# **INCREMENTAL LSTM-BASED DIALOG STATE TRACKER**

Lukas Zilka, Filip Jurcicek

Charles University in Prague, Faculty of Mathematics and Physics Malostranske namesti 25, 118 00 Prague

### ABSTRACT

A dialog state tracker is an important component in modern spoken dialog systems. We present an incremental dialog state tracker, based on LSTM networks. It directly uses automatic speech recognition hypotheses to track the state. We also present the key non-standard aspects of the model that bring its performance close to the state-of-the-art and experimentally analyze their contribution: including the ASR confidence scores, abstracting scarcely represented values, including transcriptions in the training data, and model averaging.

*Index Terms*— spoken dialog systems, dialog state tracking, recurrent neural networks, LSTM

#### 1. INTRODUCTION

A dialog state tracker is an important component of statistical spoken dialog systems. It estimates the user's goals throughout the dialog by analyzing the automatic speech recognition (ASR) outputs for the user's utterances. For example, in the restaurant information domain, the dialog state tracker can track what kind of food the user wants and which price range is he looking for, as a probability distribution over *food* and *price\_range*: P(food, price\_range).

The state-of-the-art dialog state trackers [1, 2, 3, 4, 5] achieve their top 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 [6]. However, they still possess two undesirable properties. First, they can only track the dialog state turn-by-turn (as opposed to a more complicated word-by-word approach), which limits their interaction with users: For example, in a typical dialog system [7, 8], the dialog system can

neither provide affirmative natural feedback while the user is speaking, nor can the system interpret additional information said by the user while the system is speaking, both of which is very natural in human-human communication. And second, some of the trackers use an intermediate semantic representation and a spoken language understanding (SLU) component [9]. As the representation is manually crafted, it can cause loss of information, and an SLU, if used, is an additional component of the dialog system that needs to be trained and tuned.

The main contribution of this paper is an extension of our LSTM-based [10] dialog state tracker, first described in [11], which brings its performance close to the state-of-the-art models. We refer to the tracker as LecTrack<sup>1</sup>. LecTrack naturally operates incrementally, word-by-word, and does not require an SLU. It learns from dialog sessions annotated by dialog state component labels at different time steps. The improvements consist of including the ASR confidence scores, abstracting scarcely represented values, including transcriptions in the training data, and model averaging.

The paper is organized as follows: First, we give a basic description of the dialog state tracking task in Section 2. In Section 3, the model of our LSTM dialog state tracker is described with its training procedure. The tracker is evaluated in Section 4. Related work from the literature is discussed in Section 5. Section 6 concludes the paper.

### 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* [6, 12]. Because of uncertainty in the user input, statistical dialog systems maintain a distribution over all possible 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  $n_i$  values, and we assume the com-

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, GAUK grant 2076214, SVV project number 260 224 of Charles University in Prague. 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.

<sup>&</sup>lt;sup>1</sup>(L)STM R(ec)urrent Neural Network Dialog State (Track)er.

ponents are independent:

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

Our dialog state tracker, that we describe in the following, gives the probability distribution only over one of the independent components  $p(c_i|w_1, ..., w_t)$ . A prediction for more components together is made independently by running different models, specific for each component *i*.

### 3. LSTM DIALOG STATE TRACKER

In this section we describe an extended version of the Lec-Track LSTM dialog state tracking model [11]. The task of the tracker is to map a sequence of words in the dialog to a probability distribution over the values of a dialog state component p. For example, for the dialog state component *area*,  $p_t$  is a probability distribution over values  $\{north, south, east, west\}$  at the time t. Because the input words may be preprocessed, we refer to them sometimes as tokens; a sequence of words/tokens  $a_1, ..., a_t$  from some a vocabulary  $a_i \in Vocab$ .

### 3.1. Model

Our dialog state tracking model is an encoder-classifier model: LSTM  $[10]^2$  is used to encode the information from the input word sequence into a fixed-length vector representation, and given this representation, a classifier returns a probability distribution over the values for the dialog state. The input consists of words which were recognized by ASR along with their confidence scores. The words are represented as embeddings, and before they are passed to the LSTM, a single-layer neural network is used to create new word embeddings, accounting for the confidence score. An example of the model applied to a particular input sentence is at Figure 1.

