

# Belief Tracking with Stacked Relational Trees

**Deepak Ramachandran**

Nuance Communications Inc.  
1178 E Arques, Sunnyvale, CA  
deepak.ramachandran@nuance.com

**Adwait Ratnaparkhi**

Nuance Communications Inc.  
1178 E Arques, Sunnyvale, CA  
adwait.ratnaparkhi@nuance.com

## Abstract

We describe a new model for Dialog State Tracking called a Stacked Relational Tree, which naturally models complex relationships between entities across user utterances. It can represent multiple conversational intents and the change of focus between them. Updates to the model are made by a rule-based system in the language of tree regular expressions. We also introduce a probabilistic version that can handle ASR/NLU uncertainty. We show how the parameters can be trained from log data, showing gains on a variety of standard Belief Tracker metrics, and a measurable impact on the success rate of an end-to-end dialog system for TV program discovery.

## 1 Introduction

Significant advances have been made in recent years on the problem of Dialog State Tracking or Belief Tracking. Successive iterations of the Dialog State Tracking Challenge (Williams et al., 2013; Henderson et al., 2014b; Henderson et al., 2014a) have expanded the scope of the problem to more general settings such as changing goals and domain adaptation. It has been shown that improvements in Belief Tracking metrics lead to improvements in extrinsic measures of dialog success as well (Lee, 2014). However, the underlying representations of state have almost always been propositional i.e. defined by a collection of slot-value pairs, though the probability distribution used for tracking might be quite complex (Mehta et al., 2010). These representations are good for form-filling or information collection type dialogs that are most commonly deployed e.g. airline reservation systems that fill in all the constraints a user has (such as destination and source) before doing a database lookup. However, as dialog systems get more sophisticated, complex

dialog phenomena present in human-human conversations such as common ground or conversational focus need to be supported as well.

This work is motivated by the need for a belief tracker capable of tracking conversations with the end-to-end conversational prototype for TV program discovery described in (Ramachandran et al., 2014). The prototype understands concepts at a deep relational level and supports nested subdialogs with multiple intents of different types like searches, questions, and explanations. We introduce a representation called a *Stacked Relational Tree* to represent the state of a dialog between a user and system. It uses the notion of a *relational tree*, similar to a dependency graph but constructed between entities from a Named Entity Recognizer (NER), to represent each individual intent of the user. A stack (i.e. LIFO structure) of these trees is used to model the conversational focus and the structure of subdialogs. State updates are modeled by sequences of stack and tree-editing operations. Allowable operations are defined using the language of tree-regular expressions (Lai and Bird, 2004). The use of stacks to represent intentional structure is common in dialog modeling (Grosz and Sidner, 1986) and plan recognition (Carberry, 1990). Our novel contribution is to combine it with a semantic representation and update rules that are simple enough so that the entire model can be trained from dialog data.

A system using this belief tracker was deployed in a user study and made a dramatic difference in the task success rate. We also describe a probabilistic extension of this model for handling uncertainty in input and ambiguity in understanding. We show that training the weights of this model on log data can improve its performance.

## 2 Dialog State Representation

Most commercial and research dialog systems represent the state of a conversation as a collection

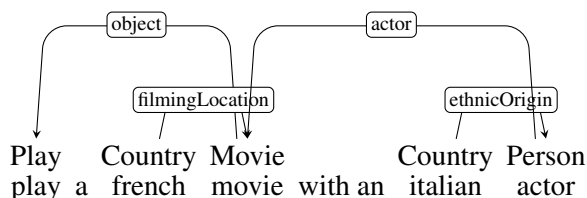


Figure 1: REL-Tree for the utterance “Play a French movie with an Italian actor.”

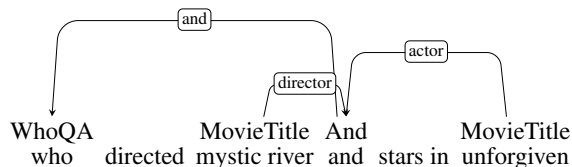


Figure 2: REL-Tree for the question “Who directed Mystic river and stars in Unforgiven?”

