Chinese Poetry Generation Based on Memory Model

Ruiqi Huang

I. Introduction

As an exquisite and concise literary form, poetry is a gem of human culture. Compared with other genres, poetry is subject to constraints of form and meter, which makes automatic poetry generation a challenging but essential step towards computer creativity. In recent years, several neural models have been designed for this task. However, these models still have difficulty in addressing the lack of coherence in the content and theme of the generated poems.

In this project, we suggest improvements on the model designed by Yi et all[1]. The model explicitly maintains information about the topic and the generation history by adding a dynamic memory module. During the generation process, the generation model reads the most relevant parts from the memory slot and consults it to generate the current line. After each line is generated, it writes the most salient parts of the previous line into memory slots. By dynamic manipulation of the memory, the model keeps a coherent information flow and learns to express each topic flexibly and naturally.

Based on the existing model, this report proposes three techniques to improve the efficiency of its training process.

(1) Inspired by de-noising auto-encoders, we propose that adding noise to the encoder input during the word embeddings pre-training can be beneficial to yield a more flexible model. By testing the convergence of the model under multiple noise ratios, we determined the noise ratio parameter.

(2) We found that the probability of replacing the ground truth with the predicted value is an important parameter in determining the speed of model convergence and the final performance. By analyzing the effect of the decreasing speed of Teaching Rate on the model performance and overfitting issue, we determined the parameter of the decay algorithm.

(3) We propose to incorporate scheduled sampling technique in the memory mechanism to facilitate the relevance of the generated sentences to the antecedent text in terms of content and topic.

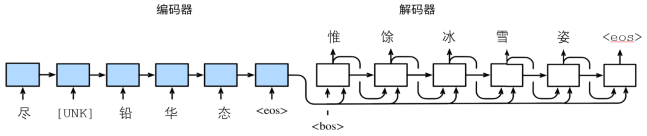
The automatic evaluation results show that the above techniques are beneficial to the content and thematic coherence of the generated poems.

II. Methodology

A. Denoising Autoencoder

Autoencoder is an unsupervised learning method that enables the encoder to learn high-dimensional features by programming the output of the decoder to approach the input of the encoder. denoising autoencoders are a modified version of autoencoders that prevent the neural network from learning identity function by adding noise to the input, thus improving the model's generalization ability.

Autoencoders are commonly used to extract image features, but it has also been shown that adding denoising autoencoders to the training of Seq2seq models can improve the performance of text generation tasks[2]. Inspired by this, the original authors added noise to the input during pre-training of the seq2seq model, replacing words in the input with UNK symbols according to a certain probability as shown in the figure.



We discuss the relationship between the noise ratio c of the encoder during pretraining and the speed of model convergence and automatic evaluation scores, and choose the appropriate setting to prepare for the subsequent training.

```
# Corrupt input to the encoder. Param: corrupt_ratio
def _do_corruption(self, inp):
    # corrupt the sequence by setting some tokens as UNK
    m = int(np.ceil(len(inp) * self._corrupt_ratio))
    m = min(m, len(inp))
    m = max(1, m)
    unk_id = self.get_UNK_ID()
    corrupted_inp = copy.deepcopy(inp)
    pos = random.sample(list(range(0, len(inp))), m)
    for p in pos:
        corrupted_inp[p] = unk_id
    return corrupted_inp
```

The relationship between the noise ratio(c) and the number of randomly polluted characters m is as follows

$$m = \lceil len(input) \cdot c \rceil$$

In our experiments, we set c to be 0, 0.1, 0.2, and 0.3, respectively. we control the noise ratio at a low value, because a higher noise ratio will result in more than half of the characters being replaced with _UNK_ID, and accordingly, a large amount of semantic information will be lost.

