Recurrent Neural Networks for Dialogue State Tracking

Ondřej Plátek, Petr Bělohlávek, Vojtěch Hudeček, and Filip Jurčíček

Charles University in Prague, Faculty of Mathematics and Physics
{oplatek,jurcicek}@ufal.mff.cuni.cz,
me@petrbel.cz,
vojta.hudecek@gmail.com,
http://ufal.mff.cuni.cz/ondrej-platek

Abstract: This paper discusses models for dialogue state tracking using recurrent neural networks (RNN). We present experiments on the standard dialogue state tracking (DST) dataset, DSTC2 [6]. On the one hand, RNN models became state of the art in DST, on the other hand, most state-of-the-art models are only turn-based and require dataset-specific preprocessing (e.g. DSTC2-specific) in order to achieve state-of-the-art results. We implemented two architectures which can be used in incremental settings and require almost no preprocessing. We compare their performance to the benchmarks on DSTC2 and discuss their properties. With only trivial preprocessing, the performance of our models is close to the state-of-the-art results.

1 Introduction

Dialogue state tracking (DST) is a standard and important task for evaluating conversational agents [16, 6, 7]. A dialogue state tracker summarizes the hidden information state (HIS)[18] of a user's goal from the conversation history. User's goals are expressed in a formal language, typically represented as a dialogue act item (DAI). It was shown that with a better dialogue state tracking of HIS, conversation agents achieve better success rate in overall completion of the their task [10]. Dialogue state tracking translates ambiguous natural language into a formal language convenient for reasoning and accessing external knowledge. Reasoning as well as the use of external knowledge are crucial to a successful conversation in a task oriented dialogue.

In our experiments, we predict the dialogue state only from ASR transcriptions of the conversation history and a word feature which fires if a word may be a property from the database. We argue that using only database values instead of the full ontology is not only simpler but also more natural because the system can inform the users only about the values in the database. Note that a database of information e.g., about Cambridge restaurants, naturally defines the task of the conversation by listing the information which the system can provide. On the other hand, the domain ontology needs to be handcrafted because it defines the task by listing all the information which users can ask about.

In our experiments, we focus only on the *goal* slot predictions because the other groups are trivial to predict[1].

We also experiment with another re-splitting of the DSTC2 data because there are considerable differences between the standard train and test data sets. The DSTC2 test set was collected from different systems [6]. Since the data of training, development and test set are distributed differently, the resulting performance difference between training and test data is rather high. Our experiments show that one can obtain better results by splitting the data randomly. As a result, we conclude that DSTC2 might suggest too pessimistic view of the state-of-the-art methods in dialogue state tracking caused by the data distribution mismatch.

Our contribution is threefold. First, in Section 3 we compare two different architectures using RNN for dialogue state tracking. Secondly, in Section 4.2 we describe state-of-the art word-by-word dialogue state trackers architectures and propose a new encoder-decoder architecture for the DST task. Finally, in Section 4.2 we show that obtaining excellent results on DSTC2 dataset is very demanding because of the training and test set dissimilarity. In contrast, we obtained much better results just by re-splitting the DSTC2 data randomly.

2 Dialogue state tracking on DSTC2 dataset

Dialogue state tracking is task which updates distribution of representation of dialogue history. In the DSTC2 dataset, the history is captured by dialogue act items and their probabilities. A Dialogue act item is a triple of the following form $(actionType, slotName, slotValue)$.

The DSTC2 is a standard dataset for DST, and most of state-of-the-art systems in DST have reported their performance on this dataset [6]. The full dataset is freely available since January 2014 and contains 1612 dialogues in the training set, 506 dialogues in the development set and 1117 dialogues in the test set. The conversations are annotated at the turn level where the hidden information state was annotated manually in form of $(actionType, slotName, slotValue)$ according to the domain ontology. The task of the domain is defined by

[1]The slots *Requested* and *Method* have accuracies 0.95 and 0.95 on the test set according to the state-of-the-art [17].

a database of restaurants and their properties[2]. The manually designed ontology which captures a restaurant domain is also distributed with the dataset.

3 Models

Our models are all based on a recurrent neural network (RNN) encoder [15]. Similarly to RNN encoder of [19], the models update their hidden states h after processing each word. The RNN encoder takes as inputs the previous state h_{t-1} representing history for first $t-1$ words and features X_t for the current word w_t. It outputs state h_t representing the whole history up to current word. We uses a GRU cell [5] as the update function and we learn its parameters during training.

