Dialog State Tracking Using Long Short-term Memory Neural Networks

Xiaohao Yang, Jia Liu

Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084
yangxiaohao@gmail.com, liuj@tsinghua.edu.cn

Abstract

Neural network based approaches have recently shown state-of-art performance in the Dialog State Tracking Challenge (DSTC). In DSTC, a tracker is used to assign a label to the state at each moment in an input sequence of a dialog. Specifically, deep neural networks (DNNs) and simple recurrent neural networks (RNNs) have significantly improved the performance of the dialog state tracking. In this paper, we investigate exploiting long short-term memory (LSTM) neural networks, which contain forgetting, input and output gates and are more advanced than simple RNNs, for the dialog state tracking task. To explicitly model the dependence of the output labels, we propose two different models on top of the LSTM un-normalized scores. One is a regression model, the other is a conditional random field (CRF) model. We also apply a deep LSTM to the task. The method is evaluated on the second Dialog State Tracking Challenge (DSTC2) corpus and the results demonstrate that our proposed models can improve the performances of the task.

Index Terms: dialog state tracking, recurrent neural networks, long short-term memory, spoken dialog systems

1. Introduction

Spoken dialog systems allow a user to achieve a task, such as booking a ticket or finding a restaurant, using natural language. A dialog manager is one key component of a spoken dialog system, which aims at producing the appropriate system responses to users. To make the dialog more natural and effective, the dialog manager should take into account not only the given user utterance itself, but also the previous and the present dialog situations and background knowledge obtained from the progress of the dialog. Dialog state tracking is a key sub-task of dialog management which maintains the dialog state at each moment. The major challenge to dialog state tracking is that the inputs of the tracker are likely to be noisy because of the errors produced by automatic speech recognition (ASR) and spoken language understanding (SLU) processes before the dialog tracking.

Therefore, many researchers have focused on improving the robustness of dialog state trackers against ASR and SLU errors. In statistical dialog state tracking, models can be roughly divided into two major aspects which are generative and discriminative. Generative models mainly exploit dynamic bayesian networks to model the transition probability \( P(S_t | S_{t-1}) \) and observation probability \( P(O_t | S_t) \), where \( S_t \) and \( O_t \) are dialog state and observations at turn \( t \) [1, 2, 3]. In discriminative models, the conditional probability \( P(S_t | O^t) \) is modeled directly, using all the observations from turn 1 to \( t \) (\( O^t \)). One problem with the generative models is that the independent assumptions of the input features are always not satisfied. For example, \( N \)-best hypotheses and their confidence scores are often assumed independent of each other, which is inaccurate in realistic situations [4]. Additionally, it is difficult for generative models to handle overlapping features, which prevent models from incorporating large feature set. Discriminative models do not suffer from the above problems and they are potentially able to exploit much larger feature sets [5]. Thus, discriminative models tend to be more data-driven and can make more accurate predictions than generative models.

Several successful discriminative models for dialog state tracking include maximum entropy models [6], deep neural networks [7], conditional random fields [8, 9], and recurrent neural networks (RNNs) [10, 11]. RNNs provide a natural model for dialog state tracking and achieve the state of art, as they are able to model and classify dynamic sequences with complex features from step to step. The dialog tracker based on a RNN structure is capable of generalising to unseen dialog state hypotheses and requires little feature engineering [10].

In this paper we apply long short-term memory recurrent neural networks (LSTM-RNN) to the dialog state tracking task. LSTM-RNN [12, 13] is a novel recurrent neural network and has some advanced properties compared to the simple RNN [14]. LSTM contains three gates. Input gate is the connection between the input and the hidden memory cell. Forgetting gate is the recurrent connection amongst the hidden memory cells. Output gate is the connection between the memory cell and the output node. Importantly, the input and output of the memory cells are modeled in a context-sensitive way, which can be used in the context of timing analysis to extract information from dialog progress. To avoid the gradient exploding and diminishing problem in the simple RNN structure, the memory cells are linearly activated and propagated between different time steps.

To model the dependence of the output labels, we further extend the basic LSTM-RNN structure to include connections between the output nodes. One approach is using a regression model on top of the LSTM-RNN, the other is modeling the top of the LSTM-RNN as the output of a linear-chain CRF. To avoid label-bias problem [15], the model exploits un-normalized scores before softmax. Additionally, we apply a deep LSTM as an extension, which consists of multiple layers of LSTMs, to the dialog state tracking task.

The remainder of paper is organized as follows. Section 2 describes the three targets of dialog state tracking task. Section 3 details our proposed method, including feature engineering, basic model of LSTM and its extensions. Section 4 presents the implement of the experiment and the discussion of the results. The paper concludes in Section 5.

2. Task definition

In DSTC2, a dialog state consists of the following three components: goals, method and requested slots [16]. The task can be
regarded as labeling each dialog state at each turn of the dialog progress.

