Learning Dialogue Management Models for Task-Oriented Dialogue with Parallel Dialogue and Task Streams

Eun Young Ha, Christopher M. Mitchell, Kristy Elizabeth Boyer, and James C. Lester
Department of Computer Science
North Carolina State University
Raleigh, NC 27695, USA
{eha,cmmitch2,keboyer,lester}@ncsu.edu

Abstract

Learning dialogue management models poses significant challenges. In a complex task-oriented domain in which information is exchanged via parallel, interleaved dialogue and task streams, effective dialogue management models should be able to make dialogue moves based on both the dialogue and the task context. This paper presents a data-driven approach to learning dialogue management models that determine when to make dialogue moves to assist users' task completion activities, as well as the type of dialogue move that should be selected for a given user interaction context. Combining features automatically extracted from the dialogue and the task, we compare two alternate modeling approaches. The results of an evaluation indicate the learned models are effective in predicting both the timing and the type of system dialogue moves.

1 Introduction

Automated dialogue systems allow users to interact with information systems in a natural and intuitive manner. With the growth of speech-enabled applications for mobile devices, the demands for practical dialogue systems have been increasing at an accelerating pace. The core tasks of automated dialogue systems include dialogue management, which is concerned with selecting system actions in response to a given user input. Traditionally, dialogue managers have been manually constructed. However, manually crafting dialogue managers is labor-intensive and yields systems that are brittle with respect to unexpected user behaviors. For rapid creation of robust and adaptive dialogue systems, data-driven approaches to dialogue management hold much appeal. Recent work on dialogue systems has explored machine learning techniques to automatically learn dialogue managers from corpora (Scheffler and Young, 2002; Hardy et al., 2006; Williams and Young, 2007; Bangalore et al., 2008; Sridhar et al., 2009).

To support more natural human-computer dialogue, earlier work on dialogue systems envisioned rich interaction environments that take into account observed user actions for selecting optimal dialogue strategies (Carberry, 1990; Rich and Sidner, 1998; Allen et al., 2001). However, recent data-driven approaches have primarily focused on application domains in which information between the user and the system are communicated solely by dialogue, such as telephone-based systems (Hardy et al., 2006; Bangalore et al., 2008) and online chat dialogues (Ivanovic, 2008; Kim et al., 2010). With increasing demands for natural human-computer interaction beyond these restricted application domains, dialogue systems are required to support more complex types of interaction, in which users perform tasks in parallel to exchanging dialogue. For instance, dialogue interfaces for task-assistance systems, such as intelligent tutoring systems, should be able to monitor users' task completion activities and incorporate the observed activities into dialogue management decisions such that the systems can provide users with spontaneous assistance (e.g., providing hints) even without an explicit request from the user.

We have been exploring data-driven approaches for a complex task-oriented application domain in which information is delivered both by exchanging dialogue with users and by observing users' task completion activities. Our previous work has focused on the automatic interpretation of user dialogue input (Boyer et al., 2010; Ha et al., 2012). Findings suggest that identifying an effective representation to combine information from dialogue and users' task completion activities is key to effective dialogue processing in a domain consisting of parallel dialogue and task streams.

As the next step in this line of investigation on complex task-oriented domains with parallel dialogue and task streams, this work proposes an approach to automatically learning dialogue management models from a human dialogue corpus. The proposed approach combines information from a dialogue stream and a task stream in order to create spontaneous dialogue interventions for users based on monitoring users' activities. Two subtasks of dialogue management are addressed: the first is to determine when to provide dialogue feedback (*timing*), and the second is to determine what kind of dialogue feedback to provide (*type*). Dialogue managers in conventional domains have primarily focused on the selection of feedback type. However, determining the appropriate timing of system moves is critical for dialogue systems that support parallel dialogue and task streams.

The work presented here makes three contributions. First, it endeavors to expand data-driven dialogue management by addressing more complex task-oriented domains consisting of parallel dialogue and task streams. Second, it proposes a timing intervention model that determines the correct time to make spontaneous system interventions. Third, it presents a maximum entropy dialogue management model and compares alternate approaches. It also compares the predictive power of the dialogue and task streams on the targeted dialogue management tasks.

