Dissertation Proposal

Dialog Management with Deep Neural Networks

Lukáš Žilka

Institute of Formal and Applied Linguistics
Faculty of Matematics and Physics
Charles University in Prague

Abstract. This document is a dissertation proposal submitted in partial fulfillment of the requirements for the doctoral exams. Its purpose is to define the dissertation goals and summarize initial experiments. We propose a novel approach to dialog state tracking in spoken dialog systems based on long short-term memory recurrent neural neural networks. The proposed model allows incremental, word-by-word dialog state tracking and is trainable from a corpus of annotated dialogs. The experiments show that the model achieves competitive performance on sub-tasks and suggest ways for improvement. Future work proposes extensions to the model and a way to use it for dialog state tracking with large knowledge-bases.

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*: P(food, price_range) The dialog state tracker also needs to deal with speech recognition errors and tries to reduce their impact on the dialog (*Williams et al., 2013*).

The standard mode of operation of a spoken dialog system is turn-by-turn, where the system always waits for the user to stop talking and only then generates its response. Also the system assumes its responses are spoken to the user as a whole without interruption. This brings simplicity to the spoken dialog system architecture, because the system can be implemented as a pipeline without feedback loops, but it also introduces some limitations. Particularly, people are used to say feedback things during the system's response as a sign of understanding, surprise or not-understanding, like "yes", "ok", "really", or "oh, wait". Also, sometimes only partial information provided by the system is enough for the user, and she will interrupt the system in the middle of its utterance to ask a follow-up question or otherwise continue with the dialog. The turn-by-turn mode of operation does not allow for utilizing these clues. We will refer to the more advanced mode of operation, where the system processes the user's input

incrementally as it is coming in and also generates the response incrementally, as word-by-word. This mode of operation allows the dialog system to be more natural and responsive with the user, because it can interpret the various signs the user gives during the system's responses and react more precisely to what the user says. A simple example of a dialog system working in this advanced mode can be found in (*Skantze and Schlangen, 2009*).

The state-of-the-art dialog state trackers (*Williams, 2014, Henderson et al., 2014b, Lee et al., 2014, Smith, 2014, Sun et al., 2014*) 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 (*Henderson et al., 2014a*). However, they possess two undesirable traits. First, they can only track the dialog state turn-by-turn (as opposed to the more complicated word-by-word approach), which limits the responsiveness of the dialog system. And, some of the trackers rely on the results from a spoken language understanding (SLU) component (*Wang et al., 2005*), which brings an additional component into the dialog system that needs to be trained and tuned. In this thesis proposal we aim to address these problems.

The contribution of this thesis proposal is a novel approach to dialog state tracking. 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.

Our approach is based on the modern deep learning techinques, particularly the long-short term memory recurrent neural network (LSTM RNN) (*Hochreiter and Schmidhuber, 1997*). We have chosen this approach due to several reasons: First, LSTMs were shown to be effective for learning sequence mappings in automatic speech recognition (*Graves and Schmidhuber, 2005*), machine translation (*Sutskever et al., 2014*), protein structure prediction (*Sønderby and Winther, 2014*), and many other sequence classification tasks. The length of the sequences successfully modelled by LSTMs is comparable to the length of 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 (*Gülçehre and Bengio, 2013*). The success of LSTM on multiple complicated and diverse tasks promises to be exploitable also in dialog state tracking.

The thesis proposal is organized as follows. First, we give an overview of the relevant spoken dialog systems and neural networks literature (Section 2). In Section 3, the model of the LSTM dialog state tracker is described along with the experiments. In Section 4, we discuss the future research directions.

2 Background

This thesis builds upon the work from the fields of statistical spoken dialog systems and deep neural networks.

2.1 Spoken Dialog Systems

A spoken dialog system needs to understand what the user says, process it and provide an answer. Usually the dialog systems are turn-based, meaning that the system listens to the user and only replies when the user stops talking (*Thomson and Young, 2010*). They are commonly built as a pipeline of several components:

1. The automatic speech recognition (ASR) component converts the user's speech to text.
2. The spoken language understanding component converts the text into structured information.
3. The dialog management component updates the dialog state and generates a system action.
4. The structured system action is converted to natural text.
5. The natural text is synthesized and spoken back to the user.