For each input token, our RNN encoder reads the word embedding [3] of this token along with several binary features. The binary features for each word are:

- the speaker role, representing either user or system,
- and also indicators describing whether the word is part of some named entity representing a value from the database.

Since DSTC2 database is a simple table with six columns, we introduce six binary features firing if the word is a substring of named entity at given column. For example, the word *indian* will not only trigger the feature for column $food$ and its value *indian* but also for column restaurant *name* and its value *indian heaven*.

Our model variants differ only in the way they predict *goal* labels, i.e. $food$, *area* and *pricerange*, from the RNN's last encoded state. The first model predicts the output slot labels independently by employing three independent classifiers (see Section 3.1) The second model uses a decoder in order to predict values one after each other from the h_T (see Section 3.2). We also experimented with single classifier which predicts the labels jointly (see Figure 1, but it suffers from data sparsity of the predicted tuples, so we focused only on the independent label prediction and encoder-decoder models.

The models were implemented using TensorFlow [1] framework. By introducing more and more complex models, we aim to overcome the data sparsity problem and incorrect independence assumptions.

3.1 Independent classifiers for each label

The independent model predicts slots $food$, *area* and *pricerange* based on the last hidden state h_T independently using three classifiers. Independent slots prediction using one classifier per each slot is straightforward to implement, but the model introduces an unrealistic assumption of uncorrelated slot properties. In case of DSTC2 and the Cambridge restaurant domain, it is hard to believe that, e.g., the slots *area* and *pricerange* are not correlated.

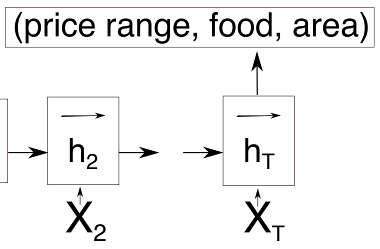

Figure 1: The joint label predictions using RNN from last hidden state h_T. The h_T represents the whole dialog history of T words. The RNN takes as input for each word i an embedding and binary features concatenated to vector X_i.

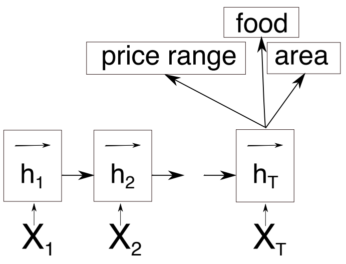

Figure 2: The RNN encodes the word history into dialogue state h_T and predicts slot values independently.

3.2 Encoder-decoder framework

The encoder-decoder model with attention [2] is the most sophisticated model that we used for slot predictions. To our knowledge, we are first who used this model for the task of slot predictions. The model is successfully used in machine translation where it is able to handle long sequences with good accuracy. In DST, it captures correlation between the decoded slots easily [2].

The disadvantage of this model is its complexity. Firstly, the model is not trivial to implement[3]. Secondly, the decoding time is quadratic in the length of the decoded sequences. However, it is trivial to cast DST as a sequence-to-sequence problem. Given the dialogue history as the input sequence, we trained the model to predict a sequence of four tokens representing the first, the second, the third slot and the end-of-string symbol. Since our target sequence is always of length four, the model does not suffer from long decoding time.

4 Experiments

We report results on the standard DSTC2 data split where we used 516 dialogues as a validation set for early stopping [13] and the remaining 1612 dialogues for training. We use 1-best ASR transcriptions on the input and measure joint slot accuracy. For the results on the standard

[2]There are six columns in the database: name, food, price_range, area, telephone, address.

[3]We modified code from the TensorFlow 'seq2seq' module.

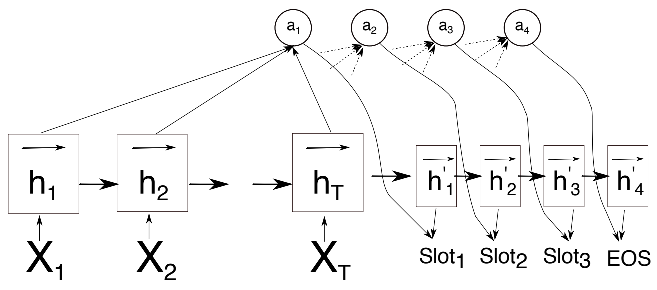

Figure 3: Encoder decoder with attention predicts goals.

split of DSTC2, see Table 1. We also evaluated the models on a randomly split DSTC2 dataset and the details are described in Section 4.3.

