Sequence Prediction with Unlabeled Data by Reward Function Learning[*]

Lijun Wu[1], **Li Zhao**[2], **Tao Qin**[2], **Jianhuang Lai**[1,3] and **Tie-Yan Liu**[2]
[1]School of Data and Computer Science, Sun Yat-sen University
[2]Microsoft Research Asia
[3]Guangdong Key Laboratory of Information Security Technology
wulijun3@mail2.sysu.edu.cn; {lizo, taoqin, tie-yan.liu}@microsoft.com; stsljh@mail.sysu.edu.cn

Abstract

Reinforcement learning (RL), which has been successfully applied to sequence prediction, introduces *reward* as sequence-level supervision signal to evaluate the quality of a generated sequence. Existing RL approaches use the ground-truth sequence to define reward, which limits the application of RL techniques to labeled data. Since labeled data is usually scarce and/or costly to collect, it is desirable to leverage large-scale unlabeled data. In this paper, we extend existing RL methods for sequence prediction to exploit unlabeled data. We propose to learn the reward function from labeled data and use the predicted reward as *pseudo reward* for unlabeled data so that we can learn from unlabeled data using the pseudo reward. To get good pseudo reward on unlabeled data, we propose a RNN-based reward network with attention mechanism, trained with purposely biased data distribution. Experiments show that the pseudo reward can provide good supervision and guide the learning process on unlabeled data. We observe significant improvements on both neural machine translation and text summarization.

1 Introduction

Sequence prediction, the task of producing a sequence of tokens given an input, attracts more and more attention in research community recently. The input is quite flexible, depending on the applications, such as source language sentences in machine translation, documents in text summarization, or speech segments in speech recognition. It has been shown that recurrent neural networks (RNNs) can deliver excellent performance in many such applications [Sutskever *et al.*, 2014; Bahdanau *et al.*, 2014] when trained to maximize the likelihood of the next output token given the input and previous output tokens.

Recently, reinforcement learning (RL) has been introduced into sequence prediction to address several shortcomings of previous training methods. One main shortcoming is that, the models are optimized to maximize the likelihood of training data, which may not well match the evaluation metrics that actually quantify the prediction quality. These evaluation metrics, such as BLEU score [Papineni *et al.*, 2002] for machine translation and ROUGE score [Lin, 2004] for text summarization, are usually defined at sequence level. These metrics are formulated as *reward* in RL for direct optimization and various RL algorithms such as REINFORCE [Ranzato *et al.*, 2016] and actor-critic [Bahdanau *et al.*, 2016] are used for training.

However, the reward in these RL works is defined on the ground-truth sequence, which limits the approach to labeled data. Labeled data is usually costly to collect and thus limited in quantity and coverage, making it appealing to exploit large-scale unlabeled data. Although unlabeled data is employed in some works to augment sequence prediction, by learning better language model [Dai and Le, 2015], by generating more pseudo labeled data [Sennrich *et al.*, 2016; Zhang and Zong, 2016], and by minimizing the reconstruction error [Cheng *et al.*, 2016], all of the above works train their models through likelihood maximization, thus suffering from the shortcomings we discussed above.

In this paper, we extend the existing RL methods for sequence prediction to further exploit unlabeled data. Since the reward is missing on unlabeled data, we propose to learn the reward function from labeled data, and use the learned reward function as the supervision signal for unlabeled data. Namely, we use the predicted reward from our learned reward function, as *pseudo reward* on unlabeled data. Our reward function takes the source sequence X and an arbitrary target sequence $\hat{Y}$ as input, and try to predict the reward $R(Y, \hat{Y})$, which is defined based on ground-truth Y. The effectiveness of reward function learning is extremely important for our method. The better reward function we learn, the better performance we can expect.

Reward function learning is challenging for sequence prediction due to two reasons: First, we need to predict reward only based on $(X, \hat{Y})$, while the true reward is computed based on $(Y, \hat{Y})$. Second, the space of all possible $\hat{Y}$'s is extremely large, and the reward $R(Y, \hat{Y})$ is zero for most $\hat{Y}$'s, making it easy to predict 0 for all $\hat{Y}$'s and ignore those important $\hat{Y}$'s with non-zero reward.

