Learning and Thinking Strategy for Training Sequence Generation Models

Yu Li[1,2,3]
liyu@ict.ac.cn
Sheng Tang[1]
ts@ict.ac.cn
Min Lin[3]
linmin@360.cn
Junbo Guo[1]
guojunbo@ict.ac.cn
Jintao Li[1]
jtli@ict.ac.cn
Shuicheng Yan[3]
yanshuicheng@360.cn

[1] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China

[2] the University of the Chinese Academy of Sciences
Beijing, China

[3] AI institute
Qihoo 360
Beijing, China

Abstract

In order to alleviate the exposure bias problem caused by the discrepancy between training and testing strategies, we propose an Inverse Reinforcement Learning (IRL) based learning and thinking strategy for sequence generation. First, a task-agnostic reward is learned to evaluate the appropriateness of the generated tokens at each time step with the knowledge of ground truth token and current RNN models. With this reward, a deep SARSA network is then designed to meditate among the whole space. Therefore, it can fill in the space that has not been exposed during training with a better policy than the original RNN model. Sequence generation experiments on various text corpus show significant improvements over strong baseline and demonstrate the effectiveness of our method.

1 Introduction

Generating sequential data that is like real data is an appealing and useful unsupervised problem that has been explored for a long time. The sequence generation approaches are a core component of many important artificial intelligence and natural language processing problems such as language modeling [21], machine translation [1, 29], speech recognition [8], image caption [14], *etc.*

Recurrent neural networks (RNN) [21] provide a natural way to model sequence. However, RNN-based methods suffer from a major exposure bias problem [4] caused by the discrepancy between the different strategies during training and inference [25]. Generally speaking, during

This work was supported by the National Key Research and Development Program of China (2017YFB1002202), the National Natural Science Foundation of China (61572472, U1703261 61571424).
Corresponding author: Sheng Tang.

training, the RNN model is trained to predict the next token given the previous ground truth as input, which is called *teacher forcing strategy* [4]; while during inference, we have to feed the previously generated token back as input at current time step to produce a whole sequence, which can quickly accumulate a serious bias, *i.e.*, exposure bias problem. Recent researches [4, 25] figured this problem out in some degree by gently transferring the training input tokens from ground truth to the previously generated tokens. Once the ground truth is not taken as input, it is no longer proper to use them as targets due to the aligning problem. More specifically, if the ground truth sequence is *"I really love English"*, while the model generates *"I love English"*, it is improper to judge "*love*" as "*really*" at the second time step. Therefore, a new judgment is desirable for sequence evaluation. Unfortunately, sequence level metrics like BLEU [24] and ROUGE-2 [16] are not differentiable which means it will be hard to optimize the network at sequence level directly. As a result, reinforcement learning (RL) methods [35] [31] are introduced to language modeling [3, 25] to change these non-differentiable metrics to reward in RL.

Once the sequence generation model is considered as an agent in RL, which can be trained with generated tokens as input, it should be noticed that we originally have no access to any reward signal. We only have plenty of sentences written by human which can be regarded as sequences sampled from an expert policy, which means it will be more natural to solve the agent in the view of inverse reinforcement learning (IRL) which attends to recover the reward given the execution traces of an expert policy [1, 23]. However, methods like [3, 25] directly construct the reward with task-specific evaluation metrics. Though they gain improvements with specific metrics, we consider that sequence generation problem is essentially unsupervised and universal, and it is somehow unnatural to bind the model with some specific tasks. Another solution starts from the view of generative adversarial networks (GAN) [9], and use the signal from a discriminator as reward [6, 36].

We follow IRL to figure out a solution. In the view of IRL, traditional RNN-based models [21] are all IRL methods since they take into account the expert behaviors to imitate the expert policy. They directly model the state-action value function Q, but the exposure bias problem proves that Q does not generalize well in the space that training data do not cover [25] due to the complexity of what Q represents (details will be discussed in Sec.3). Thus, our proposed methods try to explore and model the space not covered by the training data by taking as input the previously generated tokens.

