Deep Neural Network Approach for the Dialog State Tracking Challenge

Matthew Henderson, Blaise Thomson and Steve Young
Department of Engineering,
University of Cambridge, U.K.
{mh521, brmt2, sjy}@eng.cam.ac.uk

Abstract

While belief tracking is known to be important in allowing statistical dialog systems to manage dialogs in a highly robust manner, until recently little attention has been given to analysing the behaviour of belief tracking techniques. The Dialogue State Tracking Challenge has allowed for such an analysis, comparing multiple belief tracking approaches on a shared task. Recent success in using deep learning for speech research motivates the Deep Neural Network approach presented here. The model parameters can be learnt by directly maximising the likelihood of the training data. The paper explores some aspects of the training, and the resulting tracker is found to perform competitively, particularly on a corpus of dialogs from a system not found in the training.

1 Introduction

Statistical dialog systems, in maintaining a distribution over multiple hypotheses of the true dialog state, are able to behave in a robust manner when faced with noisy conditions and ambiguity. Such systems rely on probabilistic tracking of dialog state, with improvements in the tracking quality being important in the system-wide performance in a dialog system (see e.g. Young et al. (2009)).

This paper presents a Deep Neural Network (DNN) approach for dialog state tracking which has been evaluated in the context of the Dialog State Tracking Challenge (DSTC) (Williams, 2012a; Williams et al., 2013)[1].

Using Deep Neural Networks allows for the modelling of complex interactions between arbitrary features of the dialog. This paper shows improvements in using deep networks over networks with fewer hidden layers. Recent developments in speech research have shown promising results using deep learning, motivating its use in the context of dialog (Hinton et al., 2012; Li et al., 2013).

This paper presents a technique which solves the task of outputting a sequence of probability distributions over an arbitrary number of possible values using a single neural network, by learning tied weights and using a form of sliding window. As the classification task is not split into multiple sub-tasks for a given slot, the log-likelihood of the tracker on training data can be directly maximised using gradient ascent techniques.

The domain of the DSTC is bus route information in the city of Pittsburgh, but the presented technique is easily transferable to new domains, with the learned models in fact being domain independent. No domain specific knowledge is used, and the classifier learned does not require knowledge of the set of possible values. The tracker performed highly competitively in the 'test4' dataset, which consists of data from a dialog system not seen in training. This suggests the model is capable of capturing the important aspects of dialog in a robust manner without overtuning to the specifics of a particular system.

Most attention in the dialog state belief tracking literature has been given to generative Bayesian network models (Paek and Horvitz, 2000; Thomson and Young, 2010). Few trackers have been published using discriminative classifiers, a notable exception being Bohus and Rudnicky (2006). An analysis by Williams (2012b) demonstrates how such generative models can in fact degrade belief tracking performance relative to a simple baseline. The successful use of discriminative models for belief tracking has recently been alluded to by Williams (2012a) and Li et al. (2013), and was a prominent theme in the results of the DSTC.

[1] More information on the DSTC is available at http://research.microsoft.com/en-us/events/dstc/

2 The Dialog State Tracking Challenge

This section describes the domain and methodology of the Dialog State Tracking Challenge (DSTC). The Challenge uses data collected during the course of the Spoken Dialog Challenge (Black et al., 2011), in which participants implemented dialog systems to provide bus route information in the city of Pittsburgh. This provides a large corpus of real phonecalls from members of the public with real information needs.

Set	Number of calls	Notes
train1a	1013	Labelled training data
train1b&c	10619	Same dialog system as train1a, but unlabelled
train2	678	Similar to train1*
train3	779	Different participant to other train sets
test1	765	Very similar to train1* and train2
test2	983	Somewhat similar to train1* and train2
test3	1037	Very similar to train3
test4	451	System not found in any training set

Table 1: Summary of datasets in the DSTC

Table 1 summarises the data provided in the challenge. Labelled training sets provide labels for the caller's true goal in each dialog for 5 slots; route, from, to, date and time.

