MACHINE LEARNING FOR DIALOG STATE TRACKING: A REVIEW

Matthew Henderson

Google

matthen@google.com

ABSTRACT

Spoken dialog systems help users achieve a task using natural language. Noisy speech recognition and ambiguity in natural language motivate statistical approaches that model distributions over the user's goal at every step in the dialog. The task of tracking these distributions, termed Dialog State Tracking, is therefore an essential component of any spoken dialog system. In recent years, the Dialog State Tracking Challenges have provided a common test-bed and evaluation framework for this task, as well as labeled dialog data. As a result, a variety of machine-learned methods have been successfully applied to Dialog State Tracking. This paper reviews the machine-learning techniques that have been adapted to Dialog State Tracking, and gives an overview of published evaluations. Discriminative machine-learned methods outperform generative and rule-based methods, the previous state-of-the-art.

Index Terms— dialog systems, dialog state tracking, machine learning

1. INTRODUCTION

Spoken interaction promises a natural, effective, and hands- and eyes-free method for users to interact with computer systems. Voice-based computer systems, called *spoken dialog systems*, allow users to interact using speech to achieve a goal. Efficient operation of a spoken dialog system requires a component that can track what has happened in a dialog, incorporating system outputs, user speech, context from previous turns, and other external information. The output of this Dialog State Tracking (DST) component is then used by the *dialog policy* to decide what action the system should take next, and so DST is essential for the final performance of a complete system.

A variety of machine-learned discriminative methods for DST have been proposed in recent years, particularly since the Dialog State Tracking Challenges (DSTCs) have provided labelled dialog state tracking data and a common evaluation framework and test-bed for comparing techniques among research groups [1–4].

This paper presents an overview of recent work on DST, with a focus on machine-learned methods. Section 2 reproduces the definition of DST given in the second and third DSTCs. Rule-based and generative methods for DST found in the literature are presented in section 3. Section 4 presents how machine learning has been applied to the problem, and presents some of the techniques that give the state of the art in DST. Section 5 presents results from the second DSTC, where machine-learned methods came top in all metrics. Finally section 6 concludes.

2. PROBLEM DEFINITION

Spoken dialog systems that help a user complete a task are typically framed in terms of *slots* and are termed *slot-based* dialog systems. The slots and possible slot values of a slot-based dialog system specify its domain, i.e. the scope of what it can talk about and the tasks that it can help the user complete. The slots inform the set of possible actions the system can take, the possible semantics of the user utterances, and the possible dialog states.

The second and third DSTCs formulated a definition of slot-based dialog domains (sometimes referred to as *ontologies*), which is relatively general to dialog systems that help the user search a database of entities by specifying constraints [2, 3]. This is the definition adopted here. The set of all slots S is composed of two not-necessarily disjoint subsets – the informable slots S_{inf}, and the requestable slots S_{req}, such that $S = S_{\text{inf}} \cup S_{\text{req}}$. Informable slots are attributes of the entities in the database that the user may use to constrain their search. Requestable slots are attributes that users may ask the value of, but may not necessarily be allowed to specify a value for as a constraint. A typical example of a requestable slot that is not informable is the phone number, which the user may ask for but would not give as a constraint. For each slot $s \in S$, the set of possible values for the slot is denoted V_s.

The term *dialog state* loosely denotes a full representation of what the user wants at any point from the dialog system. The dialog state comprises all that is used when the system makes its decision about what to say next. Figure 1 gives an illustrative example of dialog state labels in the domain of restaurant information used in DSTC2, where the dialog state at a given turn consists of:

- The **goal constraint** for every informable slot $s \in S_{\text{inf}}$. This is an assignment of a value $v \in V_s$ that the user is specifying as a constraint, or a special value *Dontcare*, which means the user has no preference, or *None*, which means the user is yet to specify a valid goal for the slot.
- A set of **requested slots**, the current list of slots that the user has asked the system to inform. This is a subset of S_{req}.
- The current dialog **search method**, one of several values that encode how the user is trying to interact with the dialog system including *by constraints* (the user is attempting to issue a constraint), *by alternatives* (the user is requesting alternative suitable entities), and *finished* (the user wants to end the call).

In DSTC1 the dialog state consists only of goal constraints, and once a slot s had been constrained to a value it was considered fixed for the rest of the dialog [1].

