LecTrack: Incremental Dialog State Tracking with Long Short-Term Memory Networks

Lukáš Žilka and Filip Jurčíček*

Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics,
Malostranské náměstí 25, 118 00 Prague, Czech Republic
lukas@zilka.me,jurcicek@ufal.mff.cuni.cz

Abstract. A dialog state tracker is an important component in modern spoken dialog systems. We present the first trainable incremental dialog state tracker that directly uses automatic speech recognition hypotheses to track the state. It is based on a long short-term memory recurrent neural network, and it is fully trainable from annotated data. The tracker achieves promissing performance on the *Method* and *Requested* tracking sub-tasks in DSTC2.

Keywords: dialog systems, recurrent neural network, dialog state tracking

1 Introduction

A dialog state tracker is an essential component of modern spoken dialog systems. It maintains the user's goals throughout the dialog by looking at the automatic speech recognition (ASR) results of her utterances. For example, in the restaurant information domain, the dialog state tracker tracks what kind of food the user wants and which price range is she looking for, and provides this information as a probability distribution over *food* and *price_range*: $\mathrm{P}(\mathrm{food}, \mathrm{price\_range})$ The dialog state tracker also needs to deal with speech recognition errors and tries to reduce their impact on the dialog [1].

The state-of-the-art dialog state trackers [2,3] achieve their performance by learning from annotated data, and they were shown to work well in the restaurant information domain in the dialog state tracking challenge DSTC2 [4]. However, they possess several undesirable traits. First, they can only track the dialog state turn-by-turn (as opposed to a more complicated word-by-word approach), which limits the responsivity of the dialog system. Second, some of the trackers rely on the results from a spoken language understanding (SLU) component [5], which brings an additional component into the dialog system that needs to be trained and tuned. Third, elaborate and complicated tracking models of the trackers are difficult to reproduce and maintain. We aim to address these problems in the model proposed in this paper.

* This research was partly funded by the Ministry of Education, Youth and Sports of the Czech Republic under the grant agreement LK11221, core research funding, grant GAUK 2076214 of Charles University in Prague. This research was (partially) supported by SVV project number 260 224. This work has been using language resources distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013). Cloud computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Z GPU used for this research.

The contribution of this paper is our novel dialog state tracker, which we refer to as LecTrack[1]. It aims towards building more responsive and simpler dialog systems by proposing the first trainable dialog state tracker which naturally operates incrementally, word-by-word, and can directly learn from annotated dialogs, removing the need for an SLU unit. The word-by-word mode of tracking allows the dialog manager to be more responsive with the users. Simplicity comes from the fact that the whole dialog state tracker can be automatically optimized from data by a standard backpropagation algorithm, without requiring the user to manually tune opaque hyper-parameters or task-specific pre-processing.

Our LecTrack tracker is based on the long short-term memory recurrent neural network (LSTM RNN) [6]. We have chosen this approach due to several reasons: First, LSTMs were shown to be effective for learning sequence mappings in automatic speech recognition [7], machine translation [8], and many other sequence classification tasks. The length of the sequences successfully modelled by LSTMs is comparable to the length to the word sequences in the spoken dialog systems. Second, the sequential nature of the dialog naturally fits the LSTM's recurrent mode of operation. And finally, as the tracker processes the input, it incrementally builds an intermediate representation of the dialog. It has been shown that good intermediate representations help generalization [9]. The success of LSTM on multiple complicated and diverse tasks promises to be exploitable also in dialog state tracking.

The paper is organized as follows. First, we give a basic description of the task (Section 2). In Section 3, the model of the LSTM dialog state tracker is described with its training procedure. Section 4 shows how it performs in the benchmarks. Then, in Section 5, we discuss the results and qualities of the LSTM dialog state tracker. Finally we discuss some related work from the literature (Section 6) and conclude with Section 7.