2 Related Work

Data-driven approaches to dialogue management continue to be the subject of increasing attention within the dialogue community. Prominent among these are reinforcement learning approaches for learning dialogue policies from corpora (Henderson et al., 2008; Levin et al., 2000; Lewis and Di Fabbrizio, 2006; Roy et al., 2000; Scheffler and Young, 2002; Singh et al., 2002; Williams and Young, 2007; Young, 2002). These approaches model dialogue as Markov decision processes, either fully observable (MDPs) or partially observable (POMDPs), in which the transitions of dialogue states are associated with system actions and rewards. The goal of reinforcement learning is to learn optimal policies that maximize aggregate expected rewards, such as user satisfaction (Walker et al., 1997). Learned policies that result from RL exploration do not, by design, necessarily reflect the patterns in the bootstrap dialogue corpus. Additionally, to cover all possible state spaces, reinforcement learning typically requires a very large set of training data, which limits the complexity of the dialogue system in its representation of the dialogue states and the system actions (Young et al., 2013).

A second body of related work focuses on dialogue act classification. Classification-based approaches aim at learning the patterns of dialogue that are present in the corpus. A variety of machine learning frameworks have been exploited, including hidden Markov models (Stolcke et al., 2000; Boyer et al., 2010), maximum entropy models (Bangalore et al., 2008; Sridhar et al., 2009; Ha et al., 2012), support vector machines (Ivanovic, 2008), conditional random fields (Kim et al., 2010), and memory-based classifiers in combination with latent semantic analysis (Di Eugenio et al., 2010). Classification-based approaches incorporate rich sets of features, including not only lexical information, syntactic features, and dialogue structure, but also prosodic features in the case of spoken dialogue (Stolcke et al., 2000; Sridhar et al., 2009) and non-verbal features such as facial expressions (Boyer et al., 2011) and shifts in posture (Ha et al., 2012).

While most work on dialogue act classification has focused on either offline analysis of dialogue (Stolcke et al., 2000; Ivanovic, 2008; Kim et al., 2010; Di Eugenio et al., 2010) or interpretation of user dialogue (Boyer et al., 2010; Ha et al., 2012), Bangalore et al. (2008) utilized dialogue act classification as a mechanism for determining system dialogue moves. They proposed a unified dialogue act classification approach for both the interpretation of user utterances and selection of system dialogue moves.

Our work is similar to Bangalore et al. (2008) in that it takes a dialogue act classification approach to the task of selecting system dialogue moves. However, it addresses the problems posed by complex task-oriented application domains in which information is communicated not only by dialogue exchanges but also by monitoring users' task performance. In such domains, a user's task activities constitute a full communicative stream in its own right, separate from the dialogue stream. The challenges of parallel dialogue and task streams are addressed by exploiting automatically obtained task features combined with dialogue features. In contrast to previous work (Bangalore et al. 2008, Boyer et al., 2010), in which task information was derived from manual annotation, our work utilizes automatically computed task features.

Our work also focuses on a growing application area of dialogue systems: intelligent tutoring. In support of student learning, recent work in this area utilized human tutorial dialogue corpora to learn effective tutorial strategies using MDPs (Chi et al., 2010; Mitchell et al., 2013), to develop tutorial dialogue models that adapt to students' affective states (Forbes-Riley and Litman, 2011), and to improve robustness of a symbolic tutorial dialogue system (Dzikovska et al., 2013).

3 Task-Oriented Dialogue Corpus

To learn dialogue management models from naturally occurring human-to-human dialogue we utilize a human tutorial dialogue corpus we collected in the domain of introductory programming in Java. The corpus consists of textual dialogue exchanges between students and tutors in a web-based remote-tutoring interface, aligned with task context logs (Appendix A). A subset of the corpus was annotated with dialogue acts, which was used to train and test the dialogue management models described in this paper.

3.1 Human tutoring study

The data collection study involved forty-two undergraduate students who were paired with one of four tutors. The students were enrolled in a first-year engineering course and were pre-screened to filter out those with significant programming experience. The students were compensated for their participation with partial course credit. The tutors were graduate students with previous tutoring or teaching experience in Java programming, and the students worked with the same tutor for the entire study. Each lesson consisted of between four and thirteen distinct subtasks.

