

Task Lineages: Dialog State Tracking for Flexible Interaction

Sungjin Lee and Amanda Stent

Yahoo Research

229 West 43rd Street, New York, NY 10036, USA

{junior, stent}@yahoo-inc.com

Abstract

We consider the gap between user demands for seamless handling of complex interactions, and recent advances in dialog state tracking technologies. We propose a new statistical approach, *Task Lineage-based Dialog State Tracking* (TL-DST), aimed at seamlessly orchestrating multiple tasks with complex goals across multiple domains in continuous interaction. TL-DST consists of three components: (1) task frame parsing, (2) context fetching and (3) task state update (for which TL-DST takes advantage of previous work in dialog state tracking). There is at present very little publicly available multi-task, complex goal dialog data; however, as a proof of concept, we applied TL-DST to the *Dialog State Tracking Challenge* (DSTC) 2 data, resulting in state-of-the-art performance. TL-DST also outperforms the DSTC baseline tracker on a set of pseudo-real datasets involving multiple tasks with complex goals which were synthesized using DSTC3 data.

1 Introduction

The conversational agent era has arrived: every major mobile operating system now comes with a conversational agent, and with the announcements over the past year of messaging-based conversational agent platforms from Microsoft, Google, Facebook and Kik (among others), technology now supports the rapid development and interconnection of all kinds of dialog bots. Despite this progress, most conversational agents can only handle a single task with a simple user goal at any particular moment. There are three significant hurdles to efficient, natural task-oriented interaction with these agents. First, they lack the ability to **share slot values across tasks**. Due to the independent execution of domain-specific task scripts, information sharing across tasks is minimally supported – the user typically has to provide common slot values separately for each task. Second, these agents

lack the ability to **express complex constraints on user goals** – the user can rarely communicate in a single utterance goals related to multiple tasks, and can typically not provide multiple preferential constraints such as a boolean expression over slot values (Crook and Lemon, 2010). Third, current conversational agents lack the ability to **interleave discussion of multiple related tasks**. For instance, an agent can help a user find a restaurant, and then a hotel, but the user can't interleave these tasks to manage shared constraints.

The *dialog state tracker* (DST) is the most crucial component for addressing these hurdles. A DST constructs a succinct representation of the current conversation state, based on the previous interaction history, so that the conversational agent may choose the best next action. Recently, researchers have developed numerous DST approaches ranging from handcrafted rule-based methods to data-driven models. In particular, the series of Dialog State Tracking Challenges (DSTC) has served as a common testbed, allowing for a cycle of rigorous comparative analysis and rapid advancement (Williams et al., 2016). A consistent finding across the DSTC series is that the best performing systems are statistical DSTs based on discriminative models. The main focus of recent advances, however, has been largely confined to developing more robust approaches to other conversational agent technologies, such as automated speech recognition (ASR) and spoken language understanding (SLU), in a session-based dialog processing a single task with a relatively simple goal. Session-based, single task, simple goal dialog is easier for dialog system engineers and consistent with 25 years of commercial dialog system development, but does not match users' real-world task needs as communicated with human conversational assistants or recognized in the dialog literature (e.g. (Grosz and Sidner, 1988; Lochbaum, 1998)), and is inconsistent with the mobile-centric, always-on, conversational assistant commercial vision that has emerged over the past few years.

This gap between how humans most effectively converse about complex tasks and what conversa-

tional agent technology (including DST) permits clearly shows the direction for future research – statistical DST approaches that can seamlessly orchestrate multiple tasks with complex goals across multiple domains in continuous interaction. In this paper, we describe a new approach, *Task Lineage-based Dialog State Tracking* (TL-DST), centered around the concept of a **task lineage**, to lay a framework for incremental developments toward this vision. TL-DST consists of three components: (1) task frame parsing, (2) context fetching and (3) task state update (for which TL-DST takes advantage of previous work in dialog state tracking). As a proof of concept, we conducted a set of experiments using the DSTC2 and DSTC3 data. First, we applied TL-DST to the DSTC2 data which has a great deal of user goal changes, and obtained state-of-the-art performance. Second, in order to test TL-DST on more challenging data, we applied TL-DST to a set of pseudo-real datasets involving multiple interleaved tasks and complex constraints on user goals. To generate the datasets, we fed the DSTC3 data, which includes three different types of tasks in addition to goal changes, to simulation techniques which have been often adopted for the development and evaluation of dialog systems (Schatzmann et al., 2006; Pietquin and Dutoit, 2006; Lee and Eskenazi, 2012). The results of these experiments show that TL-DST can successfully handle complex multi-task interactions, largely outperforming the DSTC baseline tracker.

The rest of this paper is organized as follows. In Section 2 we describe TL-DST. In Section 3 we discuss our experiments. In Section 4 we present a brief summary of related work. We finish with conclusions and future work in Section 5.

2 Task Lineage-based Dialog State Tracking

We start by defining some essential concepts. Following the convention of the DSTC, we represent each utterance produced by the user or agent as a set of **dialog act items** (DAIs) of the form *dialog-act-type(slot = value)*. A DAI is produced by a SLU; TL-DST may receive input from multiple (domain-specific or general-purpose) SLUs.

Task Schema A task schema is a manually identified set of slots for which values must or may be specified in order to complete the task. For example, the task schema for a restaurant booking task will contain the required slots date/time, location, and restaurant ID, with optional slots cuisine-type, ratings, cost-rating, etc. A task schema governs the configuration of related structures such as task frame and task state.

