Dialogue State Tracking using Long Short Term Memory Neural Networks

Koichiro Yoshino, Takuya Hiraoka, Graham Neubig and Satoshi Nakamura

Abstract We propose a dialogue state tracker based on long short term memory (LSTM) neural networks. LSTM is an extension of a recurrent neural network (RNN), which can better consider distant dependencies in sequential input. We construct a LSTM network that receives utterances of dialogue participants as input, and outputs the dialogue state of the current utterance. The input utterances are separated into vectors of words with their orders, which are further converted to word embeddings to avoid sparsity problems. In experiments, we combined this system with the baseline system of the dialogue state tracking challenge (DSTC), and achieved improved dialogue state tracking accuracy.

1 Introduction

Improvements in automatic speech recognition (ASR) have led to wide-spread use of speech interfaces, and thus the importance of spoken language understanding (SLU) is increasing [4]. SLU is a grounding process from natural language expressions to machine readable expressions, and is used in a variety of systems that use speech input. In task-oriented dialogue systems, SLU is often formulated as detection of the user's intention, and a list of values, the details of which depend on the intention. The intention and its values are maintained over several dialogue turns, and change gradually over the course of the dialogue.

The dialogue state tracking challenge (DSTC) is a shared task to recognize intentions and their values in task-oriented dialogues. In the previous challenge, DSTC 3, the task was designed to recognize the intentions of the user who talks with a dialogue system, which works for a pre-defined task. From this challenge, DSTC 4, the

Koichiro Yoshino, Takuya Hiraoka, Graham Neubig and Satoshi Nakamura
Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Takayama-cho, Ikoma, Nara, 6300192, Japan e-mail: koichiro,takuya-h,neubig,s-nakamura@is.naist.jp

scenario of the provided dialogue data has been changed to human-human dialogue in a limited task, consultations in the sightseeing domain. This change expands the variety of expressions of users, because the users will be free from limitations imposed by dialogue systems.

The property of human-human dialogue, including a variety of expressions, exacerbates sparsity problems, even if we try to use only the bag-of-words features of the user utterance for dialogue state tracking. We use a distributed representation of input words to avoid the problem. Distributed representation is a method to convert words into a dense vector representation by using information about the distribution of the surrounding words. This reduces the problem of data sparsity of word vectors, because meanings of words are compressed into limited dimensional vectors.

Dialogue state tracking is a task of predicting gradual changes of the user intention with their corresponding slot values. Previously, recurrent neural networks (RNNs) have been used for the dialogue state tracking [2, 1] and they achieved good results in previous challenges (DSTC 2 and 3). However, RNNs have also been reported to lack in ability to capture long-distance dependencies due to problems of vanishing gradients. In other words, while RNNs have the ability to propagate information through time due to recurrent connections between the hidden layers in the future. The propagation is weakened by several multiplications of the weight over several turns, which makes gradients decrease during every step through time. Long-short term memory (LSTM) [3] is a variation of RNNs that can avoid the vanishing gradients problem more effectively and consider information of long distant dependencies in a sequence, by creating linear connections between states from different time steps. Thus, it can be expected that this framework can capture the long-distance dependencies between gradually changing user intentions more accurately [10].

In this paper, we evaluate the proposed dialogue state tracker on the dataset of DSTC 4. We describe the framework of the dialogue state tracker in section 2, evaluate the tracker with the evaluation metrics provided by DSTC 4 committee in section 3, conclude the paper in section 4.