Automatic Speech Recognition The task of an automatic speech recognizer is to decode the user's utterance into a structure called *ASR hypothesis*. Typically the hypothesis is either an 1-best list (*Gorin et al., 1997, Wang et al., 2003*), an n-best list (*He and Young, 2003*), a word confusion network (*Hakkani-Tür et al., 2006*) or a word lattice (*Oerder and Ney, 1993*). Word lattices are the richest but also the most complex structures to work with. It has been shown that using word lattices or word confusion networks in the spoken language understanding yields substantially better results over 1-best hypothesis (*Tür et al., 2002*). The ASR hypothesis is consumed by the other components of the dialog system.

There are open-source and commercial ASR solutions available with varying capabilities and purposes of use. The most pouplar open-source solution is Kaldi (*Povey et al., 2011*), which can be trained out-of-the-box to provide state-of-the-art ASR performance. Google Speech API[1] is a popular commercial solution that is very fast and accurate for general speech and supports a lot of languages, but is not customizable and has unclear licensing conditions. Nuance's[2] commercial recognizers are more flexible but they are paid and closed-source. ISpeech ASR API[3] is also customizable (users can select expected speech type: text messages, voice mail, dictation) but it does not allow full customization and is also paid. AT&T Watson[4] allows users to provide a custom grammar (language model) but its free use is limited.

[1] https://www.google.com/intl/en/chrome/demos/speech.html
[2] http://dragonmobile.nuancemobiledeveloper.com/public/
[3] http://www.ispeech.org/api
[4] http://developer.att.com/apis/speech

Spoken Language Understanding Spoken language understanding (SLU) unit aims to interpret the user's intention from their speech utterance (*Wang et al., 2005*). Particularly it converts the ASR hypothesis to some form of meaning representation. For example "yes" could be represented as "affirm()", or "I want to go to Brno" as "inform(task=find_connection)&inform(to_stop=Brno)" in the meaning representation used in ALEX dialog system (*Dušek et al., 2014*). The meaning representation is arbitrary and adjusted to the particular dialog system. Examples of other representations can be found in (*Skantze, 2008, He and Young, 2003*).

Historically, SLU has been done by manually writing grammars used to fill slots in semantic frames (*Ward and Issar, 1994, Dowding et al., 1993*), but this is expensive and non-flexible, because an expert needs to devise the grammar and then laborously maintain it. As the complexity of the domain grows the maintanence effort grows much more. Instead, statistical approach to SLU has been adopted, where SLU is viewed as a pattern recognition problem

$$\hat{M} = \arg\max_M \mathrm{P}(M|W) \tag{1}$$

where we are looking for the best meaning representation M of the input given the ASR hypothesis W. The main benefit of this approach is the ability of the model to leverage the training data to improve its performance. HMM can be used to model the joint probability of the words and meaning representation given the acoustic signal (*Pieraccini et al., 1992, Pieraccini and Levin, 1992*). Hidden Vector State model that learns stack operations to parse the user's utterance in a hierarchical way was proposed (*He and Young, 2003*). It is particularly appealing for its ability to infer the hierarchy in the user's utterances without explicit hierarchical annotations. Markov Logic Networks were used for SLU by (*Meza-Ruiz et al., 2008*), where first-order logic formulaes are used as templates to instantiate complex Markov network that model the slot-value representation. In (*Zettlemoyer and Collins, 2007*) an algorithm that learns to parse the text input into lambda-calculus expressions using extended combinatory categorial grammar is given.

Dialog Management Dialog management (DM) component directly influences how natural and intelligent will the dialog system be perceived among its users. Its task is to come up with the next action of the system given the previous progress of the dialog. It usually consists of two parts: dialog state tracking, and dialog policy.

Dialog state tracking watches the dialog progress and keeps track of important information in form of the dialog state, or as a probability distribution over all possible dialog states called the belief state. Usually, the dialog state is a structure that contains several components that take several values, for example a component "food" that can take values "chinese", "indian", or "thai", and a component "area" that take values "north" and "south", and only the information needed for the dialog policy is tracked.

Dialog policy acts upon the dialog state or the belief state to generate the next action. There are handcrafted approaches to the dialog policy (*Pieraccini and Huerta, 2005, Skantze, 2008*), and approaches that automatically learn from observing rewards (*Levin et al., 2000, Walker, 2000, Lemon et al., 2006, Thomson and Young, 2010*).