Participants in the DSTC were asked to report the results of their tracker on the four test sets in the form of a probability distribution over each slot for each turn. Performance was determined using a basket of metrics designed to capture different aspects of tracker behaviour Williams et al. (2013). These are discussed further in Section 4.

The DNN approach described here is referred to in the results of the DSTC as 'team1/entry1'.

3 Model

For a given slot s at turn t in a dialog, let $S_{t,\,s}$ denote the set of possible values for s which have occurred as hypotheses in the SLU for turns $\leq t$. A tracker must report a probability distribution over $S_{t,\,s} \cup \{\text{other}\}$ representing its belief of the user's true goal for the slot s. The probability of 'other' represents the probability that the user's true goal is yet to appear as an SLU hypothesis.

A neural network structure is defined which gives a discrete distribution over the $|S_{t,\,s}|+1$ values, taking the turns $\leq t$ as input.

Figure 1 illustrates the structure used in this approach. Feature functions $f_i\,(t,\,v)$ for $i = 1 \ldots M$ are defined which extract information about the value v from the SLU hypotheses and machine actions at turn t. A simple example would be $f_{\text{SLU}}\,(t,\,v)$, the SLU score that s=v was informed at turn t. A list of the feature functions actually used in the trial is given in Section 3.1. For notational convenience, feature functions at negative t are defined to be zero:
$\forall i\, \forall v, t' < 0 \Rightarrow f_i\,(t',\,v) = 0$.

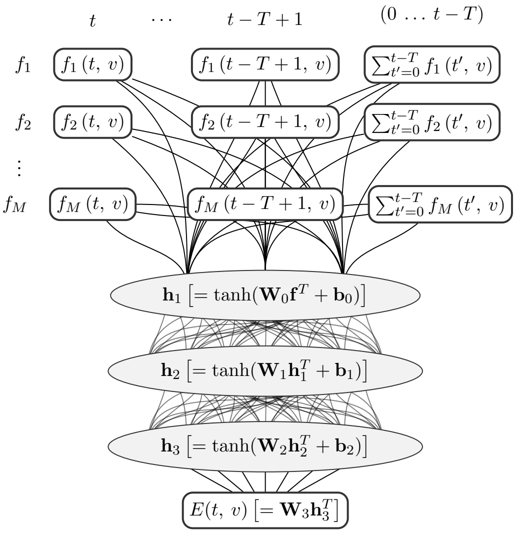
t
$\cdots$
$t-T+1$
$(0\,\ldots\,t-T)$
f_1
f_2
$\vdots$
f_M
$f_1\,(t,\,v)$
$f_1\,(t-T+1,\,v)$
$\sum_{t'=0}^{t-T} f_1\,(t',\,v)$
$f_2\,(t,\,v)$
$f_2\,(t-T+1,\,v)$
$\sum_{t'=0}^{t-T} f_2\,(t',\,v)$
$f_M\,(t,\,v)$
$f_M\,(t-T+1,\,v)$
$\sum_{t'=0}^{t-T} f_M\,(t',\,v)$
$\mathbf{h}_1\;[=\tanh(\mathbf{W}_0\mathbf{f}^T+\mathbf{b}_0)]$
$\mathbf{h}_2\;[=\tanh(\mathbf{W}_1\mathbf{h}_1^T+\mathbf{b}_1)]$
$\mathbf{h}_3\;[=\tanh(\mathbf{W}_2\mathbf{h}_2^T+\mathbf{b}_2)]$
$E(t,\,v)\;[=\mathbf{W}_3\mathbf{h}_3^T]$

Figure 1: The Neural Network structure for computing $E\,(t,\,v) \in \mathbb{R}$ for each possible value v in the set $S_{t,\,s}$. The vector $\mathbf{f}$ is a concatenation of all the input nodes.

The input layer for a given value v is fixed in size by choosing a window size T, such that the feature functions are summed for turns $\leq t-T$. The input layer therefore consists of $(T \times M)$ input nodes set to $f_i\,(t',\,v)$ for $t' = t-T+1 \ldots t$ and $i = 1 \ldots M$, and M nodes set to $\sum_{t'=0}^{t-T} f_i\,(t',\,v)$ for $i = 1 \ldots M$.