On the contrary, we propose to derive the reward R which is less complicated and can be easier to generalize from Q that is trained with the teacher forcing strategy to learn the knowledge exhibited in the expert behaviors. With this reward, the agent is able to try a large number of situations automatically and learn the best choices in different states, which can be considered as thinking since it is a self-learning process. Additionally, this reward is task-agnostic. With our proposed learning and thinking strategy, our model can perform better in the space where training data do not cover and experiments on text generation task with corpus Penn Treebank (PTB) [19], Chinese Poem [38], and Obama Political Speech show a significant improvements in perplexity and BLEU scores over strong sequence generation baseline models, which verifies the effectiveness of our proposed method.

The main contributions of this paper are as follows:

- We propose that the immediate reward should be less complicated and easier to generalize than the action value function which is a discounted expectation summation of rewards at each time step.
- According to this opinion, we propose the learning and thinking strategy for training

sequence generation models, which learns action value function from the given training data and thinks with our derived task-agnostic reward to explore spaces that training data does not cover to improve the generalization ability of our policy.

2 Related Works

Explorations on Exposure Bias Problem Standard RNN suffers from the aforementioned exposure bias problem. Once we want to avoid the problem by using as input the previously generated token during training, how to set proper targets for these frames becomes important. [10] first advocates to use model's own prediction as input during training in their proposed method SEARN. DAGGER [27] is a similar idea in an imitation learning framework. [4] proposes a method that randomly chooses ground truth words or generated words as input at each time step. Their targets remain to be the ground truth, which seems improper to force the model to predict a specific token regardless of input tokens due to the aligning problem.

Thus, a sequence level guidance are required as training target. [25] introduces REINFORCE algorithm [35] to transform evaluation metrics like BLEU [24] to reward signals for RL which is called MIXER. [3] applies an actor-critic approach [31] to directly improve task-specific metrics. The similar idea is also applied to image caption task in [17, 26] which achieve good performances. However, these methods aim at improving some certain supervised tasks such as machine translation, image caption by using sequence generation as a submodule. Sequence generation, which is essentially an unsupervised problem, has not been improved actually. Different from MIXER, SeqGAN [36] and MaliGAN [6] consider constructing the reward signal with a discriminator that judges whether a sequence is real or synthetic.

Reinforcement Learning In Markov decision processes (MDPs) [5] where the state space is very large or continuous, or the environment E (consisting of state transitions T and reward R) is unknown, reinforcement learning algorithms [35] [32] [33] [31] can be adopted to learn an optimal policy. As deep neural networks achieve great progress in computer vision field [11, 28] , they have also been utilized in RL and open a new era of deep RL [15, 22]. Deep Q Network (DQN) [22] uses deep convolutional neural networks to approximate Q-learning [34] and can achieve human-level control in many Atari games, which attracts worldwide attention. On the contrary, IRL attends to recovering reward R given an expert policy or its execution traces [1, 23]. These research are usually motivated by the need of modeling the behaviors of animals and human, who can be regarded as an expert. With the learned R, we may reproduce the optimal policy. Classical RL methods such as REINFORCE [35], actor-critic [31], SARSA [13, 18], have been employed to text generation tasks to optimize the policy via non-differentiable sequence level metrics.

3 Method

Sequence generation can be considered as a decision process. We intend to model it with an agent using RL methods. A standard model for RL is MDP, which can be represented as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$ where the state $s \in \mathcal{S}$ is Markov, as defined in [30], a state signal that succeeds in retaining all relevant information is said to be Markov, or to have the Markov property. However, in sequence generation problems, a sequence of tokens $(w_0, w_1, ..., w_T)$

