

A New Concept of Deep Reinforcement Learning based Augmented General Sequence Tagging System

Yu Wang

Samsung Research America

Abhishek Patel

Samsung Research America

Hongxia Jin

Samsung Research America

Abstract

In this paper, a new deep reinforcement learning based augmented general sequence tagging system is proposed. The new system contains two parts: a deep neural network (DNN) based sequence tagging model and a deep reinforcement learning (DRL) based augmented tagger. The augmented tagger helps improve system performance by modeling the data with minority tags. The new system is evaluated on SLU and NLU sequence tagging tasks using ATIS and CoNLL-2003 benchmark datasets, to demonstrate the new system's outstanding performance on general tagging tasks. Evaluated by F1 scores, it shows that the new system outperforms the current state-of-the-art model on ATIS dataset by 1.9 % and that on CoNLL-2003 dataset by 1.4 %.

1 Introduction

Sequence tagging/labeling is one of the general techniques required for implementing different natural/spoken language understanding (NLU/SLU) tasks, such as: Named-entity recognition (NER), Part-of-speech (POS) Tagging in NLU, or Slot filling in SLU. Though the purpose of the tasks and their sequence labels are different, most of the state-of-the-art results for these tasks are generated using neural network based architectures. For example, Ma et al. demonstrates that their BLSTM-CNN-CRF model (Ma and Hovy, 2016) can achieve state-of-the-art performance for the NER task (without using extra features) on CoNLL-2003 dataset; Liu et al. gives an attention-based bi-directional LSTM model (Liu and Lane, 2016) for the slot-filling task, which also gives the best performance on the popular ATIS dataset (Hemphill et al., 1990) for SLU tasks (This result is obtained by initial paper submission, a better result is published later by Yu et al. (Wang et al., 2018)). The details of these related works will be given in next background section. The solution for general sequence tagging tasks (like NER/Slot filling) should mainly contain two properties: the first one is that the model can handle the language sequence in a word/token level manner, *i.e.*, it can read in the sequence word by word and outputs their corresponding labels; the second property is that the model should capture the contextual information/constraints which helps understand the word/token's tags, either as named-entity or simply slot tags. In (Lample et al., 2016; Ma and Hovy, 2016), the authors use a conditional random field (CRF) to assist their basic bidirectional LSTM model to capture the contextual constraints in NER task. Similarly, Liu et al. uses an attention based mechanism to take advantages of the weighted collective information from the hidden states of all words in an utterance.

Due to the imbalance of data distribution under different labels, normally the evaluations for different models will be based on their precision and recall, or simply the F1 scores. It can be observed that, though the neural network based models are becoming more and more complicated, it becomes harder to make further improvement on these tagging tasks by using more advanced network structures. The main reasons are from two aspects:

1. Though the tagging models are designed to be more and more complicated, they also becomes more likely to be over-fitting by using more layers, hidden nodes or advance recurrent neural network structures

This work is licensed under a Creative Commons Attribution 4.0 International License. License details: <http://creativecommons.org/licenses/by/4.0/>

*: The authors' emails are {yu.wang1@samsung.com, abhishek.p@partner.samsung.com, hongxia.jin@samsung.com}.

(like bidirectional LSTM). A model's performance may easily reach a bottom-neck by simply tuning hyper-parameters or changing network structures.

2. Considering the fact that the numbers of words/tokens under some wrongly labeled tags may be quite small, it may not give us more improvement if we still use the same group of training data by simply changing the model. Normally a model will focus on finding the pattern of the data under majority tags instead of minority. Sometimes, even a new model can improve the performance on minority tags' classes, at the cost of degrading the model's performance on the majority tags with more training data, which is a common issue resulting from the imbalanced data distribution under different tags (He and Garcia, 2009; Chawla et al., 2004).

Weighted sampling is one general solution to resolve the imbalanced data issue by giving more weights to minority tags and fewer to the majority ones (He and Garcia, 2009; Sun et al., 2009). The technique, however, gives the issue of distorting the original tagging distribution, which may adversely affect model's performance on majority tags. Similar approach, like adaptive synthetic sampling (He et al., 2008), also suffers from the same problem.

This gives us the dilemma on dealing with imbalanced tagged data (either for slot filling or NER task): On the one hand, it is a non-trivial task to re-organizing/sampling the data under different tags without any distortion of information; on the other hand, the improvement by changing models has an upper bound which is mainly decided by the data pattern itself instead of the model structure. Due to these two reasons, in this paper, a new concept of augmented tagging system based on deep reinforcement learning is proposed to further improve the performances of sequence labeling tasks without sacrificing their original data distributions. This novel system will use a deep reinforcement learning based compensatory model to capture the wrong labeled tags and learn their correct labels, hence it can improve the whole model system performance without affecting the original correctly labeled tags. Detailed experiments will be further conducted on two sequence labeling tasks in the domain of SLU/NLU, as NER and slot filling, using public datasets.