A feed-forward structure of hidden layers is chosen, which reduces to a single node denoted $E\,(t,\,v)$. Each hidden layer introduces a weight matrix $\mathbf{W}_i$ and a bias vector $\mathbf{b}_i$ as parameters, which are independent of v but possibly trained separately for each s. The equations for each layer in the network are given in Figure 1.

The final distribution from the tracker is:

$$\begin{aligned} \mathbb{P}\,(s=v) &= {}^{e^{E(t,\,v)}}\!/Z \\ \mathbb{P}\,(s \notin S_{t,\,s}) &= {}^{e^{B}}\!/Z \\ Z &= e^B + \sum_{v' \in S_{t,\,s}} e^{E\left(t,\,v'\right)} \end{aligned}$$

where B is a new parameter of the network, independent of v and possibly trained separately for each slot s.

3.1 Feature Functions

As explained above, a feature function is a function $f\,(t,\ v)$ which (for a given dialog) returns a real number representing some aspect of the turn t with respect to a possible value v. A turn consists of a machine action and the subsequent Spoken Language Understanding (SLU) results. The functions explored in this paper are listed below:

1. **SLU score**; the score assigned by the SLU to the user asserting $s{=}v$.
2. **Rank score**; $1/r$ where r is the rank of $s{=}v$ in the SLU n-best list, or 0 if it is not on the list.
3. **Affirm score**; SLU score for an `affirm` action if the system just confirmed $s{=}v$.
4. **Negate score**; as previous but with `negate`.
5. **Go back score**; the score assigned by the SLU to a `goback` action matching $s{=}v$.
6. **Implicit score**; $1-$ the score given in the SLU to a contradictory action if the system just implicitly confirmed $s{=}v$, otherwise 0.
7. **User act type**; a feature function for each possible user act type, giving the total score of the user act type in the SLU. Independent of s & v.
8. **Machine act type**; a feature function for each possible machine act type, giving the total number of machine acts with the type in the turn. Independent of s & v.
9. **Cant help**; 1 if the system just said that it cannot provide information on $s{=}v$, otherwise 0.
10. **Slot confirmed**; 1 if $s{=}v'$ was just confirmed by the system for some v', otherwise 0.
11. **Slot requested**; 1 if the value of s was just requested by the system, otherwise 0.
12. **Slot informed**; 1 if the system just gave information on a set of bus routes which included a specific value of s, otherwise 0.

4 Training

The derivatives of the training data likelihood with respect to all the parameters of the model can be computed using back propagation, i.e. the chain rule. Stochastic Gradient Descent with mini-batches is used to optimise the parameters by descending the negative log-likelihood in the direction of the derivatives (Bottou, 1991). Termination is triggered when performance on a held-out development set stops improving.

Each turn t and slot s in a dialog for which $|S_{t,\,s}| > 0$ provides a non-zero summand to the total log-likelihood of the training data. These instances may be split up by slot to train a separate network for each slot. Alternatively the data can be combined to learn a slot independent model. The best approach found was to train a slot independent model for a few epochs, and then switch to training one model per slot (see Section 4.4).

This section presents experiments varying the training of the model. In each case the parameters are trained using all of the labelled training sets. The results are reported for test4 since this system is not found in the training data. They are therefore unbiassed and avoid overtuning problems.

The ROC curves, accuracy, Mean Reciprocal Rank (MRR) and l2 norm of the tracker across all slots are reported here. (A full definition of the metrics is found in Williams et al. (2013).) These are computed throughout using statistics at every turn t where $|S_{t,\,s}| > 0$ (referred to as 'schedule 2' in the terminology of the challenge.) Table 2 and Figure 3 in Appendix A show these metrics. The 'Baseline' system ('team0/entry1' in the challenge), considers only the top SLU hypothesis so far, and assigns the SLU confidence score as the tracker probability. It does not therefore incorporate any belief tracking.

4.1 Window Size

The window size, T, was varied from 2 to 20. T must be selected so that it is large enough to capture enough of the sequence of the dialog, whilst ensuring sufficient data to train the weights connecting the inputs from the earlier turns. The results suggest that $T = 10$ is a good compromise.