For the first challenge, we show it is solvable to predict re-

[*]This work was conducted at Microsoft Research Asia.

ward from $(X, \hat{Y})$. We propose a RNN-based reward network with attention mechanism. The reward function can be seen as capturing the correspondence between source sequence X and predicted target sequence $\hat{Y}$, measuring to what extent $\hat{Y}$ matches X. To the best of our knowledge, we are the first to learn a reward function that takes $(X, \hat{Y})$ as input for sequence prediction.

To tackle the second challenge, we purposely bias the training data distribution. Inspired by the REINFORCE algorithm, we show that it is unnecessary to predict reward precisely on all $\hat{Y}$'s. Instead, we train the reward network with respect to the data distribution $P(X, \hat{Y})$ generated by the current sequence predictor, which is referred as *policy* using the language of RL. As the policy changes, we change the distribution of training data correspondingly. In order to do this, we couple the policy training process with the reward function learning process. The policy function learns from the reward signal provided by the reward function, while the reward function adapts to the data distribution generated by the continuously changing policy.

The contributions of this paper are summarized as follows:

- We propose to leverage unlabeled data to augment existing RL methods for sequence prediction. Since reward is missing on unlabeled data, we learn the reward function on labeled data and use the predicted reward as pseudo reward on unlabeled data.
- In order to learn the reward function, we propose a RNN-based reward network with attention mechanism, trained with purposely biased data distribution.
- Experiments on machine translation and text summarization show that our method can leverage unlabeled data effectively, and achieve better performance than existing methods.

2 Background

In this section, we introduce background works: 1) how RNN is used for sequence prediction and trained by maximum likelihood estimation; 2) how RL methodology is introduced to directly optimize evaluation metrics of sequence prediction tasks. We build our work based on those works.

2.1 Sequence Prediction

In the sequence prediction task, we aim to learn a parameterized model $p_\phi(y_t|Y_{1...t-1}, X)$, which models the probability of next token y_t conditioned on input X and previous tokens $Y_{1...t-1}$. With such a model, we try to predict y_t with the highest conditional probability at each time step, and thus generate the tokens one by one to produce the target sequence Y. A good model $p_\phi(y_t|Y_{1...t-1}, X)$ can generate target sequence Y with high probability.

To learn a good model for the conditional probability, one should first carefully design its representation and then select the training principle. Existing works adopt Recurrent Neural Networks to represent the parameterized model $p_\phi(y_t|Y_{1...t-1}, X)$, and maximum likelihood estimation as training principle, which are introduced briefly as follows.

Recurrent Neural Network Recurrent Neural Networks (RNNs) are good candidates to model the probability $p_\phi(y_t|Y_{1...t-1}, X)$ with variable length of input. The key component of RNN is state vector s_t, which compresses the historical information of $Y_{1...t}$ and X. The state vector s_t, as shown in the following equation, is updated in a *recurrent* way, based on the last output token y_t, the context vector c_{t-1} which represents information from input X, and the last state vector s_{t-1}.

$$s_t = f(s_{t-1}, c_{t-1}, y_t), \tag{1}$$

where function f is the recurrent unit such as Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] unit or Gated Recurrent Unit (GRU) [Cho *et al.*, 2014]. Based on the state vector and the context vector, the probability of next token is modeled as follows:

$$p_\phi(y_t|Y_{1...t-1}, X) = g(s_{t-1}, c_{t-1}, y_t) \tag{2}$$

where g is a stochastic output layer (typically a softmax for discrete outputs). The context vector c_t can be defined in several ways. [Bahdanau *et al.*, 2014] proposes an attention mechanism to compute the context vector. Given a variable length representation $(h_1, ..., h_L)$ for the input X, they compute an attention weight α which determines the relative importance of each vector h_j at each time step:

$$\begin{aligned} \alpha_t &= \beta(s_t, (h_1, ..., h_L)), \\ c_t &= \sum_{j=1}^{L} \alpha_{t,j} h_j, \end{aligned} \tag{3}$$

where β is the attention mechanism that produces the attention weight α and c_t is the context vector for time step t. The attention weights are computed by a multi-layer perceptron (MLP) that takes as input the RNN state and each individual vector to focus on. The weights are typically constrained to be positive and sum to 1 by using the softmax function.

