Incremental Parsing Models for Dialog Task Structure

Srinivas Bangalore and **Amanda J. Stent**
AT&T Labs – Research, Inc., 180 Park Avenue,
Florham Park, NJ 07932, USA
{srini,stent}@research.att.com

Abstract

In this paper, we present an integrated model of the two central tasks of dialog management: interpreting user actions and generating system actions. We model the interpretation task as a classification problem and the generation task as a prediction problem. These two tasks are interleaved in an incremental parsing-based dialog model. We compare three alternative parsing methods for this dialog model using a corpus of human-human spoken dialog from a catalog ordering domain that has been annotated for dialog acts and task/subtask information. We contrast the amount of context provided by each method and its impact on performance.

1 Introduction

Corpora of spoken dialog are now widely available, and frequently come with annotations for tasks/games, dialog acts, named entities and elements of syntactic structure. These types of information provide rich clues for building dialog models (Grosz and Sidner, 1986). Dialog models can be built offline (for dialog mining and summarization), or online (for dialog management).

A dialog manager is the component of a dialog system that is responsible for interpreting user actions in the dialog context, and for generating system actions. Needless to say, a dialog manager operates incrementally as the dialog progresses. In typical commercial dialog systems, the interpretation and generation processes operate independently of each other, with only a small amount of shared context. By contrast, in this paper we describe a dialog model that (1) tightly integrates interpretation and generation, (2) makes explicit the type and amount of shared context, (3) includes the task structure of the dialog in the context, (4) can be trained from dialog data, and (5) runs incrementally, parsing the dialog as it occurs and interleaving generation and interpretation.

At the core of our model is a parser that incrementally builds the dialog task structure as the dialog progresses. In this paper, we experiment with three different incremental tree-based parsing methods. We compare these methods using a corpus of human-human spoken dialogs in a catalog ordering domain that has been annotated for dialog acts and task/subtask information. We show that all these methods outperform a baseline method for recovering the dialog structure.

The rest of this paper is structured as follows: In Section 2, we review related work. In Section 3, we present our view of the structure of task-oriented human-human dialogs. In Section 4, we present the parsing approaches included in our experiments. In Section 5, we describe our data and experiments. Finally, in Section 6, we present conclusions and describe our current and future work.

2 Related Work

There are two threads of research that are relevant to our work: work on parsing (written and spoken) discourse, and work on plan-based dialog models.

Discourse Parsing Discourse parsing is the process of building a hierarchical model of a discourse from its basic elements (sentences or clauses), as one would build a parse of a sentence from its words. There has now been considerable work on discourse parsing using statistical bottom-up parsing (Soricut and Marcu, 2003), hierarchical agglomerative clustering (Sporleder and Lascarides, 2004), parsing from lexicalized tree-adjoining grammars (Cristea, 2000), and rule-based approaches that use rhetorical relations and discourse cues (Forbes et al., 2003; Polanyi et al., 2004; LeThanh et al., 2004). With the exception of Cristea (2000), most of this research has been limited to non-incremental parsing of textual monologues where, in contrast to incremental dialog parsing, predicting a system action is not relevant.

The work on discourse parsing that is most similar to ours is that of Baldridge and Lascarides (2005). They used a probabilistic head-driven parsing method (described in (Collins, 2003)) to construct rhetorical structure trees for a spoken dialog corpus. However, their parser was

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 94–102,
Athens, Greece, 30 March – 3 April 2009.

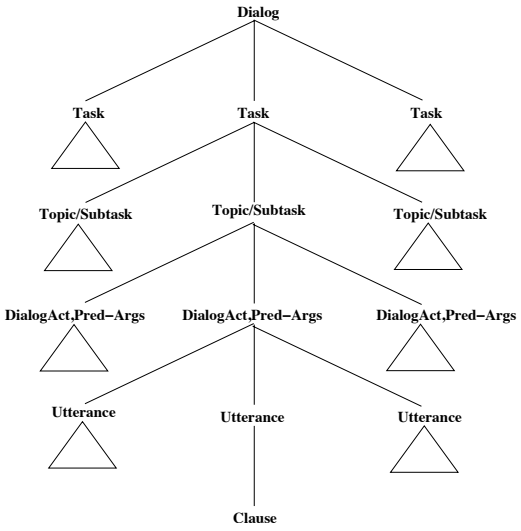

Figure 1: A schema of a shared plan tree for a dialog.

not incremental; it used global features such as the number of turn changes. Also, it focused strictly in interpretation of input utterances; it could not predict actions by either dialog partner.