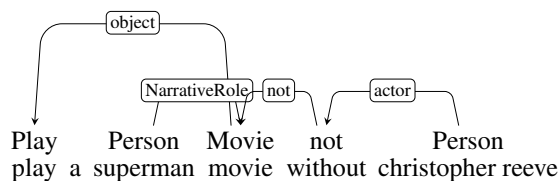


Figure 3: REL-Tree for the utterance “Play a Superman movie without Christopher Reeve.”

of slot-value pairs that define the system’s best understanding of the user’s intent e.g. an airline reservation system might have slots for destination city, arrival city, and date. Shallow NLP techniques such as Named-Entity Recognition are used to extract the relevant slot-value pairs from each spoken utterance of the user. As successive utterances accumulate, a state tracking strategy is needed to update the state given the slot-value pairs provided at each turn. Traditionally, state tracking followed a simple replacement semantics. Modern systems maintain a probability distribution over possible states, reflecting all the uncertainty and ambiguity in ASR and NLU. Recent extensions have focused on adaptation to new domains (Henderson et al., 2014b) and changing user goals (Zhu et al., 2014). However, in most cases we are aware of, the base representation of the dialog state is propositional (i.e. a collection of slot-value pairs). This reflects the simple, goal-directed nature of the dialogs supported by such systems.

## 2.1 REL-Trees

Consider an utterance like “Play a French movie with an Italian actor.” A slot-based system with a

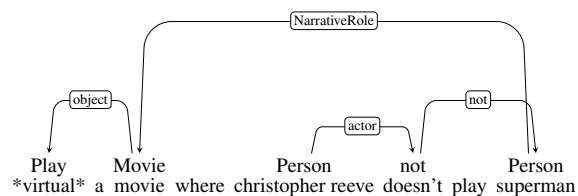


Figure 4: REL-Tree for the fragmentary utterance “A movie where Christopher Reeve doesn’t play Superman.”

slot called `Country` would not be able to distinguish between the filming location and the actor’s country of origin. A possible solution is to introduce two separate slots called `actorEthnicity` and `filmingLocation`, but scaling this approach leads to a multiplicity of slots that becomes difficult to manage and extend. A more compact representation (called a *Relational Tree* or *REL-Tree*) is shown in Fig. 1. The only entity types are `Country`, `Movie`, and `Person`. To elaborate the meaning of the utterance, “French” is attached to the `Movie` entity by the relation `filmingLocation` and “italian” is attached to `Person` by the relation `ethnicOrigin`. A REL-Tree is a rooted tree with node labels corresponding to entities and edge labels corresponding to relations between them. In most cases, a relation link is analogous to a syntactic dependency link from a dependency parser – a link from child to parent signifies that the child is a modifier of the parent. The label at the root of the tree represents the intent of the utterance (e.g., “Play”, “Who-QA”, and “ExpressPreference”) if one can be distinguished, see Fig. 2 for another example. Fragmentary utterances can have missing intents, in which case the root is simply labeled `ROOT`.

Comparing the REL-Trees in Figures 3 and 4 shows another example of the representational power of REL-Trees. The two utterances have different meanings and indeed yield different results (The 2013 movie “Man of Steel” had Christopher Reeve in a cameo role, but not as Superman). In our dialog system, REL-Trees are produced by a Relation Extraction component that operates after NER. Note that the NER is trained to label boolean connectors such as “and” and “without” as entities as well. In some cases, it adds “virtual” entities to fragmentary utterances when they are not explicit in the text (e.g. the `Play` entity in Fig. 4). For more details refer to (Ramachandran et al., 2014).

## 2.2 Stacks

The dialog example of Table 4 (see Appendix) illustrates another phenomenon not usually considered by belief trackers: multiple intents and the concept of a *conversational focus* (Grosz and Sidner, 1986). The user starts with the intention of finding a romantic movie to watch but is then led by the system response into asking a question about one of the search results (a query). He then modifies the argument of the query to ask about a different movie. Then, he gives a command to provide him with more suggestions. Finally, he goes back to the original search intent and modifies the genre. The second column of this table shows how we model multiple intents and the change in focus by a stack of REL-Trees (called a *Stacked REL-Tree* or a *Stack*). Each REL-Tree represents a separate intent of the user and the REL-Tree on top of the stack is the current focus of the conversation. Subsequent utterances are interpreted as refining or modifying this REL-Tree. If no such interpretation is possible, then either the focus is assumed to have shifted back to an earlier intent in the stack or we treat the utterance as a new intent. The allowable set of operations and the algorithm by which they are applied are fully specified in the next few sections. A REL-Tree that represents an utterance from the user will be called an *utterance REL-Tree* wherever it is necessary to make the distinction.

## 3 Update Rules

The Stacked REL-Tree representation of dialog state was introduced in the previous section and Table 4 shows how a dialog state progresses as each utterance comes in. A set of state update rules are used to specify how the REL-tree on the top of a stack is modified by the incoming utterance. To describe the update rules, we will need three definitions.