Maximum Likelihood Estimation Given labeled data (X, Y), one popular approach to train a good model is to ask the model output Y with high probability given input X, which is usually referred as maximum likelihood estimation (MLE). Given labeled data set $\{X^{(i)}, Y^{(i)}\}_{i\in\{1,...,N\}}$, the MLE objective function is

$$\mathcal{O}_{ML}(\phi) = \sum_{i=1}^{N} log P_\phi(Y^{(i)}|X^{(i)}) \tag{4}$$

where the probability can be expanded as $P_\phi(Y|X) = p(y_1|X)p(y_2|y_1, X)...p(y_T|Y_{1...T-1}, X)p(\varnothing|Y_{1...T}, X)$, and $\varnothing$ is a special end-of-sequence token.

2.2 RL for Sequence Prediction

As described above, the model is learned by maximizing the likelihood of labeled data, $P(Y|X)$, at training time. However, at test time the model is evaluated by a task-specific metric. That is, the training does not match evaluation. A more straightforward approach is to directly optimize the evaluation metric, and thus sequence prediction has been formulated as a RL problem [Bahdanau *et al.*, 2016] as follows.

The RNN model is an agent, which interacts with the environment to get reward. The parameters of this agent define a policy, whose execution results in the agent picking an action a. In this case, an action refers to generating the next token at each time step. After taking each action, the agent updates its state (the internal hidden state of RNN). A terminal reward is received once the agent finishes generating sequence. Any task-specific evaluation metric can be used as reward. In this paper, we use BLEU score for machine translation, and ROUGE score for text summarization, which are the metrics we use at test time. The reward is usually defined by generated $\hat{Y}$ and ground-truth Y, as $R(\hat{Y}, Y)$. The goal of the training is to maximize the expected total reward as follows:

$$\mathcal{O}_{RL}(\phi) = \sum_{i=1}^{N} \sum_{\hat{Y} \in \mathcal{Y}} P_{\phi}(\hat{Y}|X^{(i)}) R(\hat{Y}, Y^{(i)}) \tag{5}$$

The REINFORCE algorithm [Williams, 1992] is applied here. As discussed in [Ranzato *et al.*, 2016], the large action space here (since the vocabulary is large) makes it extremely difficult to learn with an initial random policy. In order to speed up training, two tricks are used in [Bahdanau *et al.*, 2016]: 1) they start with a policy pre-trained by maximizing log-likelihood, rather than a random policy; 2) they use reward shaping to receive intermediate reward at each step, the shaped reward are defined as $r_t(\hat{y}_t; \hat{Y}_{1...t-1}, Y) = R(\hat{Y}_{1...t}, Y) - R(\hat{Y}_{1...t-1}, Y)$. It is worth noting that the space for all possible $\hat{Y}$'s is extremely large, making it impossible to calculate the objective exactly in Eq. 5. In the REINFORCE algorithm, the objective is computed approximately by $\hat{Y}$ sampled from policy p_{ϕ}.

3 Reward Function Learning

In order to enhance the existing REINFORCE algorithm and leverage unlabeled data, we need to learn a reward function $R_{\theta}(X, \hat{Y})$ to give pseudo reward to unlabeled data. With the pseudo reward, we can learn from unlabeled data in a way similar to labeled data in the REINFORCE algorithm.

Reward function learning is challenging for sequence prediction. The first challenge lies in representation, where we need to predict the reward $R(\hat{Y}, Y)$ with a parameterized function taking $(X, \hat{Y})$ as input rather than $(\hat{Y}, Y)$. The second challenge comes from unbalanced data distribution. Since the space of all possible $\hat{Y}$'s is extremely large and $R(\hat{Y}, Y)$ is zero for most $\hat{Y}$'s, if we target on all possible $\hat{Y}$'s, the objective function will drive our model to predict 0 for all $\hat{Y}$'s and ignore those important $\hat{Y}$'s with non-zero reward.

In this section, we introduce our solutions to the challenges and propose an algorithm for reward function learning. First, we design a RNN-based reward network which takes $(X, \hat{Y})$ as input.[1] Second, we specially construct a biased data distribution for training to tackle the second challenge. Finally, we put everything together and give a detailed REINFORCE algorithm for sequence prediction with unlabeled data.