2 Overview of the Proposed System

2.1 Vectorization of Input Utterances and LSTM-based Tracker

Figure 1 shows an overview of the proposed dialogue state tracker. The tracker uses an ordered vector of words as user utterances. Because of the small amount of training data, the word vector will be sparse. Henderson et al. [2] converted an utterance to input vector expressed by a variety of classed expressions. To relieve this sparsity, we compress the utterance to a 300 dimensional vector by using doc2vec [6]. doc2vec is a variation of word2vec [7], a word embedding method, which is based on distributional similarity of words. The word embedding represents the mean-

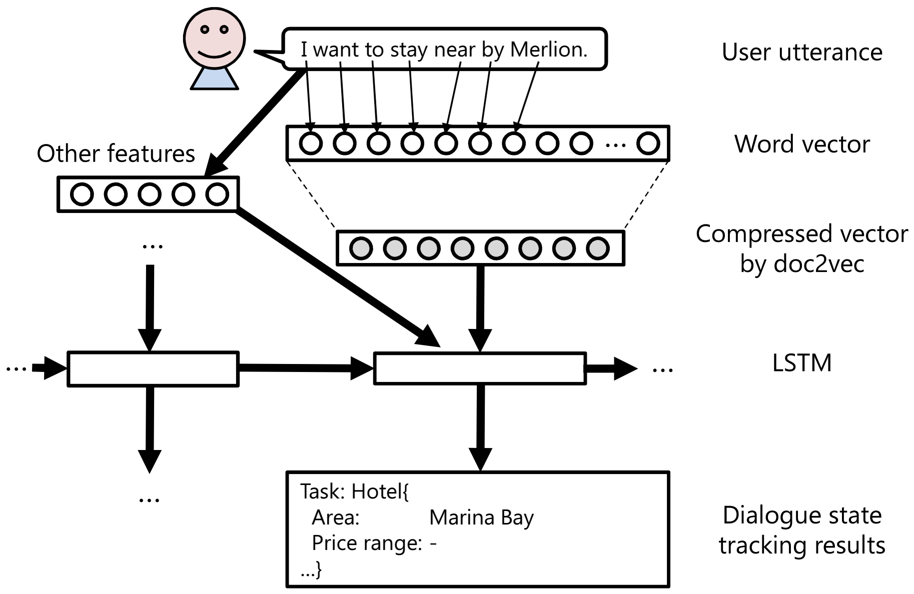
I want to stay near by Merlion.
User utterance
Word vector
Other features
Compressed vector by doc2vec
LSTM
Task: Hotel{
Area: Marina Bay
Price range: -
...}
Dialogue state tracking results

Fig. 1 An overview of the proposed dialogue state tracker.

ing of a word with the surrounding words (CBOW and Skip-gram) of the word. word2vec employs a neural network to compress the distribution of surrounding words, and the resultant vector expresses the meaning of the word in the compacted dimension. The result of doc2vec expresses the meaning of the sentence with not only bag-of-word (BOW) but also the order of words in the sentence, because it uses word order information in addition to information used in word2vec.

In addition to the compressed word vector, we use following input features:

Occurrence of frame value: This feature is represented by a vector that consists of binary elements corresponding to the occurrence of each frame value. Overlap ratio between each word in an input utterance and each frame value is calculated in the same manner with baseline tracker.

Semantic similarity between an inputted utterance and each slot: To calculate this feature, the semantic similarities between 1) each word in the input utterance and 2) slot name are calculated. This similarity represents the distance between the word and the slot name in the WordNet ontology [8]. Then, the average of similarity over all words in the input utterance is considered as the similarity of the input utterance itself.

A concatenated vector of the above features is sent to the LSTM network as input, and the LSTM network outputs the current dialogue state. Each node in the output network represents a single frame and its slot values. Each node has a probability of each single frame and its slot values. The dialogue state is affected by not only the features of the current utterance but also features of previous utterances compressed in the LSTM's hidden state. Thus, the LSTM-based tracker can propagate previous information over long distances, and the resultant tracking results are expected to keep consistency of the topic with previous sentences. We employed the LSTM implementation of the Pybrain[1] in the tracker. In the training, we constructed several LSTM-based trackers in different epochs, and integrated their trackers.

