Learning the Structure of Task-driven Human-Human Dialogs

Srinivas Bangalore
AT&T Labs-Research
180 Park Ave
Florham Park, NJ 07932
srini@research.att.com

Giuseppe Di Fabbrizio
AT&T Labs-Research
180 Park Ave
Florham Park, NJ 07932
pino@research.att.com

Amanda Stent
Dept of Computer Science
Stony Brook University
Stony Brook, NY
stent@cs.sunysb.edu

Abstract

Data-driven techniques have been used for many computational linguistics tasks. Models derived from data are generally more robust than hand-crafted systems since they better reflect the distribution of the phenomena being modeled. With the availability of large corpora of spoken dialog, dialog management is now reaping the benefits of data-driven techniques. In this paper, we compare two approaches to modeling subtask structure in dialog: a chunk-based model of subdialog sequences, and a parse-based, or hierarchical, model. We evaluate these models using customer agent dialogs from a catalog service domain.

1 Introduction

As large amounts of language data have become available, approaches to sentence-level processing tasks such as parsing, language modeling, named-entity detection and machine translation have become increasingly data-driven and empirical. Models for these tasks can be trained to capture the distributions of phenomena in the data resulting in improved robustness and adaptability. However, this trend has yet to significantly impact approaches to dialog management in dialog systems. Dialog managers (both plan-based and call-flow based, for example (Di Fabbrizio and Lewis, 2004; Larsson et al., 1999)) have traditionally been hand-crafted and consequently somewhat brittle and rigid. With the ability to record, store and process large numbers of human-human dialogs (e.g. from call centers), we anticipate that data-driven methods will increasingly influence approaches to dialog management.

A successful dialog system relies on the synergistic working of several components: speech recognition (ASR), spoken language understanding (SLU), dialog management (DM), language generation (LG) and text-to-speech synthesis (TTS). While data-driven approaches to ASR and SLU are prevalent, such approaches to DM, LG and TTS are much less well-developed. In ongoing work, we are investigating data-driven approaches for building all components of spoken dialog systems.

In this paper, we address one aspect of this problem – *inferring predictive models to structure task-oriented dialogs*. We view this problem as a first step in predicting the system state of a dialog manager and in predicting the system utterance during an incremental execution of a dialog. In particular, we learn models for predicting dialog acts of utterances, and models for predicting subtask structures of dialogs. We use three different dialog act tag sets for three different human-human dialog corpora. We compare a flat chunk-based model to a hierarchical parse-based model as models for predicting the task structure of dialogs.

The outline of this paper is as follows: In Section 2, we review current approaches to building dialog systems. In Section 3, we review related work in data-driven dialog modeling. In Section 4, we present our view of analyzing the structure of task-oriented human-human dialogs. In Section 5, we discuss the problem of segmenting and labeling dialog structure and building models for predicting these labels. In Section 6, we report experimental results on Maptask, Switchboard and a dialog data collection from a catalog ordering service domain.

2 Current Methodology for Building Dialog systems

Current approaches to building dialog systems involve several manual steps and careful crafting of different modules for a particular domain or application. The process starts with a small scale "Wizard-of-Oz" data collection where subjects talk to a machine driven by a human 'behind the curtains'. A user experience (UE) engineer analyzes the collected dialogs, subject matter expert interviews, user testimonials and other evidences (e.g. customer care history records). This heterogeneous set of information helps the UE engineer to design some system functionalities, mainly: the semantic scope (e.g. call-types in the case of call routing systems), the LG model, and the DM strategy. A larger automated data collection follows, and the collected data is transcribed and labeled by expert labelers following the UE engineer recommendations. Finally, the transcribed and labeled data is used to train both the ASR and the SLU.

This approach has proven itself in many commercial dialog systems. However, the initial UE requirements phase is an expensive and error-prone process because it involves non-trivial design decisions that can only be evaluated after system deployment. Moreover, scalability is compromised by the time, cost and high level of UE know-how needed to reach a consistent design.