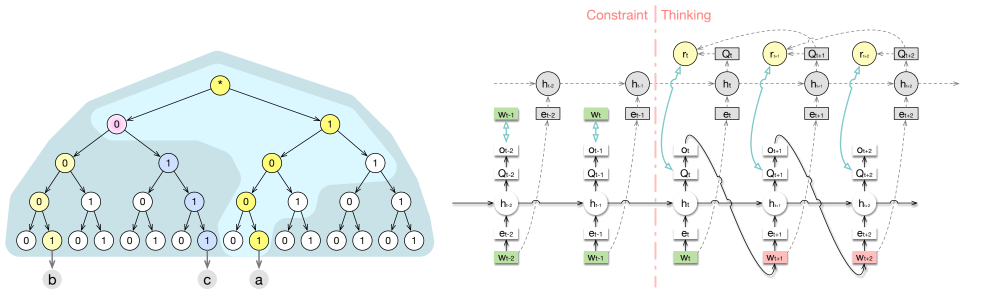

Figure 1: Left: Illustration of the searching space for different strategies. This is a generation example of a sequence whose length is 4, and the vocabulary size is 2. Sequence a is a training sequence. Original RNN training strategy models the light blue space. Our strategy enables thinking in the dark blue space while keeping the results in light blue space. Once we sample the pink 0, we may go back to a similar sequence b or generate an unseen but reasonable sequence c. Right: An illustration of our training strategy. $w_i, e_i, h_i, Q_i, o_i, r_i$ represent the input words, embedding, hidden state, action value vector, output, and reward for the i^{th} step respectively. The white boxes with shadow are calculated by $\mathbf{Q}^{\phi}$, and the gray boxes with black border are calculated by $\mathbf{Q}^{\theta}$ to provide rewards. The green boxes indicate the ground truth words, boxes in red indicate the generated words. Yellow circles represent reward signals. Double side arrows represent supervision relationships.

are not Markov because the current token w_t is not only related to the previous w_{t-1}. In natural language processing, traditional methods usually introduce an n-gram hypothesis to constrain that w_t is only decided by the previous n words $(w_{t-1}, w_{t-2}, ..., w_{t-n})$. To transform our problem into an MDP, we define the state $s_t = (w_0, w_1, ..., w_t)$. Thus, s_t can be only decided by s_{t-1} and is Markov.

The agent constructs a policy $\pi(a|s)$ to model the action distributions for each state, and takes an action a at each time step following π. π is usually derived from a state-action value function $Q(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a]$ where γ is the discount factor. As described in DQN [22], modeling the scalar $Q(s,a)$ that is calculated with a state s and an action a as input can be transformed to modeling a vector $\mathbf{Q}(s)$ that is calculated with a state s, and $Q(s,a)$ is the a_{th} element of $\mathbf{Q}(s)$, indicating as $\mathbf{Q}_a(s)$. Normally, with a learned Q, a policy can be easily derived by applying $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(s,a)$ for many RL methods like Q-learning. However, in sequence generation, the policy should be stochastic because there may exist many possible actions that are all valid in a state. Consequently, we have:

$$a \sim \pi(s) = \operatorname{softmax}(\mathbf{Q}) = \frac{\exp(\mathbf{Q}(s))}{\sum_{a_i \in \mathcal{A}} \exp(\mathbf{Q}_{a_i}(s))}. \tag{1}$$

$\mathbf{Q}$ is modeled by RNN because s is a time sequence. Policy can be derived from the modeled $\mathbf{Q}$ with $\operatorname{softmax}(\mathbf{Q})$. We intend to solve $\mathbf{Q}$ with RL methods, but we still do not have access to the reward R. We only have access to many expert behaviors (human-written sentences), which means that sequence generation is actually an IRL problem.

Traditional RNN-based methods [7, 21] usually model $\mathbf{Q}$ directly with the teacher forcing strategy and apply cross entropy loss on Eq.(1). This process does not require the estimation of reward. But these models suffer from the exposure bias problem [4] which leads to the question why the errors will accumulate along the inference trail. Why can't we return to a "correct" state once we have made some "wrong" decisions?

Recall the standard training process of RNN with a simple example(Fig.1-Left). Assume we want to generate a sequence of length 4, and the vocabulary only has two words, 0 and 1. A ground truth sequence is *1001*. During training, as we always take the ground truth words as input, we only search the space in light blue (seen space) where the training data covers. Ideally, if we model **Q** function in the light blue space, it should generalize well in the other space (unseen space), which means if we accidentally made some wrong decision and went to some unseen state, we should know how to go back to normal trace to generate a reasonable sequence. Actually, however, Q can be easily "lost" in the unseen areas. This phenomenon we observed indicates that the Q function learned from the teacher forcing strategy does not generalize well, and the model is just a mess on these blank areas.

Here, we claim the reason may be the nature of Q, a discounted expectation summation of rewards at each time step, which has a very sophisticated meaning. We propose that *reward R should be less complicated and easier to generalize* because it represents the immediate gain of the current action, which is much straightforward. We try to recover the reward function R that the expert follows, and use this R for RL to explore the unseen spaces.

