Abstract

In this paper, a deep reinforcement learning (DRL) based multimodal coaching model (DCM) for slot filling task in SLU is proposed. The DCM takes advantage of a DRL based model as a coach of the system to learn the wrong labeled slots with/without user's feedback, hence may further improve the performance of an SLU system. This DCM model is an improved model of the deep reinforcement learning based augmented tagging model as introduced in [1], by using a better DRL model with different rewards and adding in a user's feedback modal to achieve one-shot learning. The performance of DCM is evaluated on two datasets: one is the benchmark ATIS corpus dataset, another is our in-house dataset with three different domains. It shows that the new system gives a better performance than the current state-of-the-art results on ATIS by using DCM. Furthermore, we build a demo application to further explain how user's input can also be used as a real-time coach to improve model's performance even more.

1. Introduction

Slot filling is one of the most important tasks in spoken language understanding (SLU), which is normally formulated as a sequence labeling problem. Some effective algorithms for this task include conditional random fields (CRFs), recurrent neural networks (RNN), or a combination of these models [2, 3, 4, 5]. Recent works also take advantage of more advanced RNN based model like sequence-to-sequence/encoder-decoder structures, which can take the attention based contextual features to further improve the model’s performance [6]. The details of these works will be given in next section.

The performance of different models are normally evaluated by their F1 scores, which is mainly due to the imbalance data distribution in most slot filling tasks. Despite recent models demonstrate better performance, it becomes harder to make further improvement by using more advanced or complex network structures. The main reasons of this issue are from three aspects:

1. Most of the tags in a slot filling task are minority tags, which only counts a small percentage of the entire dataset. The reason is that tokens in word sequences are mostly labeled as 'no semantic tags', i.e. 'O'. Also among the misclassified tokens, we observe that most of them are with minority tags. By changing the learning model structure, it will first improve the model performance on majority tags. Sometimes a model can improve its performance on minority tags, but at a cost of degrading that on the majority tags, which can be treated as a common issue for the imbalanced data [7, 8].

2. Despite the system performance is improved by using more complicated deep learning models, the number of training parameters also increases, which is more likely to be over-fitting [9, 10, 11]. There is a bottleneck by using more complex network structures or tuning hyper-parameters.

3. Another issue need to be noticed is that even one knows a deep learning model does not perform very well on a small portion of data with specific tags, it takes a lot of time to retrain the model using a different set of hyper-parameters or even a new model structure. The robustness of dynamic online learning for most deep learning based models are not very good [12, 13].

One possible solution for the first issue is by weighted sampling the data, i.e. oversampling the minority tags and undersampling the majority ones. However, this method may distort the original distribution of data, hence may degrade model's performance [14, 15]. The second issue is even harder to overcome, as over-fitting is a direct trade-off of using a more complex model, which is the direction most recent works follow. Some techniques, like regularization, dropout or max norm constraints [16, 17, 18], can resolve the over-fitting issue to some extend, but still may not work very well if the model structure becomes even more complex. For the third issue, one possible solution is to save the trained model and the wrongly labeled results, then continue to train the model using more data under the same tags as the mislabeled data. This approach, however, may adversely affect model’s performance on correctly labeled data as well.

In this paper, a DRL based multimodal coaching model (DCM) for slot filling is proposed. The new model will use a DRL based slot filling model as introduced in [1], by adding in user's coaching as another modal, such that the correct tags of the wrongly labeled data can be taught by the users during training and the same mistakes won’t happen again in testing. The advantage of this approach is that it is a compensatory model to make up the mistakes made by the deep learning based sequence labeling models, without changing the original model's structure or re-sampling the data, hence solves the issue caused by the imbalanced data and over-fitting. Also, a new experience replay technique named as partially-fixed experience replay is proposed to achieve a faster on-line teaching without the need of retraining the entire model.

The structure of this paper is organized as following: a brief overview of our baseline RNN model is given in section 2. Section 3 will discuss the new proposed DRL based coaching system, and how it works with existing state-of-the-art RNN based models described in section 2. In section 4, the algorithm will be tested on two datasets, one is the benchmark ATIS dataset and the other is our in-house dataset with three domains. Furthermore, a user based coaching demo is given to illustrate how our model’s performance can be further enhanced by fast online training using user instructions.