The task of DST is to output a distribution over all of these components of the dialog state at each turn. The distributions output by a dialog state tracker are sometimes referred to as the tracker's *belief* or the *belief state*. The tracker may have access to the whole

		SLU output	Dialog State
Turn 1			
a:	What part of town did you have in mind? *request(area)*	0.7 *null()* 0.2 *inform(food='north african')* 0.1 *inform(area=north)*	**goal constraints** *area=north* **requested slots** – **search method** *by constraints*
u:	The North area *inform(area=north)*		
Turn 2			
a:	Which part of town? *request(area)*	0.8 *inform(area=north, pricerange=cheap)* 0.2 *inform(area=north)*	**goal constraints** *area=north* *pricerange=cheap* **requested slots** – **search method** *by constraints*
u:	A cheap place in the North *inform(area=north, pricerange=cheap)*		
Turn 3			
a:	Da Vinci Pizzeria is a cheap place in the North. *inform(name='Da Vinci Pizzeria', area=north, pricerange=cheap)*	0.7 *reqalts(area=west)* 0.3 *reqalts()*	**goal constraints** *area=west* *pricerange=cheap* **requested slots** – **search method** *by alternatives*
u:	Do you have anything else, but in the West? *requestAlternatives(area=west)*		
Turn 4			
a:	Cocum is a cheap place in the West. *inform(name=cocum, area=west, pricerange=cheap)*	0.6 *request(phone, address)* 0.3 *request(phone)* 0.1 *request(address)*	**goal constraints** *area=west* *pricerange=cheap* **requested slots** *address* *phone* **search method** *by alternatives*
u:	What is their number and address? *request(address, phone)*		

Fig. 1: Example dialog from the DSTC 2 restaurant information domain, annotated with Dialog State labels. The left column shows the actual system action a and user action u. The second column shows example SLU M-best hypotheses and their scores. In the DSTC2 data, up to 10 SLU M-best hypotheses are given. Turn 2 demonstrates an addition to the goal constraints. In turn 3, the goal constraint for the *area* slot changes, and the search method changes from *by constraints* to *by alternatives*. The last turn demonstrates a non-empty set of requested slots.

history of the dialog up to the current turn, including previous system acts, Automatic Speech Recognition (ASR) hypotheses, Spoken Language Understanding (SLU) hypotheses, and external information such as databases and models of previous interactions. The ASR hypotheses are typically given as an N-best list of sentences, a word confusion network [5], or a lattice; the SLU typically outputs an M-best list of interpretations.

3. RULE-BASED AND GENERATIVE DIALOG STATE TRACKING

3.1. Rule-based Dialog State Tracking

Early spoken dialog systems used hand-crafted rules for DST, keeping a single top hypothesis for each component of the dialog state, often with corresponding confidence scores [6, 7]. Such systems require no data to implement, and provide an accessible method for system designers to incorporate knowledge of the dialog domain. However, such rules are unable to make use of the entire ASR N-best or SLU M-best lists, and so do not account for uncertainty in a principled way. In dialog, uncertainty arises not only from errors in recognizing speech, but also from ambiguities inherent in natural language, and so it is necessary for DST to track robustly multiple competing hypotheses. The importance of maintaining distributions over what the participants of a dialog desire was argued for by Pulman in 1996, who proposed modeling dialog as a conversational game accounting for uncertainty using probabilistic frameworks [8].

The current dominant DST methods for estimating distributions of multiple competing dialog states can be split into two categories – *generative* models that jointly model both the inputs (typically the SLU hypotheses) and the dialog state, and *discriminative* models that directly capture the conditional probability over dialog states, having observed the dialog up to a certain point. These two approaches are described in the following sections.

Rule-based approaches have also been proposed for the task of tracking multiple hypotheses, using a small set of hand-crafted rules to compute dialog state distributions given observations from the SLU [9–12].

3.2. Generative Methods

Generative approaches model dialog as a dynamic Bayesian network where the true dialog state s and the true user action u are treated as unobserved random variables [13]. Bayesian inference is used to give an updated distribution over s given the system act a, and a noisy observation of u, written $\tilde{\mathbf{u}}$. Figure 2 shows the structure of the dynamic Bayesian network.

Early generative approaches used exact inference, enumerating all possible dialog states [14, 15]. This is quadratic in the number of possible states, and is intractable as the number of states can be very large. As a result, two approximations are typically used; either maintaining a *beam* of candidate dialog states [16–18], or assuming conditional independence between components of the dialog state [19–24].

Typically these techniques use parameters optimized by the system designer, though there are methods to optimize the parameters using corpora of dialog data [25, 26]. Another proposition is to use a secondary step to post-process the output of a generative model [27].