The students completed six forty-minute tutoring lessons, covering progressive topics in introductory computer science over four weeks. Each lesson consisted of four to thirteen subtasks, in which later subtasks built upon earlier ones. During each tutoring session, the paired student and tutor interacted remotely using a web-based tutoring interface. With this tutoring interface, the student and the tutor were able to exchange textual dialogue and share a synchronized view of the task.

For each lesson, students completed a pre-test and a post-test before and after the main tutoring session. The pre- and post-test consisted of the same set of questions to assess students' knowledge related to the lesson's objectives. Compared to students' pre-test results, significant learning gains were observed on the post-test, which indicates that the tutorial dialogue was effective for student learning (Mitchell et al., 2012).

3.2 Dialogue annotation

A subset of the collected data was manually annotated with dialogue acts using an annotation scheme consisting of 13 dialogue act tags for task-oriented tutorial dialogue (Table 1). The annotated corpus consists of the first of the six tutoring lessons from 21 students, which contains 2564 utterances (1777 tutor, 787 student). The average number of utterances per tutoring session was 122 (min = 74; max = 201). The average number of tutor utterances per session was 84.6 (min = 51; max = 137), and the average number of student utterances per session was 37.4 (min = 22; max = 64).

Three human annotators were trained to apply the scheme. The training consisted of an iterative process involving collaborative and independent tagging, followed by refinements of the tagging protocol. At the initial phase of training, the annotators tagged the corpus collaboratively. In later phases annotators tagged independently. To compute agreement between different annotators, 24% (5 of the 21 sessions) of the corpus were doubly annotated by two annotators. All possible pairs of the annotators participated in double annotation. The aggregate agreement was 0.80 in Cohen's Kappa (Cohen, 1960).

4 Dialogue Management Models

To support a task-oriented dialogue system capable of not only responding to users' dialogue input but also providing spontaneous system intervention during users' task activities, a dialogue manager should provide two functionalities. The first is to determine the *timing* of a system dialogue move (i.e., whether or not to provide a tutorial dialogue move at a given context). The second is to determine the *type* of dialogue move (i.e., selecting from available system dialogue acts). In this work, the problem of determining the system's next dialogue move is cast as a classification task. In previous work we found a maximum entropy approach was effective for

Tag	Description	Agreement
H	**Hint:** The tutor gives advice to help the student proceed with the task	.50
DIR	**Directive:** The tutor explicitly tells the student the next step to take	.63
ACK	**Acknowledgement:** Either the tutor or the student acknowledges previous utterance; conversational grounding	.73
RC	**Request for Confirmation:** Either the tutor or the student requests confirmation from the other participant (e.g., *"Make sense?"*)	Insufficient data
RF	**Request for Feedback:** The student requests an assessment of his performance or his work from the tutor	1.0
PF	**Positive Feedback:** The tutor provides a positive assessment of the student's performance	.90
LF	**Lukewarm Feedback:** The tutor provides an assessment that has both positive and negative elements	.80
NF	**Negative Feedback:** The tutor provides a negative assessment of the student's performance	.40
Q	**Question:** A question which does not fit into any of the above categories	.95
A	**Answer:** An answer to an utterance marked Q	.94
C	**Correction:** Correction of a typo in a previous utterance	.54
S	**Statement:** A statement which does not fit into any of the above categories	.71
O	**Other:** Other utterances, usually containing only affective content	.69

Table 1. Dialogue act annotation scheme and inter-rater agreement

classifying user dialogue acts for task-oriented dialogue with parallel dialogue and task streams (Ha, 2012). Maximum entropy outperformed both Naive Bayes and conditional random fields. Building on these results, we employ a maximum entropy classifier to learn dialogue management models that predict both the timing and the type of the system dialogue move. The following sections describe two alternate approaches to dialogue management that can both determine the timing and determine the type of system dialogue interventions.