Generally speaking, traditional RNN-based methods only have the *learning* process, during which the expert behaviors are absorbed into Q with the teacher forcing strategy and cross entropy loss. Instead of modeling Q, we propose first to obtain reward R which is easier to generalize than Q, and then the learned R guides the model to explore more on unseen states, walk through more trails, and manage to model a better Q in the whole space, which is called *thinking* because it is a self-learning process without any expert behaviors involved.

Next, we first explain how to recover the reward R, and then elaborate how to obtain a new policy with this R. In addition to this main framework, some implementation techniques are applied to ensure that the policy converges well.

3.1 Learning Reward

To recover the reward R that the expert follows, we first train an RNN model with teacher forcing strategy and cross entropy loss as in [37], which results in an RNN model $\mathbf{Q}^{\theta}$. We consider this model $\mathbf{Q}^{\theta}$ as an expert in the seen space. Therefore, the reward R can be generated from $\mathbf{Q}^{\theta}$ with Bellman Expectation Equations:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi}(s,a) \tag{2}$$

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} V^{\pi}(s') \tag{3}$$

Because the transform probability from s to s' is 1, *i.e.*,$\mathcal{P}^{a}_{ss'} = 1$ in text generation, we have:

$$\begin{aligned} R(s,a) &= Q^{\pi}(s,a) - \gamma V^{\pi}(s') \\ &= Q^{\pi}(s,a) - \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^{\pi}(s',a') \end{aligned} \tag{4}$$

In the context here, action a is w_{t+1}, and s is s_t. Bringing Eq.(1) into Eq.(4), we have:

$$R^{\theta}(s_t, w_{t+1}) = \mathbf{Q}^{\theta}_{w_{t+1}}(s_t) - \gamma \frac{\exp(\mathbf{Q}^{\theta}(s_{t+1}))}{\sum_{a_i \in \mathcal{A}} \exp(\mathbf{Q}^{\theta}_{a_i}(s_{t+1}))} \cdot \mathbf{Q}^{\theta}(s_{t+1}) \tag{5}$$

Thus, we derived the reward R with expert behaviors by resorting to teacher forcing learning. Additionally, as is described above, our reward R is obviously task-agnostic.

3.2 Thinking Policy

With the learned reward, sequence generation problem turns into a standard RL problem. Therefore, a new Q can be obtained by RL methods with learned R. But methods like Q-learning produce deterministic optimal policies which cannot meet our stochastic requirement. Thus, to produce policy as shown in Eq.(1), SARSA [13, 18] is suitable for producing stochastic policies. SARSA is an on-policy learning algorithm that interacts with the environment and updates the policy based on actions taken:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[R_t(s_t,a_t) + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)], \tag{6}$$

where α is the learning rate. For each time step, we take the previously generated sequence s_t as input, another RNN model $\mathbf{Q}^{\phi}$ calculates $\mathbf{Q}^{\phi}(s_t)$, and randomly samples the next token following $\pi(a_t|s_t)$ derived from $\mathbf{Q}^{\phi}(s_t)$ following Eq.(1). A reward signal is then given back by $\mathbf{Q}^{\theta}$ with Eq.(5). To apply SARSA to RNN, referring to Deep Q Network [22], we propose Deep SARSA Network (DSN), which can be trained by minimizing the loss functions $L_i(\phi_i)$ that changes at each iteration i,

$$L_i(\phi_i) = \mathbb{E}_{s_t,a_t\sim\pi_i}[(y_i - \mathbf{Q}_{a_t}^{\phi_i}(s_t))^2] \tag{7}$$

where $y_i = \mathbb{E}_{s_t,a_t\sim\pi_i,s_{t+1},a_{t+1}\sim\pi_{i-1}}[R(a_t,s_t) + \gamma \mathbf{Q}_{a_{t+1}}^{\phi_{i-1}}(s_{t+1})|s_t,a_t]$. Thus,

$$\bigtriangledown_{\phi_i} L_i(\phi_i) = \mathbb{E}[(R(a_t,s_t) + \gamma \mathbf{Q}_{a_{t+1}}^{\phi_{i-1}}(s_{t+1}) - \mathbf{Q}_{a_t}^{\phi_i}(s_t)) \bigtriangledown_{\phi_i} \mathbf{Q}_{a_t}^{\phi_i}(s_t)] \tag{8}$$

Then, the RNN model $\mathbf{Q}^{\phi}$ can be trained with RL method SARSA and the loss defined in Eq.(8). We should notice that while learning the SARSA network $\mathbf{Q}^{\phi}$, no ground truth text is used. It walks to many unseen spaces and is a self-learning process that only receives feedback signals R from advanced learned $\mathbf{Q}^{\theta}$, namely thinking.