4.2 Feature Set

The features enumerated in Section 3.1 were split into 4 sets. $F_1 = \{1\}$ includes only the SLU scores; $F_2 = \{1,\ ...,\ 6\}$ includes feature functions which depend on the user act and the value; $F_3 = \{1,\ ...,\ 8\}$ also includes the user act and machine act types; and finally $F_4 = \{1,\ ...,\ 12\}$ includes functions which depend on the system act and the value. The results clearly show that adding more and more features in this manner monotonically increases the performance of the tracker.

4.3 Structure

Some candidate structures of the hidden layers (h_1, h_2, ...) were evaluated, including having no hidden layers at all, which gives a logistic regression model. In Table 2 the structure is represented as a list giving the size of each hidden layer in turn.

Three layers in a funnelling [20, 10, 2] configuration is found to outperform the other structures. The l2 norm is highly affected by the use of deeper network structure, suggesting it is most useful in tweaking the calibration of the confidence scores.

	ROC	Acc.	MRR	l2
Baseline				
	⊖	0.5841	0.7574	0.5728
Window Size				
T =2	●	0.6679	0.8044	0.5405
5	△	0.6875	0.8191	**0.5164**
10	▲	**0.6922**	**0.8207**	0.5331
15	◇	0.6718	0.8107	0.5352
20	◆	0.6817	0.8190	0.5174
Feature Set				
F_1	●	0.5495	0.7364	0.6838
F_2	△	0.6585	0.7954	0.6631
F_3	▲	0.6823	0.8134	0.5525
F_4	◇	**0.6922**	**0.8207**	**0.5331**
Structure				
$[]$	●	0.6751	0.8074	0.5658
$[50]$	△	0.6679	0.8046	0.5450
$[20]$	▲	0.6656	0.8060	0.5394
$[50, 10]$	◇	0.6645	0.8045	0.5404
$[20, 2]$	◆	0.6543	0.7952	0.5514
$[20, 10, 2]$	⬠	**0.6922**	**0.8207**	**0.5331**
Initialisation				
Separate	●	0.6907	0.8206	0.5472
Single Model	△	0.6779	0.8111	0.5570
Shared Init.	▲	**0.6922**	**0.8207**	**0.5331**

Table 2: Results for variant trackers described in Section 4. By default, we train using the shared initialisation training method with $T = 10$, all the features enumerated in Section 3.1, and 3 hidden layers of size 20, 10 and 2.

4.4 Initialisation

The three methods of training alluded to in Section 4 were evaluated; training a model for each slot without sharing data between slots (*Separate*); training a single slot independent model (*Single Model*); and training for a few epochs a slot independent model, then using this to initialise the training of separate models (*Shared Initialisation*).

The method of shared initialisation appears to be the most effective, scoring the best on accuracy, MRR and l2. Training in this manner is particularly beneficial for slots which are under represented in the training data, as it initiates the parameters to sensible values before going on to specialise to that particular slot.

5 Performance in the DSTC

A DNN tracker was trained for entry in the DSTC. Training used T=10, the full feature set, a $[20,\ 10,\ 2]$ hidden structure and the shared initialisation training method. Other parameters such as the learning rate and regularisation coefficient were tweaked by analysing performance on a held out subset of the training data. All the labelled training data available was used. The tracker is labelled as ‘team1/entry1’ in the DSTC.

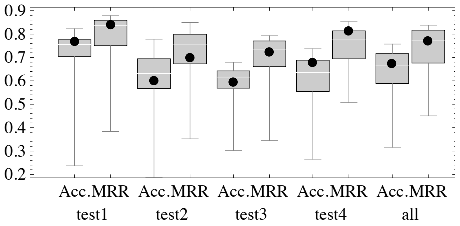

Figure 2: Accuracy and MRR of the 28 entries in the DSTC for all slots. Boxplots show minimum, maximum, quartiles and the median. Dark dot is location of the entry presented in this paper (DNN system).