**Tree Regular Expressions** A *tree regular expression* (or tree regex) is a regular expression that matches against paths in a rooted tree from a node to one of its descendants, with node and edge labels serving as the tokens of the string (Lai and Bird, 2004). The basic elements of a tree regex are:

1. **Node and Edge labels:** These are represented by a string regular expression (i.e. a regular expression over strings) surrounded by “/” e.g. `/[actor|director]/` matches a node with an actor or director label.

When labels are concatenated they represent a path from the root to a descendant node with each successive label alternatively matching node and edge labels on the path. For example, `/Movie/actor/Person/ethnicOrigin/Place` would match against the path from the “movie” node to the “italian” in Fig. 1. The empty label `//` matches any node or edge label.

2. **Node Values:** A node label followed by the expression `{V}` where `V` is a string regular expression, matches nodes where the surface text of the node equals `V`. e.g. `/Movie/narrativeRole/Person{superman}/` matches the path from the “movie” node to the “superman” node in Fig. 3.
3. **Operators:** The symbols `*`, `?`, `.` have the usual meanings for regular expressions when placed after a tree regular expression. Note however, that `*` and `+` automatically match against alternating node and edge labels along a path. Thus, the expression `//*/Place/` matches against two paths from the root in Fig. 1. The operators `^` and `$` represent the root node and a leaf node respectively.
4. **Groups:** Groups are defined by enclosing a part of a tree regex inside parentheses. Let  $M$  be a successful match of a tree regex  $P$  to the tree  $T$ , the sub-path in  $M$  matching the  $i$ th group in  $P$  can be retrieved by  $M.@i$ . For example, for the tree in Fig. 2 and the pattern `/And/./(MovieTitle)`, there are two matches  $M_1$  and  $M_2$  with  $M_1.@1$  having value “mystic river” and  $M_2.@1$  having value “unforgiven.”

**Tree Constraints** For tree regexes  $P_1$  and  $P_2$ , a *Tree constraint* on  $P_1$  and  $P_2$  is an expression of the form  $P_1.@i = P_2.@j$ ,  $P_1.@i\{\} = P_2.@j\{\}$ , or  $P_1.@i\{\} < P_2.@j\{\}$ . Here,  $x < y$  means  $x$  is a substring of  $y$ . `\{\}` retrieves the value of a node (the surface form).

**Transformations** A *transformation*  $\tau$  on tree regexes  $P_1$  and  $P_2$ , is a list of one or more of the following operations performed on paths that match against groups from  $P_1$  and  $P_2$  :

1. **Add ( $g_1, g_2$ ):** Add the matched sub-path from group  $g_2$  as a child of the head node of the matched sub-path from group  $g_1$ .

2. **Delete** ( $g$ ): Remove the head node and all descendants of the path matching group  $g$ .
3. **Unify** ( $g_1, g_2$ ): Replace the head node  $h_1$ , of  $g_1$  with the head node,  $h_2$  of  $g_2$ , and add all children of  $h_2$  as children of  $h_1$ .

An *update rule* is defined as a tuple  $(P_1, P_2, E, \tau)$  where  $P_1$  and  $P_2$  are tree regular expressions,  $E$  is a set of tree constraints on  $P_1$  and  $P_2$ , and  $\tau$  is a transformation on  $P_1$  and  $P_2$ . An update rule  $U$  is *applicable* to a dialog state tree  $T$  and an input REL-tree  $L$  if:

1.  $P_1$  has a match,  $M_1$  on  $T$
2.  $P_2$  has a match,  $M_2$  on  $L$
3.  $E$  holds for the groups in  $M_1$  and  $M_2$ .

In such case, the result of applying  $U$  on  $T$  and  $L$  are the trees  $S'$  and  $L'$  obtained by applying each operation in  $\tau$  to  $\{M_1, M_2\}$  in the order specified.

Here are some example update rules with explanations:

### 1. Head Variable Unification

$P_1$ : /object/(Program/) $P_2$ : /object/([Movie TvShow Game]/) $E$ : {} $\tau$ : {Unify( $P_1.\text{@1}$ , $P_2.\text{@1}$ )}
---

If the object of the current intent is `Program` and the current utterance from the user asks for either a movie, tv show, or game, then update the dialog state to reflect that we are searching for this kind of program (See Fig. 5 for an example).

### 2. Concept Replacement

$P_1$ : ^///(///)\$ $P_2$ : ^///(///)\$ $E$ : { $P_1.\text{@1}=P_2.\text{@1}$ } $\tau$ : {Unify( $P_1.\text{@1}$ , $P_2.\text{@1}$ ), Delete( $P_2.\text{@1}$ )}
---

This rule is applicable when the input utterance has a value for some attribute that is already present in the dialog state. In this case, the new value of the attribute replaces the old one. Note that the constraint in the utterance tree is also “consumed” by this rule (See Fig. 5 for an example).