The process of building speech-enabled automated contact center services has been formalized and cast into a scalable commercial environment in which dialog components developed for different applications are reused and adapted (Gilbert et al., 2005). However, we still believe that exploiting dialog data to train/adapt or complement hand-crafted components will be vital for robust and adaptable spoken dialog systems.

3 Related Work

In this paper, we discuss methods for automatically creating models of dialog structure using dialog act and task/subtask information. Relevant related work includes research on automatic dialog act tagging and stochastic dialog management, and on building hierarchical models of plans using task/subtask information.

There has been considerable research on statistical dialog act tagging (Core, 1998; Jurafsky et al., 1998; Poesio and Mikheev, 1998; Samuel et al., 1998; Stolcke et al., 2000; Hastie et al., 2002). Several disambiguation methods (n-gram models, hidden Markov models, maximum entropy models) that include a variety of features (cue phrases, speaker ID, word n-grams, prosodic features, syntactic features, dialog history) have been used. In this paper, we show that use of extended context gives improved results for this task.

Approaches to dialog management include AI-style plan recognition-based approaches (e.g. (Sidner, 1985; Litman and Allen, 1987; Rich and Sidner, 1997; Carberry, 2001; Bohus and Rudnicky, 2003)) and information state-based approaches (e.g. (Larsson et al., 1999; Bos et al., 2003; Lemon and Gruenstein, 2004)). In recent years, there has been considerable research on how to automatically learn models of both types from data. Researchers who treat dialog as a sequence of information states have used reinforcement learning and/or Markov decision processes to build stochastic models for dialog management that are evaluated by means of dialog simulations (Levin and Pieraccini, 1997; Scheffler and Young, 2002; Singh et al., 2002; Williams et al., 2005; Henderson et al., 2005; Frampton and Lemon, 2005). Most recently, Henderson et al. showed that it is possible to automatically learn good dialog management strategies from automatically labeled data over a large potential space of dialog states (Henderson et al., 2005); and Frampton and Lemon showed that the use of context information (the user's last dialog act) can improve the performance of learned strategies (Frampton and Lemon, 2005). In this paper, we combine the use of automatically labeled data and extended context for automatic dialog modeling.

Other researchers have looked at probabilistic models for plan recognition such as extensions of Hidden Markov Models (Bui, 2003) and probabilistic context-free grammars (Alexandersson and Reithinger, 1997; Pynadath and Wellman, 2000). In this paper, we compare hierarchical grammar-style and flat chunking-style models of dialog.

In recent research, Hardy (2004) used a large corpus of transcribed and annotated telephone conversations to develop the Amities dialog system. For their dialog manager, they trained separate task and dialog act classifiers on this corpus. For task identification they report an accuracy of 85% (true task is one of the top 2 results returned by the classifier); for dialog act tagging they report 86% accuracy.

4 Structural Analysis of a Dialog

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 that encapsulates the task structure (dominance and precedence relations among tasks), dialog act structure (sequences of dialog acts), and linguistic structure of utterances (inter-clausal relations and predicate-argument relations within a clause), as illustrated in Figure 1. As the dialog proceeds, an utterance from a participant is accommodated into the tree in an incremental manner, much like an incremental syntactic parser accommodates the next word into a partial parse tree (Alexandersson and Reithinger, 1997). With this model, we can tightly couple language understanding and dialog management using a shared representation, which leads to improved accuracy (Taylor et al., 1998).

In order to infer models for predicting the structure of task-oriented dialogs, we label human-human dialogs with the hierarchical information shown in Figure 1 in several stages: utterance segmentation (Section 4.1), syntactic annotation (Section 4.2), dialog act tagging (Section 4.3) and subtask labeling (Section 5).