Task Frame A task frame is a set of DAIs with associated confidence scores and time/sequence

information. An augmented DAI has the form (*confidence score*, $DAI_{end\ time}^{start\ time}$). Usually task frames come in a collection, called a **task frame parse**, as a result of task frame parsing when there are multiple tasks involved in the user input (see Section 2.1). The following collection of task frames shows an example task frame parse for the user input “*Connection to Manhattan and find me a Thai restaurant, not Italian*”:

$$\left[\begin{array}{l} \left[\begin{array}{ll} \mathbf{Task} & \text{Transit} \\ \mathbf{DAIs} & (0.8, \text{inform}(\text{dest}=\text{MH})_{0.7}^{0.1}) \end{array} \right] \\ \left[\begin{array}{ll} \mathbf{Task} & \text{Restaurant} \\ \mathbf{DAIs} & (0.7, \text{inform}(\text{food}=\text{thai})_{1.2}^{0.9}) \\ & (0.6, \text{deny}(\text{food}=\text{italian})_{1.7}^{1.4}) \end{array} \right] \end{array} \right]$$

Task State A task state includes essential pieces of information to represent the current state of a task under discussion, e.g., the task name, a set of belief estimates for user provided preferential constraints, DB query results, timestamps and a turn index. The following state shows an example restaurant finding task state corresponding to the user input “*Thai restaurant, not Italian*”:

$$\left[\begin{array}{ll} \mathbf{Task} & \text{Restaurant} \\ \mathbf{Constraints} & (0.7, \text{food} = \text{thai}) \\ & (0.6, \text{food} \neq \text{italian}) \\ \mathbf{DB} & [“\text{Thai To Go}”, “\text{Pa de Thai}”] \\ \mathbf{Timestamps} & 01/01/2016 : 12-00-00 \\ \dots & \dots \end{array} \right]$$

A task state is analogous to a dialog state in typical dialog systems. However, unlike in conventional dialog state tracking, we don’t assume a unique value for each slot. Instead, we adopt binary distributions for each constraint. This allows us to circumvent the exponential complexity in the number of values which otherwise would be caused by taking a power set of slot values to handle complex constraints (Crook and Lemon, 2010).

Task Lineage A task lineage is a chronologically ordered list of task states, representing the agent’s hypotheses about what tasks were involved at each time point in a conversation. A task lineage can be consulted to provide crucial pieces of information for conversation structure. For instance, the most recent task frames in a lineage can indicate the current focus of conversation. In addition, when the user switches back to a previous task, the agent can trace back the lineage in reverse order to take recency into account. However, conversational agents often cannot determine exactly what the user’s task is. For example, there may be ASR or SLU errors, or genuine ambiguities (“*want Thai*” - *food=Thai* and a restaurant finding task or *dest=Thai* and an air travel task?). Thus we maintain a N -best list of possible task lineages. Figure 1 illustrates how task lineages are extended for new user inputs.

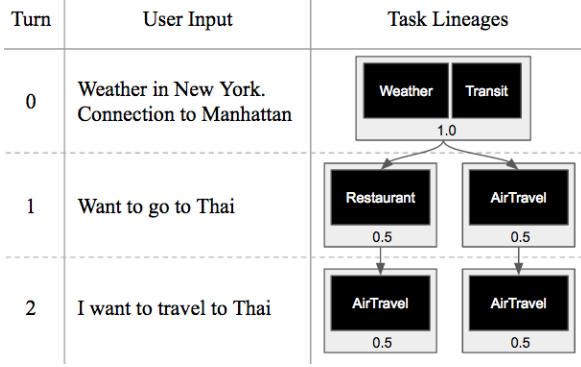


Figure 1: An example illustrating how task lineages are extended as new user inputs come in; this conversation involves multiple tasks (at turn 0) and task ambiguity (at turn 1).

Overall TL-DST Procedure Algorithm 1 describes how the overall TL-DST procedure works. At turn t , given $\tilde{\mathbf{u}}$, a set of DAI sets from one or more SLUs, we perform task frame parsing (see Section 2.1) to generate \mathcal{H} , a K -best list of task frame parses with associated belief scores, $s^{k,i}$. Then, in order to generate a set of new task states, \mathcal{T} , we consider all possible combinations of the task lineages, $l_{0:t-1}^n$, in the current N -best list of task lineages, $\mathcal{L}_{0:t-1}$, and the parses, \mathcal{A}^k , in the K -best list of task frame parses, \mathcal{H} . In a task frame parse, there may be multiple task frames, hence the i th frame in the k th parse is denoted by $f^{k,i}$. The main operation in new task state generation is task state update (see Section 2.3) which forms a new task state, $\tau^{n,k,i}$, per task frame, $f^{k,i}$, by applying belief update to the task frame, relevant information in the lineage $l_{0:t-1}^n$ and the agent’s output \mathbf{m}_t . Task state update is very similar to what is done in typical dialog state tracking except that we need to additionally identify relevant information in the task lineage since a task lineage could be a mix of different tasks. This is the role of context fetching (see Section 2.2). Given a task frame $f^{k,i}$, a task lineage $l_{0:t-1}^n$ and the agent’s output \mathbf{m}_t , the context fetcher returns a set of relevant information pieces, $c^{n,k,i} \in \mathcal{C}$. Finally we construct a new set of task lineages, $\mathcal{L}_{0:t}$, by extending each current task lineage $l_{0:t-1}^n$ with the newly formed task states, $\forall i, \tau^{n,k,i}$. The belief estimate of a new task lineage is set to the product of that of the source task lineage, s_l^n , and that of the task frame parse, s_h^k . Since the extension process grows the number of task lineages by a factor of K , we perform pruning and belief normalization at the end. Based on a N -best list of task lineages, we can then compute useful quantities for the agent’s action selection, such as marginal task beliefs (by adding the be-

¹ M sets the maximum number of samples to draw in the stochastic inference in Section 2.1.