2.2 Integrating with the Baseline Tracker

We combined the proposed tracker with the baseline tracker provided by the DSTC 4 committee [5] in two methods (**Average** and **Concat**):

Average: Both the baseline and proposed trackers have the ability to output both tracking results and probabilities of outputs. This property makes it easy to integrate multiple trackers. We combined the proposed tracker with the baseline tracker by deciding the best candidate at each turn with its probability. In this methods, the average of probabilities output by the proposed tracker and the baseline tracker is calculated, and then the hypothesis with the maximum probability is used as output.

Concat: As another way for combining two trackers, we devised a method that utilizes the union of both trackers' results. In this method, the output of each tracker is determined independently, and the output is combined so that the slot value is the union of outputs of both trackers.

From the preliminary experiment results, we determined that the Concat strategy performed superiorly for the DSTC task, and all following results use this strategy.

3 Experiments

The DSTC task provides three types of data; training, development, and test. The tracker is trained using the training dataset when testing on the development dataset. When testing on the test dataset, the training and the development dataset are used.

The doc2vec tool requires a dataset to train the document embedding model. We used the training and the development datasets for training embeddings when testing on the development dataset, and used every dataset when testing on the test dataset. Thus, these trackers assume batch processing of dialogue data, and the accuracy will decrease if we run the trackers in on-line processing.

[1] http://pybrain.org/

Table 1 Results per utterance (schedule1) for the development set.

	N	Base.	RNN	LSTM	Combi.
Accuracy	4,139	0.027	0.010	0.011	0.026
Precision	6,318	0.213	0.169	0.151	0.364
Recall	10,367	0.282	0.117	0.082	0.222
F1	14,385	0.243	0.138	0.107	0.276

Table 2 Results per sub-dialogue (schedule2) for the development set.

	N	Base.	RNN	LSTM	Combi.
Accuracy	589	0.044	0.015	0.022	0.042
Precision	1,138	0.307	0.210	0.183	0.373
Recall	1,453	0.280	0.105	0.030	0.292
F1	2,166	0.293	0.140	0.052	0.328

3.1 Results

Table 1 and **2** show development dataset results with regards to accuracy, precision, recall, and harmonic mean of precision and recall: F1. The first table (schedule1) shows results for each utterance, and the second table (schedule2) shows results for each sub-dialogue. Base. is results of the baseline tracker, LSTM is results of the proposed LSTM-based tracker, and Combi. is results of the integrated tracker. For the comparison, we also show results of the RNN-based tracker, which uses the same inputs, outputs, and hidden layers as the proposed LSTM-based tracker (=RNN).

These results show that the LSTM-based tracker did not perform well, underperforming both the baseline tracker. However, it is interesting to note that the integration result was better than the result of the baseline tracker. This indicates that the labels output by the LSTM-based tracker are mostly different from labels output by the baseline tracker, and the baseline tracker is improved by the integration with the LSTM-based tracker. Note that, the majorities of output of both trackers are “no label”, thus, the number of output is increased by the integration. This is why the precision is improved and the recall is decreased by the integration.

From these results, it is hard to find a superiority of LSTM-based tracker, by comparing to the RNN-based tracker. The accuracy of LSTM-based tracker was better than the accuracy of RNN-based tracker, however, RNN-based tracker worked better than LSTM-based tracker in F-measure score. One hypothesis of the reason of this result is that the proposed network structure is still sparse to improve the results by complicated structure of LSTM network. Especially, the output layer of the proposed system is very sparse by comparing the previous works that use RNN network [2, 1], and the size of training data was not enough to learn a good network for the huge output layer.

Table 3 Results of both schedules for the test set.

	schedule1	schedule2
Accuracy	0.027	0.040
Precision	0.340	0.358
Recall	0.201	0.263
F1	0.253	0.303

Table 3 shows results of both categories on the test dataset. The tendency of the result of testing the test dataset is similar to the results on the development dataset, indicating that our settings are not overfitting to the development data.

3.2 Adjusting Output Thresholds

We performed an error analysis [9] to provide some reasons for the worse results of the proposed LSTM-based tracker. We compare the results of LSTM-based tracker and RNN-based tracker.