4.1 One-step dialogue management model

In the first model, the two dialogue management tasks are framed as a single classification problem by treating the decision of *not to make a tutorial dialogue move* as a special dialogue act. Thus, a finite set of dialogue moves allowed for the system is defined as $M = \{m_1, m_2, \cdots, m_n\}$, in which $M = DA \cup \{NoMove\}$ and $DA = \{da_1, da_2, \cdots, da_t\}$ is the set of dialogue acts available for the system. Given M and the i^{th} step in a given user interaction history $H_i^{i-k} = h_{i-k}, h_{i-k+1}, \cdots, h_i$, the goal of the dialogue management model is to predict system's dialogue move m_{i+1} for the next step, which is determined by the following equation.

$$m_{i+1} = argmax_{m \in M} P\left(m \middle| H_i^{i-k}\right) \qquad (1)$$

The task-oriented dialogue considered in this work includes two parallel and interleaved data streams: an explicit dialogue stream, consisting of textual exchanges between a student and a tutor, and an implicit task stream, consisting of the student's problem-solving activities. Thus, a given interaction history can be decomposed into a dialogue history and a task history, rewriting equation (1) as follows,

$$m_{i+1} = argmax_{m \in M} P\left(m \middle| D_i^{i-k}, T_i^{i-k}\right) \qquad (2)$$

in which $D_i^{i-k} = d_{i-k}, d_{i-k+1}, \cdots, d_i$ and $T_i^{i-k} = t_{i-k}, t_{i-k+1}, \cdots, t_i$ represent the history of dialogue utterances and the history of student task activities, respectively.

In this work, the conditional probability distribution in Equation (2) is estimated using the maximum entropy framework (Berger et al., 1996). The maximum entropy framework selects a probability distribution that results in the highest entropy among all possible solutions. Given a vector π of feature set, the conditional probability distribution is estimated by the following equation,

$$P(X = m_i | \pi) = \frac{1}{Z(\pi)} e^{\lambda m_i \cdot \pi} \qquad (3)$$

in which λ represents weights and Z is a normalizing factor. This work used MALLET

(McCallum, 2002) to estimate this conditional distribution.

4.2 Two-step dialogue management model

A potential shortcoming of the one-step model is that the probability distribution over dialogue acts is prone to distortion depending on the portion of *NoMove* in the training data. To avoid this, the second model takes a two-step approach, treating each dialogue management task as an independent classification task. The two-step model first determines whether or not to make a dialogue move. If a decision is made to provide a dialogue move, the second classifier is called for a selection of the type of dialogue move.

In this model, system's dialogue move m_{i+1} for the next interaction step is determined by a function $f\left(H_i^{i-k}\right)$, such that

$$f\left(H_i^{i-k}\right) = NoMove, \quad \text{when } P\left(NoMove \middle| H_i^{i-k}\right) > P\left(Move \middle| H_i^{i-k}\right) \tag{4}$$

$$f\left(H_i^{i-k}\right) = argmax_{da \in DA} P\left(da \middle| H_i^{i-k}\right) \quad \text{otherwise.} \tag{5}$$

Similar to the one-step model, Equation (5) can be written as

$$f\left(H_i^{i-k}\right) = argmax_{da \in DA} P\left(da \middle| D_i^{i-k}, T_i^{i-k}\right) \tag{6}$$

This conditional probability distribution is also estimated by the maximum entropy framework.

5 Features

To learn high-performing dialogue management models for task-oriented dialogue with parallel dialogue and task streams, it is crucial to have an effective representation of user interaction state that captures information from all available data streams. The dialogue management models described in the previous section determine the system's next dialogue move based on user interaction state specified by the features extracted from the dialogue and the task streams. In contrast to previous work on task-oriented dialogue, in which task information is incorporated into dialogue utterances by manual tagging (Bangalore et al., 2008; Boyer et al., 2010), our work does not require manual effort to obtain the relevant task information. Instead, we rely on task context logs generated during students' interactions with the tutoring interface, as well as a notion of students' task progress automatically estimated by a task analysis algorithm. The same set of features is used for the prediction of both the timing and the type of system move.