2. Baseline Model

In this section, we will give a brief overview of the baseline RNN model used in our system. This model also gives the previous state-of-the-art result on ATIS dataset for slot filling task.

Slot filling task is one of the most important tasks in spo-
ken language understanding. It sequentially labels the words in an utterance using the pre-defined slot labels. The most straightforward approach is to use a single recurrent neural network (RNN) to generate sequential tags of an utterance word by word [2]. This approach has one constraint that the number of tags generated should be the same as that of words in an utterance. One possible approach to overcome this limitation is the encoder-decoder model, in which the number of decoder’s output can be different from that of the encoder’s input [6].

The current state-of-the-art model which has the best performance on ATIS dataset is an attention based bidirectional RNN model as in [6]. A brief discussion about this approach will be given in this section. The model will be used as:

1. One of the baseline models for comparison in experiment section
2. The base RNN model to coarsely process the input word sequence and then generate the filtered training data for the DRL based coaching model (DCM).

The model introduced in [6] covers both of the intent detection and slot filling tasks. Considering the focus of this paper, we only discuss the slot filling part of the model. The structure of the attention based RNN model for slot filling task is as shown in Figure 1, where a bidirectional LSTM is selected as its RNN model structure. As shown in the figure, a contextual vector $c_i(\cdot)$ is defined using the attention of hidden states $h_j$:

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

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

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

$$e_{i,k} = \phi(s_{i-1}, h_k)$$ \hspace{1cm} (2)

where $\phi(\cdot)$ is a feed-forward neural network.

This model consists of two main properties:
1. It uses a 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.

3. Deep Reinforcement Learning based Multimodal Coaching Model (DCM)

In this section, a novel deep reinforcement learning (DRL) based multimodal coaching model is proposed to address the issues as described earlier. The new system is built upon an existing recurrent neural network (RNN) based model $f_{rnn}$ for slot filling. As mentioned earlier, the deep reinforcement learning based multimodal coaching model (DCM) $f_{dcm}$, can be trained either by the given data or coached by users, in order to generate the correct tags labeled wrongly by the $f_{rnn}$ originally. Hence it can compensate the weakness of the original RNN based model.

3.1. Training Model Design

In this subsection, a detailed analysis of the design of the DRL based multimodal coaching model (DCM) for slot tagging is given. Following the definitions of reinforcement learning (RL) as in [19, 20, 21, 22], we will show the design of three main RL components, i.e. the states $s_t$, the actions $a_t$ and the rewards $r_t$. Also a brief discussion on our DRL based training algorithm using experience replay is given. Before the discussion, the entire training structure of DCM is as shown in Figure (2), which contains several steps:

Step 1: Pre-train an RNN based tagging model $f_{rnn}$ using the training data $(x_{train}, y_{train})$.

Step 2: Use the pre-trained RNN model $f_{rnn}$ to generate the predicted outputs $y_{predict}$ and compare it with the ground truth $y_{train}$. The corrected tagged data pairs are labeled as $(x_{rnn}, y_{rnn})$.

Step 3: Filter out the correctly labeled data and leave the ones with wrong labels, i.e. $x_{train} \setminus x_{rnn}$ with their correct labels $l_{i}$ in $y_{train} \setminus y_{rnn}$ to generate the state $s_t$ (as described next) for training the coaching model $f_{dcm}$.