Note that we measure accuracy without schedule 2 which skips the first turns where the believe tracker does not track any values. As a consequence, we do not evaluate the models according any standard metric.[4]

For all our experiments, we train word embeddings of size 100 and use the encoder state size of size 100, together with a dropout keep probability of 0.7 for both encoder inputs and outputs. These parameters where selected by a grid search over the hyper-parameters on development data.

4.1 Training

The training procedure minimizes the cross-entropy loss function using the Adam optimizer [11] with a batch size of 10. We train predicting goal slot values for each turn. We treat each dialogue turn as a separate training example, feeding the whole dialogue history up to the current turn into the encoder and predicting the slot labels for the current turn.

We use early stopping with patience [13], validating on the development set after each epoch and stopping if the three top models does not change for four epochs.

The predicted labels in DST task depend not only on the last turn, but on the dialogue full history as well. Since the lengths of dialogue histories vary a lot[5] and we batch our inputs, we separated the dialogues into ten buckets accordingly to their lengths in order to provide a computational speed-up. We reshuffle the data after each epoch only within each bucket.

In informal experiments, we tried to speed-up the training by optimizing the parameters only on the last turn[6] but the performance dropped relatively by more than 40%.

[4]We plan to submit the camera ready version of this article with results evaluated using the standard scripts for measuring accuracy with schedule 2.

[5]The maximum dialogue history length is 487 words and 95% percentile is 205 words for the training set.

[6]The prediction was conditioned on the full history but we back-propagated the error only in words within the last turn.

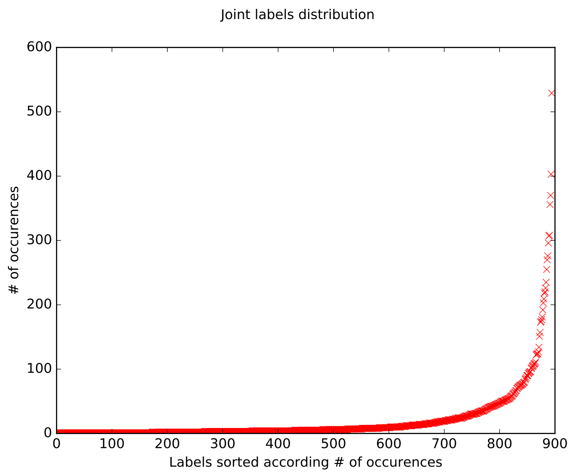

Figure 4: The number occurrences of labels in form of $(food, area, pricerange)$ triples from the least to the most frequent.

Model	Dev set	Test set
Indep	0.91	0.76
EncDec	0.94	0.73

Table 1: Accuracy on development and test set

4.2 Comparing models

Predicting the labels jointly is quite challenging because the distribution of the labels is skewed as demonstrated in Figure 4. Some of the labels combinations are very rare, and they occur only in the development and test set so the joint model is not able to predict them. During first informal experiments the joint model performed poorly arguably due to data sparsity of slot triples. We further focus on model with independent classifiers and encoder-decoder architecture.

The model with independent label prediction is a strong baseline which was used, among others, in work of [19]. The model suffers less from data set mismatch because it does not model the correlation between predicted labels. This property can explain a smaller performance drop between the test set from reshuffled data and the official test set in comparison to encoder-decoder model.

Since the encoder-decoder architecture is very general and can predict arbitrary output sequences, it also needs to learn how to predict only three slot labels in the correct order. It turned out that the architecture learned to predict quadruples with three slot values and the end-of-string (EOS) symbol quickly, even before seeing a half of the training data in the first epoch. At the end of the first epoch, the system made no more mistakes on predicting slot values in the incorrect order. The encoder-decoder system outperformed the previous models and the time needed for learning the output structure was surprisingly

Model	Dev set	Test set
Indep	0.87	0.89
EncDec	0.94	0.91

Table 2: Accuracy on the re-split DSTC2 data.

short.[7]

4.3 Data preparation experiments

The data for the DSTC2 test set were collected using a different spoken dialogue system configuration than the data for the validation and the training set.[6]. We wanted to investigate how this influences the complexity of the task so we merged all DSTC2 data together and created splits of 80%, 10% and 10% for the training, development and test sets. The results in Table 2 show that the complexity of the task dropped significantly.

5 Related work

There are numerous systems which reported on the DSTC2 dataset, we discuss only the systems which use RNN. In general, the RNN systems achieved excellent results.