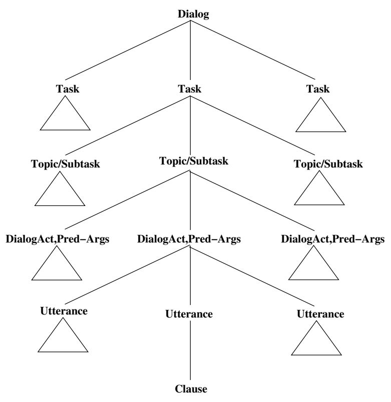

Figure 1: Structural analysis of a dialog

4.1 Utterance Segmentation

The task of ”cleaning up” spoken language utterances by detecting and removing speech repairs and dysfluencies and identifying sentence boundaries has been a focus of spoken language parsing research for several years (e.g. (Bear et al., 1992; Seneff, 1992; Shriberg et al., 2000; Charniak and Johnson, 2001)). We use a system that segments the ASR output of a user's utterance into clauses. The system annotates an utterance for sentence boundaries, restarts and repairs, and identifies coordinating conjunctions, filled pauses and discourse markers. These annotations are done using a cascade of classifiers, details of which are described in (Bangalore and Gupta, 2004).

4.2 Syntactic Annotation

We automatically annotate a user's utterance with supertags (Bangalore and Joshi, 1999). Supertags encapsulate predicate-argument information in a local structure. They are composed with each other using the substitution and adjunction operations of Tree-Adjoining Grammars (Joshi, 1987) to derive a dependency analysis of an utterance and its predicate-argument structure.

4.3 Dialog Act Tagging

We use a domain-specific dialog act tagging scheme based on an adapted version of DAMSL (Core, 1998). The DAMSL scheme is quite comprehensive, but as others have also found (Jurafsky et al., 1998), the multi-dimensionality of the scheme makes the building of models from DAMSL-tagged data complex. Furthermore, the generality of the DAMSL tags reduces their utility for natural language generation. Other tagging schemes, such as the Maptask scheme (Carletta et al., 1997), are also too general for our purposes. We were particularly concerned with obtaining sufficient discriminatory power between different types of statement (for generation), and to include an out-of-domain tag (for interpretation). We provide a sample list of our dialog act tags in Table 2. Our experiments in automatic dialog act tagging are described in Section 6.3.

5 Modeling Subtask Structure

Figure 2 shows the task structure 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 structure can be as deep as five levels. Most often the nesting is at the left or right frontier of the subtask tree.

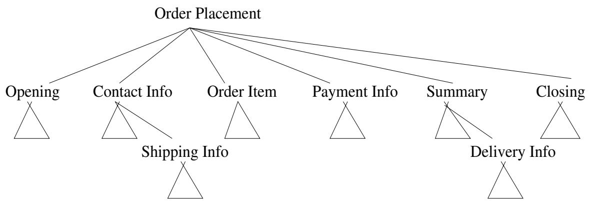

Figure 2: A sample task structure in our application domain.

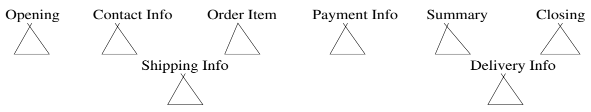

Figure 3: An example output of the chunk model's task structure

The goal of subtask segmentation is to predict if the current utterance in the dialog is part of the current subtask or starts a new subtask. We compare two models for recovering the subtask structure – a chunk-based model and a parse-based model. In the chunk-based model, we recover the precedence relations (sequence) of the subtasks but not dominance relations (subtask structure) among the subtasks. Figure 3 shows a sample output from the chunk model. In the parse model, we recover the complete task structure from the sequence of utterances as shown in Figure 2. Here, we describe our two models. We present our experiments on subtask segmentation and labeling in Section 6.4.