The paper is organized as following: In section 2, a brief overview of two common NLP/SLU tagging tasks, *i.e.* NER and Slot filling, are given. Their current state-of-the-art models will also be explained briefly. Section 3 gives a background of deep reinforcement learning (DRL) first, then illustrates our DRL based augmented general tagging system (DAT) in detail. In section 4, two different tagging tasks on different datasets are given, One is a slot filling task performed on the ATIS benchmark dataset. The other is a named-entity recognition (NER) task, tested on the CoNLL-2003 benchmark dataset. Both experiments will compare our new algorithm with their current best-performed models with state-of-the-art results.

2 Background

In this section, a brief background overview on the models for two sequence labeling tasks will be given. Their state-of-the-art neural network structures will also be discussed for further comparison purpose.

2.1 Slot filling in SLU

Slot filling task is one of the fundamental tasks in spoken language understanding. It sequentially labels the words in an utterance using the pre-defined types of attributes or slot tags. The most straight-forward approach for this task is by using a single recurrent neural network (RNN) to generate sequential tags by reading an input sentence word by word. The current state-of-the-art result on ATIS is an attention based bidirectional LSTM model (Liu and Lane, 2016). A brief overview about this approach is given for a better illustration of our new algorithm. The model introduced in (Liu and Lane, 2016) covers both of the intent detection and slot filling tasks. Considering the focus of this paper, we will only discuss the slot filling part of the model. The structure of the attention based BLSTM model for slot filling task is as shown in Figure 1.

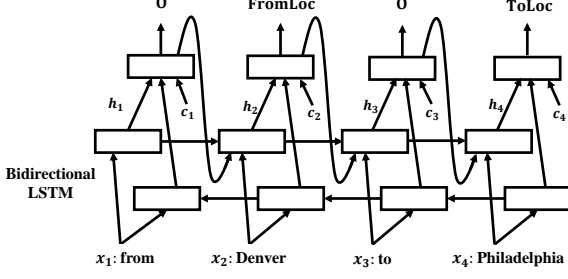


Figure 1: Attention based Bidirectional LSTM for Slot Filling

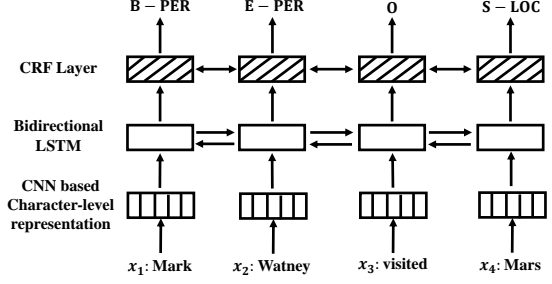


Figure 2: BLSTM-CNN-CRF Structure for Named-entity Recognition

As shown in the figure, a contextual vector $c_i(\cdot)$ is defined using the attention of hidden states h_j :

$$c_i = \sum_{j=1}^L \alpha_{i,j} h_j \quad (1)$$

where $\alpha_{i,j}$ is the attention coefficient defined as:

$$\alpha_{i,j} = \frac{e^{e_{i,j}}}{\sum_{k=1}^L e^{e_{i,k}}} \quad (2)$$

$$e_{i,k} = \phi(s_{i-1}, h_k)$$

where $\phi(\cdot)$ is a feed-forward neural network and s_{i-1} is the decoder hidden state.

This model consists of two main properties:

1. It uses the Bi-direction LSTM (BLSTM) structure to capture the long-term dependencies in both directions.
2. The attention vector c_i gives additional contextual information, for which cannot be captured by the hidden states in BLSTM.

The model gives the state-of-the-art slot filling performance on ATIS dataset. In this paper, we will also use this model as a baseline for comparison purpose. Also, this attention based model is chosen as the DNN part of our new system, *i.e.* f_{dnn} , to generate the filtered input data to the deep reinforcement learning based augmented tagger (DAT) sub-model of the system.

2.2 Named-entity recognition in NLU

Another very common NLU sequence labeling task is named-entity recognition (NER), which seeks to locate and classify entities into some pre-defined named labels. In this section, a brief overview about the neural network based NER model as in (Ma and Hovy, 2016) will be given. The model gives the state-of-the-art performance on CoNLL-2003 dataset (without lexicon features).

