dataset and Training Details

The couplet dataset we used can be found online.[2]. It contains four text files. *in.txt* contains the first line of each pair of couplets and *out.txt* contains the corresponding desired second line. *test_in.txt* contains couplet inputs for the evaluating process. *test_out.txt* contains desired couplet outputs for the evaluating process.

### A. Data preparation

Couplets were cleaned by removing useless symbols such as empty space. After the data cleaning, there remained 770 560 pairs of couplets prepared for the training process and 3 930 pairs prepared for the evaluating process, resulted in files of size 44.8 MB and 229 kB respectively. The vocabulary was created from in total 774 490 pairs of couplets, all characters were ordered by their frequency of occurrence. In addition, *PAD*, *SOS*, *EOS* and *UNK* (stands for UNKNOWN) were inserted at the beginning of the vocabulary.

[2]Dataset source: https://github.com/wb14123/couplet-dataset

### B. Data tokenization

Couplets needed be translated into integers so that the computation could be launched. Since the vocabulary contains all words in the couplet dataset, each character can be represented by its corresponding index in the vocabulary list. In such a way, all couplets were tokenized by mapping to their unique ID in the vocabulary.

### C. Training process

Tokenized couplets were firstly grouped into batches. Each batch contained 256 couplets. Therefore, we had 3 010 batches in total. Because of various lengths of couplets, couplets in a batch were padded with *PAD* token. The sequence length in a batch was determined by the maximal length of couplets in the batch. Next, we randomly chose one batch served as the validation set, namely 256 couplets. The 3 009 remaining batches were used for the training purpose.

The encoder and the decoder had hyperparameters as the following:

- Both of them had hidden size 256.
- The number of embedding features was both 256.
- The dropout layer of the decoder followed the Bernoulli distribution with probability 0.1. Namely, the probability of setting an entry of embedded vector $x_i^e$ to 0 was 0.1.
- The loss optimiser used was Adam.
- The learning rate was set to 0.001 initially. As the number of training epoch increased, the training loss might oscillate around some level, the learning rate was adapted manually in order to decrease more the training loss. The last learning rate used was 0.00005.
- The sequence to sequence model was trained for 204 epochs in total.

### D. Losses

The train losses can be seen in Fig.5. The learning rate was set to 0.001 initially for 4 first epochs. It was modified to 0.0005 for epoch from 5 to 100. Then 0.0001 was used as new learning rate from epoch 101 to epoch 194. Finally for last 10 epochs, the learning rate was set to 0.00005. The observation we can make is that when the learning rate of the loss optimiser was adapted, the loss could go down by some amount. The training losses were almost monotonically decreasing, whereas the validation losses seemed increase as time, as shown in Fig. 6. However, we still took the model of last epoch, regardless its evaluation loss was not minimal. Actually, we found that a mathematical loss could not express truly the performance of the model. Couplets generated by the model with the minimal evaluation loss were much less meaningful than models being trained for a lot of epochs. When evaluating the couplet results, the model trained for more epochs was preferred. Hence, for the final evaluation, the last model after training for 204 epochs was used.

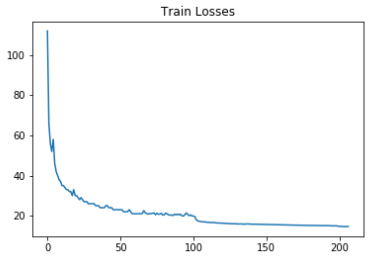


Fig. 5. This figure shows the evolution of losses during the training process. The x-axis shows the number of epochs and the y-axis shows loss values. The learning rate was set to 0.001 initially for 4 first epochs. It was modified to 0.0005 for epoch from 5 to 100. Then 0.0001 was used as new learning rate from epoch 101 to epoch 194. Finally for last 10 epochs, it was set to 0.00005. The loss could go down by some amount when the learning rate of the loss optimiser was adapted.

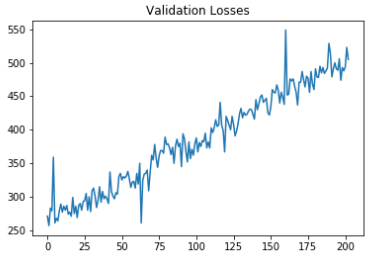


Fig. 6. This figure shows the evolution of the validation losses. The x-axis shows the number of epochs and the y-axis shows loss values.