5.1 Chunk-based model

This model is similar to the second one described in (Poesio and Mikheev, 1998), except that we use tasks and subtasks rather than dialog games. We model the prediction problem as a classification task as follows: given a sequence of utterances u_i in a dialog $U = u_1, u_2, \ldots, u_n$ and a subtask label vocabulary ($st_i \in \mathcal{ST}$), we need to predict the best subtask label sequence $ST^* = st_1, st_2, \ldots, st_m$ as shown in equation 1.

$$ST^* = \underset{ST}{argmax}\, P(ST|U) \tag{1}$$

Each subtask has *begin*, *middle* (possibly absent) and *end* utterances. If we incorporate this information, the refined vocabulary of subtask labels is $\mathcal{ST}_r = \{st_i^b, st_i^m, st_i^e \mid st_i \in \mathcal{ST}\}$. In our experiments, we use a classifier to assign to each utterance a refined subtask label conditioned on a vector of local contextual features ($\mathbf{\Phi}$). In the interest of using an incremental left-to-right decoder, we restrict the contextual features to be from the preceding context only. Furthermore, the search is limited to the label sequences that respect precedence among the refined labels (*begin* $<$ *middle* $<$ *end*). This constraint is expressed in a grammar G encoded as a regular expression ($L(G) = \bigcup_i (st_i^b\ (st_i^m)^*\ st_i^e)^*$). However, in order to cope with the prediction errors of the classifier, we approximate $L(G)$ with an n-gram language model on sequences of the refined tag labels:

$$ST_r^* = \underset{\substack{ST_r \\ ST_r \in L(G)}}{argmax} P(ST_r|U) \tag{2}$$

$$\approx \underset{\substack{ST_r \\ ST_r \in L(G)}}{argmax} \prod_i^n P(st_i|\mathbf{\Phi}) \tag{3}$$

In order to estimate the conditional distribution $P(st_i|\mathbf{\Phi})$ we use the general technique of choosing the maximum entropy (maxent) distribution that properly estimates the average of each feature over the training data (Berger et al., 1996). This can be written as a Gibbs distribution parameterized with weights λ, where V is the size of the label set. Thus,

$$P(st_i|\mathbf{\Phi}) = \frac{e^{\lambda_{\mathbf{st_i}} \cdot \mathbf{\Phi}}}{\sum_{st=1}^{V} e^{\lambda_{\mathbf{st}} \cdot \mathbf{\Phi}}} \tag{4}$$

We use the machine learning toolkit LLAMA (Haffner, 2006) to estimate the conditional distribution using maxent. LLAMA encodes multiclass maxent as binary maxent, in order to increase the speed of training and to scale this method to large data sets. Each of the V classes in the set $\mathcal{ST}_r$ is encoded as a bit vector such that, in the vector for class i, the i^th bit is one and all other bits are zero. Then, V one-vs-other binary classifiers are used as follows.

$$P(y|\Phi) = 1 - P(\bar{y}|\Phi) = \frac{e^{\lambda_y \cdot \Phi}}{e^{\lambda_y \cdot \Phi} + e^{\lambda_{\bar{y}} \cdot \Phi}} = \frac{1}{1 + e^{-\lambda'_y \cdot \Phi}} \tag{5}$$

where $\lambda_{\bar{y}}$ is the parameter vector for the *anti-label* $\bar{y}$ and $\lambda'_y = \lambda_y - \lambda_{\bar{y}}$. In order to compute $P(st_i|\mathbf{\Phi})$, we use class independence assumption and require that $y_i = 1$ and for all $j \neq i\ y_j = 0$.

$$P(st_i|\Phi) = P(y_i|\Phi) \prod_{j \neq i}^{V} P(y_j|\Phi)$$

5.2 Parse-based Model

As seen in Figure 3, the chunk model does not capture dominance relations among subtasks, which are important for resolving anaphoric references (Grosz and Sidner, 1986). Also, the chunk 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.

In this model, we are interested in finding the most likely plan tree (PT) given the sequence of utterances:

$$PT^* = \underset{PT}{argmax}\, P(PT|U) \tag{6}$$

For real-time dialog management we use a top-down incremental parser that incorporates bottom-up information (Roark, 2001).