[1]Although we do not specially design the network for the first challenge, our experiments show that the RNN-based reward network with $(X, \hat{Y})$ as input works reasonably well.

3.1 Reward Network

As reviewed in the previous section, in the REINFORCE algorithm, after taking each action (generating one token), the agent receives an immediate reward, which will be used to update its policy. For labeled data, the reward is available, which is computed by comparing the generated tokens with the ground-truth output. However, the ground-truth output for unlabeled data is missing. Thus, we need to learn a reward function g_{θ}, which provides a pseudo reward at each time step t, as an approximation for true reward $r_t(\hat{y}_t; \hat{Y}_{1...t-1}, Y)$.

This is again a sequence modeling task. Given input $(X, \hat{Y})$, the target is to generate a sequence of immediate rewards. At time step t, we only know X and $\hat{Y}_{1...t}$, which means that our parameterized reward function g_{θ} should take the form $g_{\theta}(\hat{y}_t; \hat{Y}_{1...t-1}, X)$. Clearly, the number of inputs of this function changes over time. A natural choice to represent functions with variable length of inputs is recurrent neural networks (RNNs), which compress the input sequence $\hat{Y}_{1...t}$ into a hidden state s_t, and predict the reward r_t based on both context information from X and current hidden state s_t. Mathematically,

$$r_t = h(s_t, c_t), \tag{6}$$

where current hidden state s_t and context vector c_t are computed in the same way as in Section 2.1.

Let θ denote all the parameters in the RNN-based reward network. Now we have a RNN-based parameterized model $g_{\theta}(\hat{y}_t; \hat{Y}_{1...t-1}, X)$, to represent the reward for every time step t in $\hat{Y}$.

3.2 Training Data Distribution

One common way to learn a good reward function is to minimize the Mean Square Error (MSE) on labeled data:

$$\begin{aligned}\mathcal{O}_{MSE}(\theta) = \sum_{i=1}^{N} \sum_{\hat{Y}} \sum_{t=1}^{T} \{ & g_{\theta}(\hat{y}_t; \hat{Y}_{1...t-1}, X^{(i)}) \\ & - r_t(\hat{y}_t; \hat{Y}_{1...t-1}, Y^{(i)})\}^2.\end{aligned} \tag{7}$$

As aforementioned, $\hat{Y}$'s with non-zero rewards are only the minority among all possible $\hat{Y}$'s. The MSE objective will drive g_{θ} to give zero prediction everywhere if we compute the objective on all possible $\hat{Y}$'s. Fortunately, the REINFORCE algorithm does not touch every possible $\hat{Y}$. Instead, it only needs to use the rewards on those $\hat{Y}$'s generated by the policy. A pre-trained policy is usually good enough, and its generated $\hat{Y}$'s are often of high quality. In other words, the $\hat{Y}$'s generated by a pre-trained policy (through beam search) usually have non-zero rewards. As a result, we only need to focus on those $\hat{Y}$'s, which makes reward function learning much easier.

We first generate training data $(X, Y, \hat{Y})$ based on a pre-trained policy, and pre-train our reward network with the generated data. Since the policy changes during training, the data distribution generated by the policy changes accordingly. In order to adapt to this data distribution shifting, we then couple the reward function learning process with the REINFORCE training process. The policy function learns from the reward signal provided by the reward function, while the reward function adapts to the data distribution generated by the continuously changing policy.

Now we give some intuition about the training process. The generated $\hat{Y}$ is somehow what the policy thinks as a good prediction. The reward for $\hat{Y}$, somehow as a rater, further criticizes where the policy did well, where the policy did poorly, reflected by the predicted reward at each step. Namely, we learn to criticize the existing policy. Good criticism serves as weak supervision, and guides the current policy move to the right direction. By biasing and changing the training distribution, intuitively we do not learn the standard reward function, but actually learn how to criticize the current policy.