2 Dialog State Tracking

The task of dialog state tracking is to monitor progress in the dialog and provide a compact representation of the dialog history in the form of a *dialog state* [4]. Because of uncertainty in the user input, statistical dialog trackers maintain a probability distribution over all dialog states, called the *belief state*. As the dialog progresses, the dialog state tracker updates this distribution given new observations.

In this paper, we define the dialog state at time t as a vector $s_t \in C_1 \times ... \times C_k$ of k dialog state components, sometimes called slots in the literature. Each component $c_i \in C_i = \{v_1, ..., v_{n_i}\}$ takes one of the n_i values (one of them can be the special *null* value, which denotes that this dialog state component has not been specified). Our dialog state tracker maintains a probability distribution over s_t factorized by the dialog state components:

$$P(s_t|w_1, ..., w_t) = \prod_i p(c_i|w_1, ..., w_t; \theta) \tag{1}$$

Note that all models $p(c_i|\cdot)$ share a substantial portion of the parameters, as detailed in the next section, so despite the fact that the predictions are factorized and thus in-

[1] (L)STM R(ec)urrent Neural Network Dialog State (Track)er.

dependent, they were optimized to minimize a joint objective function and therefore naturally model the dependence between the dialog state components.

3 LSTM Dialog State Tracker

Here we define our novel LSTM dialog state tracking model. Its task is to map a sequence of words in the dialog w_1, ..., w_t to predictions for each of the k dialog state components $(p_t^{(1)}, ..., p_t^{(k)})$. Each $p_t^{(i)}$ is a vector corresponding to a probability distribution over the values of the i-th dialog state component. For example, $p_t^{(area)}$ is a probability distribution over values $\{north, south, east, west\}$ at the time t.

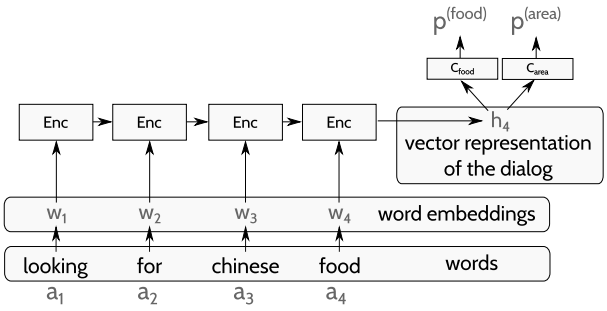

Fig. 1. A demonstration of the LSTM Dialog State Tracker applied to a user utterance "looking for chinese food". The encoding LSTM model Enc is sequentially applied to each input word and at the end, its hidden state is used to feed to the state component classifiers.

3.1 Model

Our dialog state tracking model can be seen as an encoder-classifier model, where an LSTM is used to encode the information from the input word sequence into a fixed-length vector representation. Given this representation, the classifier predicts a value for each of the dialog state components (as a probability distribution).

Formally we have an encoder that maps an input word and a previous hidden state to a new hidden state, $Enc(w, h_{t-1}) = h_t$, and a classifier that maps a hidden state to a prediction, $C(h_t) = p_t$. To encode the whole dialog, the encoder is applied sequentially on the input sequence of words. In our system, we have one encoder Enc and multiple classifiers C, one for each dialog state component (e.g. $C^{(food)}$, $C^{(area)}$, ...).

3.2 Tracker Model

LecTrack tracker is composed of two major components: the encoder Enc and the classifier C, which together form together LecTrack LSTM dialog state tracker:

$$\textbf{LecTrack} : a_1, ..., a_n \rightarrow p_1, ..., p_k \tag{2}$$

$$\forall i \in 1, ..., k : p_i = \mathrm{C_i}(\mathrm{Enc}(E \cdot a_1, ..., E \cdot a_n, h_0, c_0)) \tag{3}$$