Algorithm 1: Overall TL-DST Procedure

Input: $N > 0, K > 0, M > 0, \delta \geq 0$
 \triangleright Let $\mathcal{L}_{0:t} = [(l_{0:t}^1, s^1), \dots, (l_{0:t}^N, s^N)]$ be a N -best list of task lineages with scores at turn t
 \triangleright See task_frame_parsing in Section 2.1
 \triangleright See context_fetch in Section 2.2
 \triangleright See task_state_update in Section 2.3
 $\mathcal{L}_{0:0} \leftarrow \emptyset;$
 $t \leftarrow 1;$
while *True* **do**
 $\mathbf{m}_t \leftarrow \text{agent_output}();$
 $\tilde{\mathbf{u}}_t \leftarrow \text{user_input}();$
 $\mathcal{H} \leftarrow \text{task_frame_parsing}(\tilde{\mathbf{u}}_t, K, M);$
 $\mathcal{C} \leftarrow \{c^{n,k,i} :=$
 $\quad \text{context_fetch}(l_{0:t-1}^n, f^{k,i}, \mathbf{m}_t, \delta) \mid$
 $\quad l_{0:t-1}^n \in \mathcal{L}_{0:t-1}, \mathcal{A}^k \in \mathcal{H}, f^{k,i} \in \mathcal{A}^k\};$
 $\mathcal{T} \leftarrow \{\tau^{n,k,i} :=$
 $\quad \text{task_state_update}(c^{n,k,i}, f^{k,i}, \mathbf{m}_t) \mid$
 $\quad c^{n,k,i} \in \mathcal{C}, \mathcal{A}^k \in \mathcal{H}, f^{k,i} \in \mathcal{A}^k\};$
 $\hat{\mathcal{L}}_{0:t} \leftarrow [(l_{0:t-1}^n :: \tau^{n,k,i}, s_l^n \times s_h^k) \mid$
 $\quad (l_{0:t-1}^n, s_l^n) \in \mathcal{L}_{0:t-1}, \tau^{n,k,i} \in \mathcal{T},$
 $\quad s_h^k \in \mathcal{H}];$
 $\mathcal{L}_{0:t} \leftarrow \text{prune}(\hat{\mathcal{L}}_{0:t}, N);$
 $t \leftarrow t + 1;$
end

liefs of each task across the lineages) or marginal constraint beliefs (by weighted averaging of the beliefs of each constraint across task states with the task lineage beliefs carrying the weights).

There are a few noteworthy aspects of our TL-DST approach that depart from conventional dialog state tracking approaches. Unlike most methods where the DST keeps on overriding the content of the dialog state (hence losing past states) TL-DST adopts a dynamically growing structure, providing a richer view to later processing. This is particularly important for continuous interaction involving multiple tasks. Interestingly, this is a crucial reason behind advances in deep neural network models using the *attention mechanism* (Bahdanau et al., 2014). Also unlike some approaches that use stack-like data structures for focus management (Larsson and Traum, 2000; Ramachandran and Ratnaparkhi, 2015) where the tracker pops out the tasks above the focused task, losing valuable information such as temporal ordering and partially filled constraints, TL-DST preserves all of the past task states by viewing the focus change as a side effect of generating a new updated task state each time. This allows for flexible task switching among a set of partially fulfilled tasks.

2.1 Task Frame Parsing

In this section we formalize task frame parsing as a structure prediction problem. We use a probabilistic framework that employs a beam search technique using Monte Carlo Markov Chain (MCMC) with simulated annealing (SA) and permits a clean

integration of hard constraints to generate legitimate parses with probabilistic reasoning.

Let $d \in \mathcal{D}$ be a domain and $\tilde{\mathbf{u}}_d$ denote the SLU results from a parser for domain d for observation o , which is a set of confidence score and time information annotated DAIs $\tilde{u}_d^{i_d}, i_d \in \mathcal{I}_d = \{1 \dots N_d\}$. Let $\mathcal{F} = \{f_d^{i_d, t_d} | d \in \mathcal{D}, i_d \in \mathcal{I}_d, t_d \in \mathcal{T}_d\}$ be a collection of the sets of all possible task frames for each $\tilde{\mathbf{u}}_d$, where \mathcal{T}_d is a set of task schemas defined in domain d . We add a special frame $f_{inactive}$ to \mathcal{F} to which some DAIs may be assigned in order to generate legitimate task frame parses when those DAIs are created by either SLU errors or irrelevant pieces of information (e.g. greetings), or they have conflicting interpretations from different domains. Now we define a task frame parse $\mathcal{A}_{\tilde{\mathbf{u}}}$ to be a functional assignment of every $\tilde{u} \in \tilde{\mathbf{u}} = \bigcup_d \tilde{\mathbf{u}}_d$ to \mathcal{F} observing the following constraints: 1) one of any two DAIs overlapped in time must be assigned to the inactive frame; 2) $\tilde{u}_d^{i_d}$ cannot be assigned to any of task frame parses arising from another DAI $\tilde{u}_{d'}^{i_{d'}}$ (i.e. $f_d^{i_d, t_d}, \forall t_d$) if the start time of $\tilde{u}_{d'}^{i_{d'}}$ is later than that of $\tilde{u}_d^{i_d}$ (this constraint is necessary to get rid of spurious assignment ambiguities due to symmetry).

At a particular turn, given $\tilde{\mathbf{u}}$, the aim of task frame parsing is to return a K -best list of assignments $\mathcal{A}_{\tilde{\mathbf{u}}}^k, k \in \{1, \dots, K\}$ according to the following conditional log-linear model:

$$p_{\theta}(\mathcal{A}_{\tilde{\mathbf{u}}} | \tilde{\mathbf{u}}) = \frac{\exp \theta^T \mathbf{g}(\mathcal{A}_{\tilde{\mathbf{u}}}, \tilde{\mathbf{u}})}{\sum_{\mathcal{A}'_{\tilde{\mathbf{u}}}} \exp \theta^T \mathbf{g}(\mathcal{A}'_{\tilde{\mathbf{u}}}, \tilde{\mathbf{u}})} \quad (1)$$

where θ are the model weights, and \mathbf{g} is a vector-valued feature function. The exact computation of Eq. 1 can become very costly for a complicated user input due to the normalization term. To avoid the exponential time complexity, we adopt a beam search technique (presented below) to yield a K -best list of parses which are used to approximate the sum in the normalization term. Figure 2 presents an example of how the variables in the model are related for different parses.

Parsing Independent assignment of DAIs to task frames may result in parses that violate the rules above. To generate a K -best list of legitimate parses, we adopt a beam search technique using MCMC inference with SA as listed in Algorithm 2. After starting with a heuristically initialized parse, the algorithm draws a sample by randomly moving a single DAI from one task frame to another so as not to produce an illegal parse, until the maximum number of samples M has been reached.