Algorithm 1: REINFORCE Training for Sequence Prediction with Unlabeled Data

```
Require: A policy $p_\phi(a|\hat{Y}_{1...t}, X)$ and a reward function
    $g_\theta(a; \hat{Y}_{1...t}, X)$ with weights $\phi$ and $\theta$ respectively;
    Labeled data set $\{X^{(i)}, Y^{(i)}\}_{i\in\{1,...,N\}}$; Unlabeled data
    set $\{X^{(i)}\}_{i\in\{N+1,...,N+M\}}$.
 1: Initialize delayed policy $p'_{\phi'}$ and delayed reward
    function $g'$ with same weight: $\phi' = \phi$, $\theta' = \theta$
 2: while Not Converged do
 3:     Randomly receive a labeled or unlabeled data
 4:     Generate a sequence of actions $\hat{Y}$ from $p'$
 5:     if the received data is labeled data $(X, Y)$ then
 6:         compute shaped reward with ground-truth for all $t$
            $r_t(\hat{y}_t; \hat{Y}_{1...t-1}) = R(\hat{Y}_{1...t}, Y) - R(\hat{Y}_{1...t-1}, Y)$
 7:         Update reward function weights $\theta$ using the
            gradient for all $t$
            $\frac{d}{d\theta}(g_\theta(\hat{y}_t; \hat{Y}_{1...t-1}, X) - r_t(\hat{y}_t; \hat{Y}_{1...t-1}, Y))^2$
 8:     else
 9:         compute shaped reward with reward function for all
            $t$
            $r_t(\hat{y}_t; \hat{Y}_{1...t-1}) = \alpha g'(\hat{y}_t; \hat{Y}_{1...t-1}, X)$
10:     end if
11:     Compute value function $V_t(\hat{y}_t; \hat{Y}_{1...t-1})$ for all $t$
        $V_t(\hat{y}_t; \hat{Y}_{1...t-1}) = \sum_{\tau=t}^{T} r_t(\hat{y}_\tau; \hat{Y}_{1...\tau-1})$
12:     Update policy weights $\phi$ using the following gradient
        estimate
        $\sum_{t=1}^{T} \frac{d \log p(a=\hat{y}_t|\hat{Y}_{1...t-1}, X)}{d\phi}(V_t(\hat{y}_t; \hat{Y}_{1...t-1}) - b_t(X))$
13:     Update delayed policy and reward, with a constant $\gamma$
        $\phi' = \gamma\phi + (1-\gamma)\phi'$, $\theta' = \gamma\theta + (1-\gamma)\theta'$
14: end while
```

3.3 Overall Algorithm

With learned reward function, now we can extend the REINFORCE training algorithm for sequence prediction with unlabeled data. The training details are shown in Algorithm 1 and 2. Note that $b_t(X)$ is called *control variance* in [Bahdanau *et al.*, 2016]. We start with a pre-trained policy and a pre-trained reward function. During training, true reward is used for labeled data and predicted reward is used for unlabeled data. In order to control the relative importance of unlabeled data, we rescale the predicted reward with hyper parameter α. As the policy changes slowly, we go on training the reward function with data generated by current policy. We also apply deep RL techniques [Bahdanau *et al.*, 2016] here, by adopting delayed policy and delayed reward function, with the purpose to prevent divergence.

Some semi-supervised learning methods [Zhang and Zong, 2016] generate $\hat{Y}$ for an unlabeled X and treat $(X, \hat{Y})$ pair as pseudo labeled data to enlarge training data. Those methods have no mechanism to control the quality of generated pseudo pairs. Compared with these methods, the reward function in our method provides quality feedback about a pseudo pair $(X, \hat{Y})$. As a result, our method can leverage and learn from unlabeled data in a more precise way.

Algorithm 2: Complete Algorithm for Sequence Prediction with Unlabeled Data

```
1: Initialize policy function $p_\phi(a|\hat{Y}_{1...t}, X)$and reward
   function $g_\theta(a; \hat{Y}_{1,...,t}, X)$ with random weights $\phi$ and
   $\theta$ respectively.
2: Pre-train the policy to predict $y_{t+1}$ given $Y_{1...t}$ and $X$
   by maximizing $\log p_\phi(y_{t+1}|Y_{1...t}, X)$.
3: Run Algorithm 1 to pre-train reward function with
   fixed policy.
4: Run Algorithm 1.
```

4 Experiments

To valid our approach, we performed experiments on two sequence prediction tasks, machine translation and text summarization. In addition to MLE and REINFORCE with labeled data only, we also implement a semi-supervised learning baseline to test on machine translation task.