## V. Evaluation and analyses

### A. Evaluation method

As described previously, we need to respect some rules when dueling Chinese couplets. Constraints are incorporated as followings:

- When the model evaluates the first line of a couplet, its length is recorded. The decoder will only produce characters of the same length as the second line of couplets. *PAD*, *SOS*, *EOS* and *UNK* tokens are not allowed to appear.
- Normally, all characters in the first line of couplets are forbidden to appear in the second line. Thus, we do not allow our model to reproduce the same characters appeared in the first line. If the model predicts the forbidden character, the latter will be replaced by the next candidate character having secondly maximal probability.
- We found that the coherent and related meanings were learned by the model itself, as well as the lexical category. During the training process, a verb will always correspond to another verb. Likewise, a noun will always correspond to another noun. Consequently, when the model encounters a verb during the evaluation process, it will only search for the best candidate from the set of corresponding words encountered during the training process. As a result, the rule of having the same lexical category is usually respected.

- When there are repeated words in the first line of couplets, the second line should also have repeated words at the corresponding positions. Therefore, we forced the model answer to have the correct format. For each input couplet, the positions of repeated characters are recorded. Then the model answer was re-evaluated in the way that each character at these positions follows the character that appeared firstly.
- We have applied tone masking during the evaluation process to respect the tone pattern. The decoder evaluates one character at a time. When the model evaluates the $i$th character $y_i$ of the second line, each candidate character in the vocabulary is associated with a probability. Normally the character with the maximal probability will be chosen as described in Section III-C. Although, nothing guarantees that this character has an opposite tone as the tone of the $i$th character $x_i$ of the source sentence. As we need to respect the tone rule, what we do is to mask all the characters having the same tone as $i$th character $x_i$, then search for the character having maximal probability among the remaining elements.

Each time, searching for the best candidate for the prediction is called greedy search. We have also used beam search to find the best results. Beam search finds k best candidates for each iteration, instead of using one single candidate. By doing so, we can consider more choices and may obtain better results. The final result is taken from the best one having maximal probability among k answers. The number k is called beam width. It is set to 3 for the human evaluation.

### B. Evaluation results

Fig.7 contains some couplets results of our model. The observation we can make is that generally our model produced fair results. Sometimes it could produce even better results than the original answers. Nevertheless, our model performed less well for longer couplets. Characters tends to repeat for the last part of the answer.

### C. Attention plots

Fig.8 contains attention plots for six different couplets. Attention weights are from 0 to 1. We observed that the attention weights have a diagonal form. The $i$th character is decoded mostly based on the $i$th input character, which makes sense.

### D. BLEU score

Bilingual Evaluation Understudy (BLEU) is used as a machine language translation metric which measures the overlap of the source sentence and the output sentence. The higher the BLEU score is, the better the performance is. The Table I shows BLEU scores of our model with different beam widths, averaging over 3 930 couplets from the test set.

Note that it is natural to consider the quantity of overlaps as a metric of language translation performance as the translation is almost an one-to-one mapping. Though the overlap is less significant for Chinese couplets, since we do not translate the source sentence but duel a special poem. There can be millions of possible and good answers. That's why we required the human evaluation at the same time.

| Beam width | BLEU score |
|---|---|
| 1 | **14.15** |
| 3 | 13.76 |
| 5 | 13.54 |

TABLE I
PERFORMANCE OF OUR MODEL

### E. Human evaluation

For the human evaluation, we randomly chose 50 couplets from the test set to form a couplet questionnaire. Each question consists of the first line of the couplets and two choices. One choice is the original couplet answer, the other is the answer produced by our model. Choices are randomly ordered. The volunteers should choose the choice that they think it is better compared to the other choice. The questionnaire was answered by 17 volunteers. Overall, our model was chosen in 48.94% cases. This number was computed by the number of times volunteers chose our answers divided by the number of total answers and total volunteers. Fig.9 contains examples where our model was chosen by most of volunteers compared to the original answer and examples where original answers were mostly chosen.

For most of the time, the original answers were still preferred. Reasons may be the following:

- Our model cannot handle long couplets well in general. Characters in the couplet answer tend to repeat.
- Generally, if there are not repeated characters in the first line of couplets, then there should not have repeated characters in the second line. Nevertheless, our model produces repeated words in a couplet whereas it should not.
- Sometimes our model does not generate meaningful answers.