### 3. Boolean fragment

$P_1$ : ([or and]/[And Or])*(/or/Or)(///)\$ $P_2$ : ^(/or/Or)(///)\$ $E$ : { $P_1.\text{@3}=P_2.\text{@2}$ } $\tau$ : {Add( $P_2.\text{@2}$ , $P_1.\text{@3}$ ), Delete( $P_1.\text{@3}$ ), Add( $P_2.\text{@2}$ , $P_1.\text{@2}$ ), Delete( $P_2.\text{@2}$ )}
--

This rule is applicable when the input utterance is a boolean fragment with an attribute

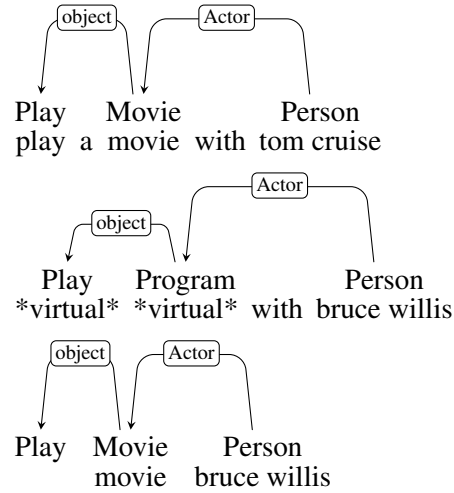


Figure 5: The tree at the bottom is the result of applying rules 1 and 2 to the trees at the top (current dialog state) and the middle (current utterance).

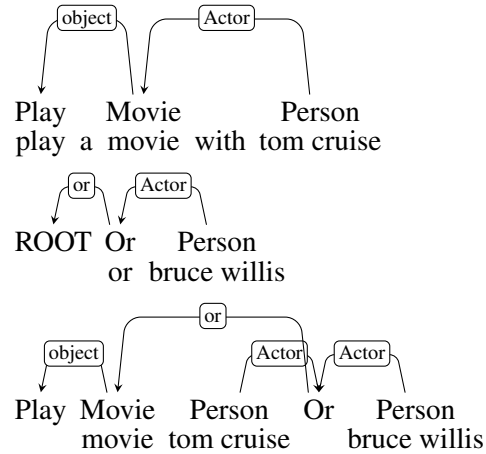


Figure 6: The tree at the bottom is the result of applying rules 1 and 3 to the trees at the top (current dialog state) and the middle (current utterance).

already present in the dialog state. The subtrees are then unified as shown in Fig. 6.

The definition of update rules and the allowable operations we have presented were tailored to our particular domain. In principle, it is possible to extend them to be more general, but care must be taken so that the operations and especially the regex matching algorithm can be efficiently implemented (Lai and Bird, 2004). For our implementation of tree regexes we adapted the `TSurgeon` package (Levy and Andrew, 2006) from the Stanford Parser.

---

**Algorithm 1:** UpdateDialogState

---

**Data:** Stacked REL-Tree  $S$ , utterance  
REL-Tree  $L$ , List of Update Rules  $R$   
Applied:=**false**,  $S_{\text{cur}} := S$ ;  
**repeat**  
   $T := S.\text{pop}()$ ;  
  **for each update rule**  $R_i \in R$  **in sequence**  
  **do**  
    **if**  $R_i$  **is applicable to**  $(T, L)$  **then**  
      Applied=**true**;  
      Apply transformation  $\tau_i$  (from  $R_i$ )  
      to  $(T, L)$ ;  
  **until**  $S.\text{empty}()$  or Applied=**true**;  
  **if not Applied then**  
     $S := S_{\text{cur}}$ ;  
     $S.\text{push}(T)$ ;  
**return**  $S$ ;

---

### 3.1 The Belief Tracking Algorithm

Recall that our state representation is a stack of REL-Trees as in Table 4. Algorithm 1 shows how we update the dialog state at each turn. It is parameterized by an ordered list of update rules as described in Section 3. We attempt to apply them in order to the REL-Tree at the top of the stack first. If no rule is applicable, this indicates that the conversational focus has shifted. We pop the top REL-Tree off the stack and try again with the REL-Tree below it. This process continues, until a rule is successfully applied or the stack is empty. In the latter case, the utterance is regarded as being a new intent, and the utterance REL-Tree is pushed on top of the old dialog state.