The DRL based structure’s design follows the model set-up as in [1], which contains several elements, including states, actions, rewards and etc. A brief overview is given as following:

States ($s_t$): The DCM model’s state $s_t$ is as shown in Figure 3. The state is defined by each word/token $w_t$ from its input $w_t \in x_{train} \setminus x_{rnn}$, it mainly contains two parts of information: the first part is the word level information represented by an $n$-gram ($n$ is odd) averaged vector $v_t$, and the other part is a given label $l_t$ of $w_t$, i.e. $s_t = [v_t, l_t]$. During the generation of training states, $l_t$ uses all possible tags for the word/token $w_t$. $l_t$ is defined as the average of the vector of word sequences from $w_{t-(n-1)/2}$ to $w_{t+(n-1)/2}$, where $w_j$ is the center of this sequence:

$$v_t = \frac{1}{n} \sum_{j=t-(n-1)/2}^{t+(n-1)/2} w_j$$ \hspace{1cm} (3)

Remarks: The reason to use an $n$-gram vector $v_t$ to substitute $w_t$ as a word level information in a state is because that we want to better capture the contextual information compared to that using a single word vector. When $n = 1$, $v_t$ is the same as word vector $w_t$. 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.

Actions ($a_t$): The DCM model’s action $a_t$ at time step $t$ is de-
financed as in Figure 4. Each action gives a transition signal such that the state will change from its current label \( l_i \) to its next label \( l'_i \) by \( \tau(l_i, \theta) \). Simply speaking, the action set \( A \) contains all possible labels/tags for the word vector \( w_i \) or its correspond n-gram substitute \( v_i \). At each time step \( t \), the action \( a_t \) with the highest predicted action-reward value is chosen as:

\[
av_t = \arg \max_a Q(s_t, a|a \in A, \theta_t)
\]

where \( Q(\cdot) \) is the estimated action-value function generated by the neural network in DCM model at each step, which follows its definition in DQN as in [22]. By taking the action \( a_t \), an agent will transit from its current state to the state with a new label that is directed by \( a_t \).

**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 \( o_t \) of \( w_i \)’s true label \( l_i^* \), the one-hot vector \( o_i^* \) of the label \( l'_i \) in current state \( s_t \), and the predicted probability \( p_i \) for the word \( w_i \) using \( f_{rn} \) as:

\[
r_t = \begin{cases} 1 & \text{if } |o_t - p_t|_2 > ||o_i^* - o_i^*||_2 \\ -1 & \text{otherwise} \end{cases}
\]

where \( || \cdot ||_2 \) is the \( L_2 \) norm.

The reward is defined by comparing the distance from \( f_{rn} \) predicted label to \( w_i \)’s true label and that from current state’s label to its true label. The insight behind this definition is that a positive reward should be assigned to a state in which its label is more closer to the true label compared with the \( f_{rn} \)’s predicted one. The reward function is one of the key factors to get further performance improvement using our augmented tagger.

**Remarks:**

1. It is worth noticing that \( p_i \) should be generated by \( f_{rn} \) 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 \( v_i \).
2. In our setup, we use multiple cascaded models (including a RNN model and a DRL model) to improve the system performance. This idea of using multiple models’ models’ collective information has been widely used in system identification [23, 24, 25, 26], reinforcement learning [27, 28, 29] and also deep neural networks/deep learning (in NLP)[30, 31]. It has been shown as an effective approach to boost the model’s performance both theoretically and empirically.

**Training DCM using Experience Replay:** One training technique we borrowed from [22, 32, 33] is the experience replay used in DQN. It improves the convergence issue in neural network-estimator based DRL 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 \( M \). Whenever an action is performed then a new state is arrived, the past experience tuple will be pushed into the replay memory queue \( M \) following 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 \( s_{t+1} \) in the tuple.

### 3.2. Model Inference

As shown in Figure 5, the inference part of DCM is a bit different from training DCM. Since there is no ground truth during inference, in order to filter those data that may be labeled wrongly by \( f_{rn} \), a threshold value \( T_r \) is defined. All the tokens with their predicted tags’ probabilities \( p_i \) below \( T_r \) are used as the inference input of DCM, i.e \( w_i \in x_{test}\setminus x_{test\_rn} \). The outputs of DCM are the actions that can transfer the tokens from their current states to the states with their target label \( l_i^* \).