We rewrite equation (6) to exploit the subtask sequence provided by the chunk model as shown in Equation 7. For the purpose of this paper, we approximate Equation 7 using one-best (or *k-best*) chunk output.[1]

$$PT^* = \underset{PT}{argmax} \sum_{ST} P(ST|U) P(PT|ST) \tag{7}$$

$$\approx \underset{PT}{argmax}\, P(PT|ST^*) \tag{8}$$

where

$$ST^* = \underset{ST}{argmax}\, P(ST|U) \tag{9}$$

6 Experiments and Results

In this section, we present the results of our experiments for modeling subtask structure.

6.1 Data

As our primary data set, we used 915 telephone-based customer-agent dialogs related to the task of ordering products from a catalog. Each dialog was transcribed by hand; all numbers (telephone, credit card, etc.) were removed for privacy reasons. The average dialog lasted for 3.71

[1]However, it is conceivable to parse the multiple hypotheses of chunks (encoded as a weighted lattice) produced by the chunk model.

minutes and included 61.45 changes of speaker. A single customer-service representative might participate in several dialogs, but customers are represented by only one dialog each. Although the majority of the dialogs were on-topic, some were idiosyncratic, including: requests for order corrections, transfers to customer service, incorrectly dialed numbers, and long friendly out-of-domain asides. Annotations applied to these dialogs include: utterance segmentation (Section 4.1), syntactic annotation (Section 4.2), dialog act tagging (Section 4.3) and subtask segmentation (Section 5). The former two annotations are domain-independent while the latter are domain-specific.

6.2 Features

Offline natural language processing systems, such as part-of-speech taggers and chunkers, rely on both *static* and *dynamic* features. Static features are derived from the local context of the text being tagged. Dynamic features are computed based on previous predictions. The use of dynamic features usually requires a search for the globally optimal sequence, which is not possible when doing incremental processing. For dialog act tagging and subtask segmentation during dialog management, we need to predict incrementally since it would be unrealistic to wait for the entire dialog before decoding. Thus, in order to train the dialog act (DA) and subtask segmentation classifiers, we use only *static* features from the current and left context as shown in Table 1.[2] This obviates the need for constructing a search network and performing a dynamic programming search during decoding. In lieu of the dynamic context, we use larger static context to compute features – word trigrams and trigrams of words annotated with supertags computed from up to three previous utterances.

Label Type	Features
Dialog Acts	Speaker, word trigrams from current/previous utterance(s) supertagged utterance
Subtask	Speaker, word trigrams from current utterance, previous utterance(s)/turn

Table 1: Features used for the classifiers.

6.3 Dialog Act Labeling

For dialog act labeling, we built models from our corpus and from the Maptask (Carletta et al., 1997) and Switchboard-DAMSL (Jurafsky et al., 1998) corpora. From the files for the Maptask corpus, we extracted the moves, words and speaker information (follower/giver). Instead of using the raw move information, we augmented each move with speaker information, so that for example, the *instruct* move was split into *instruct-giver* and *instruct-follower*. For the Switchboard corpus, we clustered the original labels, removing most of the multidimensional tags and combining together tags with minimum training data as described in (Jurafsky et al., 1998). For all three corpora, non-sentence elements (e.g., dysfluencies, *discourse markers*, etc.) and restarts (with and without repairs) were kept; non-verbal content (e.g., laughs, background noise, etc.) was removed.

As mentioned in Section 4, we use a domain-specific tag set containing 67 dialog act tags for the catalog corpus. In Table 2, we give examples of our tags. We manually annotated 1864 clauses from 20 dialogs selected at random from our corpus and used a ten-fold cross-validation scheme for testing. In our annotation, a single utterance may have multiple dialog act labels. For our experiments with the Switchboard-DAMSL corpus, we used 42 dialog act tags obtained by clustering over the 375 unique tags in the data. This corpus has 1155 dialogs and 218,898 utterances; 173 dialogs, selected at random, were used for testing. The Maptask tagging scheme has 12 unique dialog act tags; augmented with speaker information, we get 24 tags. This corpus has 128 dialogs and 26181 utterances; ten-fold cross validation was used for testing.