In contrast to other work on discourse parsing, we wish to use the parsing process directly for dialog management (rather than for information extraction or summarization). This influences our approach to dialog modeling in two ways. First, the subtask tree we build represents the functional task structure of the dialog (rather than the rhetorical structure of the dialog). Second, our dialog parser must be entirely incremental.

Plan-Based Dialog Models Plan-based approaches to dialog modeling, like ours, operate directly on the dialog's task structure. The process of task-oriented dialog is treated as a special case of AI-style plan recognition (Sidner, 1985; Litman and Allen, 1987; Rich and Sidner, 1997; Carberry, 2001; Bohus and Rudnicky, 2003; Lochbaum, 1998). Plan-based dialog models are used for both interpretation of user utterances and prediction of agent actions. In addition to the hand-crafted models listed above, researchers have built stochastic plan recognition models for interaction, including ones based on Hidden Markov Models (Bui, 2003; Blaylock and Allen, 2006) and on probabilistic context-free grammars (Alexandersson and Reithinger, 1997; Pynadath and Wellman, 2000).

In this area, the work most closely related to ours is that of Barrett and Weld (Barrett and Weld, 1994), who build an incremental bottom-up parser

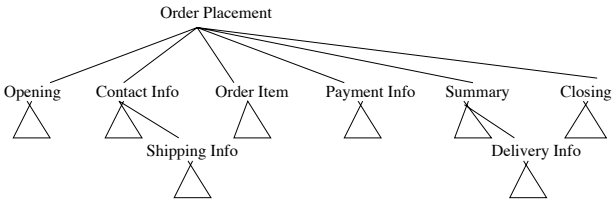

Figure 2: Sample output (subtask tree) from a parse-based model for the catalog ordering domain.

to parse plans. Their parser, however, was not probabilistic or targeted at dialog processing.

3 Dialog Structure

We consider a task-oriented dialog to be the result of incremental creation of a shared plan by the participants (Lochbaum, 1998). The shared plan is represented as a single tree T that incorporates the task/subtask structure, dialog acts, syntactic structure and lexical content of the dialog, as shown in Figure 1. A task is a sequence of subtasks $ST \in S$. A subtask is a sequence of dialog acts $DA \in D$. Each dialog act corresponds to one clause spoken by one speaker, customer (c^u) or agent (c^a) (for which we may have acoustic, lexical, syntactic and semantic representations).

Figure 2 shows the subtask tree for a sample dialog in our domain (catalog ordering). An *order placement* task is typically composed of the sequence of subtasks *opening, contact-information, order-item, related-offers, summary*. Subtasks can be nested; the nesting can be as deep as five levels in our data. Most often the nesting is at the leftmost or rightmost frontier of the subtask tree.

As the dialog proceeds, an utterance from a participant is accommodated into the subtask tree in an incremental manner, much like an incremental syntactic parser accommodates the next word into a partial parse tree (Alexandersson and Reithinger, 1997). An illustration of the incremental evolution of dialog structure is shown in Figure 4. However, while a syntactic parser processes input from a single source, our dialog parser parses user-system *exchanges*: user utterances are interpreted, while system utterances are generated. So the steps taken by our dialog parser to incorporate an utterance into the subtask tree depend on whether the utterance was produced by the *agent* or the *user* (as shown in Figure 3).

User utterances Each user turn is split into clauses (utterances). Each clause is supertagged

Interpretation of a user's utterance:

$$DAC : da_i^u = \underset{d^u \in D}{argmax}\ P(d^u | c_i^u, ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, c_{i-k}^{i-1}) \quad (1)$$

$$STC : st_i^u = \underset{s^u \in S}{argmax}\ P(s^u | da_i^u, c_i^u, ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, c_{i-k}^{i-1}) \quad (2)$$

Generation of an agent's utterance:

$$STP : st_i^a = \underset{s^a \in S}{argmax}\ P(s^a | ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, c_{i-k}^{i-1}) \quad (3)$$

$$DAP : da_i^a = \underset{d^a \in D}{argmax}\ P(d^a | st_i^a, ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, c_{i-k}^{i-1}) \quad (4)$$

Table 1: Equations used for modeling dialog act and subtask labeling of agent and user utterances. c_i^u/c_i^a = the words, syntactic information and named entities associated with the i^{th} utterance of the dialog, spoken by user/agent u/a. da_i^u/da_i^a = the dialog act of the i^{th} utterance, spoken by user/agent u/a. st_i^u/st_i^a = the subtask label of the i^{th} utterance, spoken by user/agent u/a. DA_{i-k}^{i-1} represents the dialog act tags for utterances $i-1$ to $i-k$.

and labeled with named entities[1]. Interpretation of the clause (c_i^u) involves assigning a dialog act label (da_i^u) and a subtask label (st_i^u). We use ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, and c_{i-k}^{i-1} to represent the sequence of preceeding k subtask labels, dialog act labels and clauses respectively. The dialog act label da_i^u is determined from information about the clause and (a kth order approximation of) the subtask tree so far ($T_{i-1} = (ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, c_{i-k}^{i-1})$), as shown in Equation 1 (Table 1). The subtask label st_i^u is determined from information about the clause, its dialog act and the subtask tree so far, as shown in Equation 2. Then, the clause is incorporated into the subtask tree.