The structure contains a convolutional neural network for generating character-level representations, which is used as an input to a bidirectional LSTM network followed by a conditional random field layer. The conditional random field layer is designed to capture the strong dependency across the labels, which can be formulated as:

$$s(X, y) = \sum_{i=0}^n A_{y_i, y_{i+1}} + \sum_{i=0}^n P_{i, y_i} \quad (3)$$

$$p(y|X) = \frac{e^{s(X, y)}}{\sum_{\tilde{y} \in Y_x} e^{s(X, \tilde{y})}}$$

where $s(X, y)$ is the score defined a predicted labeling sequence y given X . A_{y_i, y_j} represents transition score from the tag y_i to tag y_j , and P_{i, y_i} is the probability of the y_i output tag of the i^{th} input word

Table 1: Comparison between Minority Tags and Majority Tags on ATIS and CoNLL-2003

	ATIS			CoNLL-2003		
	Training Data		Test Data	Training Data		Test Data
	# of tag types	# of tokens	% of wrongly labeled data	# of tag types	# of tokens	% of wrongly labeled data
Minority Tags ($< 1\%$ of total # of data)	119	6,323	92%	2	2,312	78%
Majority Tags ($\geq 1\%$ of total # of data)	8	38,707	8%	7	202,255	22%
Total	127	45,030	100%	9	204,567	100%

generated by the bidirectional LSTM. Y_X represents all possible tag sequences for an input sentence X , and $p(y|X)$ is the predicted output y 's tagging probability giving an input words sequence $X = \{x_1, x_2, \dots, x_n\}$.

The model shown above gives the state-of-the-art result on the English NER task with an F1 score of 91.2 on CoNLL-2003 dataset (without extra features).

Though decent experimental results are obtained by using deep learning models on the slot filling and NER tagging tasks, the potentials of these models are still limited by the imbalanced data distribution, since most of the tokens don't have any entity names (labeled as 'O'). The model can easily cover the majority tags but not the minority ones. Table 1 gives a summary of minority tags (each tag has less than 1% of entire data) and majority tags on ATIS and CoNLL-2003 datasets, and their corresponding occurrence ratio among the wrongly labeled data in test dataset.

In Table 1, it shows that about 92% of the wrongly labeled data are from minority tags in ATIS dataset and 78% in CoNLL03 dataset. This gives us an indication that, in order to further improve the model performance, the new model needs to figure out how to better label the minority tags and wrongly labeled tags without sacrificing the model performance on majority tags.

3 Deep Reinforcement Learning

In this section, a novel deep reinforcement learning (DRL) based augmented tagging system is proposed to address the issue as described earlier. The system contains two parts: one is the original deep neural network (NN) based tagger (f_{dnn}) as in section 2.1 for slot filling and section 2.2 for NER, the second part is a deep reinforcement learning (DRL) based augmented tagger (f_{dat}), which can be trained to learn the correct tags that are labeled wrongly by the deep learning based tagger, such that it can compensate the weakness of the original single tagger.

3.1 Reinforcement Learning and Deep Q Network (DQN)

Reinforcement learning (RL) is a bit different from the supervised learning and unsupervised learning algorithm. Formulated as a markov decision process (MDP), an RL based model mainly contains several key elements: state (s_t), action (a_t), rewards (r_t) and policy (π). In a given a state of a stochastic environment, an agent is seeking its best action to perform in order to maximize its expected rewards to be obtained, by following some policy. The main target of a RL based model is to seek the best policy, hence the corresponding action, for an agent to perform. There are mainly three types of reinforcement learning algorithms: value-based, policy gradient and actor-critic. In our scenario, due to the necessity to define rewards at each state and small discrete action space, a value-based model design using Q -learning is proposed for our problem. Its optimal action-value function Q^π , which is also the maximum expected reward obtained by selecting the best policy π , is defined as:

$$Q^\pi(s_t, a_t) = \max_{\pi} \mathbb{E}[R_t | a_t, s_t, \pi] \quad (4)$$

One well-known issue of applying reinforcement learning in real problems is that, in order to compute this optimal action-value function Q^π , it is necessary to store all of the Q values for each state action pair (s_t, a_t) in a table, which is not very practical if the state or action space is large. This problem is described as "the curse of dimensionality" by Bellman (Powell, 2007). One approach to overcome it is