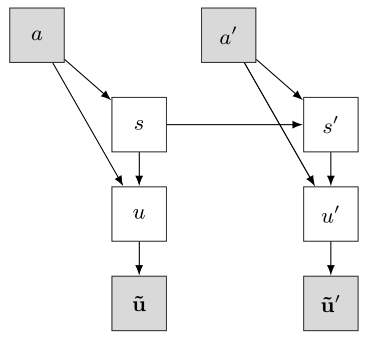

Fig. 2: Dynamic Bayesian network structure for DST, showing two consecutive time steps (dialog turns). A prime ($'$) denotes a variable in the following time step. The true dialog state s is dependent on the previous state and the machine action a. The true user's action u depends on both s and a, and a noisy observation of this is given by $\tilde{\mathbf{u}}$, whose distribution is inferred from the SLU. Observed nodes are shaded grey. The connections in the graph encode the conditional independence relationships in the joint distribution of the random variables.

4. MACHINE-LEARNED DISCRIMINATIVE DIALOG STATE TRACKING

There are several deficiencies of generative models, some of which were identified in [28]. Generative approaches must model all the correlations in the input features, so they cannot easily exploit arbitrary but potentially useful features of the dialog history. Any such features must be included into the dynamic Bayesian network, requiring new structures and dependencies to be learned or specified.

Generative models used for DST make many independence assumptions in order to make the models tractable, in terms of both computational complexity and the number of parameters that must be learned. In particular, the current implementations do not model ASR and SLU error correlations, instead assuming independence for simplicity.

Discriminative models address these issues by directly modeling the distribution over the dialog state given arbitrary and possibly correlated input features [29]. While the parameters of generative models are typically hand-crafted, the model parameters for discriminative approaches are tuned using machine learning and labeled dialog data.

Discriminative methods exploiting machine learning can be split into two categories – static classifiers that encode dialog history in the input features, and sequence models that explicitly model dialog as a sequential process.

4.1. Static Classifiers

The task of DST can be considered as specifying the distribution:

$$P(s_t \mid o_0,\ \ldots,\ o_t) \tag{1}$$

where s_t is the dialog state at the current turn t, and $o_0,\ \ldots,\ o_t$ is the sequence of observations from the SLU, ASR and machine actions up to and including the current turn. One key issue with this approach is that it is necessary to extract a fixed dimensional feature vector to summarise a sequence of arbitrary length observations $o_0,\ \ldots,\ o_t$.

Given a feature representation, this distribution has been modeled using a variety of machine-learned discriminative models including maximum entropy linear classifiers [28–32], neural networks [11, 33–35] and web-style ranking models [36].

Applying machine-learned classifiers in the first DSTC presented a difficulty, as there was no consistent set of possible values V_s for each slot s across all dialogs in the data. Instead of labeling the true values for the user's goal constraint, each SLU hypothesis was labeled as correct or incorrect. A dialog state tracker therefore had to assign probabilities to all the SLU hypotheses given so far in a dialog, a growing list of candidates. This has been achieved by using *value-specific* feature representations, and tying weights in the model.

Even when the set of possible values V_s for each slot s is known, such as in DSTC 2 and 3, value-specific feature representations have been key for learning accurate models. Some slots may have values that occur very infrequently as the constraint, such as food-type in DSTC2. Using value-specific features and sharing weights allows for better generalization to unseen or infrequent dialog states [37].

This section presents in some detail the features used by Metallinou et al. [30], which is fairly representative of this class of techniques. Consider tracking the correct value for the user's goal constraint for a given slot $s \in S_{\text{inf}}$. Metallinou et al. extract a feature vector $\mathbf{f}$ from the observation and a set of vectors $\mathbf{f}_v$ for each v that has appeared in the SLU hypotheses so far. The $\mathbf{f}_v$ features are intended to convey information about the correctness of the v hypothesis for the goal constraint on slot s. The $\mathbf{f}$ features are of general interest, and are also used to calculate the probability of the *None* hypothesis.

The features $\mathbf{f}$ provide aggregate information about the dialog history, including:

- the size of the SLU M-best list in the latest turn
- the entropy of the SLU M-best list in the latest turn
- the posterior scores from the ASR in the latest turn
- the mean and variance of the SLU M-best list lengths in the dialog history
- the number of distinct SLU results obtained so far and the entropy of the corresponding probabilities

For a value v, the value specific features $\mathbf{f}_v$ include:

- information about the latest turn:
  - the rank and probability score of the $s = v$ hypothesis in the SLU list
  - the difference between the probability score of the $s = v$ hypothesis and the top scoring hypothesis in the SLU list
- information from the dialog history:
  - the number of times the hypothesis $s = v$ has appeared in the SLU so far
  - the number of times the hypothesis $s = v$ has appeared in the SLU so far at each particular rank
  - the sum and average of the confidence scores of SLU results containing $s = v$
  - the number of possible past user negations or confirmations of $s = v$
- information about likely ASR errors and confusability:

- estimates for the likelihood of $s = v$ being correctly identified by the SLU, estimated on held-out training data
- a similar estimate for the prior probability of $s = v$ appearing in an SLU M-best list, and at specific rank positions in the list
- similar estimates for how likely $s = v$ is to be confused with other hypotheses $s = v'$

Note that the dialog history is summarized using sums, averages, and other statistics. An alternative approach is to use a sliding window that runs over features $\mathbf{f}_{v,\,t}$, which depend on the value v and turn t [33].

4.2. Sequence Models

Static classifiers models the sequential nature of the input by using feature functions that attempt to summarise the sequence. It is also possible to use sequence models that inherently model the sequential nature of the problem. A Conditional Random Field (CRF) can be used to do sequential labeling of the dialog [38, 39]. A linear-chain CRF is used to learn the conditional distribution:

$$P(s_0,\, \ldots,\, s_t \mid o_0,\, \ldots,\, o_t) \tag{2}$$

This is then marginalized to give the distribution for the latest state s_t. Features like those used in section 4.1 to extract information about hypotheses for the current turn are used, except these are calculated for all turns in the history.

One key issue with using CRF for DST is that continuous features must be quantized, as the CRF modeling techniques in all published work on DST so far require discrete features.

Recurrent Neural Networks (RNNs) have been proposed as sequential models that are able to deal with high dimensional continuous input features [37, 40]. Notably, such models can be applied to operating directly on the distribution over sentences from the ASR, without requiring SLU and instead using weighted n-gram features from the ASR N-best list. This is termed *word-based* DST, and has two key benefits: first, it removes the need for feature design, and the risk of omitting an important feature, which can degrade performance unexpectedly [36]. Second, it avoids the engineering task of building a separate SLU model.

There are several machine-learning tricks that are required to reliably train word-based RNNs for DST, which are reproduced here from Henderson et al. [37, 40, 41]. Firstly, it is useful to exploit *delexicalized* features. These are value-specific features that match values in the domain against the n-grams in the ASR. As with value-specific SLU-based features, these help both with generalization to unseen dialog states, and in the case of flexible domains.

RNNs that operate on high-dimensional n-gram features can be hard to train, and so benefit from careful initialization. Performance is improved if the weights multiplying the feature vectors are initialized with weights learned by training a denoising auto-encoder. Improvements are also found by first learning a slot-independent model for tracking goal constraints, then adapting this model for each slot.

In the case where the set of possible values V_s for each slot s is known, it is possible to improve the RNN model by adding layers that directly modify the output softmax layer given the raw n-gram features. This allows the model to associate new n-grams with dialog states, without relying on delexicalized features.

Long Short-Term Memory (LSTM) networks [42] have also been applied to the task of word-based DST, modeling not only the sequence of turns, but also the sequence of words within a turn [43]. This is attractive as it allows for incremental DST, though it is restricted to using 1-best ASR results.

4.3. System Combination

Many of the competitive results in the DSTCs were achieved by combining multiple models [12, 31, 36, 37, 40]. In Henderson et al. [2], the trackers submitted to DSTC2 from all participants were combined using *stacked ensemble learning* [44] to give a new tracker that performed significantly better than any individual tracker. This suggests that by exploiting the strengths of individual models, ensemble learning can provide better quality output.

The simplest method of system combination is *score averaging*, i.e. averaging the output distributions of multiple trackers. This has the advantage that no extra training data is required. If each tracker's output is correct more than half the time, and if the errors made by the tracker are not correlated, then this technique is guaranteed to improve the performance (since the majority vote will be correct in the limit). For example, in the RNN-based method for DST proposed by Henderson et al. [40], at least 6 individual RNNs are trained with varying hyper-parameters (such as regularization, learning rates, hidden layer sizes), and the resulting outputs are averaged to give the final distributions. In this case the final combined output consistently outperforms any individual model.

4.4. Modeling Joint Distributions

The number of possible dialog states can be enormous; every informable slot added to the domain multiplies this number by the number of possible values it can take. To deal with the resulting inevitable sparsity of examples in the training data, and the large number of possible states to consider, machine-learned methods must exploit the structure of the DST problem.

One approach is to constrain the dialog states to score to a set of candidate states [36]. In this case, value-specific features allow for generalization to unseen states, and the model is able to directly output full structured joint distributions over the most likely candidate states. In this work, an SLU component is used both to generate candidate dialog states, and provide features used in the final classifier.