### F. Additional constraint

The fact that the model generates repeated words in a couplet whereas it should not deteriorates rapidly the performance of the model. Unfortunately we have noticed that after the human evaluation. But, we still decided to fix this issue.

To avoid the model generating repeated characters, we can remove these illegal characters from the list of character candidates. During the evaluation phase, the decoder returns an output for each iteration. This output is a list of probabilities associated to all character candidates. In order to remove unwanted characters, we simply reset the probabilities of characters produced so far by the model to some minimal values. In this fashion, the model will consider other candidate with secondly maximal probability as new output, or several best candidates among the remaining list of probabilities if beam width was greater than one. The model will never reuse same characters produced before, as desired. Four improved results are shown in Fig.10.

| 对联上联 | 机器下联 | 标准下联 |
|---|---|---|
| 宝幢 | 莲花 | 莲花 |
| 窗空花雨后 | 月满柳梢头 | 秋冷珠帘中 |
| 期花不败春常在 | 望月无心夜未央 | 待月长圆夜未央 |
| 买山半晌种幽静 | 把酒一壶叙旧情 | 看竹一庭坐踽凉 |
| 春风送喜财入户 | 瑞雪迎新福临门 | 岁月更新福满门 |
| 削峰作笔云调墨 | 划地铺笺月写诗 | 划地为枰谁布局 |
| 凭栏寄语乡音远 | 把酒临屏梦语浓 | 把酒思情故里遥 |
| 岁月流诗，奋蹄快马催春步 | 风云际会，昂首豪情壮国威 | 城乡绘画，振翅大方赴锦程 |
| 五福临门，五环旗映五星闪 | 八音奏凯，八曲声扬八域欢 | 百川归海，百姓家和百事兴 |
| 吉星高照，人财两旺荣华宅 | 福地福临，福寿双全福寿门 | 喜气常临，富贵双全幸福家 |
| 国策是春风，剪裁园地千村锦 | 党恩施化雨，掀起中华一片天 | 党恩如喜雨，滋润农居万户新 |
| 发扬民主，关爱民生，公正公平为榜样 | 严守法规，清严法管，清廉清己作公 | 创造和谐，建成和美，反污反腐作标兵 |

Fig. 7. Examples of dueling couplets. The first column contains the first line of each couplet which is also the input of the model. The second column contains the couplet answer of the model. The third column contains the desired couplet answer for the second line.

## VI. Discussion

### A. Bidirectional decoder

For the decoder, inputs $y_i$s are entered one token at a time. Therefore, the results of forward pass and backward pass are the same within the iteration. The reason that we still chose to use a bidirectional decoder in our project are the following:

- Experimentally, the bidirectional decoder is faster than unidirectional decoder. When the decoder is bidirectional, GRU only needs to compute with two small matrices of shape $hidden\_size$. One for the forward pass, the other for the backward pass. When the decoder is unidirectional, its hidden size will be $2 \cdot hidden\_size$. So, GRU has to deal with a matrix which is two times bigger. A doubled $hidden\_size$ is due to the fact that the encoder with bidirectional GRU outputs its final hidden state which have $2 \cdot hidden\_size$ entries. Since the final hidden state of encoder is the first hidden state of decoder, the hidden size of decoder should set to $2 \cdot hidden\_size$ in order to be compatible for computation. The computation takes more time for one big matrix than two small matrices.
- If unidirectional decoder is used, the final hidden state of the encoder should be reshaped before fitting into the decoder. While if the decoder is bidirectional, then there is no need to reshape the hidden state. The forward pass and the backward pass are always properly separated.
- Experimentally, the attention plot of bidirectional decoder is better compared to the result of unidirectional decoder. For some unknown reasons, the attention of the first word is concentrated to the last word of the sentence.

### B. Chinese tone

Manually writing down tones for each Chinese characters in the vocabulary is impossible since it would take too much time. Therefore, we have searched for Chinese Pinyin packages in order to figure out the tone of each Chinese character. However, all packages or dictionaries we found contain errors or inconvenience that we could not use. At the end, the relatively best Pinyin package was chosen for the project.[3]

### C. Chinese segmentation and lexical category

Same as Chinese tone package, Chinese segmentation packages produces errors when segmenting couplets. Also, the same Chinese character can have different lexical categories according the the context. This is not captured by packages existing online. As a result, we have not introduced the lexical category labels into our model. Fortunately, most of time the model learned lexical categories during the training process.