There are 4,812[2] frames that are expected to be estimated, and LSTM and RNN trackers output 2,474 and 3,033 of them, respectively. The number of frames output by the LSTM is lower than the reference, indicating that the probability threshold of the LSTM output (0.5 in default) is too high to output enough hypothesized frames. We changed the probability threshold in increments of 0.1 between 0.1 and 0.5, and calculated accuracy, precision, recall, and F-measure of the output in schedule1 (per utterance). **Figure 2** shows the results, and it shows that the 0.2 threshold maximizes the F-measure of the frame, and 0.4 threshold maximizes the accuracy.

4 Conclusion

We proposed an LSTM-based dialogue state tracker for the DSTC4 shared task, and found that the tracker that by integrating the tracker with the baseline that we can improve the overall accuracy. However, the improvement provided by the proposed tracker was not large. One of main problems of the tracker is a mismatch between the training set and testing set, and we plan to use some abstraction methods based on knowledge bases to improve generalization and close this gap. Another problem is the size of the data. The proposed LSTM-base tracker uses embeddings trained on the dataset of DSTC4. However, it is possible that the data was too small to train the embedding, and we will try to use other dataset to alleviate this problem. The design of the output layer of the LSTM network is also a problem of the proposed tracker,

[2] The number is different from the N of accuracy, because some utterances are annotated with several frames.

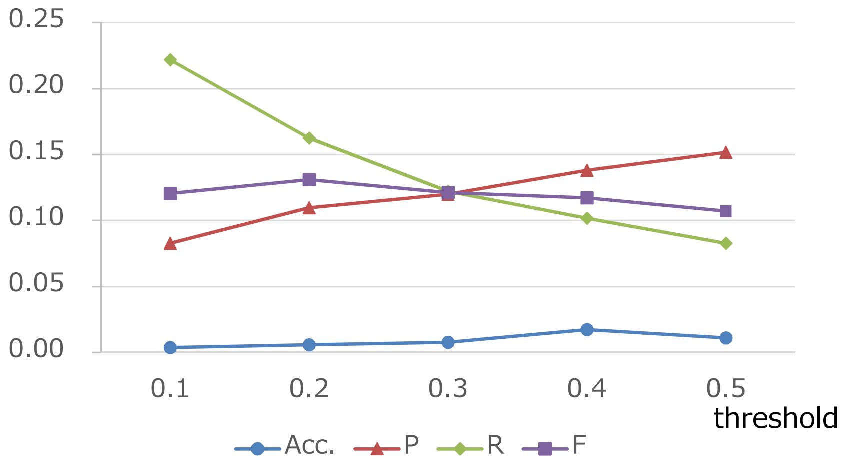

Fig. 2 Accuracy (Acc.), precision (P), recall (R) and F-measure (F) of each threshold value of LSTM-based tracker in schedule1 (per utterance).

and we will try to design a more simplified output layer to avoid the problem of sparseness.

5 Acknowledgment

This work is supported by Grant-in-Aid for Research Activity Start-up No.15660505.

References

1. Matthew Henderson, Blaise Thomson, and Steve Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *IEEE Workshop on Spoken Language Technology*, pages 360–365. IEEE, 2014.
2. Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 292, 2014.
3. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
4. Tatsuya Kawahara. New perspectives on spoken language understanding: Does machine need to fully understand speech? In *IEEE Workshop on Automatic Speech Recognition & Understanding, 2009. ASRU 2009*, pages 46–50, 2009.
5. Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason Williams, and Matthew Henderson. The Fourth Dialog State Tracking Challenge. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*, 2016.
6. Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

7. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
8. George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
9. Ronnie W Smith. Comparative error analysis of dialog state tracking. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 300–208, 2014.
10. Lukáš Žilka and Filip Jurčíček. Lectrack: Incremental dialog state tracking with long short-term memory networks. In *Text, Speech, and Dialogue*, pages 174–182. Springer, 2015.