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 2: Sample set of dialog act labels

Table 3 shows the error rates for automatic dialog act labeling using word trigram features from the current and previous utterance. We compare error rates for our tag set to those of Switchboard-DAMSL and Maptask using the same features and the same classifier learner. The error rates for the catalog and the Maptask corpus are an average of ten-fold cross-validation. We suspect that the larger error rate for our domain compared to Maptask and Switchboard might be due to the small size of our annotated corpus (about 2K utterances for our domain as against about 20K utterances for

[2]We could use dynamic contexts as well and adopt a greedy decoding algorithm instead of a viterbi search. We have not explored this approach in this paper.

Maptask and 200K utterances for DAMSL).

The error rates for the Switchboard-DAMSL data are significantly better than previously published results (28% error rate) (Jurafsky et al., 1998) with the same tag set. This improvement is attributable to the richer feature set we use and a discriminative modeling framework that supports a large number of features, in contrast to the generative model used in (Jurafsky et al., 1998). A similar obeservation applies to the results on Maptask dialog act tagging. Our model outperforms previously published results (42.8% error rate) (Poesio and Mikheev, 1998).

In labeling the Switchboard data, long utterances were split into *slash units* (Meteer et.al., 1995). A speaker's turn can be divided in one or more slash units and a slash unit can extend over multiple turns, for example:

sv B.64 utt3: C but, F uh –
b A.65 utt1: Uh-huh. /
+ B.66 utt1: – people want all of that /
sv B.66 utt2: C and not all of those are necessities. /
b A.67 utt1: Right . /

The labelers were instructed to label on the basis of the whole slash unit. This makes, for example, the dysfluency turn B.64 a Statement opinion (sv) rather than a non-verbal. For the purpose of discriminative learning, this could introduce noisy data since the context associated to the labeling decision shows later in the dialog. To address this issue, we compare 2 classifiers: the first (non-merged), simply propagates the same label to each continuation, cross turn slash unit; the second (merged) combines the units in one single utterance. Although the *merged* classifier breaks the regular structure of the dialog, the results in Table 3 show better overall performance.

Tagset	current utterance	+ stagged utterance	+ 3 previous (stagged) utterance
Catalog Domain	46.3	46.1	42.2
DAMSL (non-merged)	24.7	23.8	19.1
DAMSL (merged)	22.0	20.6	16.5
Maptask	34.3	33.9	30.3

Table 3: Error rates in dialog act tagging

6.4 Subtask Segmentation and Labeling

For subtask labeling, we used a random partition of 864 dialogs from our catalog domain as the training set and 51 dialogs as the test set. All the dialogs were annotated with subtask labels by hand. We used a set of 18 labels grouped as shown in Figure 4.

Type	Subtask Labels
1	opening, closing
2	contact-information, delivery-information, payment-information, shipping-address,summary
3	order-item, related-offer, order-problem discount, order-change, check-availability
4	call-forward, out-of-domain, misc-other, sub-call

Table 4: Subtask label set

6.4.1 Chunk-based Model

Table 5 shows error rates on the test set when predicting refined subtask labels using word n-gram features computed on different dialog contexts. The well-formedness constraint on the refined subtask labels significantly improves prediction accuracy. Utterance context is also very helpful; just one utterance of left-hand context leads to a 10% absolute reduction in error rate, with further reductions for additional context. While the use of trigram features helps, it is not as helpful as other contextual information. We used the dialog act tagger trained from Switchboard-DAMSL corpus to automatically annotate the catalog domain utterances. We included these tags as features for the classifier, however, we did not see an improvement in the error rates, probably due to the high error rate of the dialog act tagger.