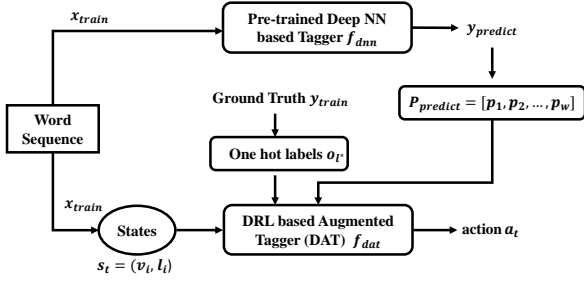


Figure 3: Training Model of DRL based Augmented Tagger

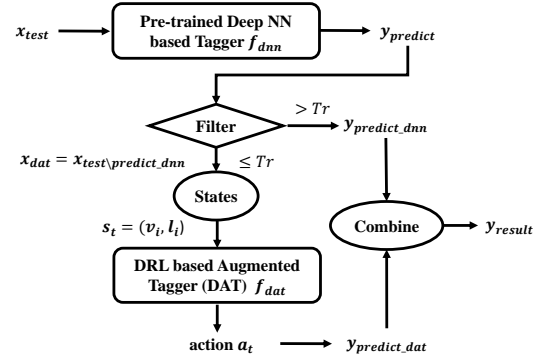


Figure 4: Inference Model of DRL based Augmented Tagger

by training a deep neural network based estimator $Q(s_t, a_t | \theta_t)$ to estimate $Q^\pi(s_t, a_t)$ at time t . However, these types of estimators tend to be unstable since the convergence cannot be fully guaranteed. Recently, the deep Q network (DQN) (Mnih et al., 2013; Mnih et al., 2015) demonstrates much better convergence performance on large state space, by taking advantage of a novel training technique called “experience replay”. In this paper, we will use DQN as our selected deep reinforcement learning structure to satisfy the model requirement due to large state space.

In DQN structure, the optimal action-value function $Q^\pi(s_t, a_t)$ is estimated by a neural network based estimator $Q(s_t, a_t | \theta_t)$, i.e.

$$Q(s_t, a_t | \theta_t) \sim Q^\pi(s_t, a_t) \quad (5)$$

where θ_t is the neural network parameter, and the state s_t serves as the input of the network. In next section we will describe how to use the techniques in DQN to build our augmented tagger from a system perspective. Also, the states, actions and rewards of the DRL based tagger are also going to be carefully designed next.

3.2 DRL based Augmented Tagger (DAT) System

The design of DRL based augmented tagging (DAT) system is as given in Figure 3 and Figure 4. Due to the differences of using augmented tagger in training and inference, we discuss these two parts separately.

3.2.1 Training DAT System

In order to train the entire tagging system, one may need to first pre-train a deep NN based tagger f_{dnn} , this tagger will follow the design as discussed in section 2.1 for slot filling task or section 2.2 for the NER task. Once f_{dnn} is fully trained, the output labels of f_{dnn} using training input x_{train} are stored as $y_{predict}$ with a probability distribution $P_{y_{predict}} = [p_1, p_2, \dots, p_w]$, where w is the total number of tags in the entire training dataset. Following is the DRL based design of our new DAT structure:

States (s_t): The DAT model’s state s_t is shown in Figure 5. The state is defined by each word/token w_i in sentences, it mainly contains two parts: the first part contains the word level information represented by an n -gram (n is odd) averaged vector v_i , and the other part is a given label l_i of w_i . During the generation of training states, l_i uses all possible tags for the word/token w_i . v_i is defined as the average of the vector of word sequences from $w_{i-(n-1)/2}$ to $w_{i+(n-1)/2}$, where w_j is the center of this sequence:

$$v_i = \frac{1}{n} \sum_{j=i-(n-1)/2}^{j=i+(n-1)/2} w_j \quad (6)$$

Remarks: The reason to use an n -gram vector v_i to substitute w_i as a word level information in a state is due to that we want to better capture the contextual information compared to that using a single word vector. When $n = 1$, v_i is the same as word vector w_i . Also, the average of fewer words/tokens can be used if the index of word sequences is out of the boundary of an input sentence.