Bayesian Networks Bayesian networks can be used for the belief state tracking (*Pulman et al., 1996, Williams, 2007, Bui et al., 2006, Thomson and Young, 2010*). The belief b is updated as

$$b'(s) = \frac{1}{Z} \cdot P(o'|s,a) \sum_{s'} P(s|s',a) b(s') \tag{2}$$

where o is the observation, typically the user utterance, s is the dialog state and a is the system action.

Recurrent Neural Network Tracker Using recurrent neural network (RNN) for dialog state tracking has been proposed in (*Henderson et al., 2014b, Henderson et al., 2013*). N-gram features from n-best lists are extracted and used as an input to the RNN model at each turn. The hidden state is updated as

$$h_t = f(x_t, h_{t-1}) \tag{3}$$

where $h_t = (p_t, m_t)$ consists of the distribution over the slot's values in form of the vector p_t and memory m_t.

Ranking-based Tracking The dialog state tracking was posed as a ranking problem by (*Williams, 2014*), where all the possible dialog states given the observed SLU hypotheses so far are enumerated and then ranked using a LambdaMART ranking algorithm. Then the best ranked dialog state is selected as the hypothesis.

Discriminative Model Tracking Generalized linear regression model f was trained to update the belief state

$$b'(s) = f(b(s), a, o) \tag{4}$$

and shown to outperfrom heuristic update rules (*Bohus and Rudnicky, 2006*).

Natural Language Generation The natural language generation (NLG) component generates the natural language utterance of the system action generated by DM. For example, if the system generates the action `confirm(food=chinese)` then the NLG should generate "Did you say chinese food?". One of the simplest approaches is a template-based NLG that has a database of templates, where for each possible system action there is a template to be used (*Dušek et al., 2014*). A more sophisticated approach is a data-driven model that given a corpus of (system action; its surface realization; alignment) tuples learns to generate the natural language utterances (*Mairesse et al., 2010*).

Speech Synthesis Speech synthesis takes the natural language utterance generated by NLG and transforms it to speech signal. Two main approaches are used: 1.) concenative speech synthesis (*Campbell and Black, 1997, Masuko et al., 1996*), and 2.) statistical parametric speech synthesis (*Zen et al., 2009*) Users still consider the concenative approach to be better, but the statistical parametric speech synthesis is catching up quickly (*Zen et al., 2009*).

The concenative speech synthesis (also called unit-selection synthesis) uses a database of speech segments, from which it selects some and concatenates them together to form the speech signal. The input utterance is converted to a sequence of speech units that describe how it should be pronounced. For each speech unit the synthesizer finds the most appropriate segment from the database, in terms of the similarity to the sound to be synthesized speech unit and the smoothness of concatenation with the other segments.

Statistical parametric speech synthesis directly models the acoustics of the speech. During the training of the model, linguistic l and acoustic features a are extracted from the training data and a conditional model of $p(a|l)$ found. During synthesis the best acoustic features for the given linguistics features are found $\hat{a} = \arg\max_a p(a|l)$, and used for generating the resulting waveform. HMM (*Yoshimura et al., 1999*), DNN (*Ze et al., 2013*), and RNN (*Zen and Sak, 2015*) were used as the conditional models.

2.2 Deep Neural Networks

Neural networks are known to be universal function approximators (*Hornik et al., 1989*), thus are able to model arbitrary measurable function with arbitrary accuracy given enough capacity. This makes them powerful for statistical modeling. The never-ending advances in the computational speed brought computer architectures capable of coping with previously inconceivable amounts of data. Because neural networks have enough capacity to leverage this data, the new neural networks applications were able to achieve impressive results on various tasks.

New applications of neural networks in almost all fields where machine learning was applied before yielded the new state-of-the-art results or came close to the state-of-the-art performance with much simpler models (e.g. automatic speech recognition (*Graves and Schmidhuber, 2005*), machine translation (*Sutskever et al., 2014*), image recognition (*Karpathy and Fei-Fei, 2014*), natural language processing (*Socher et al., 2012*), protein structure prediction (*Sønderby and Winther, 2014*)).