Agent utterances In contrast, a dialog system starts planning an agent utterance by identifying the subtask to contribute to next, st_i^a, based on the subtask tree so far ($T_{i-1} = (ST_{i-k}^{i-1}, DA_{i-k}^{i-1}, c_{i-k}^{i-1})$), as shown in Equation 3 (Table 1) . Then, it chooses the dialog act of the utterance, da_i^a, based on the subtask tree so far and the chosen subtask for the utterance, as shown in Equation 4. Finally, it generates an utterance, c_i^a, to realize its communicative intent (represented as a subtask and dialog act pair, with associated named entities)[2].

Note that the current clause c_i^u is used in the conditioning context of the interpretation model (for user utterances), but the corresponding clause for the agent utterance c_i^a is to be predicted and hence is not part of conditioning context in the generation model.

[1]This results in a syntactic parse of the clause and could be done incrementally as well.

[2]We do not address utterance realization in this paper.

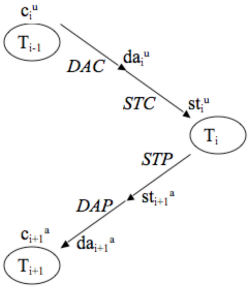

Figure 3: Dialog management process

4 Dialog Parsing

A dialog parser can produce a "shallow" or "deep" tree structure. A shallow parse is one in which utterances are grouped together into subtasks, but the dominance relations among subtasks are not tracked. We call this model a *chunk-based* dialog model (Bangalore et al., 2006). The chunk-based model has limitations. For example, dominance relations among subtasks are important for dialog processes such as anaphora resolution (Grosz and Sidner, 1986). Also, the chunk-based model is representationally inadequate for center-embedded nestings of subtasks, which do occur in our domain, although less frequently than the more prevalent "tail-recursive" structures.

We use the term *parse-based* dialog model to refer to deep parsing models for dialog which not only segment the dialog into chunks but also predict dominance relations among chunks. For this paper, we experimented with three alternative methods for building parse-based models: **shift-reduce**, **start-complete** and **connection path**. Each of these operates on the subtask tree for the dialog incrementally, from left-to-right, with access only to the preceding dialog context, as shown in Figure 4. They differ in the parsing actions and the data structures used by the parser; this has implications for robustness to errors. The instructions to reconstruct the parse are either entirely encoded in the stack (in the shift-reduce method), or entirely in the parsing actions (in the start-complete and connection path methods). For each of the four types of parsing action required to build the parse tree (see Table 1), we construct