### D. Two other models

During the semester, two other sequence to sequence models (noted as Model I, Model II for convenience) were created before the model presented in this report (noted as Model III). Model I and Model II used the same encoder. The decoder of Model I is bidirectional while the decoder of Model II is unidirectional. Model I has been trained for 73 epochs and Model II has been trained for 106 epochs. These two models could produce fair results but they are not as good as Model III.

Different from Model III, for Model I and Model II, all couplets in the training set were padded to the same length which was determined by max_length, the length of the longest couplet in the whole dataset. By doing so, even if lengths of couplets in a batch are much smaller than max_length, they are still padded to have length as max_length. In the implementation of Model I and Model II, *PAD* tokens are not ignored by GRU. Even though all *PAD* tokens are embedded to all-zero vectors in the embedding layer but they still create bias term during the computation. What's worst, couplets are usually short. They do not often have lengths as large as max_length. That means that most of the time, couplets

[3]Pinyin package: https://pypi.org/project/pypinyin/

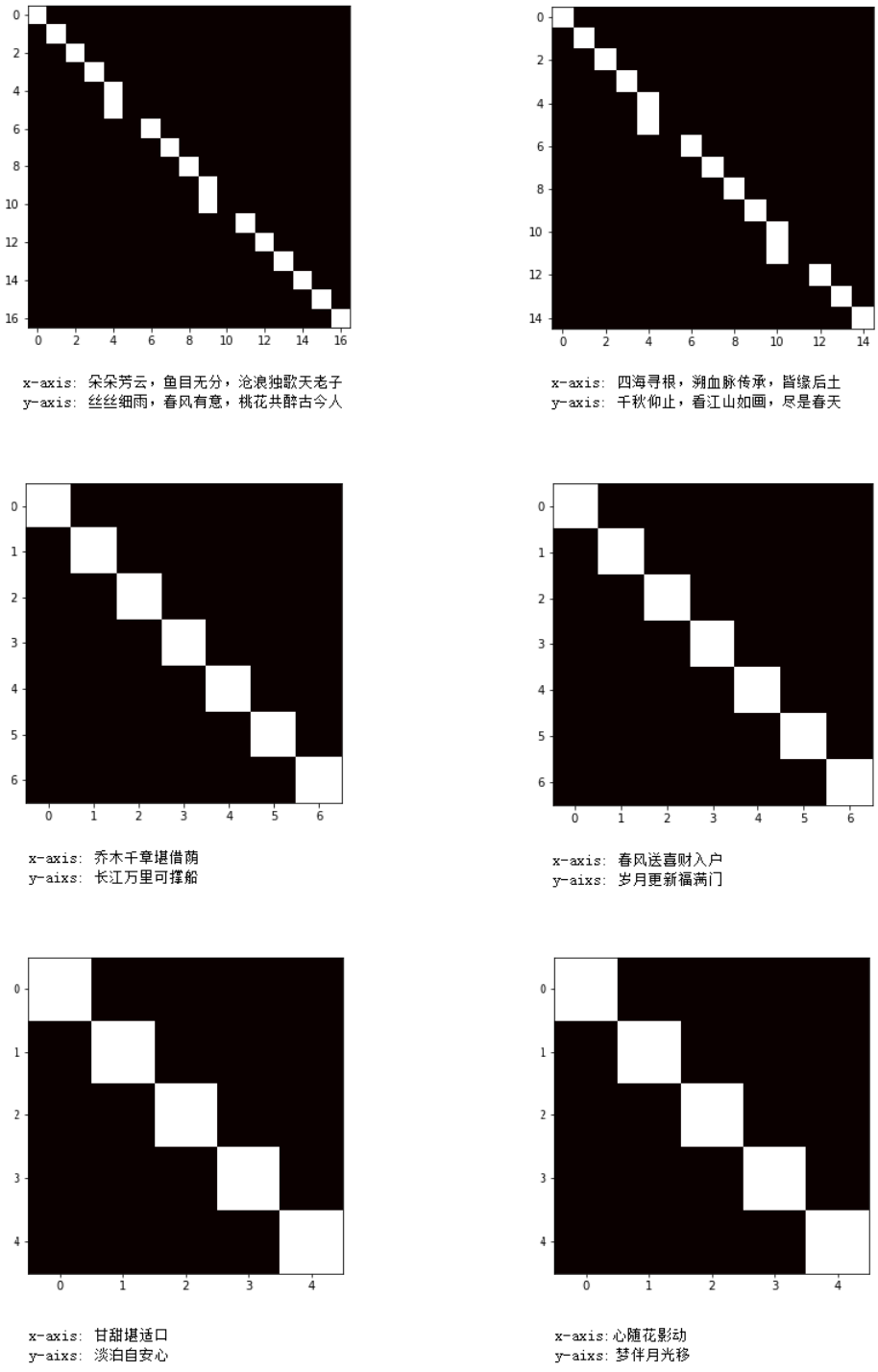


Fig. 8. Examples of attention weights. The x-axis is the first line of the couplets, the y-axis is the answer of our model. Couplet words begin from index 0 for both axes in the plot. Weights coefficients are from 0 to 1. The greater the weight is, the whiter the cell is.