Also because the fundamental algorithm that underlies the deep learning, the gradient descent algorithm, is very simple and computationally tractable, it allowed the researches to build bigger and better models. Proving once again that simple models with more data beat more complex models (*Halevy et al., 2009*).

The research in the deep neural networks is also fueled by the available open-source deep learning toolkits like Theano (*Bastien et al., 2012*), Torch7 (*Collobert*

et al., 2011), or Caffe (*Jia et al., 2014*), that provide a good programming infrastructure for building custom neural network models and for testing hypotheses fast. With these toolkits it is very easy to share experimental code, try out new models, and replicate experiments.

The currently used neural networks are called deep because of their architecutre. It consists of several layers of parametrized computational units, which are stacked in a deep hierarchy. We can classify the deep neural networks by their architecture into several conceptual classes:

- *fully-connected networks* – their layers are densely connected and their input is just a vector x that is mapped to output $y = f(x) = g(W \cdot x)$, where $g(.)$ is some activation function like tanh, σ, or ReLU, and $W \in \mathbb{R}^2$ is a matrix of parameters
- *convolutional networks* – their layers are connected through a convolutional kernel; the convolutional kernel behaves like a small fully-connected layer applied at the different positions of the input simultaniously $y_1 = f(x_{1:|k|})$, ..., $y_{n-|k|+1} = f(x_{n-|k|+1:n})$; the result $[y_1, ..., y_{n-|k|+1}]$ is usually very large so it is transformed by a pooling layer into a smaller space $y = \text{pool}(y_1, ..., y_{n-|k|+1})$; in the most general case the pooling layer sub-samples its inputs, for example selects the maximum values for each dimension of the input vectors
- *recurrent neural networks* – are almost the same as the fully-connected networks, but with the difference that they are applied to a sequence rather than a single input x_1, x_2, ..., one by one, and some of the results of the computation performed by computing an output for the input x_t (e.g. the values of the hidden layer) are passed on to the function's application on the next element of the sequence x_{t+1}; formally we have $(y_i, h_i) = f(x_i, h_{i-1})$, where h_i denotes the results of the computation passed from the previous step, and y_i is an output

However, this classification is not strict and almost every model from the literature is a combination of these classes. We aimed to give an intuitive feel for the different classes of architecture rather than their rigorous description, which can be found in the literature (*Graves, 2013*).

Long Short-Term Memory Networks An important type of the recurrent neural networks is the long short-term memory (LSTM) (*Hochreiter and Schmidhuber, 1997*) model that aims to overcome some of the practical drawbacks of the standard RNN model. Standard RNN model suffers from the vanishing gradient problem (*Bengio et al., 1994*) during training, and also the network forgets information very fast. LSTM's special architecture aims to overcome this, and

has the following form:

$$\mathbf{f} : X \times U \to U \tag{5}$$

$$f(x_t, u_{t-1}) = u_t = \begin{pmatrix} h_t \\ c_t \end{pmatrix} \tag{6}$$

$$i_t = \sigma(\mathrm{W}_{(\mathrm{i})} \cdot \begin{pmatrix} x_t \\ h_{t-1} \\ c_{t-1} \end{pmatrix} + b_{(i)}) \tag{7}$$

$$f_t = \sigma(\mathrm{W}_{(\mathrm{f})} \cdot \begin{pmatrix} x_t \\ h_{t-1} \\ c_{t-1} \end{pmatrix} + b_{(f)}) \tag{8}$$

$$x'_t = \tanh(\mathrm{W}_{(\mathrm{x})} \cdot \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_{(x)}) \tag{9}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot x'_t \tag{10}$$

$$o_t = \sigma(\mathrm{W}_{(\mathrm{o})} \cdot \begin{pmatrix} x_t \\ h_{t-1} \\ c_t \end{pmatrix} + b_{(o)}) \tag{11}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{12}$$

The vectors i_t, f_t, o_t represent the input, forget and output gates, respectivelly, x'_t is the modulated input, the vector c_t is the cell state and the vector h_t denotes the output, jointly denoted by u_t.

3 LSTM Dialog State Tracker

In this section we will describe our dialog state tracking model based on the long short-term memory units. Our model fits in the big picture of a spoken dialog system described in subsection 2.1 as a spoken language understanding and dialog state tracker together. Thus it connects between the automatic speech recognizer and the dialog policy, getting the ASR hypotheses as its input and provides the belief state as its output to the dialog policy.