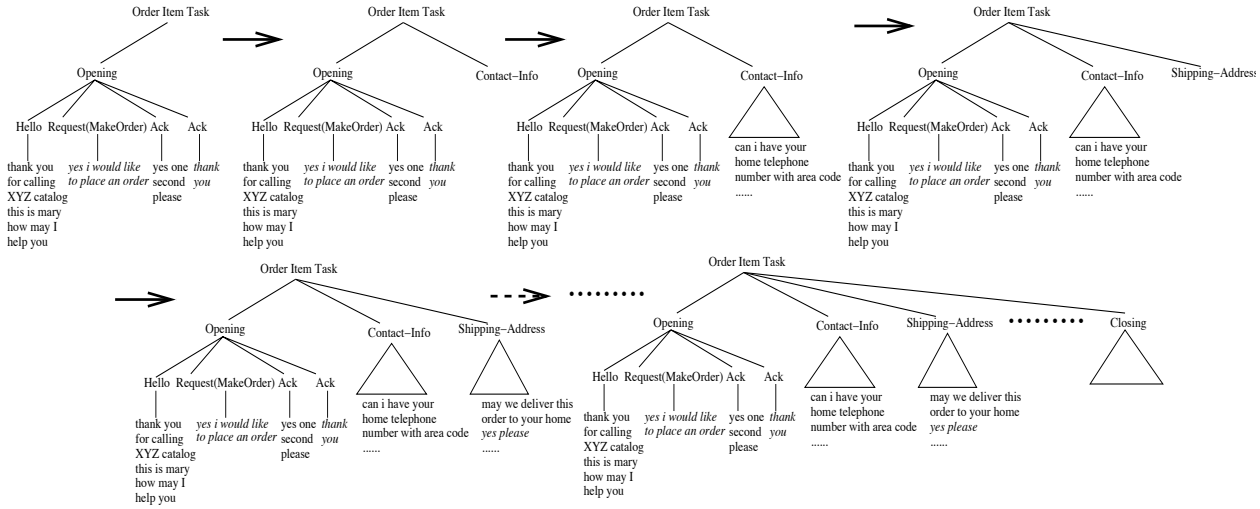

Figure 4: An illustration of incremental evolution of dialog structure

a feature vector containing contextual information for the parsing action (see Section 5.1). These feature vectors and the associated parser actions are used to train maximum entropy models (Berger et al., 1996). These models are then used to incrementally incorporate the utterances for a new dialog into that dialog's subtask tree as the dialog progresses, as shown in Figure 3.

4.1 Shift-Reduce Method

In this method, the subtask tree is recovered through a right-branching shift-reduce parsing process (Hall et al., 2006; Sagae and Lavie, 2006). The parser shifts each utterance on to the stack. It then inspects the stack and decides whether to do one or more reduce actions that result in the creation of subtrees in the subtask tree. The parser maintains two data structures – a stack and a tree. The actions of the parser change the contents of the stack and create nodes in the dialog tree structure. The actions for the parser include *unary-reduce-X*, *binary-reduce-X* and *shift*, where X is each of the non-terminals (subtask labels) in the tree. *Shift* pushes a token representing the utterance onto the stack; *binary-reduce-X* pops two tokens off the stack and pushes the non-terminal X; and *unary-reduce-X* pops one token off the stack and pushes the non-terminal X. Each type of *reduce* action creates a constituent *X* in the dialog tree and the tree(s) associated with the reduced elements as subtree(s) of *X*. At the end of the dialog, the output is a binary branching subtask tree.

Consider the example subdialog *A: would you like a free magazine? U: no*. The processing of this dialog using our shift-reduce dialog parser would proceed as follows: the STP model predicts *shift* for st^a; the DAP model predicts *YNP(Promotions)* for da^a; the generator outputs *would you like a free magazine?*; and the parser shifts a token representing this utterance onto the stack. Then, the customer says *no*. The DAC model classifies da^u as *No*; the STC model classifies st^u as *shift* and *binary-reduce-special-offer*; and the parser shifts a token representing the utterance onto the stack, before popping the top two elements off the stack and adding the subtree for *special-order* into the dialog's subtask tree.

4.2 Start-Complete Method

In the shift-reduce method, the dialog tree is constructed as a side effect of the actions performed on the stack: each reduce action on the stack introduces a non-terminal in the tree. By contrast, in the start-complete method the instructions to build the tree are directly encoded in the parser actions. A stack is used to maintain the global parse state. The actions the parser can take are similar to those described in (Ratnaparkhi, 1997). The parser must decide whether to join each new terminal onto the existing left-hand edge of the tree, or start a new subtree. The actions for the parser include *start-X*, *n-start-X*, *complete-X*, *u-complete-X* and *b-complete-X*, where X is each of the non-terminals (subtask labels) in the tree. *Start-X* pushes a token representing the current utterance onto the stack; *n-start-X* pushes non-terminal X onto the stack; *complete-X* pushes a token representing the current utterance onto the stack, then

pops the top two tokens off the stack and pushes the non-terminal X; *u-complete-X* pops the top token off the stack and pushes the non-terminal X; and *b-complete-X* pops the top two tokens off the stack and pushes the non-terminal X. This method produces a dialog subtask tree directly, rather than producing an equivalent binary-branching tree.

Consider the same subdialog as before, *A: would you like a free magazine? U: no*. The processing of this dialog using our start-complete dialog parser would proceed as follows: the STP model predicts *start-special-offer* for st^a; the DAP model predicts *YNP(Promotions)* for da^a; the generator outputs *would you like a free magazine?*; and the parser shifts a token representing this utterance onto the stack. Then, the customer says *no*. The DAC model classifies da^u as *No*; the STC model classifies st^u as *complete-special-offer*; and the parser shifts a token representing the utterance onto the stack, before popping the top two elements off the stack and adding the subtree for *special-order* into the dialog's subtask tree.

4.3 Connection Path Method

In contrast to the shift-reduce and the start-complete methods described above, the connection path method does not use a stack to track the global state of the parse. Instead, the parser directly predicts the *connection path* (path from the root to the terminal) for each utterance. The collection of connection paths for all the utterances in a dialog defines the parse tree. This encoding was previously used for incremental sentence parsing by (Costa et al., 2001). With this method, there are many more choices of decision for the parser (195 decisions for our data) compared to the shift-reduce (32) and start-complete (82) methods.