3.3 Implementation Techniques

Initialization During thinking, though it is ideally feasible to learn an entirely meditated model, the policy model $\mathbf{Q}^{\phi}$ is actually very hard to converge well in reality if it is trained from scratch without any extra conditions because the searching space is often too large. Consequently, motivated by MIXER, certain conditions are needed to ensure effective thinking in searching space where reasonable sequences may exist. We initialize $\mathbf{Q}^{\phi}$ with well-trained $\mathbf{Q}^{\theta}$ to ensure it starts off with a much better policy than a random one. This initialization shrinks the searching space sharply because $\mathbf{Q}^{\theta}$ already focuses on the good part of space.

Constraint Our reward R is derived from the hypothetical expert policy $\mathbf{Q}^{\theta}$, which is actually not a real expert. Thus, the reward R is biased from the optimal R^*, but is still able to provide better knowledge for $\mathbf{Q}^{\phi}$ in the unseen space. However, if we only use the biased R for thinking, the accurate estimation of Q in the training data space (that can be well modeled by the teacher forcing strategy) could also be biased and reduce the performance. So, the teacher forcing constraint should be added in the training data space to ensure the estimation of Q in the seen space while we meditate Q in the unseen space to fill in the blank. Also inspired by MIXER, we design learning and thinking strategy with constraint as shown in Algorithm.1 A gradually curtate length of head is trained with teacher forcing, and the rest of the sequence is trained with thinking strategy. The structure is shown in Fig.1-Right.

```
Data: a set of sequences
Result: $\mathbf{Q}^{\phi}$
Initialize $\mathbf{Q}^{\theta}$ at random, training schedule $\mathbb{S}$;
Train $\mathbf{Q}^{\theta}$ following $\mathbb{S}$;
Initialize $\mathbf{Q}^{\phi}$ with $\mathbf{Q}^{\theta}$, set constants $\Delta, l, s, e$, learning rate $lr$, and $decay$, set variable
  $n$.
for $n = l : s : -\Delta$ do
    $lr = lr \times decay^{(l-n)/\Delta}$;
    Train $\mathbf{Q}^{\phi}$ for $e$ epochs with $lr$: use cross entropy loss and ground truth as input for
      teacher forcing learning in the first $n$ steps; use SARSA loss and generated word
      as input for thinking in the remaining $(T-n)$ steps;
end
```

Algorithm 1: Learning and thinking strategy with constrain for language modeling.

Transpose embedding For RNN-based sequence models, we always need to map a one hot vocabulary vector to RNN space with an embedding matrix $M_e \in \mathbf{R}^{N\times m}$, in which each vector represents a corresponding token, where N is the size of vocabulary W, and m is the dimension of embedding vectors. For output, another transformation (usually implemented with a fully connected layer with weight parameter $M_o \in \mathbf{R}^{m\times N}$) is needed to map the hidden state $\mathbf{h}_t \in \mathbf{R}^m$, which has the same dimension as embedding vector does, to a vocabulary vector that indicates the probabilities of all words in the vocabulary. If we consider $\mathbf{h}_t$ as a demand vector that expresses what kind of word is needed now in embedding space, we can generate the probabilities over vocabulary by calculating the similarity between the demand vector and each embedding vector. We choose to use inner product, which is correlated with cosine similarity, to neatly reach this target by replacing M_o with $M_e^{\mathbf{T}}$. The transpose embedding approach not only reduces a great number of parameters thus prevents the model from overfitting, but also facilitates the gradient flow among all the calculation graph that helps the model to converge better. This small technique is coincidentally similar to that of [12], but we propose it from a totally different motivation.

4 Experiments

We mainly conduct our experiments on text generation task with Penn Treebank, Obama Political Speech, and Chinese Poem datasets respectively.

Penn Treebank [19] [1], has 930K words for training, 74K for validation, and 82K for testing. The vocabulary size is 10K. Words out of the vocabulary are replaced with *. Perplexity (PPL) is utilized for model evaluation as is standard in language modeling.