2.1. Goals tracking

Goals represent the constraint values which are truly intended by a user at each turn of the dialog. These values are predefined as the following 4 categories: area, food, name and price range. Assuming that the possible value set for each category (slot) is fixed, the task of goals tracking can be considered as a classification problem of determining the distribution over these hypotheses. While the previous challenge (DSTC1) aims at tracking a fixed goal for each dialog session, the models for DSTC2 need to handle changing goals during a session.

2.2. Method tracking

Method is the way of requesting information by a user. In DSTC2, method is divided into the following 4 categories: by constraints, by alternatives, by name, and finished. Method tracking is also a classification problem, computing the probability distribution over these four hypotheses at each turn.

2.3. Requested slots tracking

The third component for dialog state tracking is to determine whether the slots are requested by a user. The tracker should output the binary distributions of each slot and compute the probabilities whether the slot is requested or not. The slots which can be requested are predefined as follows: area, food, name, price range, address, phone, postcode and signature. Thus 8 different distributions are obtained at each turn.

3. Method

3.1. Models

3.1.1. LSTM

RNNs are able to incorporate discrete-time dynamics. The long short-term memory (LSTM) RNN has been shown to perform better at modeling and exploiting long range dependencies in the data than the simple RNN. The most important improvement is that the LSTM uses a memory cell with linear activation function to store information. In a simple RNN, during the gradient back-propagation phase, the gradient signal can end up being multiplied a large number of times by the weight matrix associated with the connections between the cells of the recurrent hidden layer. This means that, the magnitude of weights in the transition matrix can have a strong impact on the learning process. Using linear activation functions allows the LSTM to maintain the value of errors because its derivative with regard to the error is one. This to some extent avoids the error diminishing and exploding problems as the linear memory cells preserve unscaled activation and error derivatives across arbitrary time lags.

In this paper, we implemented the LSTM using the following composition functions:

\[ i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \]  \hspace{1cm} (1)
\[ f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \]  \hspace{1cm} (2)
\[ c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \]  \hspace{1cm} (3)
\[ o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \]  \hspace{1cm} (4)
\[ h_t = o_t \cdot \tanh(c_t) \]  \hspace{1cm} (5)

where \( i, f, o \) and \( c \) are respectively the input gate, forget gate, output gate, and memory cell activation vectors, all of which have the same size as the hidden vector \( h_t \). \( \sigma \) is the logistic sigmoid function. \( W_{xi}, W_{hi}, W_{ci}, W_{xf}, W_{hf}, W_{cf}, W_{xo}, W_{ho}, W_{co} \) are weight matrices and \( b_i, b_f, b_c, b_o \) are bias vectors. The weight matrices from the cell to gate \( (W_{ci}, W_{cf}, W_{co}) \) are diagonal, so the element \( j \) in each gate vector only receives input from element \( j \) of the cell vector. However, the weight matrices from inputs, hidden nodes, and outputs are not diagonal.

In the task of dialog state tracking, LSTM-RNNs operate in an online fashion to label the dialog state as soon as the feature is seen in a given turn, rather than after seeing the whole sequence. In this paper, we further improve the performance of a LSTM-RNN by incorporating elements of the CRF model. Specifically, we explicitly model the dependencies of output-labels with transition features. Because this new model is a CRF with features generated by a LSTM-RNN, we call it LSTM-CRF for short. Figure 3 shows the graphical structure of the LSTM-CRF.

In a LSTM-CRF, the LSTM is used to generate the input features for a CRF. The features used are the values \( h_t \) before softmax normalization. Using the activation before the softmax can avoid the label bias problem. The LSTM-CRF naturally incorporates dependencies between output labels via the CRF transition parameters by optimizing the objective function of the model.

For simplicity of notation, we denote the input-output pair sequence during training as \((x(t : 1), y(t : 1))\). In the dialog tracking task, \( x_t \) represents the features extracted from turn \( t \) and \( y_t \) represents the output targets such as the user’s true Goal, system’s Method or Requested slots at turn \( t \). With this notation, the objective function for a single training example is defined as

\[ P = \frac{1}{Z} \exp \sum_{t=1}^{T} (\lambda y_t \cdot \Delta y(t) + h_{y_t}(t)) \]  \hspace{1cm} (6)
Q = \log P = \sum_{t=1}^{T} (\lambda a y^*(t-1) y(t) + h y^*(t)) - \log Z \quad (7)

where \([y^*(1), ..., y^*(T)]\) denotes the correct output sequence and \(Z\) is the normalization factor. \(a y^*(t-1) y(t)\) is the transition feature from label \(y(t-1)\) to \(y(t)\). \(\lambda\) is a parameter used to weight the transition feature and is usually set to 1.0.