Model Training Having training data consisting of SLU results-parse pairs $(\tilde{\mathbf{u}}^{(i)}, \mathcal{A}_{\tilde{\mathbf{u}}}^{(i)})$, we maximize the log-likelihood of the correct parse. Formally,

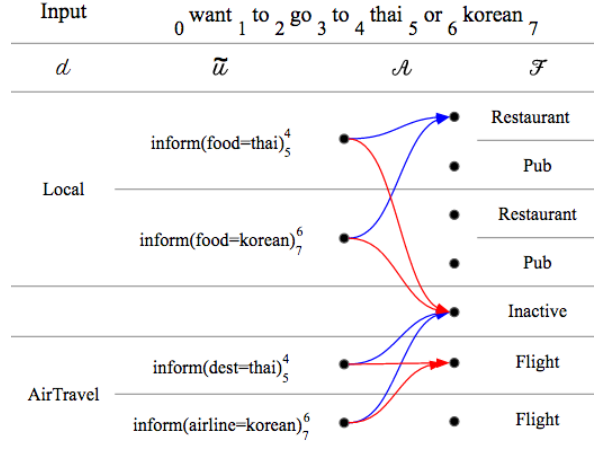


Figure 2: An example illustrating task frame parsing. Here we assume that there are two related domains, *Local* and *AirTravel*, pertinent to the user input “want to go to Thai or Korean”. Time information is annotated as word positions in the input.

Algorithm 2: MCMC-SA Beam Parsing

Input: $K > 0, M > 0, \tilde{\mathbf{u}}, p_{\theta}$ from Eq. 1
Result: $\mathcal{H} = [(\hat{\mathcal{A}}_{\tilde{\mathbf{u}}}^1, s^1), \dots, (\hat{\mathcal{A}}_{\tilde{\mathbf{u}}}^K, s^K)]$, a K -best list of assignments with scores
 $\mathcal{A}_{\tilde{\mathbf{u}}} \leftarrow \text{initialize}(\tilde{\mathbf{u}}), s \leftarrow p_{\theta}(\mathcal{A}_{\tilde{\mathbf{u}}} | \tilde{\mathbf{u}});$
 $\text{insert_and_sort}(\mathcal{H}, \mathcal{A}_{\tilde{\mathbf{u}}}, s);$
 $c \leftarrow 0, \text{acc_rate} \leftarrow 1;$
while $c < M$ **do**
 $\hat{\mathcal{A}}_{\tilde{\mathbf{u}}}, \hat{s} \leftarrow \text{random_choice}(\mathcal{H});$
 $\mathcal{A}_{\tilde{\mathbf{u}}} \leftarrow \text{sample}(\hat{\mathcal{A}}_{\tilde{\mathbf{u}}}, s \leftarrow p_{\theta}(\mathcal{A}_{\tilde{\mathbf{u}}} | \tilde{\mathbf{u}});$
 if $s > \hat{s}$ **or** $\text{random}(0,1) < \text{acc_rate}$ **then**
 $\text{insert_and_sort}(\mathcal{H}, \mathcal{A}_{\tilde{\mathbf{u}}}, s);$
 end
 $c \leftarrow c + 1, \text{acc_rate} \leftarrow \text{acc_rate} - \frac{1}{M};$
end
return \mathcal{H}

our training objective is:

$$\mathcal{O}(\theta) = \sum_i \log p_{\theta}(\mathcal{A}_{\tilde{\mathbf{u}}}^{(i)} | \tilde{\mathbf{u}}^{(i)}) \quad (2)$$

We optimize the objective by initializing θ to 0 and applying *AdaGrad* (Duchi et al., 2011) with the following per-feature stochastic gradient:

$$\frac{\partial \mathcal{O}(\theta, \mathcal{A}_{\tilde{\mathbf{u}}}^{(i)}, \tilde{\mathbf{u}}^{(i)}, \mathcal{H}^{(i)})}{\partial \theta_j} = g_j(\mathcal{A}_{\tilde{\mathbf{u}}}^{(i)}, \tilde{\mathbf{u}}^{(i)}) p_{\theta}(\mathcal{A}_{\tilde{\mathbf{u}}}^{(i)} | \tilde{\mathbf{u}}^{(i)}) - \sum_{\mathcal{A}'_{\tilde{\mathbf{u}}} \in \mathcal{H}^{(i)}: \mathcal{A}'_{\tilde{\mathbf{u}}} \neq \mathcal{A}_{\tilde{\mathbf{u}}}^{(i)}} g_j(\mathcal{A}'_{\tilde{\mathbf{u}}}, \tilde{\mathbf{u}}^{(i)}) p_{\theta}(\mathcal{A}'_{\tilde{\mathbf{u}}} | \tilde{\mathbf{u}}^{(i)})$$

In our experiments we use the features in Table 1, which are all sparse binary features except those marked by †.

- The number of task frames in the parse
- The number of task frames in the parse conjoined with the agent’s DA type
- The number of DAIs in the inactive task frame
- The pair of the total number of DAIs and the number of DAIs in the inactive task frame
- All possible pairs of delexicalized agent DAIs and delexicalized user DAIs in the inactive task frame
- All possible pairs of delexicalized user DAIs for each task frame
- The average confidence score of all DAIs assigned to active task frames†
- The average number of DAIs per active task frame†
- The conjunction of the number of DAIs assigned to active task frames and the number of active task frames
- The fraction of the number of gaps to the number of DAIs assigned to active task frames (a gap happens when two DAIs in the same task frame instance have an intermediate DAI in a different task frame instance)
- The entropy of DAI distribution across active task frames †
- The number of active task frames with only one DAI
- An indicator testing if the parse is the same as a heuristically initialized parse
- The degree of deviation of the parse from a heuristically initialized parse in terms of the number of gaps †

Table 1: Features used in model for task frame parsing

2.2 Context Fetching

There are a variety of phenomena in conversation in which context-dependent analysis plays a crucial role, such as ellipsis resolution, reference resolution, cross-task information sharing and task resumption. In order to successfully handle such phenomena, TL-DST must fetch relevant pieces of information from the conversation history. In this section, we mainly focus on modeling the context fetching process for belief update, ellipsis resolution and task resumption, but a similar technique can be used for handling other phenomena. We first formally define the context fetching model and then introduce a set of feature functions that allow the model to capture general patterns of different context-dependent phenomena.