4.1 Experimental Settings

Machine Translation For the machine translation experiment, we use data from the German-English machine translation track of the IWSLT 2014 evaluation campaign [Cettolo *et al.*, 2014], as used in [Ranzato *et al.*, 2016] and [Bahdanau *et al.*, 2016]. For a fair comparison, we exactly follow the preprocessing described in above two works. The data consists of training/dev/test corpus with 153326, 6969 and 6750 sentence pairs respectively. The English vocabulary has 22822 words, while the German has 32009 words, all other words were replaced with a special token. The maximum sentence length in our data set is 50. For unlabeled data, we only take source sentences from IWSLT 2015 evaluation campaign [Cettolo *et al.*, 2015] data set, the number of unlabeled data is 153326, equal to that of labeled data.

Our policy is a GRU with 256 hidden units, and we use the same convolutional attentive encoder architecture as [Ranzato *et al.*, 2016] and [Bahdanau *et al.*, 2016] to make our results more comparable. The reward network is similar to the policy function except the concatenation of decoder hidden state and context vector are fed into another multilayer perceptron

Model	Greedy	Beam search
LL∗ [Ranzato *et al.*, 2016]	17.74	20.3
MIXER [Ranzato *et al.*, 2016]	20.73	21.8
LL [Bahdanau *et al.*, 2016]	19.33	21.46
RF [Bahdanau *et al.*, 2016]	20.92	21.35
Semi-supervised baseline	20.10	21.65
Our work	**21.64**	**22.35**

Table 1: BLEU scores on German-English translation test set. LL, RF stands for log-likelihood, REINFORCE. The asterisk identifies results from [Ranzato *et al.*, 2016]. LL and LL∗ both denote maximum log-likelihood training. LL is implemented by Blocks [Van Merriënboer *et al.*, 2015] while LL∗ is implemented by Torch [Collobert *et al.*, 2011]. Our work is significantly better than other models ($\rho < 0.01$).

(MLP) with 256 hidden units, and finally output a real number as predicted reward. Hyper parameter α is 0.5, and the delay constant γ is 0.1.

Text Summarization The data set we use to train and evaluate our model on text summarization is from a subset of Gigaword Corpus [Graff and Cieri, 2003]. We use the first sentence of a new article as source input, and the headline as target output, as described in [Rush *et al.*, 2015]. We preprocess the data in the same way as above work. After preprocessing there are 42212 words in the source dictionary and 19014 words in the target dictionary. The number of sentence pairs in the training set, validation set and test set are 189295, 18475 and 10000 respectively [2]. For unlabeled data, we use another subset of 189295 training data and only take the source article.

The policy and reward network are same as used in machine translation except the dimension of embedding and hidden state are reduced to 128. The hyper parameters are same as in machine translation.

4.2 Experimental Results

We compute BLEU score on the machine translation task and ROUGE score on text summarization task. For decoding we use greedy search and beam search with a beam size of 10.

Machine translation results are reported in Table 1. We first compare our work with previous models, the MIXER in [Ranzato *et al.*, 2016] and REINFORCE in [Bahdanau *et al.*, 2016]. Results are reported from the best setting in the corresponding paper. The MIXER in [Ranzato *et al.*, 2016] combines REINFORCE and MLE for curriculum training. The RF in [Bahdanau *et al.*, 2016] is exactly the same as our algorithm without unlabeled data. Our model has an improvement of 0.7/0.9 BLEU points when using greedy search, and an improvement of 1.0/0.5 BLEU points when using beam search. We also implement a semi-supervised baseline model with MLE training [Zhang and Zong, 2016]. The pre-trained policy generates translations $\hat{Y}$ for unlabeled data X, then we put $(X, \hat{Y})$ with labeled data together for better training.

[2]Note the data set is different from [Ranzato *et al.*, 2016] because their data is not released, but we pre-process the data in a same way.