Particularly, the task of our tracker 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.

3.1 Dialog State

In this thesis, 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. Each component $c_i \in C_i =$

$\{v_1, ..., v_{n_i}\}$ takes one of the n_i values. Our dialog state tracker maintains a probability distribution over s_t factorized by the dialog state components:

$$P(s_t|w_1, ..., w_t) = \prod_i p(c_i|w_1, ..., w_t; \theta) \tag{13}$$

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

3.2 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 produces a probability distribution for each of the dialog state components.

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. $C^{(food)}$, $C^{(area)}$, ...).

The model of the encoder Enc is a LSTM RNN (Section 2.2). In case of a recursive application of Enc, we write

$$Enc(w_1, ..., w_n, h, c) \tag{14}$$

instead of

$$Enc(w_n, ...Enc(w_1, h, c)) \tag{15}$$

to simplify the notation. The model of the classifier C is a single-layer neural network with rectified linear units in the hidden layer and the softmax output layer. The architecture is illustrated in Figure 1.

The encoder and classifier form together the LSTM dialog state tracker that we call *LecTrack*:

$$\textbf{LecTrack} : a_1, ..., a_n \rightarrow p_1, ..., p_k$$
$$\forall i \in 1, ..., k : p_i = \mathrm{C_i}(\mathrm{Enc}(E \cdot a_1, ..., E \cdot a_n, h_0, c_0))$$

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, $w_i = E \cdot a_i$. The literature suggests many ways for obtaining the embedding matrix E (*Mikolov et al., 2013, Kim, 2014, Stratos et al.,*), but

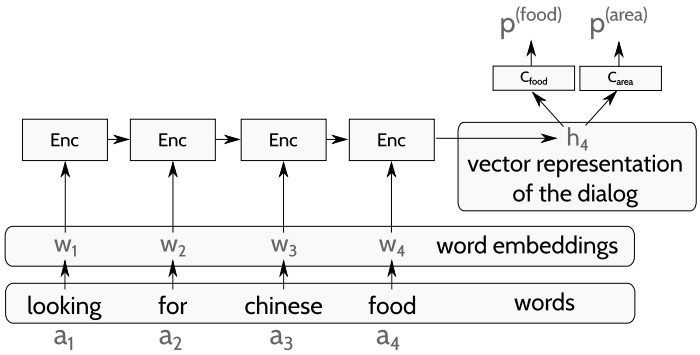

Fig. 1. A demonstration of the LSTM Dialog State Tracker applied to a user utterance "looking for chinese food". The encoding LSTM model *Enc* is sequentially applied to each input word and at the end, its hidden state is used to feed to the state component classifiers.

in the experiments in this thesis, we treat it just as another set of parameters for the sake of simplicity.[5]

3.3 Training

The model is trained using the standard cross-entropy criterion (*Rubinstein and Kroese, 2004*) in the vanilla stochastic gradient descent scenario (*Bottou, 2010*):

$$l(a_1, ..., a_n, y_1, ..., y_k; \theta) = \sum_{i=0}^{k} \log \text{LecTrack}(a_1, ..., a_n)^i_{y_i} \tag{16}$$

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

After each optimization epoch, we monitor the performance [6] 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.

3.4 Experiments

Dataset To train and evaluate our model, we use the DSTC2 (*Henderson et al., 2014a*) 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

[5] Informal experiments with different types of initialisaion of word embedings, such as using word2vec (*Mikolov et al., 2013*) embedings estimated from a large out-of-domain corpus, did not suggest any advantage over randomly initialised embeddings in this relatively limited domain. Therefore, we decided not to explore this direction further.

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

domain, each dialog is 10 turns long on average. The data is split into training, development and test sets. 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.

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 (*Henderson et al., 2014a*) and we report the results of the *focus* baseline, which was the best among them.

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[7]. The resulting tokens are added to the vocabulary of the model side by side with the words from the user utterances, but in such a way that they are still differentiated from the user's words of the same form. For example, the system act `inform(food=chinese)` results in the tokens `(inform, food)` and `chinese`. But the user utterances also contain the word `chinese`, so if we put both the system tokens and the user words into one vocabulary, they would be mixed. We care to keep them separate because we empirically found that mixing user words and system tokens to be harmful to the performance of the tracker.