### 3.3. Coaching the System by Users

Besides the DRL based modal in DCM can improve a conventional RNN tagging model by using the wrongly labeled data, another important modal of DCM is the user coaching modal, in which the system can also be further coached by users in an online manner. During the inference, the model allows a user to select the correct labels of the words from its input, i.e \( x_{test\_rn} \), and then retrain the model, which is as shown in the dashed box in Figure 5. The retraining step is the same as that in section 3.1, except two aspects:

1. A new state \( s_{user} = [\text{user}, \hat{t}_{user}] \) is generated based on user input, and put together with the selected words’ current state \( s_t = [\text{user}, \hat{t}] \) as the experience replay tuple \( (s_t, r_t, a_t, s_{user}) \).
2. During experience replay, the model fixes the tuple(s) \( (s_t, r_t, a_t, s_{user}) \) containing user instructions in its replay memory \( M \), and only push in/pop out other states. Since the tuple(s) storing user instruction(s) is/are always in the memory, it has a higher chance to be selected and further trained. In practice, it gives us a much faster online training performance, hence one-shot learning can be achieved.

### 4. Experiment

The experiment consists of two set-ups: one set-up uses DCM only to coach the system without user’s feedback, and the other set-up takes the user’s instruction and retraining the DCM model during the inference. The performance comparison is based on their F1 scores.
4.1. Data Sets

Two datasets are used in the experiments, one is the the public ATIS dataset [34] containing utterances of flight reservations, and the other is our self-collected dataset in three different domains: Food, Home and Movie. The ATIS dataset used in this paper follows the same format as in [6, 3, 35, 4], which contains 4978 utterance in training set and 893 utterance in test set, the total number of slot tags is 127. For our self-collected dataset, it contains three domains: food, home and movie. There are 15 semantic tags in food domain, 16 semantic tags in home domain, 14 semantic tags in movie domain.

4.2. Training Setup

The neural network structure of the DRL in our DCM is chosen as LSTM, the number of LSTM states is chosen as 200. The averaged word vector in a reinforcement learning state is chosen as a trigram, i.e. n=3. The discount factor is chosen as $\gamma = 0.9$ and the threshold $T_r$ for inference is set as 0.9. The size of word embedding is 128, which are initialized randomly at the beginning of the experiment. The pre-trained RNN tagging model $f_{rnn}$ is chosen as the attention based BLSTM structure as illustrated in section 2, with a set-up follows [6]: the number of state in is 128, the drop out rate is 0.5 and batch size is 16. The models are trained individually on single Nvidia M40 GPU.

4.3. Performance of DCM with/without user coaching

The experiment is conducted using two different set-ups: one is a self-learning DCM model only based on the filtered states generated by $f_{rnn}$ and the threshold $T_r$, the other set-up takes the user/coach’s input such that one-shot learning can be achieved. For the second scenario, in order to interact with users, a coaching app is also developed on a android mobile as shown in Figure 6. During the inference, a small number of k sentences (here is chosen as $k = 5$) with tags below the threshold $T_r$ are selected from the entire data input of DCM, and to be corrected by the user, as shown in Figure 6(a) and 6(b). Then the DCM on our server is retrained using the corrected user inputs for 30 seconds. Figure 6(c) shows the outputs using the same and a similar input after DCM is retrained, which gives the correct output tags correspondingly.

To future evaluate the DCM model in a more quantitative manner, the test result on ATIS by using DCM with/without user’s feedback for slot filling is designed. The system uses DCM to coach the entire tagging model when “unsatisfied results” (below a threshold $T_u$) are generated. Furthermore, the model is designed in a manner that users can also teach the system based on their knowledge, which can be learned by the system in one-shot. The results generated by the DCM with/without user’s input outperform the state-of-the-art models on public ATIS dataset and our own in-house dataset. More importantly, as shown in the demo, the DCM has great potential as a fast on-line coaching model for general labeling tasks.

5. Conclusion

In this paper, a new DRL based coaching model with/without user’s feedback for slot filling is designed. The system uses DCM to coach the entire tagging model when “unsatisfied results” (below a threshold $T_u$) are generated. Furthermore, the model is designed in a manner that users can also teach the system based on their knowledge, which can be learned by the system in one-shot. The results generated by the DCM with/without user’s input outperform the state-of-the-art models on public ATIS dataset and our own in-house dataset. More importantly, as shown in the demo, the DCM has great potential as a fast on-line coaching model for general labeling tasks.

6. References