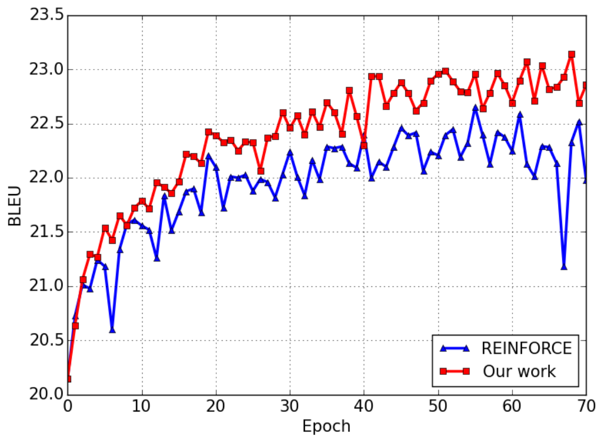

Figure 1: Progress of REINFORCE and our method with BLEU score on the validation set. The curves start from the epoch of log-likelihood pretraining from which the parameters were initialized.

Model	Greedy	Beam search
Log-Likelihood	8.85	10.22
REINFORCE	12.15	12.87
Our work	**12.89**	**13.21**

Table 2: ROUGE-2 scores compared to REINFORCE on text summarization test set. Our work is significantly better than Log-Likelihood and RFINFORCE ($\rho < 0.01$).

α	Test BLEU	Valid BLEU
0	20.92	22.6
0.5	21.64	23.1
1	21.41	22.9

Table 3: BLEU scores for different α in our work on German-English translation test set and validation set.

Semi-supervised model has a significant improvement over log-likelihood baseline, 20.10 compared to 17.74/19.33, and our model has additional 1.5 BLEU points improvement.

Moreover, Figure 1 shows that our work has better progress on the validation set. The BLEU curve is clearly stronger than REINFORCE without unlabeled data.

Text summarization results are reported in Table 2. We compare to [Bahdanau *et al.*, 2016] [3] and observe an improvement of 0.7 ROUGE points.

Impact of Hyper Parameter We try different hyper parameters by setting $\alpha = \{0, 0.5, 1\}$, to see how it impacts the performance of our method on machine translation task. The BLEU scores of greedy search on test and validation set are reported in Table 3. We find that non-zero α achieves similar superior performance compared with zero α, which demonstrates the effectiveness and robustness of our method.

[3]We use open source code from [Bahdanau *et al.*, 2016]: https://github.com/rizar/actor-critic-public, with BLEU metric replaced by ROUGE.

reward setting	Test	Validation
REINFORCE	20.92	22.6
Random reward	19.44	21.4
Our reward function	21.64	23.1
True reward	22.10	23.9

Table 4: BLEU scores under different settings of pseudo reward for unlabeled data on German-English translation task.

The Effectiveness of Learned Reward Function Finally, we want to demonstrate the effectiveness of our learned reward function, by replacing the pseudo reward in our method. To be specific, we test our method on machine translation task by setting the pseudo reward for unlabeled data as: 1. random reward, 2. our reward function (as we used in previous experiment), 3. true reward (computed by using the target sentence for unlabeled data). We want to see how good supervision can our reward function provide, with comparison to random reward and true reward.

The results are reported in Table 4. Random reward gives bad supervision signal and misleads the learning process, and thus hurts the performance. Our reward function gives a fairly good supervision signal, with a performance not too far away from that of the true reward, which demonstrates the effectiveness of our learned reward function.

5 Related Work

Our work is related to three different research topics: reinforcement learning for sequence prediction, semi-supervised sequence prediction, and inverse reinforcement learning.

Reinforcement Learning for Sequence Prediction RL is introduced to sequence prediction aiming at addressing the problem of the gap between the training objective and the final evaluation metric. [Daumé *et al.*, 2009] is the first to formulate structure prediction problem as a special case of reinforcement learning. [Ranzato *et al.*, 2016] proposes to leverage RL to directly optimize the sequence-level evaluation metric, such as BLEU or ROUGE. A terminal reward is received when the policy finishes generating sequence. [Shen *et al.*, 2016] propose the minimum risk training to minimize the expected loss on the training data, which is in the same spirit as RL formulation to directly optimize the evaluation metric. [Bahdanau *et al.*, 2016] argues that immediate reward is good for faster convergence, and makes reward signal less sparse. They also propose to apply actor-critic for less variance in value function estimation. Although they learn a value function in their model which is somewhat analogous to our reward function, their value function takes $(Y, \hat{Y})$ as input which makes their function much easier to learn. While our function takes $(X, \hat{Y})$ as input, and thus has the ability to learn with unlabeled data. The following two statements distinguish our method from all above methods: 1. None of these methods employ unlabeled data. 2. We are the first to learn the reward function taking $(X, \hat{Y})$ as input in sequence prediction.