Here, n is the length of the input sequence, k the number of the dialog state components, a_1, ..., a_n is the input word sequence encoded in a one-hot encoding, E is a word embedding matrix, and $h_0 = c_0 = \mathbf{0}$ are zero vectors. As shown, each token a_i is mapped to its corresponding embedding vector through the embedding matrix E, $w_i = E \cdot a_i$, which is treated just as another parameter. The model of the encoder $Enc(w_t, (h_{t-1}; c_{t-1}))$ is a standard LSTM RNN [6,10], where h_{t-1} is the previous LSTM output and c_{t-1} is the previous LSTM cell's state. In case of a recursive application of Enc, we write $Enc(w_1, ..., w_n, h, c)$ instead of $Enc(w_n, ...Enc(w_1, h, c))$ to simplify the notation. The model of the classifier C is a neural network with a single layer composed of rectified linear units and a softmax output layer.

3.3 Training

The model is trained using the standard cross-entropy criterion [11] in the vanilla stochastic gradient descent scenario [12]:

$$l(a_1, ..., a_n, y_1, ..., y_k; \theta) = \sum_{i=0}^{k} \log \mathrm{LecTrack}(a_1, ..., a_n)_{y_i}^{i} \tag{4}$$

Here, $\mathrm{LecTrack}(.)_n^m$ denotes the probability of the n-th value in the m-th dialog state component.

After each optimization epoch, we monitor the performance [2] of the model on a held-out set D. When the performance stops increasing for several iterations, we terminate the training and select the best-performing model.

4 Experiments

To train and evaluate our model, we use the DSTC2 [4] data, which is a common data set for dialog state tracking evaluation. The DSTC2 data consists of about 3,000 dialogs from the restaurant information domain, each dialog is 10 turns long on average. This data allows us to measure the performance of our tracker on turn-based dialogs. Ideally we would run the evaluation on a dataset where we could also measure the incremental capabilities of the tracker, but to the best of our knowledge, no such dataset is publicly available yet, and we shall address this in our future work.

A baseline system for this domain has been provided by the DSTC2 organizers. It uses the SLU results and confidence to rank hypotheses for the values of the individual dialog state components. There were several baselines described in [4] and we report the results of the *focus* baseline, which was the best among them.

We follow the DSTC2 methodology [4] and measure the accuracy and L2 norm of the joint slot predictions. The joint predictions are grouped into the following groups, and the results of each group is reported separately: Goals, Requested, Method. For each dialog state component in each dialog the measurements are taken *at the end of*

[2] See the experiments section for the description of the featured metrics.

each dialog turn, provided the component has already been mentioned in some of the SLU n-best lists in the dialog[3]

4.1 Data Preprocessing

Each dialog turn contains the system utterance and the user utterances, which we need to serialize into a stream of words as the input to our model. The system utterance undergoes a simple preprocessing detailed below, and the user utterance is directly fed to the model word-by-word without any further preprocessing. There is no difference between the system and user utterance in the eyes of our model, both are seen together as one long sequence of words.

System Input: To get the system input, we perform a simple preprocessing. We flatten the system dialog acts of the form `act_type(slot_name=slot_value)` into a sequence of two tokens t_1, t_2, where $t_1 = (\text{act\_type}, \text{slot\_name})$ and $t_2 = \text{slot\_value}$. For example `request(slot=food)` is flattened as $(request, slot), food$, which the model then sees as a word sequence of length two.

User Input: For the sake of simplicity, we use only the best live-ASR[4] hypothesis and ignore the rest of the n-best list. We plan to extend our model for processing multiple ASR hypotheses in the near future.

Out-of-Vocabulary Words are randomly mixed into the training data to give the model a chance to cope with unseen words: At training time, a word in the user input word is replaced by a special out-of-vocabulary token with probability α. At test time, this token is used for all unknown words.

4.2 Results

The results of LecTrack on the DSTC2 data are summarized in Table 1. For the groups *Method* and *Requested* LecTrack's accuracy is better than the baseline and comes closer to the state-of-the-art. Within these groups the handcrafted preprocessing present in the baseline and the state-of-the-art models is not as effective as for the *Goal* group.