Consider the same subdialog as before, *A: would you like a free magazine? U: no*. The processing of this dialog using our connection path dialog parser would proceed as follows. First, the STP model predicts *S-special-offer* for st^a; the DAP model predicts *YNP(Promotions)* for da^a; the generator outputs *would you like a free magazine?*; and the parser adds a subtree rooted at *special-offer*, with one terminal for the current utterance, into the top of the subtask tree. Then, the customer says *no*. The DAC model classifies da^u as *No* and the STC model classifies st^u as *S-special-offer*. Since the right frontier of the subtask tree has a subtree matching this path, the parser simply incorporates the current utterance as a terminal of the *special-offer* subtree.

Type	Task/subtask labels
Call-level	call-forward, closing, misc-other, opening, out-of-domain, sub-call
Task-level	check-availability, contact-info, delivery-info, discount, order-change, order-item, order-problem, payment-info, related-offer, shipping-address, special-offer, summary

Table 2: Task/subtask labels in CHILD

Type	Subtype
Ask	Info
Explain	Catalog, CC_Related, Discount, Order_Info
	Order_Problem, Payment_Rel, Product_Info
	Promotions, Related_Offer, Shipping
Convers--ational	Ack, Goodbye, Hello, Help, Hold, YoureWelcome, Thanks, Yes, No, Ack,
	Repeat, Not(Information)
Request	Code, Order_Problem, Address, Catalog,
	CC_Related, Change_Order, Conf, Credit,
	Customer_Info, Info, Make_Order, Name,
	Order_Info, Order_Status, Payment_Rel,
	Phone_Number, Product_Info, Promotions,
	Shipping, Store_Info
YNQ	Address, Email, Info, Order_Info,
	Order_Status,Promotions, Related_Offer

Table 3: Dialog act labels in CHILD

5 Data and Experiments

To evaluate our parse-based dialog model, we used 817 two-party dialogs from the CHILD corpus of telephone-based dialogs in a catalog-purchasing domain. Each dialog was transcribed by hand; all numbers (telephone, credit card, etc.) were removed for privacy reasons. The average dialog in this data set had 60 turns. The dialogs were automatically segmented into utterances and automatically annotated with part-of-speech tag and supertag information and named entities. They were annotated by hand for dialog acts and tasks/subtasks. The dialog act and task/subtask labels are given in Tables 2 and 3.

5.1 Features

In our experiments we used the following features for each utterance: (a) the speaker ID; (b) unigrams, bigrams and trigrams of the words; (c) unigrams, bigrams and trigrams of the part of speech tags; (d) unigrams, bigrams and trigrams of the supertags; (e) binary features indicating the presence or absence of particular types of named entity; (f) the dialog act (determined by the parser); (g) the task/subtask label (determined by the parser); and (h) the parser stack at the current utterance (deter-

mined by the parser). Each input feature vector for agent subtask prediction has these features for up to three utterances of left-hand context (see Equation 3). Each input feature vector for dialog act prediction has the same features as for agent subtask prediction, plus the actual or predicted subtask label (see Equation 4). Each input feature vector for dialog act interpretation has features a-h for up to three utterances of left-hand context, plus the current utterance (see Equation 1). Each input feature vector for user subtask classification has the same features as for user dialog act interpretation, plus the actual or classified dialog act (see Equation 2).

The label for each input feature vector is the parsing action (for subtask classification and prediction) or the dialog act label (for dialog act classification and prediction). If more than one parsing action takes place on a particular utterance (e.g. a shift and then a reduce), the feature vector is repeated twice with different stack contents.

5.2 Training Method

We randomly selected roughly 90% of the dialogs for training, and used the remainder for testing.

We separately trained models for: user dialog act classification (DAC, Equation 1); user task/subtask classification (STC, Equation 2); agent task/subtask prediction (STP, Equation 3); and agent dialog act prediction (DAP, Equation 4). In order to estimate the conditional distributions shown in Table 1, we use the general technique of choosing the MaxEnt distribution that properly estimates the average of each feature over the training data (Berger et al., 1996). We use the machine learning toolkit LLAMA (Haffner, 2006), which encodes multiclass classification problems using binary MaxEnt classifiers to increase the speed of training and to scale the method to large data sets.

5.3 Decoding Method

The decoding process for the three parsing methods is illustrated in Figure 3 and has four stages: STP, DAP, DAC, and STC. As already explained, each of these steps in the decoding process is modeled as either a prediction task or a classification task. The decoder constructs an input feature vector depending on the amount of context being used. This feature vector is used to query the appropriate classifier model to obtain a vector of labels with weights. The parser action labels (STP and STC) are used to extend the subtask tree. For example, in the shift-reduce method, *shift* results in a push action on the stack, while *reduce-X* results in popping the top two elements off the stack and pushing *X* on to the stack. The dialog act labels (DAP and DAC) are used to label the leaves of the subtask tree (the utterances).