## 4 A Probabilistic Extension

The State Tracker described above is able to model relational representations and shifting conversational focus. However, it is deterministic and thus unable to handle ambiguity caused by multiple applicable rules. Consider the third user turn in Table 4. We interpret “How about The Notebook?” as a modification to the question intent, but it is possible that the user intended it to be a refinement of his search intent i.e. he wants to watch “The Notebook”. Furthermore, in most practical dialog systems the output of the ASR and NLU components will have multiple hypotheses with associated confidence scores or probabilities.

To represent this uncertainty in a compact way,

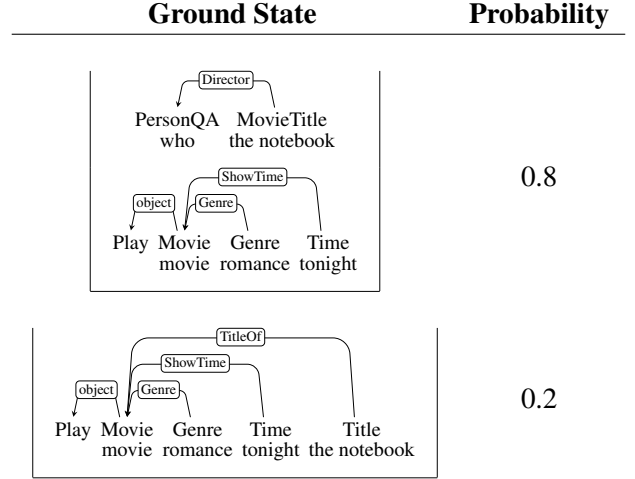


Figure 7: A sample belief state after turn 3 of the dialog in Table 4. The first ground state is the result of a merge of the utterance REL-Tree with the top of the stack. The second ground state is the result of a pop followed by a merge.

we will expand our representation of dialog state to a *dialog belief state* that is a *probability distribution over Stacked REL-Trees*. An example belief state for the case above is shown in Fig. 7, having two ground dialog states (i.e. Stacked REL-Trees) with probability 0.8 and 0.2. The belief state,  $B_t$ , for a particular turn  $t$ , is constructed from the belief state of the previous turn  $B_{t-1}$ , by trying every combination of Stacked REL-Tree  $S_{t-1}$  in the support of  $B_{t-1}$ , utterance REL-Tree  $L$ , and sequence of applicable rule  $\{R_i\}$  to yield a different Stacked REL-Tree  $S_t$ . The probability of  $S_t$  is given by:

$$Pr_{B_t}(S_t|S_{t-1}, L, \{R_i\}) = Pr_{B_{t-1}}(S_{t-1}) \cdot Pr_L(L) \cdot \prod_i Pr(R_i|S_{t-1}^{i-1}, L)$$

where  $S_t^i$  is obtained by applying  $R_i$  to  $S_{t-1}^{i-1}$ , and

$$Pr(R_i|S, L) \propto e^{-w_i \cdot \mathbf{f}(S, L, R_i)} \quad (1)$$

Here,  $\mathbf{f}(S, L, R_i)$  is a feature-generating function. It uses a combination of structural tree features such as number of children and depth from root and features from the surface text (e.g., functional words/phrases such as “and” or “instead of”). We also have special rules for *pushing* a REL-Tree on top of the stack, *popping* the top REL-Tree, and rules marked *terminal* indicating that no more rules are to be applied. The weights for all rules are trained by logistic regression.

---

**Algorithm 2:** UpdateBeliefState

---

**Data:** Belief State of previous turn  $B_{t-1}(S)$ ,  
Distribution over utterance REL-Trees  
 $P_L(L)$ , List of Update Rules  $R$

**for** each stack  $S$  in the support of  $B_{t-1}$  **do**  
    **for** each tree  $L$  in the support of  $P_L$  **do**  
         $W := B_{t-1}(S) \cdot P_L(L)$   
         $B_t := B_t \cup \text{UpdateI-State}(S, L, W)$

Prune  $B_t$  down to the top  $K$  elements;  
Normalize the weights to 1.  
**return**  $B_t$

---

---

**Algorithm 3:** UpdateI-State

---

**Data:** Stacked REL-Tree  $S$ , Utterance  
REL-Tree  $L$ , List of Update Rules  $R$ ,  
Prior Weight  $W$

$S = \{\}$

**for** each update rule  $R_i \in R$  applicable to  
 $(S, L)$  **do**  
    Apply transformation  $\tau_i$  (from  $R_i$ ) to  
     $(S, L)$  to get  $(S', L')$   
     $W_i := W \cdot Pr(R_i|S, L)$   
    **if**  $R_i$  is terminal **then**  
         $S := S + (S', W_i)$   
    **else**  
         $S := S \cup \text{UpdateI-State}(S', L', W_i)$

**return**  $S$ ;

---

The full probabilistic belief tracking algorithm is shown in Algorithm 2. It uses a recursive helper method (Algorithm 3) to apply rules successively to stacks in the input distribution. The intermediate states of this process are called *I-States*. To prevent a combinatorial explosion in the size of the belief state over successive turns, it is pruned down at the end to a distribution over at most  $K$  stacks ( $K = 50$ ).