Formally, we have an input neural network that maps the word a and its ASR confidence score r to a joint representation u:

$$u = NN(a, r)$$

The representation u is used by the LSTM encoder along with the previous hidden state  $q_{t-1} = (c_{t-1}, h_{t-1})^3$  to create a new hidden state  $q_t$ :

$$q_t = \operatorname{Enc}(u, q_{t-1})$$

The classifier, represented by a single softmax layer, then maps the hidden state to a probability distribution over all possible values:

$$p_t = \mathcal{C}(h_t)$$



**Fig. 1.** A demonstration of the LecTrack 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 its hidden state is used to feed to the state component classifiers.

Put together, these components make up LecTrack, which maps an input ASR word and score sequence into a sequence of dialog state estimates

**LecTrack** : 
$$(a_1, r_1)..., (a_n, r_n) \to p_1, ..., p_n$$
  
 $\forall i \in 1, ..., n : q_i = (c_i, h_i) = \text{Enc}(\text{NN}(a_1, r_1), q_{i-1})$   
 $\forall i \in 1, ..., k : p_i = \text{C}(h_i)$ 

where *n* is the length of the input sequence.

#### 3.2. Improvements

In this subsection, we describe the modifications we make to the original tracker [11].

Note that contrary to the original model, the new model described in this paper is factored by dialog state components. We empirically found that it converges faster and more reliably, and it also makes the model simpler and standard (because now the model is a standard multi-class classification, as opposed to a multi-target (tracking multiple state components at once) classification before, which allows the use of the standard neural-network toolkits available on Internet for implementation). Tracking of more dialog state components is achieved simply by instantiating more LSTM models in parallel.

### 3.2.1. Including ASR scores

The original model did not make use of the ASR 1-best hypothesis confidence scores. When the model does not have this information, the only possible way to learn not to trust the input is by learning which word patterns correspond to well-recognized speech and which are typical for erroneously-recognized speech. Therefore, we decided to include the confidence score of the input hypothesis as an additional dimension to each input word embedding, and add one fully-connected non-linear layer between this input and the LSTM, so that the model can learn to transform the embeddings according to the confidence score.

<sup>&</sup>lt;sup>2</sup>Contrary to the original LSTM formulation we use *tanh* activation instead of *sigmoid* for the input gate.

<sup>&</sup>lt;sup>3</sup>The state of a standard LSTM model consists of two components.

#### 3.2.2. Including Transcriptions in Training Data

Our training data are noisy due to ASR errors, and it is a common practice to expand the training set to reduce the noise. We thus decided to mix the ASR 1-best hypotheses with the true manually-transcribed user utterances to form an expanded training set, which should reduce the amount of noise.

#### 3.2.3. Model Averaging

Following [1, 6], where the authors successfully use a simple model averaging strategy to boost the performance of their models, we train 10 different models from 10 different random initializations and average their predictions.

### 3.2.4. Abstracting low-occurring values

Our model has little chance of learning to properly predict state component values that do not occur frequently in the training data set. We thus decided to substitute the ones that occur less than 40 times in the training set by an abstract value. This threshold was chosen empirically from the development set, by looking at how many examples of a particular value are needed to achieve a reasonable accuracy. As a result, we replace each occurrence of such a low-occurring value by an abstract token, e.g. jamaican for *#food1*. Occurrences of the same value are replaced by the same abstract token, and if a different value is encountered we create another abstract token, e.g. *#food2*. For each of these abstract tokens we need to add a new class to the classifier. During tracking, the classifier output is post-processed and these values are substituted back, e.g. prediction *#food1* for *jamaican*.

This modification makes the tracker able to track values that it has never seen in the training data, by manually putting them in the abstraction dictionary. Also, because the frequent values are not being abstracted the tracker can still learn ASR error patterns for them. This idea is similar to [6] who abstracted out everything and included both abstracted and nonabstracted features as the input to his model.

#### 3.3. Training

The training criterion is a cross-entropy loss [13] for a dialog example, which is annotated by true lables at some points in time:

$$l(\theta) = -\sum_{t \in Y} \log \operatorname{LecTrack}(a_1, r_1, ..., a_n, r_n)_{y_1}^t$$

Here,  $y_i$  denotes a label for the dialog state at time *i*, and *Y* is a set of times where the label  $y_i$  exists (times that correspond to the end of turns, because in our experiments we have labels only for them). LecTrack $(.)_n^m$  denotes the probability of the *n*-th value at time *m*.