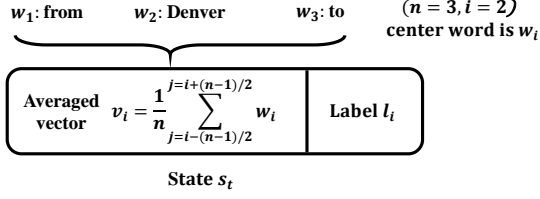


Figure 5: State of DAT

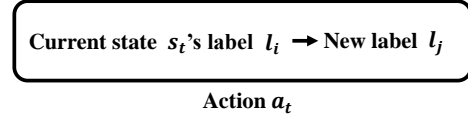


Figure 6: Action of DAT

Actions (a_t): The DAT model's action a_t at time step t is defined as in Figure 6. The action gives a transition signal such that the state will change from its current label l_i to its next label l_j by keep the same n-gram vector v_i . Simply speaking, the action set A contains all possible labels/tags for the word vector w_i or its correspond n-gram substitute v_i , i.e. $A = \{l_1, \dots, l_k\}$. At each time step t , the action a_t with highest predicted action-reward value is chosen as:

$$a_t = \arg \max_a Q(s_t, a | a \in A, \theta_t) \quad (7)$$

Then the state will transit from its current label to the label directed by action a_t .

Rewards (r_t): The reward defined at a state s_t containing an n-gram vector v_i (with a center word/token w_i) will use the one-hot representation $o_{l_i^*}$ of w_i 's true label l_i^* , the one hot vector o_{l_i} of the label l_i in current state s_t , and the predicted probability p_i for the word w_i using f_{dnn} as:

$$r_t = \tanh(\log(\frac{\|o_{l_i^*} - p_i\|_2}{\|o_{l_i} - o_{l_i^*}\|_2 + \epsilon})) \quad (8)$$

where $\tanh(\cdot)$ is used to normalize the reward to be within -1 and 1, $\|\cdot\|_2$ is the \mathcal{L}_2 norm, and ϵ is a very small value added to avoid zero denominator.

The insight behind our reward design is using the ratio of the distance from f_{dnn} predicted label to w_i 's true label and that from current state's label to its true label. Based on our definition, a higher reward will be assigned to a state in which its label is more closer to the true label compared with the f_{dnn} 's predicted one. The reward function is one of the key factors for further improvement using our new augmented tagger.

Remarks: It is worth noticing that p_i should be generated by f_{dnn} using the same word sequence as preparing v_i . Similarly, l_i^* is the true label of w_i in the same sentence of preparing p_i and v_i .

Training: The training algorithm for DAT is as shown in Algorithm 1. The loss function is defined based on the difference of two estimated expected rewards at state s_t using an unsupervised approach as:

$$\begin{aligned} \mathcal{L}_{s_t} &= (\hat{Q}(s_t, a_t | \theta_t) - Q(s_t, a_t | \theta_t))^2 \\ &= (r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} | \theta_t) - Q(s_t, a_t | \theta_t))^2 \end{aligned} \quad (9)$$

where $Q(s_t, a_t | \theta_t)$ is the predicted expected reward using (s_t, a_t) generated by the neural network-based estimator in DAT directly, and $\hat{Q}(s_t, a_t | \theta_t)$ is an estimation of $Q(s_t, a_t | \theta_t)$ using the current state reward r_t and its neighbors' estimations $Q(s_{t+1}, a_{t+1} | \theta_t)$ as in (9).

One training technique we borrowed from (Mnih et al., 2013) is the experience replay used in DQN. It improves the convergence issue in neural network-estimator based Q -learning by storing the state s_t visited before, action performed a_t , state's reward r_t and the next state s_{t+1} after performing the action a_t in an experience tuple (s_t, r_t, a_t, s_{t+1}) . This tuple is then pushed into the experience replay memory queue M . The size of M is pre-defined based on experiment.

Whenever an action is performed then a new state is arrived, the past experience tuple will be pushed into the replay memory queue M if M is not full, otherwise M will pop out the first tuple and push in the latest one as First-in First-out (FIFO). At each training iteration, a random tuple is selected from M and the loss function value is calculated based on the s_t, r_t, a_t and s_{t+1} given by the tuple.