| 对联上联 | 机器下联 | 标准下联 |
|---|---|---|
| 攒攒 | 熙熙 | 纷纷 |
| 举杯歌幸福 | 挥笔颂和谐 | 向党报恩情 |
| 清露洒兰蕙 | 春风拂柳梢 | 幽香拂袖风 |
| 长夜梦回情似水 | 小桥流水月如钩 | 短笺书就意尤甜 |
| 裁万里风云，有色有声长播远 | 染千秋岁月，无私无意不留名 | 雏三门父老，一诗一画总传真 |
| 情结翰墨缘，一度春秋，辽土楹花呈硕果 | 笔绘宏图画，千秋翰墨，滑河画卷绘宏图 | 爱注中华网，两行文字，宁心联海驾长风 |
| 云锁青山风引路 | 日升旭日日熔金 | 雾迷津度月行舟 |
| 老子坛前谁论道 | 秦皇殿上我开心 | 如来殿下我说佛 |
| 集里桥边，长兴骏业 | 中央路上，再展鸿图 | 道吾山下，大展鸿图 |
| 改伪存真，革故标新，改革潮中兴改革 | 为民为本，为民造福，为为本本葆为为 | 东风入律，西霞出彩，东西湖里有东西 |
| 水秀山青，鱼跃鸢飞，真万顷江南景色 | 风清气正，清明正正，大千秋大大风流 | 天高云淡，花香鸟语，好一派北国风光 |
| 鄂南名校，钟灵毓秀，欣继往开来，代有英才登甲榜 | 华夏英才，创业辉煌，喜腾蛟起凤，更期奥运占鳌头 | 儒学府园，卧虎藏龙，喜与时俱进，前无俊业胜今朝 |

Fig. 9. Some results of the couplet questionnaire. The first column contains the first line of the couplet. The second column stands for the couplet answer of our model. The third column shows the original second line of the couplet. Answers that were mostly chosen by volunteers are colored red.

| 对联上联 | 改善前机器下联 | 改善后机器下联 |
|---|---|---|
| 执法为民扬正气 | 秉公办事秉公心 | 秉公办事倡廉风 |
| 湛淡清湖波潋滟 | 闳幽秀水水嵯峨 | 闳幽秀水气恢弘 |
| 风过南楼，竹画北墙，花飞西径，月上东坡不肯去 | 雪融北岭，雪凝梅岭，雪映北山，梅妆北苑好相宜 | 雪融岭壑，水流天际，帆伫长江，舟横野渡欲何眠 |
| 足立八闽，情系八荒，八秩续茶经，总凭科技新风劲 | 心怀百姓，梦圆百业，百州臻大业，更借春风好梦圆 | 心怀百姓，梦圆百海，百州兴骏业，先奋帜扬帆展潮 |

Fig. 10. Improved results by adding the additional constraint mentioned in Section V-F. The first column contains the first line of the couplet. The second column contains the original couplet answer of the model. The third column shows the improved answer.

are facing useless padding and annoying bias addition, which could generate a lot of noise.