We fit the model using ADAM optimization algorithm [14]. All parameters are initialized randomly from a zero-mean Gaussian with  $\sqrt{\frac{2.0}{d}}$  variance [15] (where *d* is the dimensionality of the input for the layer), apart from the biases of the LSTM forget gates, which are initialized to 1.0 [16].

After each optimization epoch, we monitor the performance<sup>4</sup> 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

### 4.1. Dataset

To train and evaluate our model, we use the DSTC2 [6] data set. The DSTC2 data consists of about 3,000 dialogs from the restaurant information domain, each dialog is 10 turns long on average. The data is split into training, development and test sets.

Our model is incremental and does not explicitly represent turns, but the DSTC2 data set contains only turn-based dialogs. So we treat each dialog in DSTC2 as a sequence of words in time, where the dialog state labels are always attached to the last word of the turn. Ideally we would run the evaluation on a data set where we could also measure the incremental capabilities of the tracker, but to the best of our knowledge, no such data set is publicly available yet, and we leave the collection and experimentation on such a data set for future work.

In our experiments, the word embeddings have 170 dimensions, the input network has 300 output units with ReLU on top, LSTM encoder has 100 cells, and we train using full network unrolling in time in mini-batches of 10 dialogs.

### 4.2. Baseline

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 [6]; we report the results of the *focus* baseline, which was the best among them.

#### 4.3. Data Preprocessing

Each dialog turn consists of the system and the user utterance. We serialize both of them into a stream of pairs (word/token, ASR confidence score) as the input to our model.

**System Utterance Preprocessing:** 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)

<sup>&</sup>lt;sup>4</sup>See Subsection 4.4 for the description of the featured metrics.

into a sequence of three tokens  $t_1, t_2, t_3$ , where  $t_1 = \text{act_type}$ ,  $t_2 = \text{slot_name}$  and  $t_3 = \text{slot_value}$ . For example request (slot=food) is converted into *request*, *slot*, *food*, which the model then sees as a word sequence of length three.

**User Utterance Preprocessing:** For the sake of simplicity, we use only the best live-ASR hypothesis<sup>5</sup> (we refer to it as ASR 1-best) and ignore the rest of the n-best list. It is not obvious how to incorporate more ASR hypotheses into an incremental dialog state tracker in a good way and we plan to address this issue in our future work.

**Out-of-Vocabulary Words:** 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 is replaced by a special out-of-vocabulary token with a probability  $\alpha^6$ . At test time, this token is used to represent all unknown words.

### 4.4. Evaluation Metrics

We follow the DSTC2 methodology [6] and measure the accuracy and L2 norm of the joint slot predictions. The joint predictions are grouped into the following groups: Goals, Requested, Method. The results of each group are reported separately.

For each dialog state component in each dialog, the measurements are taken at the end of each dialog turn<sup>7</sup>.

To asses the effect of the individual improvements over the base model described in Subsection 3.2, we evaluate the following configurations that cumulatively add the different improvements on top of each other:

(base) Base model without the proposed improvements [11].

(score) Include scores.

(transcr) Include transcriptions.

(abstract) Abstract low-occurring classes.

(model avg) Model Averaging.

(dontcare oracle) Don't care oracle (detailed later).

### 4.5. Results

The results of all evaluated LecTrack configurations on the DSTC2 data are summarized in Table 1. The results from DSTC2 are publicly available along with the output of the

trackers on test data set so we try to compare our tracker to [2], which we refer to as RNNTrack, to see in greater detail where are our strengths and weaknesses.

LecTrack's accuracy in its strongest configuration (model avg) is better than the baseline and comes close to the state-of-theart, with the exception of the *Goal* group on the test data set. Note that the model has never seen test data set during training, and development data set was used for selecting the best model seen during the training.

#### 4.5.1. Test set performance difference

We attribute the performance difference for the Goal group between the development and test data sets to a substantial difference between the dialog systems used for the two data sets. The training and development data sets were collected using a different dialog system than the test data set.

The dialog system in the test dataset produces on average about 25% longer system utterances (measured in the number of input words/tokens), which can influence the stability of the LSTM predictions due to increased number of time steps. Also, the distributions of the slot values differ substantially, particularly for the dont care value.