LSTM is used for our basic RNN framework. We conduct two sets of experiments on different levels of the number of parameters. The small LSTM model base-s and the large LSTM model base-l have exactly the same structure as is in [37]. Our implemented base-l reproduce the performance of LSTM-l in [37]. The dimension of embedding vectors is the same as the LSMT cell. During training $\mathbf{Q}^{\theta}$, we follow the training schedule in [37]. During training $\mathbf{Q}^{\phi}$, we increase the batch size to 100. $lr = 0.005, e = 5$ for both models. For small model, we set $l = 10, s = 4, \Delta = 2, decay = 0.32$. For large model, we set $l = 35, s = 26, \Delta = 3, decay = 0.5$. Base models (base-$x$) are trained by teacher forcing strategy

[1]http://www.fit.vutbr.cz/ imikolov/rnnlm/simple-examples.tgz

Table 1: (1) Performance of our models versus other neural language models on PTB(P) and Obama speech(O) datasets. PPL indicates perplexity (smaller is better). -T and -w/oT means with and without transpose embedding introduced in Sec.3.3. (2) BLEU-2(B2), perplexity(PPL) performances and their relative improvements over their own baselines(+%) on Chinese Poem-5 and Poem-7 Corpus.

Table.1(1)

Set	Model	PPL Val	PPL Test
P	KN-5[20]	148.0	141.2
	RNN-LDA[20]	–	113.7
	LSTM-l[37]	82.2	78.4
	MaliGAN[6]	128.0	123.8
	base-*s*	119.8	111.3
	base-*s*-extra	118.9	110.8
	base-*s*-T	114.0	109.0
	ours-*s*-w/oT	112.7	105.5
	ours-*s*	**110.8**	**104.3**
	base-*l*	82.9	78.7
	base-*l*-T	79.0	75.0
	ours-*l*-w/oT	80.8	75.4
	ours-*l*	**78.7**	**73.5**
O	base-*l*	128.2	108.3
	ours-*l*	**121.9**	**100.7**

Table.1(2)

Model	Poem-5 B2	+%	PPL	+%	Poem-7 B2	+%	PPL	+%
RNNLM[38]	–	–	–	–	–	–	145.0	–
RNNPG[38]	–	–	–	–	–	–	93.0	35.9
MLE[6]	0.69	–	564.1	–	0.32	–	192.7	–
SeqGAN[36]	0.74	6.8	–	–	–	–	–	–
MaliGAN[6]	0.76	10.1	542.7	3.8	0.55	**71.9**	180.2	6.5
base-*l*	0.67	–	166.4	–	0.69	–	134.9	–
ours-*l*	**0.80**	**19.4**	**137.1**	**17.6**	**0.77**	11.6	**82.3**	**39.0**

while our models (ours-x) are trained by learning and thinking strategy with techniques stated in Sec.3.3 without specific states. All the results are shown in Table.1.

4.1 Main Results

On PTB dataset, we list the results of the classical LMs [20] and a current sequence generation method MaliGAN-full [6] that reports there results on PTB as a reference line to show how this dataset is solved by methods that focusing on sequence generation. [37] is our baseline which concentrates on how to add regularization to large LSTM networks. Comparing with base-s, ours-s improves 7.0 PPL which is by 6.3% relatively. Base-l reproduces LSTM-l in [37]. Ours-l outperforms the state-of-the-art single model LSTM-l over 6.6% relatively.

To prove the generalization ability of our proposed methods further, we conduct experiments on Obama Political Speech Corpus[2] and Chinese Poem Corpus [38][3]. Obama Political Speech Corpus is consists of 11,092 paragraphs with 32530 vocabularies (much more than the vocab size of PTB) from Obama's political speeches. Comparing with base-l baseline, our model also improves PPL of 7.1% relatively.

On Chinese poem generation task, we split the dataset as described in [38], and train 2 models for Poem-5 and Poem-7 (poems consisting of 5 or 7 Chinese characters) respectively with training set. Following [36] and [6], we report the BLEU-2 scores and PPL. Our method obtains the best BLEU-2 scores and PPL on both datasets. As the baseline models perform differently, we consider it will be more fair to compare the relative improvements over baseline models. RNNLM is the baseline of RNNPG, MLE is the baseline for SeqGAN and MaliGAN-full, and base-l is our baseline. As shown in +%, though our baseline model is very strong and hard to improve, our proposed method achieves relatively higher improvement