The DNN approach performed competitively in the challenge. Figure 2 summarises the performance of the approach relative to all 28 entries in the DSTC. The results are less competitive in test2 and test3 but very strong in test1 and test4.

The performance in test4, dialogs with an unseen system, was probably the best because the chosen feature functions forced the learning of a general model which was not able to exploit the specifics of particular ASR+SLU configurations. Features which depend on the identity of the slot-values would have allowed better performance in test2 and test3, allowing the model to learn different behaviours for each value and learn typical confusions. It would also have been possible to exploit the *system-specific* data available in the challenge, such as more detailed confidence metrics from the ASR.

For a full comparison across the entries in the DSTC, see Williams et al. (2013). In making comparisons it should be noted that this team did not alter the training for different test sets, and submitted only one entry.

6 Conclusion

This paper has presented a discriminative approach for tracking the state of a dialog which takes advantage of deep learning. While simple Gradient Ascent training was tweaked in this paper using the ‘Shared Initialisation’ scheme, a possible promising future direction would be to further experiment with more recent methods for training deep structures e.g. initialising the networks layer by layer (Hinton et al., 2006).

Richer feature representations of the dialog contribute strongly to the performance of the model. The feature set presented is applicable across a broad range of slot-filling dialog domains, suggesting the possibility of using the models across domains without domain-specific training data.

A ROC Curves

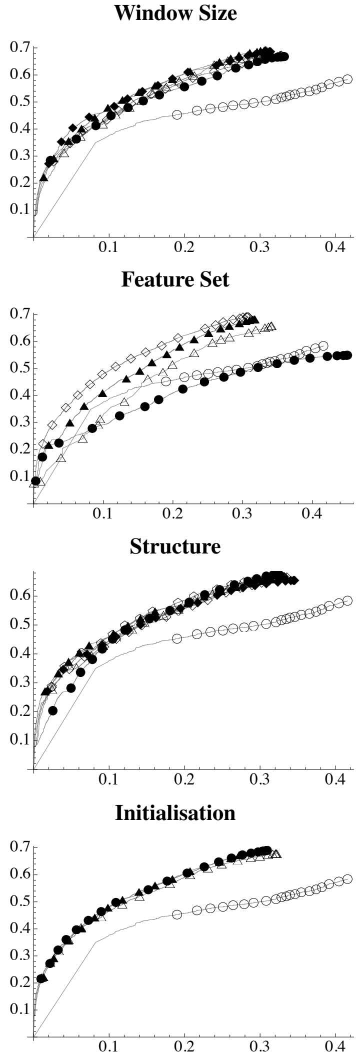

Figure 3: ROC (Receiver Operating Characteristic) Curves x-axis and y-axis are false acceptance and true acceptance respectively. Lines are annotated as per Table 2.

Acknowledgments

The authors would like to thank the organisers of the DSTC. The principal author was funded by a studentship from the EPSRC.

References

Alan W. Black, Susanne Burger, Alistair Conkie, Helen Wright Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, Jason D. Williams, Kai Yu, Steve Young, and Maxine Eskenazi. 2011. Spoken dialog challenge 2010: Comparison of live and control test results. In *SigDIAL*.

Dan Bohus and Alex Rudnicky. 2006. A K-hypotheses+ Other Belief Updating Model. *Proc. of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems*.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*, Nîmes, France. EC2.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*.

Deng Li, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael L Seltzer, Geoff Zweig, Xiaodong He, Jason D. Williams, Yifan Gong, and Alex Acero. 2013. Recent Advances in Deep Learning for Speech Research at Microsoft. In *ICASSP*.

Tim Paek and Eric Horvitz. 2000. Conversation as action under uncertainty. In *The Sixteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The Dialogue State Tracking Challenge. In *SigDIAL*.

Jason D. Williams. 2012a. A belief tracking challenge task for spoken dialog systems. In *NAACL HLT 2012 Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data*. Association for Computational Linguistics.

Jason D. Williams. 2012b. Challenges and opportunities for state tracking in statistical spoken dialog systems: Results from two public deployments. *J. Sel. Topics Signal Processing*, 6(8):959–970.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2009. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*.