To train the LSTM-CRF model, we need to optimize the objective function \(Q\) for each dialog session in training data. Thus we take the gradient over the vertex feature \(\alpha\) and matrices \(\lambda a\) as the regression matrices on the pre-transforms. The regression model can score ending at turn \(t\), with output label \(p\). The backward pass is defined as the sum of path scores starting at turn \(t - 1\), with output label \(q\). With the forward and backward scores, we can compute gradients with respect to \(h y(t)=h(t)\) and \(a_{ij}\) as follows:

\[
\frac{\partial Q}{\partial h_{y(t)=k}(t)} = \delta(y(t) = y^*(t)) - \alpha(t,k) \beta(t,k) \sum_{j} \alpha(t,j) \beta(t,j) \quad (8)
\]

\[
\frac{\partial Q}{\partial a_{ij}} = \lambda \sum_{t} \delta(y(t-1) = j, y(t) = i) - \lambda \sum_{t} \alpha(t-1,j) \beta(t,i) \exp(\lambda a_{ij} + h_{i}(t)) \sum_{j} \alpha(t,j) \beta(t,j) \quad (9)
\]

The model parameters are updated using stochastic gradient ascent (SGA) over training data multiple passes.

3.1.4. Deep LSTM

The deep LSTM is created by stacking multiple LSTMs on top of each other. The input for the upper LSTM is the output of the lower LSTM. For example, the input \(x_t\) of the upper LSTM takes \(h_t\) from the lower LSTM. A matrix is applied on the \(h_t\) to transform it to \(x_t\) and the matrix can be designed so that the upper and lower LSTMs have different hidden layer dimensions. This paper investigates deep LSTMs with two hidden layers. Figure 5 shows the graphical structure of the deep LSTM.

3.2. Features

In this paper, we combine two kinds of feature set: designed feature functions and \(n\)-gram features extracted from utterances and dialog acts.

3.2.1. Feature functions

To train a tracking model, a set of feature functions are predefined based on the \(N\)-best hypotheses of user actions obtained from the ASR and SLU results at a given turn and the system actions corresponding to the previous system output [9]. The major actions of the user and the system in DSTC2 can be divided into 8 categories: ‘inform’, ‘confirm’, ‘deny’, ‘affirm’, ‘negate’, ‘request’, ‘reps’ and ‘canhelp’. The feature function for each action type is defined as follows (\(f\) represents the feature function, \(S\) represents the confidence score of the SLU results, \(s\) is the target slot, \(v\) is its value, \(U\) represents the \(N\)-best list of SLU results of user actions, and \(S\) represents the previous system actions):

- ‘inform’, ‘confirm’ and ‘deny’ function type:

\[
f(\text{in}/\text{co}/\text{de}, s, v) = \begin{cases} S(\text{in}/\text{co}/\text{de}, s, v), & \text{if } \text{in}/\text{co}/\text{de}(s, v) \in U \\ 0, & \text{otherwise} \end{cases}
\]
affirm' and 'negate' function type (‘expl-conf’ and ‘impl-conf’ are the previous system actions):
\[ f(\text{aff/neg}, s, v) = \begin{cases} \max(S(\text{aff/neg})), & \text{if expl-c}(s, v) \in S \\ 0, & \text{or impl-c}(s, v) \in S \end{cases} \]  
\[ \text{otherwise} \]  
\[ \text{(13)} \]

‘request’ and ‘reqalts’ function type:
\[ f(\text{req/reqalts}, s) = \begin{cases} S(\text{req/reqalts}(s)), & \text{if req/reqalts}(s) \in U \\ 0, & \text{otherwise} \end{cases} \]  
\[ \text{(14)} \]

‘canthelp’ function type:
\[ f(\text{canthelp}, s, v) = \begin{cases} 1, & \text{if canthelp}(s, v) \in S \\ 0, & \text{otherwise} \end{cases} \]  
\[ \text{(15)} \]

These 8 feature functions can compose a 8-dimensional feature vector \( F \). Prepared for the combination with \( n \)-gram features.

3.2.2. \( n \)-gram features
In addition to the designed feature functions above, \( n \)-grams extracted from utterances and dialog acts also provide important feature representations for the input to the LSTM-RNN [10]. For \( n \)-gram features extracted from the ASR \( N \)-best hypotheses, unigram, bigram and trigram features are computed for each hypothesis. They are then weighted by the \( N \)-best list probabilities and summed to give a single feature vector \( G_n \). For the dialog acts in the form of ‘acttype(slot=value)’, the extracted \( n \)-gram features are unigram (‘acttype’, ‘slot’, ‘value’), bigram (‘acttype slot’, ‘slot value’) and trigram (acttype slot value). Some acts (‘acttype( )’) do not have value or slot, so the corresponding \( n \)-gram feature is just unigram (‘acttype’). The \( n \)-gram features extracted from the SLU \( N \)-best dialog act list are also encoded in the same way. They are weighted by the corresponding posterior probabilities and summed to give a single feature vector \( G_n \). The \( n \)-gram feature extracted from the last machine act is represented as \( G_m \).