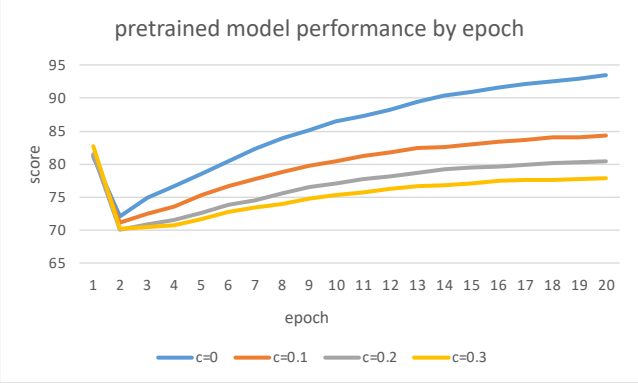


Fig. 1. The effect of noise on the performance of pre-trained models

Figure 1 shows the change of model score against epoch number for different noise ratio settings. As shown by the trend of the lines, pre-training with the addition of noise produces a more stable and effective generative model. Counterintuitively, the data show that higher noise leads to better model performance. This may be related to the short length of the poems and the unconstrained use of language.

To preserve more semantic information and to take into account the robustness of the model to diverse inputs, we used a moderate noise ratio setting of 0.2 for the subsequent training. As shown in the figure, pretraining for too long will make the model performance deteriorate, probably because of the overfitting of the model parameters to the training set, so in this subsequent experiment, we take the pretrained model with epoch=10 for adjustment.

B. Teacher Forcing and Scheduled Sampling

Teacher forcing is a training mode of Seq2Seq model, which is proposed to circumvent the inefficiency of the traditional free running training, where the output of the previous state is directly used as the input of the next state. Obviously, using the wrong output as sequence input is bad for the training of the model when the model has low accuracy.

Teacher forcing requires the model to use the corresponding ground truth directly as the input for the next time step. Specifically, at moment t of the training process, the true value $y(t)$ of the training dataset is used instead of using the model-generated output $h(t)$ as the input $x(t+1)$ at the next time step.

There are two problems with using teacher forcing.

(1) Exposure bias. using Teacher Forcing leads to inconsistency in the behavior of decoding during training and prediction, i.e., the probability prediction of the generated values is inferred from different distributions.

(2) Over correctness. teacher forcing kills the generative diversity of the model to some extent. The use of teacher forcing makes the model overfit to the training set.

Scheduled sampling[3] was proposed to solve the problem of inconsistent input distribution between training and testing of the model in the teacher forcing training mode. Scheduled sampling feeds the model with the both true values and the predicted values as input according to a certain teaching rate(tr). This randomness is designed based on the experience that early in training, the model generates less correct values, so more true values are used as input; later in training, the predicted values are more reliable, so more predicted values are used as input.

```
# 2. teacher forcing in pre-training. Param: teach_ratio
is_teach = random.random() < teach_ratio
if is_teach or (not self.training):
    inp = trgs[:, t].unsqueeze(1)
else:
    normed_out = F.softmax(out, dim=-1)
    top1 = normed_out.data.max(1)[1]
    inp = top1.unsqueeze(1)
```

Inverse Sigmoid Decay[4] is used as the decay schedule, and we want to find the parameter k that is most suitable for the generation task of this topic. The experiments are set to k = 5, 7, 10, and 15 to broadly cover the slow-decay and rapid-decay cases, respectively. After 14 epochs were completed, the most conservative setting (k = 15) used the true values with a probability close to 0.8, and the most aggressive setting (k = 5) used only 20% of the true values. Figures 2 show how different values of k affect the decay of tr.