5.1 Automatic task analysis

In order to provide a measure of students' task progress through each of the tutoring sessions, an edit distance metric was implemented. This metric computes the minimum edit distance between a student's program at a particular time t and a representative solution for a given programming task, in order to estimate how far away the student is from completing the task. Because our tutors were experienced in the subject matter and were familiar with the lesson structures, we can safely assume that they knew what this final state of the code would be and thus had an intuitive metric of student progress, which is analogous to our edit distance metric. As this value changes over a session, one can observe how the student's progress is affected by tutor dialogue acts.

Because a character-based edit distance would not capture the relative functional importance of each part of the student's program, our edit distance metric is based on tokenized versions of the program, as generated by the Java compiler, and the output is the number of tokens that differ from the solution for that task. In this way, comments, variable names, or string literals with many characters are all treated as single tokens and do not artificially inflate the edit distance. This tokenization also allows for abstraction of these comments, variable names, and string literals into generalized tokens so that they can be more easily compared between students.

5.2 Dialogue features

Previous work on dialogue act classification has shown that lexical features extracted from dialogue utterances are good predictors of dialogue acts (Bangalore et al., 2008; Boyer et al., 2010a; Kim et al., 2010). However, this finding does not apply when the goal of dialogue act classification is to learn dialogue management models because determining system moves precedes system utterance generation. Instead, this work exploits features that represent local interaction structure within dialogue streams, which includes *current student dialogue act*, *current tutor dialogue act*, *previous tutor dialogue act*, and *tutor utterance count.*

- **Current student dialogue act** represents the interpreted dialogue act for the previous user dialogue input. Student dialogue act interpretation is not addressed in this paper, assuming the existence of an external module that carries out user dialogue interpretation (e.g., Ha et al., 2012).

- **Current tutor dialogue act** represents the type of system dialogue act at the current interaction step. In our tutorial dialogue corpus, we observed tutors often made several dialogue utterances in succession, such as a feedback (*"Great Job."*) followed by a question (*"Do you have any questions?"*). Thus, the value of the current tutor dialogue act impacts the probability distribution over the tutor's next dialogue move. This feature captures such temporal patterns present in tutor dialogue moves as observed in the corpus.
- **Previous tutor dialogue act** represents the type of system dialogue act generated for the previous interaction step. This is similar to the *current tutor dialogue act* feature, but models longer temporal patterns by extending the size of interaction history.
- **Tutor utterance count** represents the number of system dialogue acts generated in succession without interruption until the current interaction step. In our corpus, it was observed that the tutor dialogue turns often consist of multiple utterances. This feature is included to model system dialogue turns consisting of multiple utterances.

5.3 Task features

To create a rich representation of task context, a number of features were automatically extracted from task streams. Three groups of task information were considered, including types of task activity taken by user, the amount of time taken between certain activities, and the user's task progress estimated by the task analysis algorithm (Section 5.1). Alternate representations of these features were empirically compared, resulting in the following task features incorporated in current dialogue management models.

- **Current log type** represents the type of activity taken at the current interaction step either by the user or the system. A complete list of log types is shown in Appendix B.
- **Previous log type** represents the type of activity taken at the previous interaction step. Analogous to *previous tutor dialogue act* in dialogue stream, this feature models temporal patterns among task activities.
- **Same log type** is a binary feature indicating the type of activities at the current and previous interaction step is identical.
- **Previous and current log type** is a feature that combines the current and previous log types (i.e., a bigram of log types).
- **Elapsed time** is the amount of time since the last logged activity, which represents the duration of the user's inactivity. This feature is included to enable the learned dialogue management model to make spontaneous dialogue interventions when a user has been detected to be inactive for an excessive period of time.
- **Elapsed coding time** specifies the amount of time the user has taken since the beginning of current coding task.

6 Evaluation

The dialogue act models were trained and tested using the manually annotated portion of the task-oriented tutorial dialogue corpus described in Section 3. The textual dialogue exchanges in the corpus were aligned with the logged task-completion activities based on the timestamp, resulting in 6919 total interaction logs. Table 2 shows the distribution of different types of activities in the resulting interaction logs. It was observed that tutors made a dialogue move in 26.5% of the total logged interactions (Table 3).