Feature Context	Utterance Context		
	Current utt/with DA	+prev utt/with DA	+three prev utt/with DA
Unigram	42.9/42.4 (53.4/52.8)	33.6/34.1 (43.0/43.0)	30.0/30.3 (37.6/37.6)
Trigram	41.7/41.7 (52.5/52.0)	31.6/31.4 (42.9/42.7)	30.0/29.1 (37.6/37.4)

Table 5: Error rate for predicting the refined subtask labels. The error rates without the well-formedness constraint is shown in parenthesis. The error rates with dialog acts as features are separated by a slash.

6.4.2 Parsing-based Model

We retrained a top-down incremental parser (Roark, 2001) on the plan trees in the training dialogs. For the test dialogs, we used the k-best (k=50) refined subtask labels for each utterance as predicted by the chunk-based classifier to create a lattice of subtask label sequences. For each dialog we then created n-best sequences (100-best for these experiments) of subtask labels; these were parsed and (re-)ranked by the parser.[3] We combine the weights of the subtask label sequences assigned by the classifier with the parse score assigned by the parser and select the top

[3]Ideally, we would have parsed the subtask label lattice directly, however, the parser has to be reimplemented to parse such lattice inputs.

Features	Constraints		
	No Constraint	Sequence Constraint	Parser Constraint
Current Utt	54.4	42.0	41.5
+ DA	53.8	40.5	40.2
Current+Prev Utt	41.6	27.7	27.7
+DA	40.0	28.8	28.1
Current+3 Prev Utt	37.5	24.7	24.7
+DA	39.7	29.6	28.9

Table 6: Error rates for task structure prediction, with no constraints, sequence constraints and parser constraints

scoring sequence from the list for each dialog. The results are shown in Table 6. It can be seen that using the parsing constraint does not help the subtask label sequence prediction significantly. The chunk-based model gives almost the same accuracy, and is incremental and more efficient.

7 Discussion

The experiments reported in this section have been performed on transcribed speech. The audio for these dialogs, collected at a call center, were stored in a compressed format, so the speech recognition error rate is high. In future work, we will assess the performance of dialog structure prediction on recognized speech.

The research presented in this paper is but one step, albeit a crucial one, towards achieving the goal of inducing human-machine dialog systems using human-human dialogs. Dialog structure information is necessary for language generation (predicting the agents' response) and dialog state specific text-to-speech synthesis. However, there are several challenging problems that remain to be addressed.

The structuring of dialogs has another application in call center analytics. It is routine practice to monitor, analyze and mine call center data based on indicators such as the average length of dialogs, the task completion rate in order to estimate the efficiency of a call center. By incorporating structure to the dialogs, as presented in this paper, the analysis of dialogs can be performed at a more fine-grained (task and subtask) level.

8 Conclusions

In order to build a dialog manager using a data-driven approach, the following are necessary: a model for labeling/interpreting the user's current action; a model for identifying the current subtask/topic; and a model for predicting what the system's next action should be. Prior research in plan identification and in dialog act labeling has identified possible features for use in such models, but has not looked at the performance of different feature sets (reflecting different amounts of context and different views of dialog) across different domains (label sets). In this paper, we compared the performance of a dialog act labeler/predictor across three different tag sets: one using very detailed, domain-specific dialog acts usable for interpretation and generation; and two using general-purpose dialog acts and corpora available to the larger research community. We then compared two models for subtask labeling: a flat, chunk-based model and a hierarchical, parsing-based model. Findings include that simpler chunk-based models perform as well as hierarchical models for subtask labeling and that a dialog act feature is not helpful for subtask labeling.

In on-going work, we are using our best performing models for both DM and LG components (to predict the next dialog move(s), and to select the next system utterance). In future work, we will address the use of data-driven dialog management to improve SLU.