The decoder can use n-best results from the classifier to enlarge the search space. In order to manage the search space effectively, the decoder uses a beam pruning strategy. The decoding process proceeds until the end of the dialog is reached. In this paper, we assume that the end of the dialog is given to the decoder[3].

Given that the classifiers are error-prone in their assignment of labels, the parsing step of the decoder needs to be robust to these errors. We exploit the state of the stack in the different methods to rule out incompatible parser actions (e.g. a *reduce-X* action when the stack has one element, a *shift* action on an already shifted utterance). We also use n-best results to alleviate the impact of classification errors. Finally, at the end of the dialog, if there are unattached constituents on the stack, the decoder attaches them as sibling constituents to produce a rooted tree structure. These constraints contribute to robustness, but cannot be used with the connection path method, since any connection path (parsing action) suggested by the classifier can be incorporated into the incremental parse tree. Consequently, in the connection path method there are fewer opportunities to correct the errors made by the classifiers.

5.4 Evaluation Metrics

We evaluate dialog act classification and prediction by comparing the automatically assigned dialog act tags to the reference dialog act tags. For these tasks we report accuracy. We evaluate subtask classification and prediction by comparing the subtask trees output by the different parsing methods to the reference subtask tree. We use the labeled crossing bracket metric (typically used in the syntactic parsing literature (Harrison et al., 1991)), which computes recall, precision and crossing brackets for the constituents (subtrees) in a hypothesized parse tree given the reference parse tree. We report F-measure, which is a combination of recall and precision.

For each task, performance is reported for 1, 3,

[3]This is an unrealistic assumption if the decoder is to serve as a dialog model. We expect to address this limitation in future work.

5, and 10-best dynamic decoding as well as oracle (Or) and for 0, 1 and 3 utterances of context.

5.5 Results

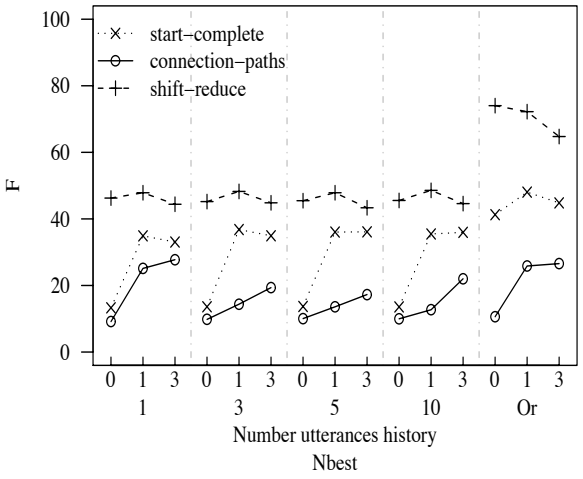

Figure 5: Performance of parse-based methods for subtask tree building

Figure 5 shows the performance of the different methods for determining the subtask tree of the dialog. Wider beam widths do not lead to improved performance for any method. One utterance of context is best for shift-reduce and start-join; three is best for the connection path method. The shift-reduce method performs the best. With 1 utterance of context, its 1-best f-score is 47.86, as compared with 34.91 for start-complete, 25.13 for the connection path method, and 21.32 for the chunk-based baseline. These performance differences are statistically significant at $p < .001$. However, the best performance for the shift-reduce method is still significantly worse than oracle.

All of the methods are subject to some 'stickiness', a certain preference to stay within the current subtask rather than starting a new one. Also, all of the methods tended to perform poorly on parsing subtasks that occur rarely (e.g. *call-forward, order-change*) or that occur at many different locations in the dialog (e.g. *out-of-domain, order-problem, check-availability*). For example, the shift-reduce method did not make many *shift* errors but did frequently *b-reduce* on an incorrect non-terminal (indicating trouble identifying subtask boundaries). Some non-terminals most likely to be labeled incorrectly by this method (for both agent and user) are: *call-forward, order-change, summary, order-problem, opening* and *out-of-domain*.

Similarly, the start-complete method frequently mislabeled a non-terminal in a *complete* action, e.g. *misc-other, check-availability, summary* or *contact-info*. It also quite frequently mislabeled nonterminals in *n-start* actions, e.g. *order-item, contact-info* or *summary*. Both of these errors indicate trouble identifying subtask boundaries.