We hypothesize that the accuracy on the *Goal* group does not achieve the state of the art because of two reasons. First, LecTrack needs to see examples for each value of each dialog state component. But the distribution of the individual values in the data has a heavy tail, and thus the baseline method and state-of-the-art methods that use various kinds of handcrafted abstraction to make the data denser and leverage hand-crafted generalization beat LecTrack. Second, our model does not utilize the information in the

[3] Note we do not use the SLU n-best list in our model at all, but we adapt this metric to be able to compare to the other trackers in DSTC2.

[4] There are batch and live ASR results in the DSTC2 data. We use the live ones and refer to them as live-ASR.

n-best lists, thus loses useful information in the uncertain cases where more hypotheses than the first one are useful.

For the frequently seen values from the group *Goal* the performance of LecTrack is much better than the baseline, as is shown in Table 2. We looked at the sub-goal *food* and compared the classification accuracy of its individual values. The top of the table contains 9 values, which occur more than 100 times in the test set, as the representatives of the classes that are well-represented in the data; the bottom of the table contains the representatives of the under-represented classes, and we selected values which occur at least 10 times in the test set to get meaningful accuracy estimates. For the well-represented classes, LecTrack's performance is stable and usually beats the baseline by a large margin, however for the under-represented classes LecTrack's performance is much worse than the baseline. This suggests that some form of abstraction should improve the results for the under-represented cases.

To keep the model simple we did not use any form abstraction, such as gazeteers to preprocess our data, and only used 1-best hypothesis as an input. Gazeteers offer a cheap solution to data sparsity for English but are difficult to gather and maintain for other languages where one word can have many forms. In our future experiments we plan to introduce some form of abstraction. Also, it is not obvious from the machine learning literature how an n-best list could be used in the model to improve the performance. This is another aspect that will be addressed in our future experiments.

	Dev						Test					
	Goal		Method		Requested		Goal		Method		Requested	
model	Acc.	L2	Acc.	L2	Acc.	L2	Acc.	L2	Acc.	L2	Acc.	L2
baseline	0.61	**0.63**	0.83	0.27	0.89	0.17	0.72	0.46	0.90	0.16	0.88	0.20
LecTrack	0.62	0.79	0.87	0.24	0.95	0.09	0.60	0.79	0.91	0.17	0.96	0.07
state-of-the-art [2]	**0.71**	0.74	**0.91**	**0.13**	**0.97**	**0.05**	**0.78**	**0.35**	**0.95**	**0.08**	**0.98**	**0.04**

Table 1. Performance on the DSTC2 data.

	chinese	indian	korean	asian o.	don't care	european	italian	spanish	thai	::	traditional	steakhouse	romanian	german
baseline	0.53	0.49	0.67	0.54	0.98	0.61	0.41	0.69	0.14		0.17	0.14	0.35	0.28
LecTrack	0.82	0.79	0.93	0.86	0.88	0.80	0.79	0.73	0.64		0.17	0.07	0.21	0.07

Table 2. Accuracy for the most frequent values for the *food* dialog state component which have at least 100 test examples in the test set, and for some that contain between 10 to 20 examples in the test set.

5 Discussion

Our LSTM dialog state tracker is capable of learning from raw dialog text, annotated with true dialog state component values at some timesteps. No spoken language understanding unit is needed to pre-process the input for our model. In addition, the model performance does not suffer if the input word sequences are long, which is in accordance with other LSTM applications [8].

Our model naturally handles the inter-slot dependence by projecting the input sequence into a fixed-length vector from which all the dialog state component predictions are made. However, the predictions are made independently for all of the state components and the joint distribution is not explicitly modelled.

Provided the ASR decodes also non-speech events, e.g., the affirmative ”hmm” or ”oh” or the information that the user is silent, the model can naturally learn to interpret them and provide hints to the dialog manager, such as whether the user seems to be confused, or if they started saying something and the dialog manager should interrupt its speech production and listen instead. In noisy conditions, waiting for silence is very limiting for the dialog system. The tracker's ability to process the input incrementally can overcome this issue and signal to the dialog manager when the incoming speech starts to make sense. This can lead to more human-like and interactive dialogs and simpler dialog managers. Our model was designed to be able to predict at arbitrary time in the dialog the full distribution over the dialog state components, and this mode of operation costs no additional computation as opposed to other trackers.