**Training** For training data, we use conversations with a full dialog-system. Each turn of the dialog is annotated with the sequence of update rules that are applied to the belief state of the previous turn to get the correct belief state for the current turn. From these, we can compute the sequence of I-States for that turn. Then, for each rule that is applicable to each of these I-States, a training instance is added to the classifier for that rule, along with a binary label indicating whether the rule was applied in that I-State or not. The classifier (using logistic

regression) then learns to distinguish I-States where the rule should be used, from those where it should not. Note that this training protocol requires very strong labels from the annotator (a sequence of operations for every turn). This limits its scalability to larger training sets, but nevertheless we present it as a proof of concept that training this model is possible in principle. Exploring ways to ease this constraint is a topic we plan to explore in future work.

## 5 Evaluation

We present two evaluations of the tracking approaches described above. The first one measures the impact of using the deterministic algorithm as part of a larger conversational prototype for TV Program Discovery, in contrast to a system with no belief tracking (stateless). In the second, we show the additional value gained by the probabilistic version, trained on dialogs from developer logs. The framework for the second evaluation was made to be as close as possible to the methods in the DSTC competition.

### 5.1 User Study

An implementation of Algorithm 1 with 16 update rules and 4 kinds of user intents (search requests, questions, commands, and preference statements) was included as a component of a Spoken Dialog System for TV Program Discovery on an iPad. The system had an NER and a Relation Extractor as described in Section 2 as well as a dialog manager that operated on Stacked REL-Trees and a back-end for program search that used both structured database queries and graph inference on Freebase. For more details, see (Ramachandran et al., 2014). This system was evaluated in a user study with 14 subjects to determine how much the statefulness of the dialog model impacted success and usability. Subjects were presented with 7 scenarios to imagine themselves in and asked to find a suitable program to watch using the prototype, for example:

You are at home and have young nieces  
and nephews coming over. Find a pro-  
gram to watch with them.

The subject was asked to continue speaking with the system until he/she either found a suitable program (in which case the scenario was recorded as a *success*) or gave up (in which case a *failure* was recorded). For this evaluation, the subject was

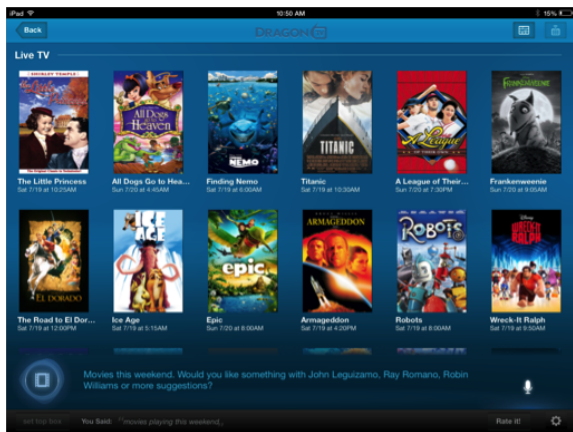


Figure 8: A Screenshot of our iPad Conversational Prototype after the query “Movies playing this weekend”.

System	Succ. Rate	Avg. # of turns	SUS Score
Stateful	85.72 %	4.81	84.29 (15.7)
Stateless	63.27%	5.38	85.71 (15.5)

Table 1: Comparison of dialog system performance for 14 real users with and without the state tracker. SUS score is a industry-standard usability metric.

asked to do these tasks using both the complete stateful dialog system and a stateless *repeated one-shot* version where the belief tracker and dependent components were deactivated. Both systems were otherwise identical and the subjects were not told what the differences between them was. Table 1 summarizes the result of the comparison. The use of belief tracking was found to make a 22% (absolute) difference to the objective success rate of the system. The stateful model allowed subjects to pose more complex requests that are difficult to formulate as a single utterance as a series of simpler ones; or allowed them to explore the space of options more efficiently. This was also reflected in a lower average number of turns per dialog. Interestingly, this did not lead to a significant difference in the perceived usability of the system, as measured by the industry standard SUS score (Brooke, 1996). Indeed, users barely noticed the stateful/stateless distinction between the systems. The overall usability was still pretty high, corresponding to an adjective rating of “Good” verging on “Excellent.”