Another popular approach is to assume conditional independence between components of the dialog state. For example, the joint distribution over goal constraints can be estimated as a product over distributions for each individual slot. This was the approach taken by all but one participant in DSTC2.

However, in general the independence assumption among slots is not valid. For example in DSTC1, which concerned bus information, the destination is correlated with the chosen bus route. And in DSTC3, which concerned tourist information, it only made sense to constrain by *food=italian* if the user also had the constraint *type=restaurant* and not *type=pub*.

RNNs for tracking individual slots can be combined using a special softmax combination layer [45]. This allows for tuning a factorized model to output an accurate joint distribution, giving improvements in the DSTC3 data. The basic idea is to order the slots, and add a recurrent layer that selects an assignment for each slot, given the assignments selected so far [41].

4.5. Unsupervized Adaptation and Generalization

As with all applications of machine learning, generalization from the training set is a key problem for DST [1, 2]. The ability of methods to generalize to unseen data was explicitly tested in DSTC3. In this challenge, no training data in the testing dialog domain was released, though a large amount of data in a similar but smaller domain was available.

Rule-based methods that do not require training data, or which have very few parameters to learn, can give strong generalization ability [12, 24]. Also recall that value-specific features give inherent generalization ability, as they allow tracking of unseen dialog states; in theory it is not necessary to see a value in the training data, so long as these features can be calculated for the value when it does appear. Delexicalized features for RNN word-based tracking can be used to share training data across disparate domains, improve performance in each individual domain [46]. If a small amount of training exists, then multi-domain learning can be exploited when adapting to the new domain [32].

The RNN framework allows for tuning parameters with an unsupervized cost function that tries to propagate information backward through the dialog. This cost function minimizes the cross entropy between predictions of consecutive turns, weighted by the entropy of a baseline model, which serves to propagate information from later in the dialog so far as the baseline model was uncertain. This has been shown to improve DST performance when adapting an out-of-domain model to a new domain, without requiring any training data [37].

5. EVALUATION

Almost all of the machine-learned methods for DST described in the previous section were developed for and evaluated in the DSTCs. Table 1 gives a summary of the DST techniques and some of the results of the DSTC2 evaluation. As well as rule-based baselines described in [2], a generative dynamic Bayesian network is also included in the table under *Bayesian net.* (as reported in [41]). The results from DSTC2 are presented as it is the most simple set-up of the three challenges and the most amenable to machine learning; DSTC1 had multiple test sets and hundreds of evaluation metrics, and DSTC3 had no in-domain training data. Nevertheless, full results for all the DSTCs can be found in the relevant summary papers [1–3] and in the overview paper [4].

The domain of DSTC2 was restaurant information. In this domain, crowd-sourced users were asked to find a restaurant matching a given set of constraints, and to find some information about the suggested restaurant. In some cases, a matching restaurant would not exist, and the user was asked to relax their constraints to find an alternative. The informable slots S_{inf} were food type, part of town, and price range; the requestable slots S_{req} additionally included address, phone number, and post code.

Prior to the DSTC2 evaluation, a training set was published containing 2,118 dialogs with transcriptions, SLU labels and dialog state labels. During the week-long evaluation phase, an unlabeled dataset of 1,117 dialogs was released. Participants were required to run their DST algorithms over this data, and submit the output (distributions over components of the dialog state) for evaluation.

Two metrics were featured in the evaluation. The *accuracy* metric measures the quality of the top hypothesis for each dialog state component, and is equal to the fraction of times the top hypothesis is correct. The *L2* metric measures the quality of the whole distribution, and is equal to the square of the L2 distance between the reported distribution and the label distribution (equal to zero at all points except at the correct label, where it is 1).

The discriminative machine-learned systems notably outperform the generative and rule-based systems. The top accuracy and L2 scores achieved by a generative model were 0.699 and 0.498 respectively [26]. In contrast, a method using web-style ranking classifiers was able to get 0.784 in accuracy [36], and an RNN method achieved 0.346 in L2 score.

Machine-learned methods for DST also performed top in the DSTC1 and DSTC3 evaluations. In no case has a rule-based or generative model achieved best performance in any of the DSTCs. Note however that the *focus* baseline, the top performing rule-based system that uses a very simple method of updating the dialog state [2], outperforms several of the machine-learned methods. This demonstrates a deficiency in generalization from the training data to the test set, particularly since most of these methods were predicted to outperform the focus baseline on the development set [2].