Algorithm 1 DAT Training using Experience Replay

```
1: Epochs:  $N$ 
2: Batch Size:  $K$ 
3: Initialize a replay memory  $M$  with size  $\mu$ 
4: for  $n=1 \rightarrow N$  do
5:   Time step in state space:  $t \leftarrow 0$ 
6:   Randomly select an initial state  $s_t = (v_i, l_i)$ ,  $l_i^* \leftarrow$  true label of  $v_i$ 
7:   while  $l_i \neq l_i^*$  do
8:      $Q(s_t, a_t | \theta_t) \leftarrow \text{DAT}(s_t)$ 
9:      $a_t = \arg \max_a Q(s_t, a | \theta_t)$ 
10:    Perform action  $a_t$ , generating next state  $s_{t+1}$ 
11:    Push tuple  $(s_t, r_t, a_t, s_{t+1})$  into replay memory  $M$ 
12:    for  $b=1 \rightarrow K$  do
13:      Randomly select a tuple  $m$  from  $M$ 
14:       $s_t \leftarrow m(0)$ ,  $r_t \leftarrow m(1)$ ,  $s_{t+1} \leftarrow m(3)$ 
15:       $Q(s_t, a_t | \theta_t) \leftarrow \text{DAT}(s_t)$ 
16:       $Q(s_{t+1}, a_{t+1} | \theta_t) \leftarrow \text{DAT}(s_{t+1})$ 
17:      Obtain  $p_i$  from  $f_{dnn}$  and  $o_{l_i}$  in state  $s_t$ ,  $o_{l_i}^*$  from the ground truth in training data
18:       $r_t \leftarrow \tanh(\log(\frac{\|o_{l_i}^* - p_i\|_2}{\|o_{l_i} - o_{l_i}^*\|_2 + \epsilon}))$ 
19:      Update  $\hat{Q}(s_t, a_t | \theta_t)$  in (9)
20:      Update the network by minimizing loss  $\mathcal{L}_{s_t}$  in (9)
21:    end for
22:     $t \leftarrow t + 1$ 
23:  end while
24: end for
```

Remarks: 1. It is worth noticing that it is necessary to use reinforcement learning (RL) in our problem, since RL based model can learn the long-term dependency much better than the “conventional” deep learning approach. In our case, since we split a sentence into multiple word level vectors and select them randomly during training, not only we need to consider what the states current label is, but also how this state/word level vector is affected by its connected state in the network, such that it wont have any difficulty to find the correct label from its current wrong label.

2. In our model, we use a cascaded multiple model structure to boost the system performance. The concept of using information from multiple models (with the same or different structures) to achieve better performance has been widely used in deep learning ((Dean et al., 2012; Ngiam et al., 2011; Wang et al., 2018)), system identification ((Narendra et al., 2014; Narendra et al., 2015b; Murray-Smith and Johansen, 1997; Narendra et al., 2015c; Wang, 2017a; Wang, 2017b)) and also reinforcement learning field ((Narendra et al., 2015a; Wang and Jin, 2018; Narendra et al., 2016)) recently . Instead of using collective information, in this paper, we takes advantage of two models (DNN-based and DRL-based) to compensate each other, such that minor tags can be taken care of and learned.

3.2.2 Model Inference

As shown in Figure 4, the inference part of the DAT model is a bit different from training the DAT. Since during the training procedure, the entire training dataset (x_{train}, y_{train}) is used to train both DNN and DAT models. Comparatively, during inference, only partial of test data with “unsatisfied performance” will be collected and further evaluated by DAT. This is mainly because that we use DAT as an augmented tagger to compensate the minority cases when the original DNN based tagger f_{dnn} does not perform well. For the majority test data $x_{predict_dnn}$, on which f_{dnn} can perform very well, we still use f_{dnn} to generate their tags $y_{predict_dnn}$. In order to filter those data with “unsatisfied performance”, a threshold value T_r is defined. All the tokens with their predicted tags’ probabilities below T_r are filtered as the

minority cases and further used as the inference input of DAT, *i.e.* $x_{dat} = x_{test \setminus predict_dnn}$. The outputs of DAT are the actions that will transfer the states from their current labels l_i to the target label l_i^* , which gives the output of minority cases, *i.e.* $y_{predict_dat}$.

Remarks: The choice of T_r will slightly affect the performance of DAT as different percentage of data will be filtered. A general recommended T_r is to use a similar percentage of data under minority tags in training data as shown in section 2.2. For example, we use a T_r to filter around 14% (=6,323/45,030) percent of test data in ATIS for DAT since the training tokens under minority tags in ATIS are roughly around this ratio, as shown in Table 1.

4 Experiment

4.1 Data Sets

In our experiment, we will evaluate our deep reinforcement learning based augmented tagging system on two sequence labeling tasks: Slot filling task and NER task.

Slot Filling: For Slot filling task, we use the public ATIS dataset which follows the same format as in (Liu and Lane, 2015; Mesnil et al., 2015; Xu and Sarikaya, 2013; Liu and Lane, 2016). The training set contains 4,978 utterances/sequences, and the test dataset contains 893 utterances. There are totally 127 slot tags.