6 Related Work

The only incremental dialog system in the literature that we are aware of is [13]. In this paper, the authors describe an incremental dialog system for number dictation as a specific instance of their incremental dialog processing framework. To track the dialog state, they use a discourse modelling system, which keeps track of the confidence scores from the semantic parses of the input. The semantic parses are produced by a grammar-based semantic interpreter with a hand-coded context-free grammar. While their system is mostly handcrafted, ours is trained using annotated dialog data, so we do not need the handcrafted grammar and an explicit semantic representation of the input.

Using RNN for dialog state tracking has been proposed before [3,14]. The dialog state tracker in [3] uses an RNN, with a very elaborate architecture, to track the dialog state turn-by-turn. Similarly to our model, their model does not need an explicit semantic representation of the input. However, unlike our model, they use tagged n-gram features, which allows them to perform better generalization on rare but well-recognized values. Our model is capable of such generalization, too, but it needs more data. We refrain from using the tagged features because they introduce a preprocessing effort, and we are interested in a model that can learn from the data directly without assuming any correspondence between the names and values of the dialog state components and their surface forms that occur in the dialog (e.g. that value “chinese” of the dialog state component “food” will typically be represented as “chinese food” in the dialog). In English dialog systems, it might be perceived as an unneccessary complication not to leverage

these tagged features, but when we consider other languages, where a word often has a lot of forms, it pays off, because the effort spent on producing quality tagged features is non-trivial.

7 Conclusion

We presented a first trainable incremental dialog state tracker that directly uses automatic speech recognition hypotheses to track the state. It is based on a long short-term memory recurrent neural network and fully trainable from the dialog utterances annotated at certain points in time by the dialog state information. It represents the history of the whole dialog as a low-dimensional real vector, which is on its own used for the prediction of the whole dialog state. We evaluated our dialog state tracker on the data from Dialog State Tracking Challenge 2, where we showed that it achieves a promissing performance on the *Method* and *Requested* tracking sub-tasks. We believe that the simplicity, ease of use, and the incremental tracking capability of LecTrack make it a first good step on the way towards more responsive dialog systems.

References

1. Williams, J., Raux, A., Ramachandran, D., Black, A.: The Dialog State Tracking Challenge. In: Proceedings of the SIGDIAL 2013 Conference. (2013) 404–413
2. Williams, J.D.: Web-style ranking and SLU combination for dialog state tracking. In: 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue. (2014) 282
3. Henderson, M., Thomson, B., Young, S.J.: Word-based Dialog State Tracking with Recurrent Neural Networks. In: Proceedings of SIGdial. (2014)
4. Henderson, M., Thomson, B., Williams, J.: The second dialog state tracking challenge. In: 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue. (2014) 263
5. Wang, Y.Y., Deng, L., Acero, A.: Spoken language understanding. Signal Processing Magazine, IEEE **22**(5) (2005) 16–31
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997) 1735–1780
7. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks **18**(5) (2005) 602–610
8. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems. (2014) 3104–3112
9. Gülçehre, Ç., Bengio, Y.: Knowledge matters: Importance of prior information for optimization. arXiv preprint arXiv:1301.4083 (2013)
10. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
11. Rubinstein, R.Y., Kroese, D.P.: The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer Science & Business Media (2004)
12. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer (2010) 177–186
13. Skantze, G., Schlangen, D.: Incremental dialogue processing in a micro-domain. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics (2009) 745–753
14. Henderson, M., Thomson, B., Young, S.J.: Deep Neural Network Approach for the Dialog State Tracking Challenge. In: Proceedings of SIGdial. (2013)