[2]https://github.com/samim23/obama-rnn
[3]http://homepages.inf.ed.ac.uk/mlap/Data/EMNLP14/

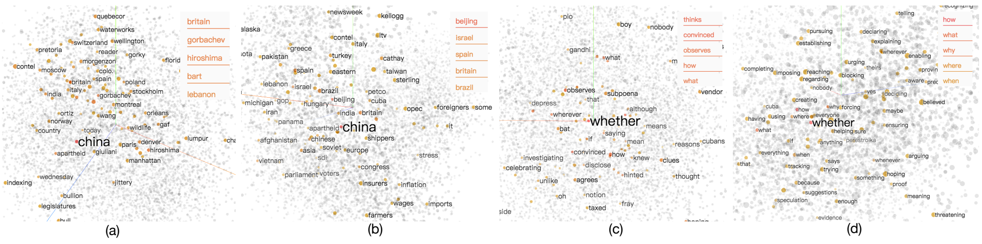

Figure 2: Visualization of embedding parameters after PCA. We highlight the most similar words with our selected words (china and whether). The redder the color is, the closer the word is to the selected word. Top-5 words are listed on the upper right of each subfigure. (a)(c) are the embeddings of base-s model; (b)(d) are that of base-s-T model.

over baselines. The B2+% of MaliGAN [6] is high because their baseline 0.32 is very low.

As is shown on these 3 datasets, we conjecture that the learning and thinking strategy can obtain a better policy in the unseen space, and that the reward R is easier to generalize than Q.

4.2 Ablation Study

Extra Iterations Since our learning and thinking strategy involves extra iterations for thinking phase, for fair comparison, we train the base-s model with the same extra epochs, which is base-s-extra. As is shown, base-s-extra can hardly improve PPL since base-s is already converges well, which means the improvement of our model is caused by our training strategy but not the extra iterations.

Implementation techniques Without initialization and constraint, the model is easily trapped in a state where the samples are recurrent, like the original LSTM with a very low temperature for softmax, which leads to poor generation performance. Without transpose-embedding, our model ours-s-w/oT still gains 5.8 PPL improvement over base-s. Another 1.2 PPL is obtained comparing ours-s with ours-s-w/oT. Similar results can be derived from large models.

4.3 Discussions

Transpose embedding This approach increases PPL while decreasing the number of parameters (61% for base-s, and 77% for base-l). Linking embedding parameter with the output provides a shortcut for gradient flow across RNN cell, which enables the model to converge better. As shown in Fig.2, by using transpose embedding (Fig.2(b)(d)) to replace the original FC layer (Fig.2(a)(c)) for output projection, the embedding parameters learn more semantic informations. The closest words (*thinks, convinced, observes*) of the interrogative *whether* turn into (*how, what, why*), which are more reasonable, which verifies that properly cutting and reusing parameters can prevent overfitting in some degree.

Text generation We feed the same context to base-l model and ours-l model respectively to generate sentences. For normal heads, base-l and ours-l perform almost the same, while sometimes the sentences generated by our model seem to be more natural. Furthermore, the generated sentences have not appeared in training set, which means the model learns to produce novel sentences rather than remembers all the patterns. Some examples are listed in the following box. To prove that our strategy provides better guidance in the unseen spaces than original teacher forcing strategy, we deliberately feed some wrong heads to the model. The wrong heads are totally a mess, which ensures that we start generating at a position in the unseen area of original strategy. We should notice that in many cases base-l cannot find a way back to normal space and generate messy code continuously, while ours-l can still generate good sequences. As shown in the box, our model saves the situation by returning to a normal state, ending the mess, and starting a new sensible sequence.

(Normal heads):
we 're talking about years ago ***but two big by theater and the representatives***
we 're talking about years ago ***when our market was here***
but for now they 're looking forward ***on a series of occasions***
but for now they 're looking forward ***to the federal budget agencies***
the new company said it believes there are ***questions that over she***
the new company said it believes there are ***some progress in cooperation and development***
(Wrong heads):
result purchasing direct but new the a of and holder if on at for ***the moment path during the new york first nine months that might take effect dec. N with the federal reserve board***
result purchasing direct but new the a of and holder if on at for ***a year he said mr. luzon also said investors are worried about changing his balance of roughly N billion***
tailored premium charge be to N address for will sponsor regular ***old of the funds directly while the * of such major assets is considerably lower yields but will face those metropolitan***
tailored premium charge be to N address for will sponsor regular ***performance yesterday mr. * said *of new requirements would be recommended available for storage***
but bros. america in you be years N ago people forward ***with the shapiro in N or N fiat for example in the country responsible for throughout desktop national deal is***
but bros. america in you be years N ago people forward ***with guys individual advertisers will charge over capacity that loans would end up abroad***

Pleas note: We provide the same context sequences for base-l model and ours-l model as inputs to generate the whole sentences. The ***bold italic words*** are generated by base-l. The ***bold italic words*** are generated by ours-l.