Our system is related to the RNN tracker of Žilka and Jurčíček [19], which reported near state-of-the art results on the DSTC2 dataset and was the first incremental system which was able to update the dialogue state word-by-word with such accuracy. In contrast to work of [19], we use no abstraction of slot values. Instead, we add the additional features as described in Section 3. The first system which used a neural network for dialogue state tracking [8] used a feed-forward network and more than 10 manually engineered features across different levels of abstraction of the user input, including outputs of the spoken language understanding component (SLU). In our work, we focus on simplifying the architecture, hence we used only features which were explicitly given by the dialogue history word representation and the database.

The system of Henderson et al. [9] gives state-of-the-art results and, similarly to our system, it predicts the dialogue state from words by employing a RNN. On the other hand, their system heavily relies on the user input abstraction. Another dialogue state tracker with LSTM was used in reinforcement setting but the authors also used information from the SLU pipeline [12].

An interesting approach is presented in the work of Vodolán et al. [14], who combine a rule-based and a machine learning based approach to belief state tracking. Their work clearly separates the handcrafted features which can be easily replaced or omitted. The features are fed to an LSTM-based RNN which performs a dialog-state update. However, unlike our work, their system requires SLU parses on the input.

It is worth noting that there are first attempts to train an end-to-end dialogue system even without explicitly modeling the dialogue state [4], which further simplifies the architecture of a dialogue system. However, the reported end-to-end model was evaluated only on artificial dataset and cannot be compared to DSTC2 dataset directly.

6 Conclusion

We presented and compared two dialogue state tracking models which are based on state-of-the-art architectures using recurrent neural networks. To our knowledge, we are the first to use an encoder-decoder model for the dialogue state tracking task, and we encourage others to do so because it is competative with the standard RNN model. Note that the standard RNN model achieves almost state-of-the-art results.

We evaluated the models on DSTC2 dataset containing task-oriented dialogues in restaurant domain. We trained the models using only ASR 1-best transcriptions and task-specific lexical features defined by the task database.

We also observed that obtaining high accuracy for dialogue state tracking on DSTC2 test set is notoriously hard and that the task becomes substantially easier if the data is reshuffled.

The code for our experiments is published under a permissive Apache license on Github at `https://github.com/oplatek/e2end/`.[8]

In future work, we plan to evaluate the models using the standard DSTC2 scripts to compare them to other trackes on this dataset. In addition, we would like to investigate how much the introduced database features improved the performance of the models. Our future plans also concern collecting and annotating more data which would allow us to evaluate the models on another task-oriented dataset.

Acknowledgment

We would like to thank Mirek Vodolán for help with evaluation scripts and Ondřej Dušek for useful comments. 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 1915/2015, and also partially supported by SVV project number 260 333. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40c GPU used for this research. 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.

[7]The best model weights were found after 18 to 23 epochs for all model architectures.

[8]Some informal experiments were conducted within the Statistical Dialogue Systems course at Charles University in Prague and can also be found on Github at `https://github.com/oplatek/sds-tracker`.

References

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A Neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[4] Antoine Bordes and Jason Weston. Learning End-to-End Goal-Oriented Dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014.

[6] Matthew Henderson, Blaise Thomson, and Jason Williams. The second dialog state tracking challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, volume 263, 2014.

[7] Matthew Henderson, Blaise Thomson, and Jason D Williams. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 324–329. IEEE, 2014.

[8] Matthew Henderson, Blaise Thomson, and Steve Young. Deep neural network approach for the dialog state tracking challenge. *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471, 2013.

[9] Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, 2014.

[10] Filip Jurčíček, Blaise Thomson, and Steve Young. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 26(3):168–192, 2012.

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Byung-Jun Lee and Kee-Eung Kim. Dialog History Construction with Long-Short Term Memory for Robust Generative Dialog State Tracking. *Dialogue & Discourse*, 7(3):47–64, 2016.

[13] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[14] Miroslav Vodolán, Rudolf Kadlec, and Jan Kleindienst. Hybrid Dialog State Tracker. *CoRR*, abs/1510.03710, 2015.

[15] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[16] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, 2013.

[17] Jason D Williams. Web-style ranking and SLU combination for dialog state tracking. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, volume 282, 2014.

[18] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

[19] Lukas Zilka and Filip Jurcicek. Incremental LSTM-based dialog state tracker. *arXiv preprint arXiv:1507.03471*, 2015.