Context Sets At turn t , given a task lineage $l_{0:t-1}$ and context window δ , the context fetcher constructs three context sets:

- $\mathcal{B}(l_{0:t-1})$: A set of δ -latest belief estimates for each constraint that appears in l_t . The δ -latest belief estimate means the latest belief estimate before $t - \delta$.
- $\mathcal{U}(l_{0:t-1})$: A set of all previous SLU results within δ , $\{\tilde{\mathbf{u}}_{t-\delta}, \dots, \tilde{\mathbf{u}}_{t-1}\}$.
- $\mathcal{M}(l_{0:t-1})$: A set of all previous agent DAIs within δ , $\{\mathbf{m}_{t-\delta}, \dots, \mathbf{m}_{t-1}\}$.

By varying δ , the context fetcher controls the ratio of summarized estimates to raw observations it will use to generate new estimates for the current turn.

Context Fetching Conditioned on the task lineage $l_{0:t-1}$ and the new pieces of information at the current turn such as the task frame f and the agent output \mathbf{m}_t , the context fetcher determines which elements from the context sets will be used. We cast the decision problem as a set of binary classifications for each element using logistic regression. For the sake of simplicity, in this work, we focus on the case where δ is 0 which in effect makes the context fetcher use only the latest belief estimates

for each constraint, $\mathcal{B}(l_{0:t-1})$:

$$p_{\psi}(R(b_{\tau,c}) \mid l_{0:t-1}, f, \mathbf{m}_t) = \frac{1}{1 + \exp -\boldsymbol{\psi}^T \mathbf{h}(b_{\tau,c}, l_{0:t-1}, f, \mathbf{m}_t)}$$

where $b_{\tau,c} \in \mathcal{B}(l_{0:t-1})$ denotes the belief estimate for constraint c at turn τ , R is a binary indicator of fetching decision, $\boldsymbol{\psi}$ are the model weights, and \mathbf{h} is a vector-valued feature function.

Model Training As before, we optimize the log-likelihood of the training data using *AdaGrad*. To construct training data, we construct an oracle task lineage based on dialog state labels, SLU labels and SLU results, which allows us to build corresponding context sets and label each element in them by checking if the element appears in the oracle task state. In our experiments we use the features listed in Table 2, which are all sparse binary features except those marked by †.

2.3 Task State Update

In this section, we describe the last component of TL-DST, task state update. A nice property of TL-DST is its ability to exploit alternative methods for dialog state tracking. For instance, by setting a large value to δ for the context fetcher, one can adopt various discriminative models that take advantage of expressive feature functions extracted from a collection of raw observations (Lee, 2013; Henderson et al., 2014c; Williams, 2014). On the other hand, with δ being 0, one can employ a method from a library of generative models which only requires to know the immediately prior belief estimates (Wang and Lemon, 2013; Zilka et al., 2013). Unlike in previous work, instead of predicting a unique goal value for a slot, we perform belief tracking for each individual slot-value constraint to allow complex goals. For the experiments presented here, we chose to use the rule-based algorithm from Zilka et al. (2013) for constraint-level belief tracking. The use of a rule-based algorithm

- A continuation bias feature. This feature indicates if constraint c is present in any of the task states at the previous turn. This feature allows to model the general tendency to continue.
- Adjacency pair features. These features indicate if c comes from the previous turn when the second half of an adjacency pair (e.g. request/inform and confirm/affirm) is present at the current turn.
- Deletion features based on explicit cues. For example, the user informs alternative constraints after the agent’s unfulfilment notice (e.g. *canthelp* in the DSTC) or the user chooses an alternative to c at the agent’s selection prompt.
- Deletion features based on implicit cues. For instance, the user informs alternative constraints for a slot which is unlikely to admit multiple constraints or after the agent’s explicit or implicit confirmation request. For these features we use the confidence score of the user’s DAI. †
- Task switching features based on agent-initiative cues. Upon the completion of a task, the agent is likely to resume a previous task, thus the context fetcher needs to retrieve the state of the resumed task. Since our experiments are corpus-based, there is no direct internal signal from the agent action selection module, so these features indirectly capture the agent’s initiative on task switching based on which task the agent’s action is related to, and indicate if c is present in the agent’s action or belongs to the agent’s addressed task.
- Task switching features based on user-initiative cues. These features test if c is present in the user’s input or belongs to the user’s addressed task.

Table 2: Features used in model for context fetching

allows us to focus our analysis only on the new aspects of TL-DST.

We present the formal description of the dialog state tracking algorithm. Let $\Sigma_{t,c}^+$ ($\Sigma_{t,c}^-$) denote the sum of all the confidence scores associated with *inform* or *affirm* (*deny* or *negate*) for constraint c at turn t . Then the belief estimate of constraint c at turn t , $b_{t,c}$, is defined as follows:

- For informing or affirming,

$$b_{t,c} = b_{\tau,c}(1 - \Sigma_{t,c}^+) + \Sigma_{t,c}^+$$

- For denying or negating,

$$b_{t,c} = b_{\tau,c}(1 - \Sigma_{t,c}^-)$$

where $b_{\tau,c}$ is the latest available belief estimate for constraint c fetched from a task state at turn τ .

3 Experiments

In order to validate TL-DST, we conducted a set of corpus-based experiments using the DSTC2 and DSTC3 data. The use of DSTC data makes it possible to compare TL-DST with numerous previously developed methods. We first applied TL-DST on the DSTC2 data. DSTC2 was designed to broaden the scope of dialog state tracking to include user goal changes. TL-DST should be able to process user goal changes without any special handling – it should fetch unchanged goals from the previous task state and incorporate new goals from the user’s input to construct a new task state.