User: football games this weekend  
 Sys: Football games this weekend. Would you like College Football or Classic College Football?  
 User: or next weekend  
 Sys: Football games this weekend or next weekend. Would you like Classic College Football, College Football, or Rose Bowl?  
 User: romantic movies tomorrow  
 Sys: Ok starting a new search. Romantic movies tomorrow. Would you like something with Sandra Bullock, Nicole Kidman, Kristen Stewart, or more suggestions?  
 User: more suggestions  
 Sys: How about Robert Pattinson, Kate Winslet, Katherine Heigl or more suggestions?  
 User: Winslet  
 Sys: Romantic movies with Kate Winslet tomorrow. Would you like Titanic or Little Children?  
 User: the first one  
 Sys: You chose Titanic on Bravo at 10:30 AM on Saturday  
 User: movies about the gulf war  
 Sys: Movies set during the Gulf war. Would you like Courage Under Fire, Jarhead, or The Finest Hour?  
 User: explain the second one  
 Sys: This film is an adaptation of the book “Jarhead: a Marine’s Chronicle of the Gulf War and Other Battles”, whose subject is the Gulf War. Would you like Courage under Fire, Jarhead, or The Finest Hour?

Table 2: Example dialog with our system. System responses are accompanied by movie or TV thumbnail images on the iPad UI.

## 5.2 Probabilistic Belief Tracking

The Dialog State Track Competitions (Williams et al., 2013; Henderson et al., 2014b; Henderson et al., 2014a) introduced a shared evaluation task for belief tracking on a corpus of dialog data from various domains. Unfortunately, the data is purely slot-based so it cannot be used to evaluate our methods directly. However, the competitions also introduced a rubric for evaluation that we endeavoured to follow as closely as possible in this section.

Algorithm 2 was implemented with 16 update rules similar to the deterministic tracker described above. The weight vectors for each rule were trained by logistic regression as described. The training data came from the developer logs of our system.<sup>1</sup> Each turn of dialog was labelled by us with the correct dialog-state (i.e. stacked REL-tree) and the sequence of updates rule that were applied to progress to the next state. The training protocol of Section 4 was then followed. Overall there were 673 dialogs with 1726 turns of speech and 3642 I-states. After training, the belief tracking algorithm (Algorithm 2) was evaluated on a held out test set of 50 dialogs with 142 turns.

The DSTC competitions identified 4 clusters of evaluation metrics that tended to rank various tracking algorithms equivalently. In Table 3 we show the performance of the trained tracker and the deter-

<sup>1</sup>Logs of conversations involving testing and bug fixing were removed.

System	Accuracy	L2	ROC.V2.CA20	ROC.V1.EER
Deterministic-Test Set	0.743	0.264	0.82	0.25
Trained-Test Set	0.788	0.237	0.73	0.22
Deterministic-User Study	0.661	0.348	0.75	0.35
Trained-User Study	0.680	0.335	0.72	0.33

Table 3: Comparison of belief tracker performance with and without training using DSTC metrics.

ministic baseline on one metric from each cluster: *Accuracy* measures the percent of turns where the top-ranked hypothesis is correct. *L2* measures the  $L^2$  distance between the vector of scores for each hypothesis, and a vector of zeros with 1 in the position of the correct hypothesis. The other two measures relate to receiver-operating characteristic (ROC) curves, which measure the discrimination of the score for the highest-ranked state hypothesis. *ROC.V2.CA20* is the Correct acceptance-rate for the highest ranked hypothesis when the false-acceptance rate is set to 20%, for correctly classified utterances only. *ROC.V1.EER* is the Equal-error rate i.e. where false-acceptance rate equals false-reject rate, for all utterances. In addition to the test data-set, performance was also measured on all dialogs from the user study of Section 5.1. This gives a measure of generalization to dialogs from outside the training distribution. The results show that the trained belief tracker outperformed the handcrafted on all measures, though not by large amounts. As expected, performance was uniformly worse on the (out-of-sample) user study data but there was still some improvement.

## 6 Conclusions and Future Work

In this paper, we present the first (to our knowledge) Belief Tracking approach that represents the dialog state with a probabilistic relational and multi-intent model. We show that this model is effective when measured on standard metrics used for belief tracking, as well as making a marked difference in the task success rate of a complete dialog system.