User Input: For the sake of simplicity, we use only the best live-ASR[8] 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.

[7] The whole system utterances could be used and result in a similar performance under the measured metrics, but also increase the training time of the model and contain no more information than the stringified dialog acts. Therefore, we only use the flattened dialog acts.

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

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.

Experimental Methodology We follow the DSTC2 methodology (*Henderson et al., 2014a*) 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 each dialog turn*, provided the component has already been mentioned in some of the SLU n-best lists in the dialog[9]

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 close 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 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

[9] 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.

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
(*Williams, 2014*)[10]	**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.

value	baseline	LecTrack
chinese	0.53	0.82
indian	0.49	0.79
korean	0.67	0.93
asian oriental	0.54	0.86
dontcare	0.98	0.88
european	0.61	0.80
italian	0.41	0.79
spanish	0.69	0.73
thai	0.14	0.64
	...	
traditinal	0.17	0.17
steakhouse	0.14	0.07
romanian	0.35	0.21
german	0.28	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.

3.5 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 (*Sutskever et al., 2014*).

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.

3.6 Related Work

The only incremental dialog system in the literature that we are aware of is (*Skantze and Schlangen, 2009*). In this paper, the authors describe an incremental dialog system for number dictation as a specific instance of their incremental dialog processing framework (*Schlangen and Skantze, 2009*). To track the dialog state, they use a discourse modelling system (*Skantze, 2008*), 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 (*Skantze and Edlund, 2004*) 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 (*Henderson et al., 2014b, Henderson et al., 2013*). The dialog state tracker in (*Henderson et al., 2014b*) 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.

4 Future Work

There are several directions that we would like to explore further.

4.1 LSTM Dialog State Tracker

As suggested by the conducted experiments, the tracker seems to have problems with the under-represented classes. A straightforward way how to address this issue is by replacing the values in the input by placeholders, effectively delexicalizing the output classifier, and then replacing the values back. This will allow us to learn a general model on a somehow aggregated dataset, thus we will have more training data for less parameters, and it will allow us to extend the range of values recognized without having to retrain the model. The cost of this approach is that it will not be able to learn characteristics of specific values, and react on specific value confusions by the recognizer.

Our model does not account for the probability of the 1-best hypothesis, thus it cannot ignore bad recognition results, which results in a bad performance in noisy conditions. One approach to introduce the ASR confidence score into our model is through confidence embeddings. We can bin the confidence scale into several bins, for example bin1=0.0-0.1, bin2=0.1-0.2, etc., and add to each input word embedding a vector that corresponds to the bin of the word's confidence. There are various ways how to do it (vector addition, vector multiplication, or vector concatenation). We will explore these possibilities and see how it influences the model's performance.

Also, we just used the 1-best hypothesis of the ASR, but the literature suggests that using the whole n-best list, the confusion network or the lattice representation of the ASR hypothesis brings improvements in model's recall. Thus we decided to extend our model to use the confusion networks. Ideally we would use the lattice, but because the confusion networks are available in the DSTC2 dataset and simpler to work with we will work with them at the beginning.

4.2 Knowledge Base Tracking

Similarly to Wikipedia that gathers information about the real world in textual form, the Wikidata (*Vrandečić, 2012*) aims to gather the information about the real world in a structured form, in the form of an entity graph. Each entity in the graph represents something from the real world, for example a person, a country, an animal, and the edges between the entities represent the relationships these entities have among themselves. So far the dialog systems have always worked with databases in the order of thousands of entities, however, with Wikidata, the magnitude grows to millions entities and billions relationships. Thus the dialog state tracking in dialog systems that will work with such a huge database will

need a special approach to be computationally tractable. Inspired by (*Sutskever et al., 2014*), we will try to modify our LSTM tracker to handle such databases well. The basic idea is to replace the output classifier layer for an output generator, effectively letting our state tracker to generate the answer word-by-word as people would do, rather than as a million-dimensional vector. Then some standard information retrieval approach can retrieve the entity of interest from the database.