9 Acknowledgments

We thank Barbara Hollister and her team for their effort in annotating the dialogs for dialog acts and subtask structure. We thank Patrick Haffner for providing us with the LLAMA machine learning toolkit and Brian Roark for providing us with his top-down parser used in our experiments. We also thank Alistair Conkie, Mazin Gilbert, Narendra Gupta, and Benjamin Stern for discussions during the course of this work.

References

J. Alexandersson and N. Reithinger. 1997. Learning dialogue structures from a corpus. In *Proceedings of Eurospeech'97*.

S. Bangalore and N. Gupta. 2004. Extracting clauses in dialogue corpora : Application to spoken language understanding. *Journal Traitement Automatique des Langues (TAL)*, 45(2).

S. Bangalore and A. K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

J. Bear et al. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Proceedings of ACL'92*.

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

D. Bohus and A. Rudnicky. 2003. RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Proceedings of Eurospeech'03*.

J. Bos et al. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *Proceedings of SIGdial*.

H.H. Bui. 2003. A general model for online probabalistic plan recognition. In *Proceedings of IJCAI'03*.

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

J. Carletta et al. 1997. The reliability of a dialog structure coding scheme. *Computational Linguistics*, 23(1).

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of NAACL'01*.

M. Core. 1998. Analyzing and predicting patterns of DAMSL utterance tags. In *Proceedings of the AAAI spring symposium on Applying machine learning to discourse processing*.

M. Meteer et.al. 1995. Dysfluency annotation stylebook for the switchboard corpus. Distributed by LDC.

G. Di Fabbrizio and C. Lewis. 2004. Florence: a dialogue manager framework for spoken dialogue systems. In *ICSLP 2004, 8th International Conference on Spoken Language Processing*, Jeju, Jeju Island, Korea, October 4-8.

M. Frampton and O. Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *Proceedings of the 4th IJCAI workshop on knowledge and reasoning in practical dialogue systems*.

M. Gilbert et al. 2005. Intelligent virtual agents for contact center automation. *IEEE Signal Processing Magazine*, 22(5), September.

B.J. Grosz and C.L. Sidner. 1986. Attention, intentions and the structure of discoursep. *Computational Linguistics*, 12(3).

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

H. Hardy et al. 2004. Data-driven strategies for an automated dialogue system. In *Proceedings of ACL'04*.

H. Wright Hastie et al. 2002. Automatically predicting dialogue structure using prosodic features. *Speech Communication*, 36(1–2).

J. Henderson et al. 2005. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. In *Proceedings of the 4th IJCAI workshop on knowledge and reasoning in practical dialogue systems*.

A. K. Joshi. 1987. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

D. Jurafsky et al. 1998. Switchboard discourse language modeling project report. Technical Report Research Note 30, Center for Speech and Language Processing, Johns Hopkins University, Baltimore, MD.

S. Larsson et al. 1999. TrindiKit manual. Technical report, TRINDI Deliverable D2.2.

O. Lemon and A. Gruenstein. 2004. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction*, 11(3).

E. Levin and R. Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of Eurospeech'97*.

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

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

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'98*.

D.V. Pynadath and M.P. Wellman. 2000. Probabilistic state-dependent grammars for plan recognition. In *In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*.

C. Rich and C.L. Sidner. 1997. COLLAGEN: When agents collaborate with people. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2).

K. Samuel et al. 1998. Computing dialogue acts from features with transformation-based learning. In *Proceedings of the AAAI spring symposium on Applying machine learning to discourse processing*.

K. Scheffler and S. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT'02*.

S. Seneff. 1992. A relaxation method for understanding spontaneous speech utterances. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, CA.

E. Shriberg et al. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32, September.

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

S. Singh et al. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16.

A. Stolcke et al. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3).

P. Taylor et al. 1998. Intonation and dialogue context as constraints for speech recognition. *Language and Speech*, 41(3).

J. Williams et al. 2005. Partially observable Markov decision processes with continuous observations for dialogue management. In *Proceedings of SIGdial*.