The most serious shortcoming of this approach is the reliance on very strong labels for the training. To relax this requirement, we are exploring the possibility of training our model using weak labels (such as query results) in the manner of (Berant et al., 2013). Another direction to explore is the representation of distributions over Stacked REL-trees in compact forms.

## References

- J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- J. Brooke. 1996. SUS: A quick and dirty usability scale. In *Usability Evaluation in Industry*.
- S. Carberry. 1990. *Plan recognition in natural language dialogue*. MIT press.
- B. J. Grosz and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.
- M. Henderson, B. Thomson, and J. Williams. 2014a. The third dialog state tracking challenge. *Proceedings of IEEE Spoken Language Technology*.
- M. Henderson, B. Thomson, and J. Williams. 2014b. The second dialog state tracking challenge. In *Proceedings of the SIGDIAL 2014 Conference*, page 263.
- C. Lai and S. Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian language technology workshop*, pages 139–146.
- S. Lee. 2014. Extrinsic evaluation of dialog state tracking and predictive metrics for dialog policy optimization. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 310.
- R. Levy and G. Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 2231–2234. Citeseer.
- N. Mehta, R. Gupta, A. Raux, D. Ramachandran, and S. Krawczyk. 2010. Probabilistic ontology trees for belief tracking in dialog systems. In *Proceedings of the SIGDIAL 2010 Conference*, pages 37–46. Association for Computational Linguistics.
- D. Ramachandran, P. Yeh, W. Jarrold, B. Douglas, A. Ratnaparkhi, R. Provine, J. Mendel, and A. Emfield. 2014. An end-to-end dialog system for tv program discovery. In *SLT*.
- J. Williams, A. Raux, D. Ramachandran, and A. Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.
- S. Zhu, L. Chen, K. Sun, D. Zheng, and K. Yu. 2014. Semantic parser enhancement for dialogue domain extension with little data. In *Spoken Language Technology Workshop*.

## A Dialog Example

In Table 4 we show the belief tracking process using a Stacked REL-Tree for a sample conversation.



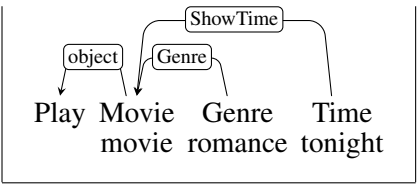
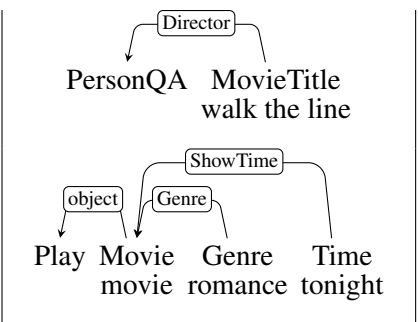
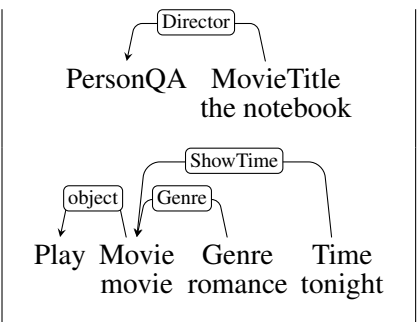
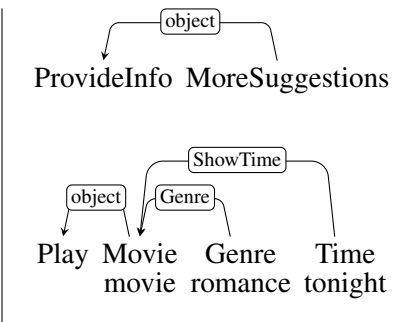
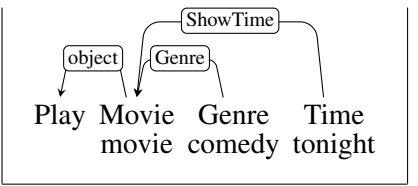
Utterance	System state after utterance	Operation performed on stack
User: I want a romance movie tonight.		Initial Search Intent
System: Ok how about The Notebook or Walk the Line? User: Who directed walk the line?		New question intent put on top of stack
System: James Mangold User: How about The Notebook?		Modification to question on top of stack.
System: Nick Cassavetes. User: Give me more suggestions.		Utterance is a command for more suggestions, gets placed on top of the stack replacing the question.
System: No more suggestions. User: Ok well, let's try a comedy then.		Command is popped off, comedy replaces romance in the original search intent.

Table 4: Dialog State updates of the deterministic tracker (Algorithm 1) for each turn of a sample dialog.