The main challenge here will be to come up with training data that will allow us to train such a tracker in a way that for different things from the real world it learns how people refer to them and their relationships.

References

Bastien et al., 2012. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bengio et al., 1994. Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Bohus and Rudnicky, 2006. Bohus, D. and Rudnicky, A. (2006). A k-hypotheses+ other belief updating model. In *Proc. of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems*.

Bottou, 2010. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Bui et al., 2006. Bui, T. H., Poel, M., Nijholt, A., and Zwiers, J. (2006). A tractable ddn-pomdp approach to affective dialogue modeling for general probabilistic frame-based dialogue systems.

Campbell and Black, 1997. Campbell, N. and Black, A. W. (1997). Prosody and the selection of source units for concatenative synthesis. In *Progress in speech synthesis*, pages 279–292. Springer.

Collobert et al., 2011. Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

Dowding et al., 1993. Dowding, J., Gawron, J. M., Appelt, D., Bear, J., Cherny, L., Moore, R., and Moran, D. (1993). Gemini: A natural language system for spoken-language understanding. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 54–61. Association for Computational Linguistics.

Dušek et al., 2014. Dušek, O., Plátek, O., Žilka, L., and Jurcícek, F. (2014). 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*, page 79.

Gorin et al., 1997. Gorin, A. L., Riccardi, G., and Wright, J. H. (1997). How may i help you? *Speech communication*, 23(1):113–127.

Graves, 2013. Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves and Schmidhuber, 2005. Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Gülçehre and Bengio, 2013. Gülçehre, Ç. and Bengio, Y. (2013). Knowledge matters: Importance of prior information for optimization. *arXiv preprint arXiv:1301.4083*.

Hakkani-Tür et al., 2006. Hakkani-Tür, D., Béchet, F., Riccardi, G., and Tur, G. (2006). Beyond asr 1-best: Using word confusion networks in spoken language understanding. *Computer Speech & Language*, 20(4):495–514.

Halevy et al., 2009. Halevy, A., Norvig, P., and Pereira, F. (2009). The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12.

He and Young, 2003. He, Y. and Young, S. (2003). A data-driven spoken language understanding system. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 583–588. IEEE.

Henderson et al., 2014a. Henderson, M., Thomson, B., and Williams, J. (2014a). The second dialog state tracking challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 263.

Henderson et al., 2013. Henderson, M., Thomson, B., and Young, S. J. (2013). Deep Neural Network Approach for the Dialog State Tracking Challenge. In *Proceedings of SIGdial*.

Henderson et al., 2014b. Henderson, M., Thomson, B., and Young, S. J. (2014b). Word-based Dialog State Tracking with Recurrent Neural Networks. In *Proceedings of SIGdial*.

Hochreiter and Schmidhuber, 1997. Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hornik et al., 1989. Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Jia et al., 2014. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Karpathy and Fei-Fei, 2014. Karpathy, A. and Fei-Fei, L. (2014). Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*.

Kim, 2014. Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Lee et al., 2014. Lee, B.-J., Lim, W., Kim, D., and Kim, K.-E. (2014). Optimizing generative dialog state tracker via cascading gradient descent. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 273.

Lemon et al., 2006. Lemon, O., Georgila, K., Henderson, J., and Stuttle, M. (2006). An isu dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, pages 119–122. Association for Computational Linguistics.

Levin et al., 2000. Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing, IEEE Transactions on*, 8(1):11–23.

Mairesse et al., 2010. Mairesse, F., Gašić, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., and Young, S. (2010). Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561. Association for Computational Linguistics.

Masuko et al., 1996. Masuko, T., Tokuda, K., Kobayashi, T., and Imai, S. (1996). Speech synthesis using hmms with dynamic features. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 389–392. IEEE.

Meza-Ruiz et al., 2008. Meza-Ruiz, I. V., Riedel, S., and Lemon, O. (2008). Spoken language understanding in dialogue systems, using a 2-layer markov logic network: Improving semantic accuracy. *Semantics and Pragmatics of Dialogue (LONDIAL)*, page 191.

Mikolov et al., 2013. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Oerder and Ney, 1993. Oerder, M. and Ney, H. (1993). Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 119–122. IEEE.

Pieraccini and Huerta, 2005. Pieraccini, R. and Huerta, J. (2005). Where do we go from here? research and commercial spoken dialog systems. In *6th SIGdial Workshop on Discourse and Dialogue*.