In this paper, the dialog state tracker takes an ASR \( N \)-best list, a SLU \( N \)-best list, the last machine act and the defined feature functions as input features for each turn. The combined feature representation is obtained by concatenating the vectors as follows:
\[ F_{\text{all}} = F \oplus G_n \oplus G_m \]  
\[ \text{(16)} \]

4. Experiments and discussions
4.1. Dataset and evaluation metrics
To demonstrate the effectiveness of proposed models, we performed experiments on the DSTC2 dataset which consists of 3235 dialog sessions on restaurant information domain [16]. The results of ASR and SLU, the true goal of the user, the method and the requested slots are annotated in the dataset for evaluation. We exploit the dataset following the original division into training, development and test sets, which have 1612, 506 and 1117 sessions, respectively. The models were implemented using Theano [18].

Among the various types of evaluation metrics listed in the results of the evaluation script [16], the following two featured metrics are selected to report the final performances of the tracker in this paper: Accuracy and L2 norm. These two metrics are calculated for the predicted joint goals, method and requested slots.

4.2. Results
Table 1 compares the performances of various models. The results indicate that our proposed LSTM-RNN outperforms the simple RNN in [10]. The extensions of the LSTM achieve better results than basic LSTM-RNN for most cases. The model LSTM-CRF which incorporates both LSTM and CRF outperforms the simple CRF [8, 9] or the simple RNN. The deep LSTM-CRF significantly improves the performance and achieves the best result. In our experiments, the deep model has two hidden layers and we use 100 hidden layer dimension for the lower LSTM and 200 dimension for the upper LSTM.

Table 1: Comparisons of dialog state tracking performances.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev set Acc</th>
<th>L2</th>
<th>Test set Acc</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF [8, 9]</td>
<td>0.643</td>
<td>0.049</td>
<td>0.960</td>
<td>0.064</td>
</tr>
<tr>
<td>RNN [10]</td>
<td>0.686</td>
<td>0.047</td>
<td>0.667</td>
<td>0.051</td>
</tr>
<tr>
<td>LSTM-RNN</td>
<td>0.688</td>
<td>0.046</td>
<td>0.675</td>
<td>0.042</td>
</tr>
<tr>
<td>LSTM-CRF</td>
<td>0.695</td>
<td>0.046</td>
<td>0.680</td>
<td>0.047</td>
</tr>
<tr>
<td>Deep LSTM-CRF</td>
<td>0.698</td>
<td>0.045</td>
<td>0.689</td>
<td>0.047</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Dev set Acc</th>
<th>L2</th>
<th>Test set Acc</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF</td>
<td>0.875</td>
<td>0.202</td>
<td>0.904</td>
<td>0.155</td>
</tr>
<tr>
<td>RNN</td>
<td>0.913</td>
<td>0.147</td>
<td>0.901</td>
<td>0.195</td>
</tr>
<tr>
<td>LSTM-RNN</td>
<td>0.914</td>
<td>0.145</td>
<td>0.907</td>
<td>0.153</td>
</tr>
<tr>
<td>LSTM-RNN-reg</td>
<td>0.921</td>
<td>0.142</td>
<td>0.910</td>
<td>0.149</td>
</tr>
<tr>
<td>LSTM-CRF</td>
<td>0.921</td>
<td>0.135</td>
<td>0.916</td>
<td>0.148</td>
</tr>
<tr>
<td>Deep LSTM-CRF</td>
<td>0.927</td>
<td>0.139</td>
<td>0.919</td>
<td>0.143</td>
</tr>
</tbody>
</table>

4.3. Discussions
The results above indicate that LSTMs with three gates are more advanced than simple RNNs for dialog state tracking task. The incorporation of the output-label dependence is beneficial for making correct predictions of output sequence. Additionally, deep LSTM models obtain a better feature learning ability than basic LSTM models thus they can achieve better results.

Therefore, we can try to improve the performance of dialog state tracking from two aspects: one is the employment of sequence modeling such as regression model and CRF model, the other is using deeper network structure to learn better features.

5. Conclusions
In this paper we have presented an application of LSTMs to dialog state tracking task. The LSTMs provided competitive performance on DSTC2 dataset. We further extended the model by performing regressions or CRF transitions on the output of LSTMs and built deep LSTMs. We observed that the extensions improve the performance on DSTC2 dataset and the deep LSTM-CRF achieves the state-of-the-art result.

6. Acknowledgements
This work is supported by National Natural Science Foundation of China under Grant No. 61370034, No. 61273268, No. 61005019 and No. 61005017.
7. References