NER: For NER task, we use the public CoNLL-2003 dataset as in (Lample et al., 2016; Chiu and Nichols, 2016; Ma and Hovy, 2016; Xu et al., 2017; Huang et al., 2015). The training set contains 14,987 sentences, and the test dataset contains 3,684 sentences. The dataset contains four different types of named entities: *PERSON*, *LOCATION*, *ORGANIZATION*, and *MISC*. By using the BIO2 tagging scheme, the total number of different labels is 9. There are a total of 204,567 tokens in training set, and 46,666 tokens in test set.

4.2 Training Setup

Slot Filling: For slot filling task, the pre-trained DNN model f_{dnn} has the same set-up as in (Liu and Lane, 2016), by using an attention based bi-directional LSTM. The number of states in LSTM cell is 128. The randomly initialized word embedding size is also 128. The batch size is 16 and the dropout rate for non-recurrent connection is 0.5.

NER: For NER task, the pre-trained DNN model f_{dnn} follows the BLSTM-CNNS-CRF structure as in (Ma and Hovy, 2016), which gives the current state-of-the-art result on CoNLL-2003 dataset. The word embedding is chosen as the GloVe 100-dimensional embedding (Pennington et al., 2014). The CNN’s window size is chosen as 3 and the number of filters is 30. The number of states in LSTM cell is 200. The batch size is 10 and the dropout rate is 0.5.

The DAT model f_{dat} used for Slot fill and NER tasks are almost the same except the batch size. The network structure chosen to estimate the action-value function Q is an LSTM structure with 100 states. The averaged word vector in a reinforcement learning state is chosen as a trigram, *i.e.* $n=3$. The discount factor γ in (9) is selected as 0.5, 0.7 and 0.9 for difference experiments. The minibatch size is $K = 16$ for slot filling task and $K = 10$ for NER in order to keep the same training batch size as f_{dnn} in different tasks, and the replay memory size is pre-defined as $\mu = 5,000$. The thresholds for both experiment are set as $T_r = 0.95$.

4.3 Performance on ATIS dataset

Our first experiment is performed on ATIS dataset and compared with other existing approaches, by evaluating their slot filling $F1$ scores. A detailed comparison is given in Table 2.

By using the same DNN based model for f_{dnn} as in (Liu and Lane, 2016), our new model surpassed the previous state-of-the-art result by 0.9%, 1.2% and 1.9% for $\gamma=0.5, 0.7$ and 0.9 separately. It is also worth noticing that, a larger discount factor γ gives a better performance as shown in Table. One empirical explanation is that when a larger discount factor is used, the model puts more weights on future states during training, hence it can search the correct label for the current word within a state in a faster manner.

Table 2: Performance of Different Models on ATIS Dataset

Model	F1 Score
Recursive NN (Guo et al., 2014)	93.96%
RNN with Label Sampling (Liu and Lane, 2015)	94.89%
Hybrid RNN (Mesnil et al., 2015)	95.06%
RNN-EM (Peng and Yao, 2015)	95.25%
CNN CRF (Xu and Sarikaya, 2013)	95.35%
Encoder-labeler Deep LSTM (Kurata et al., 2016)	95.66%
Attention Encoder-Decoder NN (Liu and Lane, 2016)	95.87%
Attention BiRNN (Liu and Lane, 2016)	95.98%
DRL based Augmented Tagging System ($\gamma = 0.5$)	96.85%
DRL based Augmented Tagging System ($\gamma = 0.7$)	97.23%
DRL based Augmented Tagging System ($\gamma = 0.9$)	97.86%

Table 3: Performance of Different Models on CoNLL-2003 Dataset

Model	F1 Score
NN+SLL+LM2(Collobert et al., 2011)	88.67%
NN+SLL+LM2+Gazetter* (Collobert et al., 2011)	89.59%
BI-LSTM-CRF* (Huang et al., 2015)	90.10%
ID-CNN-CRF (Strubell et al., 2017)	90.54%
FOFE (Xu et al., 2017)	90.71%
BLSTM-CNN+emb (Chiu and Nichols, 2016)	90.91%
BLSTM-CRF(Lample et al., 2016)	90.94%
BLSTM-CNN-CRFs (Ma and Hovy, 2016)	91.21%
BLSTM-CNN+emb+lex* (Chiu and Nichols, 2016)	91.62%
DRL based Augmented Tagging System ($\gamma = 0.5$)	91.92%
DRL based Augmented Tagging System ($\gamma = 0.7$)	92.23%
DRL based Augmented Tagging System ($\gamma = 0.9$)	92.67%

Table 4: Change on Tags' Distribution of Wrongly Labeled Data on ATIS and CoNLL-2003 Test Datasets