Other trackers from the literature do not have this issue because they all extract features from the complete turn and thus are not influenced by the length of the utterances.

We believe we can address the different lengths of the system utterances by considering the system utterance separately and injecting it into the stream of user words just as a single special token. This way the system input is always long one token, regardless of the dialog system used.

#### 4.5.2. Improvements

Our improvements mostly affect the performance of the tracker on the Goal group. The base tracker already performs well on the Method and Requested groups so the improvements there are modest.

Model averaging proved to provide a substantial improvement in performance. This is in accordance with other approaches that also combine multiple models to produce the final predictions. There is a body of work on compressing the model ensemble back into a single model [17, 18, 19], which appears as an interesting future research direction.

Including true transcriptions in the training yields almost as big improvement to the results as model averaging. The size of the training corpus is quite small (only about 1500 dialogs) and without the transcriptions, some values were never seen in their correct form in the training data. Moreover, with the transcriptions, the tracker is more biased towards learning the correct generalization patterns and can learn to correct some typical ASR mistakes.

Including ASR confidence scores only gives a modest performance improvement on the test set. This fact is surprising because we believe the ASR confidence scores are very

<sup>&</sup>lt;sup>5</sup>There are batch and live ASR results in the DSTC2 data. We use the live ones and refer to them as live-ASR.

<sup>&</sup>lt;sup>6</sup>Throughout this paper we use  $\alpha = 0.1$ .

<sup>&</sup>lt;sup>7</sup>The measurements are taken at the end of each dialog turn, provided the component has already been mentioned in some of the SLU n-best lists in the dialog. 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.

	Development set						Test set					
	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 (base)	0.63	0.74	0.90	0.19	0.96	0.08	0.62	0.75	0.92	0.15	0.96	0.07
LecTrack (score)	0.63	0.73	0.89	0.20	0.96	0.07	0.64	0.73	0.92	0.16	0.96	0.07
LecTrack (transcr)	0.66	0.69	0.90	0.20	0.97	0.07	0.67	0.65	0.92	0.15	0.97	0.07
LecTrack (abstract)	0.67	0.65	0.90	0.20	0.97	0.07	0.68	0.64	0.93	0.14	0.97	0.06
LecTrack (model avg)	0.69	0.71	0.90	0.19	0.97	0.07	0.72	0.64	0.93	0.14	0.97	0.06
LecTrack (dontcare oracle)							0.75	0.50				
turn-based RNN [2]	0.70	0.46	0.92	0.14	0.97	0.06	0.77	0.35	0.94	0.10	0.98	0.04
state-of-the-art [1]	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.

important for tracking, otherwise the tracker does not have a clear signal about the correctness of the user utterance.

#### 4.5.3. Slot Food

The slot *food* is arguably the most difficult slot for the tracker because it takes 91 values and is frequently talked about. Therefore, we chose it for a more detailed analysis of the tracker's results.

The most frequent value in the slot *food* is dont care value, which is the result of the user saying "I don't care" after the system prompted him for the type of food he wants. However, the decision whether the user does not care about food or something else is dependent on the system prompt, and the tracker must make use of this information. Our system achieves 81% accuracy for the dont care value, whereas RNNTrack [2] achieves 91%. This suggests that our tracker is not able to properly learn the dependency between the system and user utterances. A brief manual examination of other prediction errors confirms this.

The dont care values makes up 25% of the correct labels in the test data but only 15% in the development data, which is another reason for the difference in performance between the two datasets. Indeed, when we treat RNNTrack as an oracle to provide the dont care and null predictions (for all slots, not just slot *food*), we beat the baseline and come close to state-of-the art (LecTrack (dontcare oracle) line in Table 1).

## 5. RELATED WORK

The only other incremental dialog trackers known to us are [20, 21]. In [20] the tracking is performed by a parsing algorithm with a manually prepared grammar. In [21], 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 modeling system which keeps track of confidence scores from semantic parses of the input; these are produced by a grammar-based semantic interpreter with a hand-coded context-free grammar. Unlike our system, both discussed trackers require handcrafted grammar and an explicit semantic representation of the input.