The major difference between Model I and Model II was that Model I did not perform a masking for *PAD* tokens when computing the attention. Model I seemed to pay attention on the *PAD* tokens for each couplets. Model II did perform *PAD* masking for attention. However, for some unknown reasons the attention of the first word is concentrated to the last word of the sentence, or to the end of the first sub-sentence. Examples of attention plots can been seen in Fig.11

| Models | Beam width | BLEU score |
|---|---|---|
| I | 1 | 13.95 |
| I | 3 | 13.86 |
| I | 5 | 13.90 |
| II | 3 | 13.87 |
| II | 5 | 13.80 |

TABLE II
PERFORMANCE OF MODEL I AND MODEL II

## VII. CONCLUSION

During the semester, we have created a Chinese couplets dueling machine which can produce more or less fair, sometimes human-like results based on the sequence to sequence model. Couplet constraints are incorporated during both training and evaluation phases to make legal couplets.

Some further improvements or refinements are still needed to be done for the reason that the results are far from perfect. We can probably add local attention into the decoder model so that the relevance between consecutive characters can be enforced. Or try the algorithm for the global attention to compare the performance. The meaningfulness of couplet answers should be improved as well. Another drawback of the model is that when characters in the source sentence are repeated, what we did was to force the output sentence to have repeated words. Sometimes this can make fair answers but sometimes not. We need to consider more alternatives to solve this concern.

## REFERENCES


[1] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[2] I Sutskever, O Vinyals, and QV Le. Sequence to sequence learning with neural networks. *Advances in NIPS*, 2014.

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

[4] Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. Generating chinese classical poems with rnn encoder-decoder. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 211–223. Springer, 2017.

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

[6] Rui Yan, Cheng-Te Li, Xiaohua Hu, and Ming Zhang. Chinese couplet generation with neural network structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2347–2357, 2016.

[7] Jiyuan Zhang, Zheling Zhang, Shiyue Zhang, and Dong Wang. Vv-couplet: An open source chinese couplet generation system. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1756–1760. IEEE, 2018.

[8] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

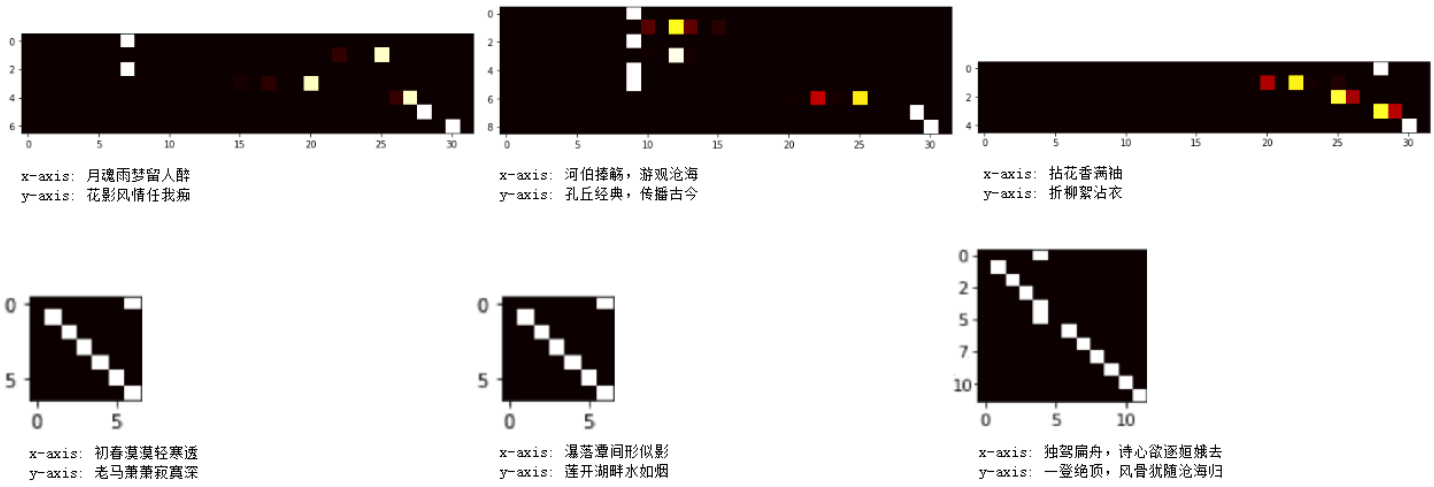


Fig. 11. Some examples of attention plots of Model I and Model II. The first line contains three attention plots of Model I and the second line contains three attention plots of Model II. Model I seemed to pay attention on the *PAD* tokens for each couplets. For model II, the attention of the first word is concentrated to the last word of the sentence, or to the end of the first sub-sentence.