However, due to the lack of multi-task conversations in the DSTC2 data, we could not evaluate the performance of task frame parsing. There are also many other aspects of our proposed approach that are hard to investigate without appropriate dialog data. We address this problem by applying simulation techniques to the DSTC3 data. Although there are no DSTC3 dialogs handling multiple task

instances in a single conversation, the DSTC3 extended the DSTC2 to include multiple task types, i.e. restaurant, pub and coffee shop finding tasks. This property of the DSTC3 data allows us to generate a set of pseudo-real dialogs involving multiple tasks with complex goals in longer interactions. The generated corpus helped us evaluate additional aspects of TL-DST.

3.1 DSTC2

In the DSTC2, the user is asked to find a restaurant that satisfies a number of constraints such as food type or area. The data was collected from Amazon Mechanical Turkers using dialog systems developed at Cambridge University. The corpus contains 1612 training dialogs, 506 development dialogs and 1117 test dialogs.

Per DSTC2, the dialog state includes three elements – the user’s goal (slot values the user has supplied), requested slots (those the user has asked for) and search method. In this work, we focus on tracking the user’s goal. Since TL-DST estimates belief for each constraint rather than assigning a distribution over all of the values per slot, we aggregated the constraint-level beliefs for each slot and took the value with the largest belief. We trained the context fetcher on the training data and saved models whenever the performance on the development data was improved. We set the learning rate to 0.1 and used L2 regularization with regularization term 10^{-4} , though the system’s performance was largely insensitive to these settings.

Table 3 shows the performance of TL-DST on the test data in accuracy and L2 along with that of other top performing systems in the literature². The result clearly demonstrates the effectiveness of TL-DST, showing higher accuracy and lower L2 than other state-of-the-art systems. This result is

²In order to make evaluation results comparable, we considered only those systems that used only the provided SLU output, not also ASR information.

particularly interesting in that all of the other systems achieved their best performance through a system combination of various non-linear models such as neural nets, decision trees, or statistical models combined with rules, whereas our system used a lightweight linear model. With the structure among the components of the TL-DST approach, it suffices to use a single linear model to handle sophisticated phenomena such as user goal changes. TL-DST achieved this result without any preprocessing steps such as SLU result correction or the use of lexical features to compensate for relatively poor SLU performance (Kadlec et al., 2014; Zhu et al., 2014). Lastly, we used a generative rule-based model for task state update which is known to be suboptimal for the DSTC2 task. Though it is not the focus of this paper, we expect that one can employ a discriminative model to get further improvements. In particular, there is plenty of room to improve the L2 metric through machine-learned discriminative models.

Entry	Acc.	L2
1-best baseline	0.619	0.738
Sun et al. (2014)	0.735	0.433
Williams (2014)	0.739	0.721
Henderson et al. (2014c)	0.742	0.387
Vodolan et al. (2015)†	0.745	0.433
TL-DST†	0.747	0.451

Table 3: DSTC2 joint goal tracking results. The post DSTC2 systems are marked by †.

3.2 Complex Interactions

In order to evaluate TL-DST on more challenging data, we generated a set of pseudo-real dialogs from the DSTC3 data that contain multiple tasks with complex user goals (Schatzmann et al., 2006; Pietquin and Dutoit, 2006). First, we constructed a repository of user goals (basically, a dictionary mapping mined goals from DSTC3 to their associated turns in the source dialog logs and labels). Then, we simulated dialogs with complex user goals by merging additional goals and the associated turns to a backbone dialog which was randomly drawn from the original DSTC3 dialogs. We randomly sampled additional goals from the goal repository according to a set of per-slot binary distributions, P_{slot}^{add} . For negative constraint generation, we flipped the polarity of an additional goal according to another set of per-slot binary distributions, P_{slot}^{neg} , and correspondingly altered the dialog act type of the relevant DAIs, e.g. *inform* to *deny*. We iterated the goal addition process up to a configured number of iterations, N^{iter} , to cover cases where more than two constraints exist for a slot. The merge process employs a set of heuristic rules so as to preserve natural discourse segments

(e.g., a subdialog for confirming a value) in the backbone dialog. One can simulate dialogs with different complexities by varying the binary distributions and the number of iterations. After this step, the value for each slot is no longer a single value but a set of constraints.

Finally, to construct a multi-task dialog, we randomly drew a backbone dialog from the corpus and decided whether to sample an additional dialog according to a binary distribution, P^{task} . Then we merged the first turns of each selected dialog to ensure the existence of multiple tasks in a single turn. We arranged the remainder of the selected dialogs in order, so as to simulate task resumption. After this process, the label of a dialog state consists of a list of task state labels. An example pseudo-real dialog might contain: A user searches for an Italian or French restaurant in the north area. (S)he also looks for a coffee shop to go to after lunch that is in a cheap price range and provides internet (See Appendix A for example dialogs).

When two turns from different dialogs have to be merged during the dialog synthesis process, we produce a list of new SLU hypotheses by taking the Cartesian product of the two source SLU hypotheses - confidence scores are also multiplied together. For time information annotation, we use the position of the DAI in the SLU hypothesis instead of the real start and end times detected by the ASR component since the DSTC3 data does not have time information. Due to space limitations, we present evaluation results only for the following dialog corpora generated with three different representative settings:

1. No complex user goals and no multiple tasks:
 $P_{food}^{add} = 0.0$, $P_{area}^{add} = 0.0$, $P_{pricerange}^{add} = 0.0$, $P_{food}^{neg} = 0.0$, $P_{area}^{neg} = 0.0$, $P_{pricerange}^{neg} = 0.0$, $N^{iter} = 0$, $P^{task} = 0.0$
2. Complex user goals and no multiple tasks:
 $P_{food}^{add} = 0.5$, $P_{area}^{add} = 0.2$, $P_{pricerange}^{add} = 0.2$, $P_{food}^{neg} = 0.2$, $P_{area}^{neg} = 0.2$, $P_{pricerange}^{neg} = 0.2$, $N^{iter} = 2$, $P^{task} = 0.0$
3. Complex user goals and multiple tasks:
 $P_{food}^{add} = 0.5$, $P_{area}^{add} = 0.2$, $P_{pricerange}^{add} = 0.2$, $P_{food}^{neg} = 0.2$, $P_{area}^{neg} = 0.2$, $P_{pricerange}^{neg} = 0.2$, $N^{iter} = 2$, $P^{task} = 1.0$