Using RNN for dialog state tracking has been proposed before [2, 22]. The dialog state tracker in [2] 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. They also use a similar abstraction of low-occurring values (they call the technique "tagged n-gram features"), which should result in better generalization on rare but well-recognized values.

We use only 1-best ASR hypothesis and achieve near state-of-the-art results, while the other tracking models from the literature [1, 2, 3, 5] typically use the whole ASR/SLU n-best list as an input.

# 6. CONCLUSION

We presented new improvements in our LecTrack incremental LSTM-based dialog state tracker [11], which make the tracker close to state-of-the-art results on the DSTC2 data set. The tracker works incrementally word-by-word and does not need a separate SLU component. The largest improvement was achieved by including the transcriptions in the training data set, and by using an ensemble of models. Minor improvements were brought by including ASR hypothesis scores and value abstraction. We also demonstrated that it is enough to use 1-best hypothesis only to achieve near state-of-the-art results in dialog state tracking on DSTC2 data set.

In future, we would like to investigate why the ASR hypothesis confidence score does not play a bigger role in our model, what techniques to employ to reduce the need for the model averaging, and how to use the tracker in a real incremental dialog system.

## 7. REFERENCES

- 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, 2014, p. 282.
- [2] M. Henderson, B. Thomson, and S. J. Young, "Wordbased Dialog State Tracking with Recurrent Neural Networks," in *Proceedings of SIGdial*, 2014.
- [3] Byung-Jun Lee, Woosang Lim, Daejoong Kim, and Kee-Eung Kim, "Optimizing generative dialog state tracker via cascading gradient descent," in 15th Annual Meeting of the SIGDD, 2014, p. 273.
- [4] Ronnie W Smith, "Comparative error analysis of dialog state tracking," in 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2014, p. 300.
- [5] Kai Sun, Lu Chen, Su Zhu, and Kai Yu, "The sjtu system for dialog state tracking challenge 2," in 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2014, p. 318.
- [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, 2014, p. 263.
- [7] Ondrej Dušek, Ondrej Plátek, Lukáš Žilka, and Filip Jurcícek, "Alex: Bootstrapping a spoken dialogue system for a new domain by real users," in 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2014, p. 79.
- [8] Stephanie Young, Milica Gasic, Blaise Thomson, and John D Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.
- [9] Ye-Yi Wang, Li Deng, and Alex Acero, "Spoken language understanding," *Signal Processing Magazine*, *IEEE*, vol. 22, no. 5, pp. 16–31, 2005.
- [10] Sepp Hochreiter and Jürgen Schmidhuber, "Long shortterm memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [11] Lukas Zilka and Filip Jurcicek, "LecTrack: Recurrent Neural Network Dialog State Tracker," in *Proceedings* of the TSD 2015 Conference (to appear), 2015.
- [12] Lukas Zilka, David Marek, Matej Korvas, and Filip Jurcicek, "Comparison of bayesian discriminative and generative models for dialogue state tracking," in *Proc. of the SIGDIAL 2013 Conf.*, 2013.

- [13] Reuven Y Rubinstein and Dirk P Kroese, The crossentropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning, Springer Science & Business Media, 2004.
- [14] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," .
- [15] David Sussillo, "Random walks: Training very deep nonlinear feed-forward networks with smart initialization," *CoRR*, vol. abs/1412.6558, 2014.
- [16] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever, "An empirical exploration of recurrent network architectures," pp. 2342–2350.
- [17] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.
- [18] Cristian BuciluÇŐ, Rich Caruana, and Alexandru Niculescu-Mizil, "Model compression," in *Proceedings* of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006, pp. 535–541.
- [19] Jimmy Ba and Rich Caruana, "Do deep nets really need to be deep?," in Advances in Neural Information Processing Systems, 2014, pp. 2654–2662.
- [20] Mikio Nakano, Noboru Miyazaki, Jun-ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata, "Understanding unsegmented user utterances in real-time spoken dialogue systems," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 200–207.
- [21] Gabriel Skantze and David Schlangen, "Incremental dialogue processing in a micro-domain," in *Proc. of the 12th Conf. of EC-ACL*. Association for Computational Linguistics, 2009, pp. 745–753.
- [22] M. Henderson, B. Thomson, and S. J. Young, "Deep Neural Network Approach for the Dialog State Tracking Challenge," in *Proceedings of SIGdial*, 2013.