Semi-supervised Sequence Prediction Since it is cost to collect labeled data but cheap to collect unlabeled data, there are many works focusing on semi-supervised sequence prediction. These methods can roughly be divided into three categories: 1. multi-task learning by sharing parameter/model component [Dai and Le, 2015]; 2. generating pseudo labeled data [Sennrich *et al.*, 2016; Zhang and Zong, 2016]; 3. reconstruction on unlabeled data [Cheng *et al.*, 2016]. Under the view of semi-supervised sequence prediction, our model can be seen as generating pseudo labeled data $(X, \hat{Y})$ and learning to evaluate to what extent should we trust this pseudo data, namely, the predicted reward in our RL setting. We are the first to introduce this novel way for semi-supervised sequence prediction under the RL framework.

Inverse Reinforcement Learning Inverse Reinforcement Learning (IRL) [Ng and Russell, 2000] is the problem of learning a reward function given observed optimal behaviour. In IRL, we have no access to the reward signal. While in our case, the reward signal is defined on labeled data, but is missing on unlabeled data. Due to the access to reward on labeled data, we can learn the reward function directly by function approximation, rather than under the IRL framework.

Another closely related work is dual learning [He *et al.*, 2016], which also leverages unlabeled data and has the reward definition. In their work, the reward is defined on unlabeled data heuristically. While in our case, we learn a reward function for $(X, \hat{Y})$ explicitly, by approximating reward function with true reward on labeled data. Another difference is that dual learning takes a pair of dual sequence predictors, which is not necessary in our work.

6 Conclusion

In this paper, we have proposed to learn reward function for sequence prediction, in order to extend existing REINFORCE training to unlabeled data. Reward function learning is challenging for sequence prediction due to the missing of ground-truth Y and the sparsity of non-zero $\hat{Y}$'s. We have addressed the second challenge through the purposely designed training data distribution, and found that RNN reward network works quite well for the first challenge, although it is not specially designed for the challenge. Experiments have demonstrated the effectiveness of our approach for neural machine translation and text summarization.

For future work, we plan to apply our method to more sequence prediction applications, such as image caption and speech recognition. We will investigate the first challenge and study how to better handle it. Furthermore, we will try various methods to learn better reward functions. For example, one candidate method is to leverage prior knowledge such as dictionary while learning reward function for machine translation. Another promising research direction is to predict reward with confidence, rather than just a real number. Pseudo reward with confidence interval can better guide the utilization of unlabeled data in training.

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2014.

[Bahdanau *et al.*, 2016] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

[Cettolo *et al.*, 2014] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, 2014.

[Cettolo *et al.*, 2015] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. The iwslt 2015 evaluation campaign. *Proc. of IWSLT, Da Nang, Vietnam*, 2015.

[Cheng *et al.*, 2016] Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Collobert *et al.*, 2011] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[Dai and Le, 2015] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015.

[Daumé *et al.*, 2009] Hal Daumé, Iii, John Langford, and Daniel Marcu. Search-based structured prediction. *Mach. Learn.*, 75(3), June 2009.

[Graff and Cieri, 2003] David Graff and C Cieri. English gigaword, linguistic data consortium, 2003.

[He *et al.*, 2016] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8), November 1997.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

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

[Rush *et al.*, 2015] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[Sennrich *et al.*, 2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Shen *et al.*, 2016] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, Cambridge, MA, USA, 2014. MIT Press.

[Van Merriënboer *et al.*, 2015] Bart Van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*, 2015.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4), May 1992.

[Zhang and Zong, 2016] Jiajun Zhang and Chengqing Zong. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, November 2016. Association for Computational Linguistics.