It is harder to analyze the output from the connection path method. This method is more likely to mislabel tree-internal nodes than those immediately above the leaves. However, the same non-terminals show up as error-prone for this method as for the others: *out-of-domain, check-availability, order-problem* and *summary*.

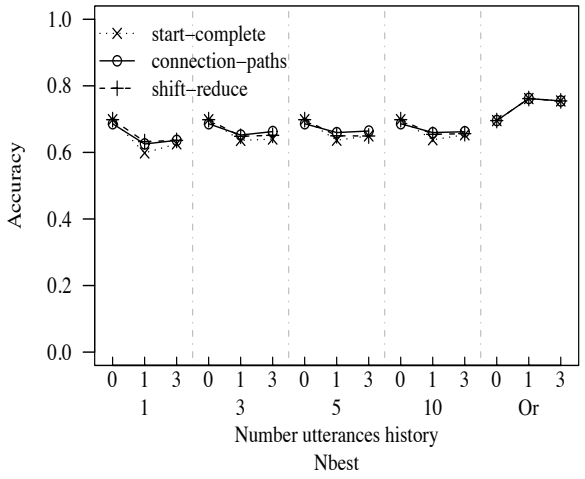

Figure 6: Performance of dialog act assignment to user's utterances.

Figure 6 shows accuracy for classification of user dialog acts. Wider beam widths do not lead to signficantly improved performance for any method. Zero utterances of context gives the highest accuracy for all methods. All methods perform fairly well, but no method significantly outperforms any other: with 0 utterances of context, 1-best accuracy is .681 for the connection path method, .698 for the start-complete method and .698 for the shift-reduce method. We note that these results are competitive with those reported in the literature (e.g. (Poesio and Mikheev, 1998; Serafin and Eugenio, 2004)), although the dialog corpus and the label sets are different.

The most common errors in dialog act classification occur with dialog acts that occur 40 times or fewer in the testing data (out of 3610 testing utterances), and with *Not(Information)*.

Figure 7 shows accuracy for prediction of agent dialog acts. Performance for this task is lower than

Speaker	Utterance	Shift-Reduce	Start-Complete	Connection Path
A	This is Sally	shift, *Hello*	start-opening, *Hello*	opening_S, *Hello*
A	How may I help you	shift, binary-reduce-out-of-domain, *Hello*	complete-opening, *Hello*	opening_S, *Hello*
B	Yes	*Not(Information)*, shift, binary-reduce-out-of-domain	*Not(Information)*, complete-opening	*Not(Information)*, opening_S
B	Um I would like to place an order please	*Rquest(Make-Order)*, shift, binary-reduce-opening	*Rquest(Make-Order)*, complete-opening, n-start-S	*Rquest(Make-Order)*, opening_S
A	May I have your telephone number with the area code	shift, *Acknowledge*	start-contact-info, *Acknowledge*	contact-info_S, *Request(Phone-Number)*
B	Uh the phone number is [number]	*Explain(Phone-Number)*, shift, binary-reduce-contact-info	*Explain(Phone-Number)*, complete-contact-info	*Explain(Phone-Number)*, contact-info_S

Table 4: Dialog extract with subtask tree building actions for three parsing methods

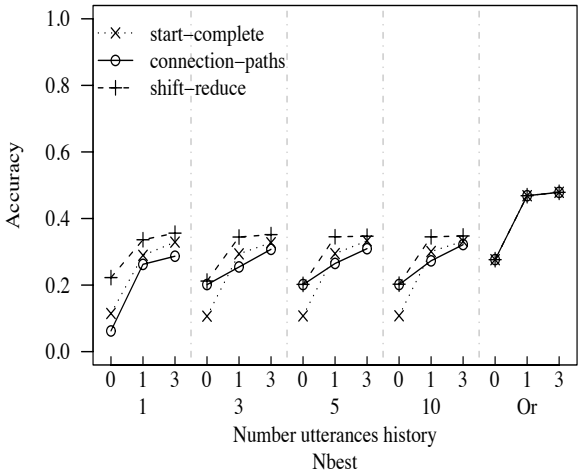

Figure 7: Performance of dialog act prediction used to generate agent utterances.

that for dialog act classification because this is a prediction task. Wider beam widths do not generally lead to improved performance for any method. Three utterances of context generally gives the best performance. The shift-reduce method performs significantly better than the connection path method with a beam width of 1 ($p < .01$), but not at larger beam widths; there are no other significant performance differences between methods at 3 utterances of context. With 3 utterances of context, 1-best accuracies are .286 for the connection path method, .329 for the start-complete method and .356 for the shift-reduce method.