While the web-style ranking classifier obtained the top accuracy figures, it performed notably poorly in terms of the probability qualities as measured by the L2 score. The ranker is trained to rank the correct hypothesis first, but the training procedure does not attempt to further separate the correct and incorrect hypotheses once a training example has been ranked correctly. This leads to high accuracy, but less meaningful confidence scores as measured by the L2 score. The sequential RNN models performed top for this metric, having been trained to maximize the log-probability and so minimize the L2 score. In DSTC1, a sequential CRF model outperformed the static classifier approaches [31]. In DSTC3, a sequential RNN-based method also outperformed the other approaches [37].

Discriminative machine-learned methods are able to incorporate arbitrary, potentially correlated features, such as those derived from the ASR. The top performing methods in table 1 use ASR features, all of which are machine learned; no generative method has been presented capable of doing this.

In general, the DSTCs have brought DST into focus in recent years, and many research teams have improvements over the previous state-of-the-art as measured by the metrics on off-line corpus-based evaluations such as these. However, it is important to assess the performance in an end-to-end dialog system. The RNN-based method for DST presented in [40] was evaluated in a live user trial, and it was found to significantly improve the end-to-end performance relative to a generative baseline method [41]. This evaluation found significant and substantial gains in employing machine-learned RNN based dialog state trackers in every metric evaluating the end-to-end dialog quality, including measures of task completion, and the users' subjective impression that the system understood them well.

6. CONCLUSIONS

The availability of labeled training data for DST has prompted interest in and development of a wide-range of machine learning for the problem. Discriminative machine-learned methods are now the state-of-the art in DST, as demonstrated in the offline corpus-based evaluations of the DSTCs, and in on-line trials with live users.

Top performing approaches use ASR features, exploiting the ability of discriminative methods to use arbitrary features. It is also key to model the dialog as a sequence, either by using a sequential model such as an CRF or RNN, or by using rich features of the dialog history.

Poor generalization and over-tuning to the training data is still a key issue for machine-learned methods, with rule-based baselines showing better robustness in some cases. This can be counteracted using some of the techniques presented in section 4 such as system combination, regularization, value-specific features for generalization including delexicalized features, and word-based tracking.

	Features		Joint Goals		Search Method		Requested	
Entry	**ASR**	**SLU**	**Acc.**	**L2**	**Acc.**	**L2**	**Acc.**	**L2**
Generative and rule-based methods								
Bayesian net.		✓	0.675	0.550	0.880	0.210	**0.885**	0.197
1-best baseline [2]		✓	0.619	0.738	0.879	0.209	0.884	**0.196**
focus baseline [2]		✓	**0.719**	**0.464**	0.867	0.210	0.879	0.206
HWU baseline [9]		✓	0.711	0.466	**0.897**	**0.158**	0.884	0.201
team8 entry1		✓	0.699	0.498	0.899	0.153	0.939	0.101
Machine-learned discriminative methods								
team1 entry0		✓	0.601	0.648	0.904	0.155	0.960	0.073
team3 entry0		✓	0.729	0.452	0.878	0.210	0.889	0.188
team4 entry2		✓	**0.742**	**0.387**	**0.922**	**0.124**	**0.957**	**0.069**
team6 entry2		✓	0.718	0.437	0.871	0.210	0.951	0.085
team7 entry4		✓	0.735	0.433	0.910	0.140	0.946	0.089
team9 entry0		✓	0.499	0.760	0.857	0.229	0.905	0.149
team2 entry2	✓		0.668	0.505	**0.944**	0.095	0.972	0.043
team4 entry0	✓		**0.768**	**0.346**	0.940	**0.095**	**0.978**	**0.035**
team7 entry0	✓		0.750	0.416	0.936	0.105	0.970	0.056
team2 entry1	✓	✓	**0.784**	0.735	**0.947**	**0.087**	0.957	0.068
team2 entry3	✓	✓	0.771	**0.354**	0.947	0.087	0.941	0.090
team5 entry4	✓	✓	0.695	0.610	0.927	0.147	**0.974**	**0.053**
SLU-based oracle [2]		✓	0.850	0.300	0.986	0.028	0.957	0.086