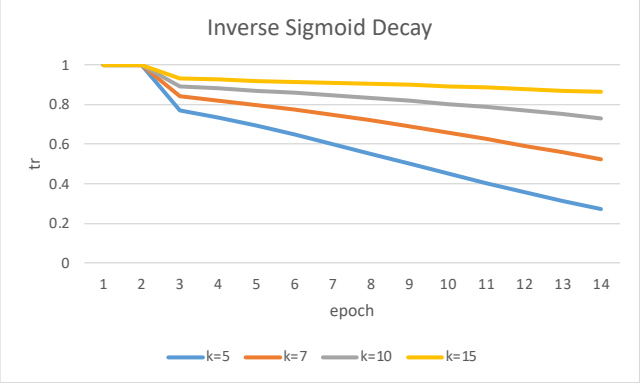


Fig. 2. Decay function: $\epsilon_i = k/(k + exp(i/k))$

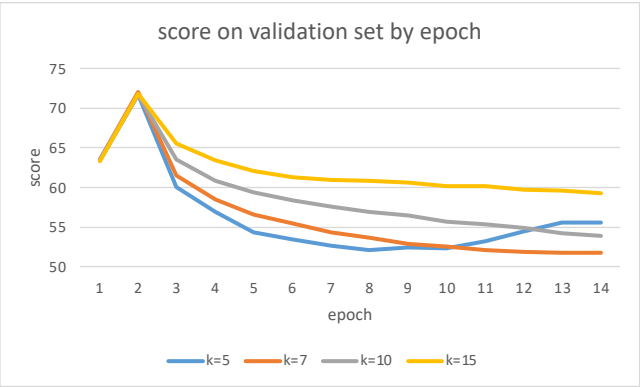


Fig. 3. The variation of the scores with k.

From Figure 3, it can be observed that:

1. the higher the tr is, the slower the model training converges and the worse the performance becomes.

2. tr being too low may cause the model to learn the wrong parameters. The blue curve in the figure (k=5) shows a failure

in model optimization after 10 epochs of training. Referring to Figures 2, we can find that tr is approximately equal to 0.5 at around the 10th epoch, which reminds us that tr less than 0.5 is risky.

With the above information, we believe that taking k=7 and setting the lower bound of tr to 0.5 is the best choice.

During the training period, we also observed the score difference between the training and validation sets of the model, in hopes to understand the overfitting of the generative model. A higher tr gives extra reference to information from the training set that cannot be found in the validation set, inevitably leading to a larger gap in the model's performance during training and testing.

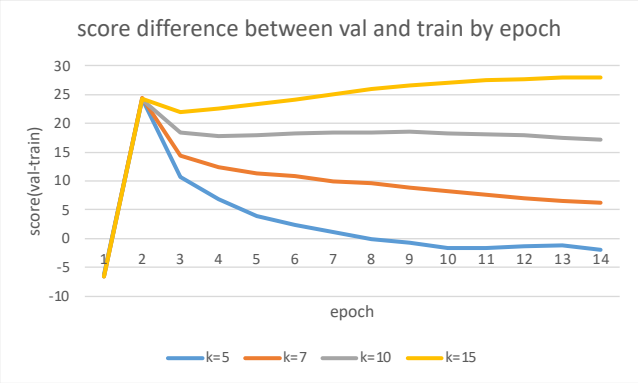


Fig. 4. Score difference between the validation and training sets

In Figure 4 we can see that for larger values of k, the gap between the scores of the training and validation sets is maintained and even widens with the training process. This indicates that a higher learning rate leads to a less flexible model that tends to overfit the training set.

When the k value is small, the model performance of the training and test sets will keep getting closer. For example, the blue curve (k=5) approaches 0 at the 8th epoch, when the model has the smallest difference between the scores in the training and validation sets. From Figure 4, we can see that the model also achieves the best results on the test set at this point, which indicates that the scores of the validation and training sets of the model can reflect the performance of the model in some way, and remind us of whether the model has reached the limit of optimization in the current setting.

C. Scheduled Sampling in Local Memory

In addition to its role in sentence generation, scheduled sampling is also used in the generation model to deal with inter-sentence relationships, using more true values for the contextual information used to generate the current line in the early stages of training and more predicted values in the later stages of training. It is conjectured that this mechanism will allow the model to generate more coherent sentences and make the formulation of topics more fluent.

```
# 3. teacher forcing in local memory.
   Param: teach_ratio
def rebuild_inps(self, ori_inps, last_outs,
    teach_ratio):
    # ori_inps: (B, L)
    # last_outs: (B, L, V)
    inp_len = ori_inps.size(1)
    new_inps = torch.ones_like(ori_inps) *
self.pad_idx
    mask = get_non_pad_mask(ori_inps, self.
pad_idx, self.device).long()
    if teach_ratio is None:
        teach_ratio = self._teach_ratio
    for t in range(0, inp_len):
        is_teach = random.random() < teach_
ratio
        if is_teach or (not self.training):

            new_inps[:, t] = ori_inps[:, t]

        else:
            normed_out = F.softmax(last_out
s[:, t], dim=-1)
            new_inps[:, t] = normed_out.dat
a.max(1)[1]
    new_inps = new_inps * mask
    return new_inps
```