5 Conclusions

In this paper, we propose a learning and thinking strategy for sequence generation, in which the reward is first learned by the teacher forcing strategy, and then deep SARSA network is designed to guide the model when wandering in the unseen space, and some implementation techniques are presented to ensure a good convergence of our model. PPL and BLEU score improvement is obtained on the corpus Penn Treebank, Obama political speech, and Chinese poem respectively according to our experiments. Detailed discussions and insight analysis further clarify that our approach can model the expert policy better in the unseen space, which ensures us to generate novel and meaningful sequences.

References

[1] Pieter Abbeel and Andrew Y Ng. Inverse reinforcement learning. In *EML*, pages 554–558. Springer, 2011.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and er al. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[3] Dzmitry Bahdanau, Philemon Brakel, and et al. An actor-critic algorithm for sequence prediction. *ICLR*, 2017.

[4] Samy Bengio, Oriol Vinyals, and et al. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015.

[5] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[6] Tong Che, Yanran Li, and et al. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[7] Kyunghyun Cho, Bart Van Merriënboer, and et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Jan K Chorowski, Dzmitry Bahdanau, and et al. Attention-based models for speech recognition. In *NIPS*, pages 577–585, 2015.

[9] Ian Goodfellow, Jean Pouget-Abadie, and et al. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[10] III Hal Daumé, John Langford, and et al. Search-based structured prediction as classification.

[11] Kaiming He, Xiangyu Zhang, and et al. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[12] Hakan Inan, Khashayar Khosravi, and et al. Tying word vectors and word classifiers: A loss framework for language modeling. *ICLR*, 2017.

[13] Leslie Pack Kaelbling, Michael L Littman, and et al. Reinforcement learning: A survey. *JAIR*, 4: 237–285, 1996.

[14] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137, 2015.

[15] Sergey Levine, Chelsea Finn, and et al. End-to-end training of deep visuomotor policies. *JMLR*, 17(39):1–40, 2016.

[16] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *ACL*, pages 71–78. Association for Computational Linguistics, 2003.

[17] Siqi Liu, Zhenhai Zhu, and et al. Optimization of image description metrics using policy gradient methods. *CVPR*, 2017.

[18] Francis Maes, Ludovic Denoyer, and et al. Structured prediction with reinforcement learning. *ML*, 77(2):271–301, 2009.

[19] Mitchell P Marcus, Mary Ann Marcinkiewicz, and et al. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[20] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *SLT*, 12:234–239, 2012.

[21] Tomas Mikolov, Martin Karafiát, and et al. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[22] Volodymyr Mnih, Koray Kavukcuoglu, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[23] Andrew Y Ng, Stuart J Russell, and et al. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000.

[24] Kishore Papineni, Salim Roukos, and et al. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. Association for Computational Linguistics, 2002.

[25] Marc'Aurelio Ranzato, Sumit Chopra, and et al. Sequence level training with recurrent neural networks. *ICLR*, 2016.

[26] Steven J Rennie, Etienne Marcheret, and et al. Self-critical sequence training for image captioning. *CVPR*, 2017.

[27] Stéphane Ross, Geoffrey J Gordon, and et al. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, page 6, 2011.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[29] Ilya Sutskever, Oriol Vinyals, and et al. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[31] Richard S Sutton, David A McAllester, and et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, 2000.

[32] Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.

[33] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38 (3):58–68, 1995.

[34] Christopher JCH Watkins and Peter Dayan. Q-learning. *ML*, 8(3-4):279–292, 1992.

[35] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *ML*, 8(3-4):229–256, 1992.

[36] Lantao Yu, Weinan Zhang, and et al. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[37] Wojciech Zaremba, Ilya Sutskever, and et al. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[38] Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *EMNLP*, pages 670–680, 2014.