Corpora 2 and 3 were divided into 1,000 training dialogs, 500 development dialogs and 1,000 test dialogs. For corpus 1, since we do not generate any new dialogs, we just partitioned the 2,264 DSTC3 dialogs into 846 training dialogs, 418 development dialogs and 1,000 test dialogs. We trained the task frame parser and the context fetcher and saved models whenever the performance on the development data was improved. We set the learning rate

Parameters	System	Avg. Acc.	Joint Acc.	L2
No complex user goals and no multiple tasks	baseline	0.837	0.575	0.864
	TL-DST	0.850	0.594	0.737
Complex user goals and no multiple tasks	baseline	0.720	0.315	1.324
	TL-DST	0.819	0.455	0.972
Complex user goals and multiple tasks	baseline	0.411	0.029	1.893
	TL-DST	0.784	0.338	1.208
	TL-DST-OP	0.833	0.466	0.984
	TL-DST-O	0.928	0.607	0.752

Table 4: Evaluation on complex dialogs with simulated data. The exact parameter settings for each simulation condition can be found in the text.

to 0.1 and used L2 regularization with regularization term 10^{-4} .

Table 4 shows how performance varies on different simulation settings. As expected, the performance of the baseline tracker, which is the DSTC3’s default tracker, drops sharply as the dialogs get more complicated. On the contrary, the performance of TL-DST decreases more gently. Note that joint goal prediction gets exponentially harder as multiple tasks are involved, since we can get each task wrong if we have any of one task’s constraints in another’s state. Thus this gentle performance reduction is in fact a significant win.

As noted before, there is an upper bound to achievable performance due to the limitation of the provided SLU results. Thus we also present the performance of the system with different oracles: 1) TL-DST-OP uses oracle task frame parses; 2) TL-DST-O additionally uses an oracle context fetcher. The comparative results suggest that there is much room for improvement in both the task frame parser and the context fetcher. Given the good performance on Avg. Accuracy, despite imperfect joint prediction, a TL-DST based agent should be able to successfully complete the conversation with extra exchanges. This also matches our empirical analysis of the tracker’s output; the tracker missed only a couple of constraints in its incorrect joint prediction.

4 Related Work

TL-DST aims to extend conventional approaches for dialog state tracking. A variety of approaches have been proposed, for instance, generative models (Thomson et al., 2010; Wang and Lemon, 2013; Zilka et al., 2013; Sun et al., 2014; Kadlec et al., 2014) and discriminative models (Lee and Eskenazi, 2013; Henderson et al., 2014c; Williams, 2014). The series of DSTCs have played a crucial role in supplying essential resources to the research community such as labeled dialog corpora, baseline systems and a common evaluation framework (Williams et al., 2013; Henderson et al., 2014a; Henderson et al., 2014b). For more information about this line of research, we refer to the

recent survey by Williams et al. (2016).

The closest work to our task frame parsing is frame semantic parsing task in NLP (Das, 2014). Differences include that the input here is a collection of potentially conflicting semantic hypotheses from different domain-specific SLUs. Also we are more interested in obtaining a N -best list of parses with well calibrated confidence scores than in getting only a top hypothesis.

Recently there has been growing interest in multidomain and multitask dialog (Crook et al., 2016; Sun et al., 2016; Ramachandran and Ratnaparkhi, 2015; Gašić et al., 2015; Wang et al., 2014; Hakkani-Tür et al., 2012; Nakano et al., 2011). To our knowledge, however, there is no previous work that provides a holistic statistical approach for complex dialog state tracking that can cover the wide range of problems discussed in this paper.