Pieraccini and Levin, 1992. Pieraccini, R. and Levin, E. (1992). Stochastic representation of semantic structure for speech understanding. *Speech Communication*, 11(2):283–288.

Pieraccini et al., 1992. Pieraccini, R., Tzoukermann, E., Gorelov, Z., Levin, E., Lee, C.-H., and Gauvain, J.-L. (1992). Progress report on the chronus system: Atis benchmark results. In *Proceedings of the workshop on Speech and Natural Language*, pages 67–71. Association for Computational Linguistics.

Povey et al., 2011. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit.

Pulman et al., 1996. Pulman, S. G. et al. (1996). Conversational games, belief revision and bayesian networks. In *Computational Linguistics in the Netherlands*. Citeseer.

Rubinstein and Kroese, 2004. Rubinstein, R. Y. and Kroese, D. P. (2004). *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media.

Schlangen and Skantze, 2009. Schlangen, D. and Skantze, G. (2009). A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718. Association for Computational Linguistics.

Skantze, 2008. Skantze, G. (2008). Galatea: A discourse modeller supporting concept-level error handling in spoken dialogue systems. In *Recent Trends in Discourse and Dialogue*, pages 155–189. Springer.

Skantze and Edlund, 2004. Skantze, G. and Edlund, J. (2004). Robust interpretation in the higgins spoken dialogue system. In *COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*.

Skantze and Schlangen, 2009. Skantze, G. and Schlangen, D. (2009). Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 745–753. Association for Computational Linguistics.

Smith, 2014. Smith, R. W. (2014). Comparative error analysis of dialog state tracking. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 300.

Socher et al., 2012. Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Sønderby and Winther, 2014. Sønderby, S. K. and Winther, O. (2014). Protein secondary structure prediction with long short term memory networks. *arXiv preprint arXiv:1412.7828*.

Stratos et al., . Stratos, K., Kim, D.-k., Collins, M., and Hsu, D. A spectral algorithm for learning class-based n-gram models of natural language.

Sun et al., 2014. Sun, K., Chen, L., Zhu, S., and Yu, K. (2014). The sjtu system for dialog state tracking challenge 2. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 318.

Sutskever et al., 2014. Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Thomson and Young, 2010. Thomson, B. and Young, S. (2010). Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Tür et al., 2002. Tür, G., Wright, J. H., Gorin, A. L., Riccardi, G., and Hakkani-Tür, D. Z. (2002). Improving spoken language understanding using word confusion networks. In *INTERSPEECH*. Citeseer.

Vrandečić, 2012. Vrandečić, D. (2012). Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 1063–1064. ACM.

Walker, 2000. Walker, M. A. (2000). An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, pages 387–416.

Wang et al., 2003. Wang, Y.-Y., Acero, A., and Chelba, C. (2003). Is word error rate a good indicator for spoken language understanding accuracy. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 577–582. IEEE.

Wang et al., 2005. Wang, Y.-Y., Deng, L., and Acero, A. (2005). Spoken language understanding. *Signal Processing Magazine, IEEE*, 22(5):16–31.

Ward and Issar, 1994. Ward, W. and Issar, S. (1994). Recent improvements in the cmu spoken language understanding system. In *Proceedings of the workshop on Human Language Technology*, pages 213–216. Association for Computational Linguistics.

Williams et al., 2013. Williams, J., Raux, A., Ramachandran, D., and Black, A. (2013). The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Williams, 2007. Williams, J. D. (2007). Applying pomdps to dialog systems in the troubleshooting domain. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, pages 1–8. Association for Computational Linguistics.

Williams, 2014. Williams, J. D. (2014). Web-style ranking and slu combination for dialog state tracking. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 282.

Yoshimura et al., 1999. Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., and Kitamura, T. (1999). Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis.

Ze et al., 2013. Ze, H., Senior, A., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7962–7966. IEEE.

Zen and Sak, 2015. Zen, H. and Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Acoustics, Speech and Signal Processing*.

Zen et al., 2009. Zen, H., Tokuda, K., and Black, A. W. (2009). Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064.

Zettlemoyer and Collins, 2007. Zettlemoyer, L. S. and Collins, M. (2007). Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.