The most common errors in dialog act prediction occur with rare dialog acts, *Not(Information)*, and the prediction of *Acknowledge* at the start of a turn (we did not remove grounding acts from the data). With the shift-reduce method, some *YNQ* acts are commonly mislabeled. With all methods, dialog acts pertaining to *Order-Info* and *Product-Info* acts are commonly mislabeled, which could potentially indicate that these labels require a subtle distinction between information pertaining to an order and information pertaining to a product.

Table 4 shows the parsing actions performed by each of our methods on the dialog snippet presented in Figure 4. For this example, the connection path method's output is correct in all cases.

6 Conclusions and Future Work

In this paper, we present a parsing-based model of task-oriented dialog that tightly integrates interpretation and generation using a subtask tree representation, can be trained from data, and runs incrementally for use in dialog management. At the core of this model is a parser that incrementally builds the dialog task structure as it interprets user actions and generates system actions. We experiment with three different incremental parsing methods for our dialog model. Our proposed shift-reduce method is the best-performing so far, and performance of this method for dialog act classification and task/subtask modeling is good enough to be usable. However, performance of all the methods for dialog act prediction is too low to be useful at the moment. In future work, we will explore improved models for this task that make use of global information about the task (e.g. whether each possible subtask has yet been completed; whether required and optional task-related concepts such as *shipping address* have been filled). We will also separate grounding and task-related behaviors in our model.

References

J. Alexandersson and N. Reithinger. 1997. Learning dialogue structures from a corpus. In *Proceedings of Eurospeech*.

J. Baldridge and A. Lascarides. 2005. Probabilistic head-driven parsing for discourse. In *Proceedings of CoNLL*.

S. Bangalore, G. Di Fabbrizio, and A. Stent. 2006. Learning the structure of task-driven human-human dialogs. In *Proceedings of COLING/ACL*.

A. Barrett and D. Weld. 1994. Task-decomposition via plan parsing. In *Proceedings of AAAI*.

A. Berger, S.D. Pietra, and V.D. Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

N. Blaylock and J. F. Allen. 2006. Hierarchical instantiated goal recognition. In *Proceedings of the AAAI Workshop on Modeling Others from Observations*.

D. Bohus and A. Rudnicky. 2003. RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Proceedings of Eurospeech*.

H.H. Bui. 2003. A general model for online probabalistic plan recognition. In *Proceedings of IJCAI*.

S. Carberry. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1–2):31–48.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–638.

F. Costa, V. Lombardo, P. Frasconi, and G. Soda. 2001. Wide coverage incremental parsing by learning attachment preferences. In *Proceedings of the Conference of the Italian Association for Artificial Intelligence (AIIA)*.

D. Cristea. 2000. An incremental discourse parser architecture. In *Proceedings of the 2nd International Conference on Natural Language Processing*.

K. Forbes, E. Miltsakaki, R. Prasad, A. Sarkar, A. Joshi, and B. Webber. 2003. D-LTAG system: Discourse parsing with a lexicalized tree-adjoining grammar. *Journal of Logic, Language and Information*, 12(3):261–279.

B.J. Grosz and C.L. Sidner. 1986. Attention, intentions and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

P. Haffner. 2006. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 48(3–4):239–261.

J. Hall, J. Nivre, and J. Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *Proceedings of COLING/ACL*.

P. Harrison, S. Abney, D. Fleckenger, C. Gdaniec, R. Grishman, D. Hindle, B. Ingria, M. Marcus, B. Santorini, and T. Strzalkowski. 1991. Evaluating syntax performance of parser/grammars of English. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems, ACL*.

H. LeThanh, G. Abeysinghe, and C. Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of COLING*.

D. Litman and J. Allen. 1987. A plan recognition model for subdialogs in conversations. *Cognitive Science*, 11(2):163–200.

K. Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572.

M. Poesio and A. Mikheev. 1998. The predictive power of game structure in dialogue act recognition: experimental results using maximum entropy estimation. In *Proceedings of ICSLP*.

L. Polanyi, C. Culy, M. van den Berg, G. L. Thione, and D. Ahn. 2004. A rule based approach to discourse parsing. In *Proceedings of SIGdial*.

D.V. Pynadath and M.P. Wellman. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of UAI*.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of EMNLP*.

C. Rich and C.L. Sidner. 1997. COLLAGEN: When agents collaborate with people. In *Proceedings of the First International Conference on Autonomous Agents*.

K. Sagae and A. Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of COLING/ACL*.

R. Serafin and B. Di Eugenio. 2004. FLSA: Extending latent semantic analysis with features for dialogue act classification. In *Proceedings of ACL*.

C.L. Sidner. 1985. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL/HLT*.

C. Sporleder and A. Lascarides. 2004. Combining hierarchical clustering and machine learning to predict high-level discourse structure. In *Proceedings of COLING*.