Entry	**Reference**	**Description**
team1 entry0	[39]	Linear CRF
team3 entry0	[10]	Discourse rules + dialog act bigrams
team4 entry2	[37]	Recurrent neural network
team6 entry2	anon	Maximum entropy Markov model, with DNN output distribution
team7 entry4	[11]	System combination of a deep neural network and maximum entropy model
team8 entry1	[26]	Hidden information state model + Goal change handling model + System-user action pair weighting model
team9 entry0	anon	Baseline, augmented with priors from a confusion matrix
team2 entry2	[36]	Recurrent neural network
team4 entry0	[37]	Recurrent neural network
team7 entry0	[11]	System combination of a Deep neural network, maximum entropy model, and rules
team2 entry1	[36]	Web-style ranking
team2 entry3	[36]	Web-style ranking
team5 entry4	anon	ASR/SLU re-ranking

Table 1: DSTC2 entries and results. References are given where teams identified their entry in a published paper. Description based on survey collected from participants. The top performing trackers from each team are selected. The best results in each section for a given choice of input features are made bold. Note that only results obtained during the official evaluation phase are included. The SLU-based oracle shows the top possible performance for systems restricted to giving weight to hypotheses suggested by the SLU. For full results of DSTC 1, 2, and 3, see the DSTC overview paper [4].

This paper has focused on task-driven slot-based dialog domains, such as those featured in DSTC 1, 2, and 3. The next instance of the DSTC will focus on state tracking in less restricted *human-human* conversation [47], which will present a variety of new issues and challenges. Solving these issues will be another step in the direction of open domain dialog, and the ultimate goal of a universal spoken dialog system that can converse naturally on any subject.

7. ACKNOWLEDGEMENTS

The author is grateful to the organizers of the DSTCs – Jason D. Williams, Antoine Raux, Deepak Ramachadran, Alan Black, and Blaise Thomson.

8. REFERENCES

[1] J. Williams, A. Raux, D. Ramachadran, and A. Black, "The Dialog State Tracking Challenge," in *SIGdial*. August 2013, Association for Computational Linguistics.

[2] M. Henderson, B. Thomson, and J. Williams, "The Second Dialog State Tracking Challenge," in *SIGdial*. 2014, Association for Computational Linguistics.

[3] M. Henderson, B. Thomson, and J. D. Williams, "The Third Dialog State Tracking Challenge," in *Spoken Language Technology Workshop*. 2014, IEEE.

[4] J. D. Williams, A. Raux, and M. Henderson, "The Dialog State Tracking Challenge Series: A Review," *Dialogue and Discourse*, 2015.

[5] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus among words: lattice-based word error minimisation," *Computer Speech & Language*, pp. 373–400, 2000.

[6] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington, "JUPlTER: a telephone-based conversational interface for weather information," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 1, pp. 85–96, Jan 2000.

[7] S. Larsson and D. R. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, no. 3&4, pp. 323–340, Sept. 2000.

[8] S. G. Pulman, "Conversational games, belief revision and Bayesian networks," in *CLIN VII: Proceedings of 7th Computational Linguistics in the Netherlands meeting, Nov 1996*, J. L. et al., Ed., 1997, pp. 1–25.

[9] Z. Wang and O. Lemon, "A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information," in *SIGdial*. 2013, Association for Computational Linguistics.

[10] R. Smith, "Comparative error analysis of dialog state tracking," in *SIGdial*. 2014, Association for Computational Linguistics.

[11] K. Sun, L. Chen, S. Zhu, and K. Yu, "The SJTU system for Dialog State Tracking Challenge 2," in *SIGdial*, Philadelphia, PA, U.S.A., June 2014, Association for Computational Linguistics.

[12] K. Sun, L. Chen, S. Zhu, and K. Yu, "A generalized rule based tracker for dialogue state tracking," in *Spoken Language Technology Workshop*. 2014, IEEE.

[13] J. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.

[14] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," in *ACL*, 2000.

[15] E. Horvitz and T. Paek, "A computational architecture for conversation," in *Proceedings of the Seventh International Conference on User Modeling*, Secaucus, NJ, USA, 1999, UM '99, pp. 201–210, Springer-Verlag New York, Inc.

[16] S. Young, J. Schatzmann, K. Weilhammer, and H. Ye, "The Hidden Information State approach to dialog management," in *ICASSP 2007*, Honolulu, Hawaii, 2007.

[17] J. D. Williams, "Using particle filters to track dialogue state," in *Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Kyoto, Japan*, 2007.

[18] J. Henderson and O. Lemon, "Mixture model POMDPs for efficient handling of uncertainty in dialogue management," in *Proc Association for Computational Linguistics Human Language Technologies (ACL-HLT), Columbus, Ohio*, 2008.

[19] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The Hidden Information state model: A practical framework for pomdp-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, Apr. 2010.

[20] T. Bui, M. Poel, A. Nijholt, and J. Zwiers, "A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems," *Nat. Lang. Eng.*, vol. 15, no. 2, pp. 273–307, 2009.