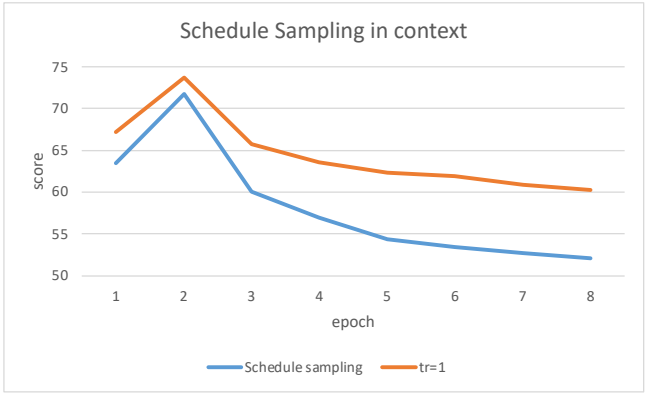


Fig. 5. The blue curve makes tr decrease with epoch according to the Inverse Sigmoid Decay algorithm described in the previous section, while the orange curve keeps tr=1.

As shown in Figure 5, the modest replacement of the input line with the generated word leads to a significant improvement of the model, which verifies our guess.

III. Result

Based on the above experiments, we arrive at the optimal setup: first set c=0.2 to pre-train 10 epochs, and then train 14 epochs at k=7 with Schedule Sampling in local memory.

We use the resulting model for testing. All tests specified the rhythm of the poem as “七绝一（平起首句不入韵）”, the rhyme foot was set to “上平八齐”, and the content in Table 1 was generated under the requirements of three different sets of keywords.

Keywords	银河 铁道	星际 旅客	葡萄 橘子
	崩腾铁马银河落	旅客迢迢千里远	卢橘葡萄一荔子
	嶰谷千岩万壑低	乱山深处有孤栖	菘肥菰黍荐珍鸡
	谁道昆仑山下路	无端星斗连天际	鲥鱼蒓菜新醅熟
	白云深处有人栖	十二峰前一夜低	赪颊樽前琥珀脐

Tab. 1. Generating examples

The above results show that although the poems generated by the model are obviously inferior to the poems written by humans, they can express and connect the topics smoothly, and the content can relate to the preceding and following texts, while basically complying with the metrical requirements. However, we also found that the choice of topic words would greatly affect the quality of the generated poems, and too rare words would limit the flexibility of the generation model.

IV. Outlook

Our analysis of hyperparameters still leaves some room for improvement. We have not considered the impact of the pretraining settings on the subsequent training, such as whether the proportion of noise added to the pretraining will have an impact on the peak performance achieved by the model in the subsequent training, whether the pretraining is too long will have a negative impact on the subsequent training, and the optimal tr setting in the pretraining.

References

[1] Z. Guo, X. Yi, M. Sun, W. Li, C. Yang, J. Liang, H. Chen, Y. Zhang, R. Li, Jiuge: A human-machine collaborative chinese classical poetry generation system, in: ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Syst. Demonstr., 2019: pp. 25–30.

[2] L. Wang, W. Zhao, R. Jia, S. Li, J. Liu, Denoising based sequence-to-sequence pre-training for text generation, in: EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf., 2020: pp. 4003–4015.

[3] W. Zhang, Y. Feng, Q. Liu, Bridging the gap between training and inference for neural machine translation, in: IJCAI Int. Jt. Conf. Artif. Intell., 2020: pp. 4790–4794.

[4] S. Bengio, O. Vinyals, N. Jaitly, N. Shazeer, Scheduled sampling for sequence prediction with recurrent neural networks, in: Adv. Neural Inf. Process. Syst., 2015: pp. 1171–1179.