Interaction Type	Frequency (%)
Programming	38.2
Compiling the Program	10.8
Running the Program	12.2
Progressing to Next Task	4.2
Exchanging Dialogue	34.6

Table 2. Distribution of interaction types

Tutor Dialogue Move	Frequency (%)
Move	26.5
NoMove	73.5

Table 3. Distribution of system dialogue move

Among the thirteen dialogue acts in the original annotation scheme (Section 3.2), four rarely occurring dialogue acts were combined into other categories, which include LF (*lukewarm feedback*) merged with NF (*negative feedback*) and RC (*request for confirmation*), RF (*request for feedback*), and C (*correction*) merged to O (*other*). A new category, GREET (*greetings*) was added to distinguish conventional expressions for greetings and thanks from more general statements and questions. Table 4 shows the resulting distribution of tutor dialogue acts in the corpus.

Dialogue Act	Frequency (%)
S (Statement)	35.4
PF (Positive Feedback)	19.8
Q (Question)	16.0
H (Hint)	8.0
DIR (Directive)	6.6
A (Answer)	5.7
GREET (Greetings)	3.1
ACK (Acknowledgement)	2.3
NF (Negative Feedback)	1.5
O (Other)	1.6

Table 4. Distribution of tutor dialogue acts

The performance of the dialogue act models were evaluated in a ten-fold cross validation. In the cross validation, the corpus was partitioned to ten non-overlapping sets and each set was used as testing data exactly once, while models were trained using the remaining nine sets.

6.1 Results

The first study compared the accuracies of the dialogue management models on predicting the timing and the type of tutor dialogue moves. The accuracy of the timing prediction was calculated for all user interaction logs in the data, including both dialogue exchanges and task-completion activities. The accuracy of the type prediction was calculated for dialogue activities by tutors only. The results are shown in Table 5.

Model	Timing	Type
Baseline	73.5	35.4
One-step	79.2[*]	40.5[*]
Two-step	**80.3**[*§]	**49.7**[*§]

Table 5. Model accuracy (%) on dialogue management tasks ([*]statistical significance over baseline, [§]statistical significance over one-step model)

Both the one-step ($t(9) = 4.14$, $p = 0.0013$) and the two-step ($t(9) = 6.26$, $p < .0001$) models performed significantly better than the majority baseline in predicting the timing of tutorial dialogue moves. The two-step model achieved higher accuracy than the one-step model. The difference between the two models was statistically significant ($t(9) = 2.17$, $p = 0.0291$).

The one-step ($t(9) = 2.68$, $p = 0.0126$) and the two-step ($t(9) = 10.93$, $p < 0.0001$) models achieved significantly higher accuracies over the baseline for the task of predicting the type of tutorial dialogue moves, as well. Again, the two-step model performed significantly better than the one-step model ($t(9) = 4.22$, $p = .0011$).

6.2 Comparing dialogue and task streams

The second study compared the predictive power of the dialogue stream and the task stream on the given two dialogue management tasks. In this study, the accuracy of the two-step model was compared in three conditions: using the dialogue features only (*Dialogue*), using the task features only (*Task*), and using all features (*All*). Table 6 reports the results.

Features	Timing	Type
Dialogue	79.6	45.0
Task	80.1	44.9
All	**80.3**[*]	**49.7**[*§]

Table 6. Comparison of features on dialogue management tasks ([*]statistical significance over *Dialogue*, [§]statistical significance over *Task*)

For determining when to intervene, the dialogue and the task features exhibited similar predictive power. No statistical significance was found for the difference between the dialogue and the task conditions. The highest accuracy was achieved by the *All* condition. Compared to the *All* condition, the *Dialogue* condition showed statistically significant decrease in accuracy ($t(9) = 2.21$, $p = 0.0272$), which implies the task stream provided important features for the dialogue management model in determining the timing of tutorial dialogue moves.

A similar trend was observed for determining what type of dialogue move to make. The *Dialogue* and the *Task* conditions achieved similar accuracies, with the highest accuracy achieved by the *All* condition. The drops in accuracy compared to the *All* condition were statistically significant for both the *Dialogue* ($t(9) = 3.38$, $p = 0.0040$) and the *Task* conditions. ($t(9) = 4.36$, $p = 0.0009$). The results imply that the prediction of the type of tutorial dialogue moves required information from both the dialogue and the task streams.