[21] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech & Language*, vol. 24, no. 4, Oct. 2010.

[22] J. Williams, "Incremental partition recombination for efficient tracking of multiple dialog states," in *International Conference on Acoustics, Speech, and Signal Processing*. 2010, IEEE.

[23] R. Kadlec, J. Libovickỳ, J. Macek, and J. Kleindienst, "IBMs belief tracker: Results on Dialog State Tracking Challenge datasets," *EACL 2014*, p. 10, 2014.

[24] R. Kadlec, M. Vodolan, J. Libovicky, J. Macek, and J. Kleindienst, "Knowledge-based dialog state tracking," in *Spoken Language Technology Workshop*. 2014, IEEE.

[25] B. Thomson, F. Jurcicek, and G. M., "Parameter learning for POMDP spoken dialogue models," *Spoken Language Technology Workshop*, 2010.

[26] B.-J. Lee, W. Lim, and K.-E. Kim, "Optimizing generative dialog state tracker via cascading gradient descent," in *SIGdial*. 2014, p. 273–281, Association for Computational Linguistics.

[27] D. Kim, C. J. Choi, K.-E. Kim, J. Lee, and J. Sohn, "Engineering statistical dialog state trackers: A case study on DSTC," in *SIGdial*. 2013, Association for Computational Linguistics.

[28] J. Williams, "A critical analysis of two statistical spoken dialog systems in public use," in *Spoken Language Technology Workshop*. 2012, IEEE.

[29] D. Bohus and A. Rudnicky, "A K-hypotheses + other belief updating model," 2006.

[30] A. Metallinou, D. Bohus, and J. Williams, "Discriminative state tracking for spoken dialog systems," in *ACL*. 2013, The Association for Computer Linguistics.

[31] S. Lee and M. Eskenazi, "Recipe for building robust spoken dialog state trackers: Dialog State Tracking Challenge system description," in *SIGdial*. 2013, Association for Computational Linguistics.

[32] J. Williams, "Multi-domain learning and generalization in dialog state tracking," in *SIGdial*. 2013, Association for Computational Linguistics.

[33] M. Henderson, B. Thomson, and S. Young, "Deep neural network approach for the Dialog State Tracking Challenge," in *SIGdial*. 2013, Association for Computational Linguistics.

[34] H. Ren, W. Xu, and Y. Yan, "Markovian discriminative modeling for dialog state tracking," in *SIGdial*, Philadelphia, PA, U.S.A., June 2014, Association for Computational Linguistics.

[35] H. Ren, W. Xu, and Y. Yan, "Markovian discriminative modeling for cross-domain dialog state tracking," in *Spoken Language Technology Workshop*. 2014, IEEE.

[36] J. Williams, "Web-style ranking and SLU combination for dialog state tracking," in *SIGdial*. 2014, Association for Computational Linguistics.

[37] M. Henderson, B. Thomson, and S. Young, "Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation," in *Spoken Language Technology Workshop*. 2014, IEEE.

[38] S. Lee, "Structured discriminative model for dialog state tracking," in *SIGdial*. 2013, Association for Computational Linguistics.

[39] S. Kim and R. E. Banchs, "Sequential labeling for tracking dynamic dialog states," in *SIGdial*, Philadelphia, PA, U.S.A., June 2014, Association for Computational Linguistics.

[40] M. Henderson, B. Thomson, and S. Young, "Word-based dialog state tracking with recurrent neural networks," in *SIGdial*. 2014, Association for Computational Linguistics.

[41] M. Henderson, *Discriminative Methods for Statistical Spoken Dialogue Systems*, Ph.D. thesis, University of Cambridge, 2015.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, Nov. 1997.

[43] L. Zilka and F. Jurcicek, "Incremental LSTM-based dialog state tracker," *ArXiv e-prints*, July 2015.

[44] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[45] S. Bengio and Y. Bengio, "Taking on the curse of dimensionality in joint distributions using neural networks," *Neural Networks, IEEE Transactions on*, vol. 11, no. 3, May 2000.

[46] N. Mrksic, D. Ó. Séaghdha, B. Thomson, M. Gasic, P. Su, D. Vandyke, T. Wen, and S. J. Young, "Multi-domain dialog state tracking using recurrent neural networks," *CoRR*, vol. abs/1506.07190, 2015.

[47] S. Kim, L. F. DHaro, R. E. Banchs, J. Williams, and M. Henderson, "Dialog State Tracking Challenge 4," `http://www.colips.org/workshop/dstc4/`, 2015.