5 Conclusions

In this paper, we have proposed the TL-DST approach toward the goal of seamlessly orchestrating multiple tasks with complex goals across multiple domains in continuous interaction. The proposed method’s state-of-the-art performance on common benchmark datasets and purposefully simulated dialog corpora demonstrates the potential capacity of TL-DST. In the future, we want to apply TL-DST to conversational agent platforms for further evaluation with real world multi-domain dialog. There are many opportunities for technical improvements, including: 1) scheduled sampling for context fetcher training to avoid the mismatch between oracles and runtime conditions (Bengio et al., 2015); 2) using discriminative (sequential) models instead of generative rule-based models for task state update; and 3) learning with weak supervision from real time interactions. Future research can include the extension of TL-DST for other conversational phenomena such as reference resolution. It would also be interesting to study the potential impact on other dialog system components of providing more comprehensive state representations to SLU and action selection.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.
- Paul Crook and Oliver Lemon. 2010. Representing uncertainty about complex user goals in statistical dialogue systems. In *Proceedings of SIGDIAL*.
- Paul Crook, Alex Marin, Vipul Agarwal, Khushboo Aggarwal, Tasos Anastasakos, Ravi Bikkula, Daniel Boies, Asli Celikyilmaz, Senthilkumar Chandramohan, Zhaleh Feizollahi, Roman Holenstein, Minwoo Jeong, Omar Khan, Youngbum Kim, Elizabeth Krawczyk, Xiaohu Liu, Danko Panic, Vasiliy Radostev, Nikhil Ramesh, Jean-Phillipe Robichaud, Alexandre Rochette, Logan Stromberg, and Ruhi Sarikaya. 2016. Task completion platform: A self-serve multi-domain goal oriented dialogue platform. In *Proceedings of NAACL*.
- Dipanjan Das. 2014. Statistical models for frame-semantic parsing. In *Proceedings of the ACL*.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.
- M Gašić, N Mrkšić, P-H Su, D Vandyke, T-H Wen, and S Young. 2015. Policy committee for adaptation in multi-domain spoken dialogue systems. In *Proceedings of ASRU*.
- Barbara J Grosz and Candace L Sidner. 1988. Plans for discourse. Technical report, DTIC Document.
- Dilek Z Hakkani-Tür, Gökhan Tür, Larry P Heck, Ashley Fidler, and Asli Celikyilmaz. 2012. A discriminative classification-based approach to information state updates for a multi-domain dialog system. In *Proceedings of INTERSPEECH*.
- Matthew Henderson, Blaise Thomson, and Jason Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of SIGDIAL*.
- Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The third dialog state tracking challenge. In *Proceedings of SLT*.
- Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of SIGDIAL*.
- Rudolf Kadlec, Miroslav Vodolan, Jindrich Libovicky, Jan Macek, and Jan Kleindienst. 2014. Knowledge-based dialog state tracking. In *Proceedings of SLT*.
- Staffan Larsson and David R Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3&4):323–340.
- Sungjin Lee and Maxine Eskenazi. 2012. An unsupervised approach to user simulation: toward self-improving dialog systems. In *Proceedings of SIGDIAL*.
- Sungjin Lee and Maxine Eskenazi. 2013. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of SIGDIAL*.
- Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *Proceedings of SIGDIAL*.
- Karen E Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572.
- Mikio Nakano, Shun Sato, Kazunori Komatani, Kyoko Matsuyama, Kotaro Funakoshi, and Hiroshi Okuno. 2011. A two-stage domain selection framework for extensible multi-domain spoken dialogue systems. In *Proceedings of SIGDIAL*.
- Olivier Pietquin and Thierry Dutoit. 2006. A probabilistic framework for dialog simulation and optimal strategy learning. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(2):589–599.
- Deepak Ramachandran and Adwait Ratnaparkhi. 2015. Belief tracking with stacked relational trees. In *Proceedings of SIGDIAL*.
- Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(02):97–126.
- Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014. A generalized rule based tracker for dialogue state tracking. In *Proceedings of SLT*.
- Ming Sun, Yun-Nung Chen, and Alexander Rudnicky. 2016. An intelligent assistant for high-level task understanding. In *Proceedings of IUI*.
- Blaise Thomson, F Jurčićek, M Gašić, Simon Keizer, François Mairesse, Kai Yu, and Steve Young. 2010. Parameter learning for pomdp spoken dialogue models. In *Proceedings of SLT*.
- Miroslav Vodolán, Rudolf Kadlec, and Jan Kleindienst. 2015. Hybrid dialog state tracker. *arXiv preprint arXiv:1510.03710*.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of SIGDIAL*.

Zhuoran Wang, Hongliang Chen, Guanchun Wang, Hao Tian, Hua Wu, and Haifeng Wang. 2014. Policy learning for domain selection in an extensible multi-domain spoken dialogue system. In *Proceedings of EMNLP*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of SIGDIAL*.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Jason D Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of SIGDIAL*.

Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu. 2014. Semantic parser enhancement for dialogue domain extension with little data. In *Proceedings of SLT*.

Lukas Zilka, David Marek, Matej Korvas, and Filip Jurcicek. 2013. Comparison of bayesian discriminative and generative models for dialogue state tracking. In *Proceedings of SIGDIAL*.

A Example Simulated Dialogs

The following dialogs show the surface form of simulated complex interactions. The dialog state tracker uses the corresponding SLU results and dialog state annotations to the parts in the labeled DSTC3 logs of which the dialog is composed.

A.1 Dialog 1

System: Thank you for calling the Cambridge Information system. Your call will be recorded for research purposes. You may ask for information about a place to eat, such as a restaurant, a pub, or a cafe. How may I help you?

User: Yes. I’m looking for a seafood or north european restaurant. And I’m looking for a moderately priced coffee shop or in expensive price range.

System: Are you looking for somewhere near a particular venue? For example, Cambridge Contemporary Art, Holy Trinity Church, or Trinity College.

User: In the expensive price range.

System: Loch Fyne, which serves seafood, is an outstanding restaurant. It’s in the expensive price range.

User: What’s the address?

System: Loch Fyne is located on 37 Trumpington Street. The price for main courses ranges from 9 pounds to 20 pounds.

User: Ah what’s the phone number?

System: The phone number of Loch Fyne is 01223902103.

User: And what’s the price range?

System: Loch Fyne is in the expensive price range. The price for main courses ranges from 9 pounds to 20 pounds.

User: Ok thanks.

System: For a coffee shop, what kind of price range are you looking for? For example, free, or moderate.

User: Expensive price range.

System: Are you looking for somewhere near a particular venue? For example, Cambridge Contemporary Art, Great Saint Mary’s Church, or Primavera.

User: No I’m not.

System: Cafe Eleven, which is in the moderate price range, is an outstanding cafe.

User: Can I get the phone number and the post-code?

System: The phone number of Cafe Eleven is 01223369157. It’s in the area with the postcode CB1 1DG.

User: Thank you goodbye.

A.2 Dialog 2

System: Thank you for calling the Cambridge Information system. Your call will be recorded for research purposes. You may ask for information about a place to eat, such as a restaurant, a pub, or a cafe. How may I help you?

User: I’m looking for a cheap Chinese restaurant or a Turkish restaurant. But I don’t want a French restaurant. And it should be in the Riverside area. And I would like a cafe.

System: What part of town do you have in mind? For example, something in the Barnwell, Chesterton, or King’s Hedges area.

User: Chesterton.

System: There is no restaurant in the Chesterton area in the cheap price range.

User: Cheap Chinese take away restaurant.

System: Would you like a Chesterton area, or you don’t care?

User: I don’t care.

System: HK Fusion is an outstanding restaurant, it serves chinese food and it’s in the cheap price range.

User: Phone number?

System: The phone number of HK Fusion is 01223355909.

User: Address?

System: HK Fusion is on 21 Burleigh Street.

User: Thank you.

System: What type of food would you like? For example, Chinese take away, American, or English.

User: Cafe.

System: What type of food would you like? For example, Greek, Mediterranean, or Middleeastern.

User: I don't care.

System: What kind of price range are you looking for? For example, cheap, or moderate.

User: I don't care.

System: The Cambridge Blue is a great cafe.

User: What is the address and phone number?

System: The Cambridge Blue is on 24 Regent Street and its phone number is 01223369222.

User: Thank you goodbye.