	ATIS Test Dataset		CoNLL-2003 Test Dataset	
	% of wrongly labeled data using f_{dnn}	% of wrongly labeled data using $f_{dnn}+f_{dat}$	% of wrongly labeled data using f_{dnn}	% of wrongly labeled data using $f_{dnn}+f_{dat}$
Minority Tags ($< 1\%$ of total # of data)	92%	56%	78%	48%
Majority Tags ($\geq 1\%$ of total # of data)	8%	44%	22%	52%
Total	100%	100%	100%	100%

4.4 Performance on CoNLL-2003 dataset

Our second experiment is conducted on the CoNLL-2003 dataset and compared with current existing neural network-based approaches for the NER tasks. The metric is also using their F1 scores, and the result is as shown in Table 3 (The results are collected before initial submission, some better results are released during final submission (Peters et al., 2018), despite our model’s performance is still better).

The methods marked using * are using extra features like lexicons and etc. It can be observed that our DAT system’s result outperforms the state-of-the-art results in (Ma and Hovy, 2016) (without lexicon features) and (Chiu and Nichols, 2016) (lexicon features) by 1% and 1.4% using $\gamma = 0.9$ separately. Similar to the SLU task, a larger discount factor γ also boosts our system’s performance.

4.5 Change on Result Distribution

Another observation from our experiment’s result is that the tags’ distribution of wrongly labeled data changed. Table 4 gives a summary about these changes on ATIS and CoNLL-2003 two test dataset.

It can be observed that the percentage of wrongly labeled data with minority tags decreases for both ATIS and CoNLL-2003 datasets by adding the DAT model f_{dat} to the DNN based model f_{dnn} . It indicates that the DAT model f_{dat} can help improve the performance of general sequence tagging model by correctly labeling data with minority tags, which is also the weakness of a single DNN based model f_{dnn} .

5 Conclusion

In this paper, a new DRL based augmented general tagging system is designed. The system use two sequence labeling models: one is a “conventional” DNN based tagger, and the other is a novel DRL based augmented tagger, *i.e.* DAT. By filtering DNN’s output and picking out the “unsatisfied data”, DAT can further improve sequence labeling tasks’ performance by correctly relabeling the data below the threshold T_r , especially for those under minority tags. The experiment results on two sequence labeling tasks, *i.e.* Slot fillings and NER in SLU and NLU, both outperform the current state-of-the-art models. Besides the decent experimental performance obtained, more importantly, the new augmented approach can be generalized to more general sequence labeling models without changing their original model setups.

References

- Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. 2004. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6.
- Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.
- Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 554–559. IEEE.
- Haibo He and Eduardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.
- Haibo He, Yang Bai, Eduardo A Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.
- Charles T Hemphill, John J Godfrey, George R Doddington, et al. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentencelevel information with encoder lstm for natural language understanding. *arXiv preprint*.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.
- Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.
- Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.
- Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(3):530–539.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.
- Roderick Murray-Smith and T Johansen. 1997. *Multiple model approaches to nonlinear modelling and control*. CRC press.
- Kumpati S Narendra, Yu Wang, and Wei Chen. 2014. Stability, robustness, and performance issues in second level adaptation. In *American Control Conference (ACC), 2014*, pages 2377–2382. IEEE.
- Kumpati S Narendra, Snehasis Mukhopadhyay, and Yu Wang. 2015a. Improving the speed of response of learning algorithms using multiple models. *arXiv preprint arXiv:1510.05034*.

- Kumpati S Narendra, Yu Wang, and Wei Chen. 2015b. Extension of second level adaptation using multiple models to siso systems. In *American Control Conference (ACC), 2015*, pages 171–176. IEEE.
- Kumpati S Narendra, Yu Wang, and Wei Chen. 2015c. The rationale for second level adaptation. *arXiv preprint arXiv:1510.04989*.
- Kumpati S Narendra, Yu Wang, and Snehasis Mukhopadhyay. 2016. Fast reinforcement learning using multiple models. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 7183–7188. IEEE.
- Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.
- Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.
- Warren B Powell. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.
- Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.
- Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. 2009. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719.
- Yu Wang and Hongxia Jin. 2018. A boosting-based deep neural networks algorithm for reinforcement learning. In *Proceedings of the 2018 American Control Conference*.
- Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*, volume 2, pages 309–314.
- Yu Wang. 2017a. *Adaptive control and learning using multiple models*. Ph.D. thesis, Yale University.
- Yu Wang. 2017b. A new concept using lstm neural networks for dynamic system identification. In *American Control Conference (ACC), 2017*, pages 5324–5329. IEEE.
- Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE.
- Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1237–1247.