7 Discussion

The experiments presented in Section 6 compared two alternate approaches to learning dialogue management models for two given subtasks: determining when to provide the user with a dialogue move, and determining which type of dialogue move to choose. The results suggest that the two-step approach, which models the two subtasks as separate classifiers, was more effective than the alternate one-step approach, which combined the two subtasks into a single classification problem. The two-step model achieved higher performance than the one-step model in both the timing and the type prediction. However, the difference in the performance of the two models was more apparent in the type prediction, with the two-step model achieving over 22% higher accuracy than the one-step model. One possible explanation for the superiority of the two step-model over the one-step model is that the corpus used to train the models was highly skewed. For more than 73% of the total interaction logs in the corpus, the tutors did not provide any dialogue feedback. Since the one-step model treated *NoMove* as a special dialogue act, the skewed distribution over *NoMove* and *Move* impacted the learned distribution over dialogue acts.

Two previous investigations reported the accuracies of dialogue act classification on system utterances. Bangalore et al. (2008) reported a prediction accuracy of 55% for system dialogue acts when a flat task model was used in a catalogue-ordering domain. When a hierarchical task structure was used in the same domain, the achieved prediction accuracy for system dialogue acts was 35.6% (Bangalore and Stent, 2009). Boyer (2010) achieved accuracy of 57% for system dialogue acts in a task-oriented tutorial dialogue. While both of these lines of investigation employed task structure features that were manually annotated, our best-performing two-step dialogue management model resulted in comparable performance utilizing only automatic features, achieving an accuracy of 49.7%.

A crucial distinction between user and system dialogue act classification is that lexical features for a given dialogue turn are not available for system dialogue act classification because a system utterance is generated after a system dialogue act is selected. The absence of lexical features poses a significant challenge to system dialogue act classification, given that lexical features have been among the most predictive features for this task. To address this challenge, future research should continue exploring larger spaces of features to improve prediction accuracies of learned models.

8 Conclusions and Future Work

Automatically learning dialogue management models for complex task-oriented domains with separate dialogue and task streams poses significant challenges. Effective dialogue management models in such domains should be able to proactively intervene by making spontaneous dialogue moves based on the observed history of both the dialogue and the user's task activities. With the overarching goal of creating a data-driven automated dialogue system that incorporates parallel dialogue and task streams, this paper has presented classification-based dialogue management models that integrate a rich set of features automatically extracted from parallel dialogue and task streams. Two subtasks of dialogue management were considered: *when* the system should provide user with a dialogue move and what *type* of system dialogue act the system should select for a given user interaction context. An evaluation found that a two-step approach that modeled the two subtasks as separate classifiers were effective, achieving significantly higher performance than an alternate approach that modeled the two subtasks with a single classifier.

The results suggest several promising directions for future work. First, incorporating richer features may improve the accuracies of learned models, such as more global interaction histories and deeper dialogue structures. Second, developing more sophisticated task analyses will inform the learned models with a representation of the user task context, guiding the models to make more context-appropriate decisions. Finally, it will be important to evaluate the learned models by incorporating them into a dialogue management system and validating their effectiveness in interactions with users in rich task-oriented dialogue.

Acknowledgments

This research was supported by the National Science Foundation under Grant DRL-1007962. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Allen, J., Ferguson, G., & Stent, A. (2001). An architecture for more realistic conversational systems. *Proceedings of Intelligent User Interfaces* (pp. 1–8). Santa Fe, NM.

Bangalore, S., Di Fabbrizio, G., & Stent, A. (2008). Learning the structure of task-driven human-human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(7), 1249–1259.

Bangalore, S., & Stent, A. J. (2009). Incremental parsing models for dialog task structure. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 94–102). Athens, Greece.

Berger, A. L., Della Pietra, V. J., & Della Pietra, S. A. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

Boyer, K. E. (2010). Structural and Dialogue Act Modeling in Task-Oriented Tutorial Dialogue. Ph.D. Dissertation. Department of Computer Science, North Carolina State University.

Boyer, K. E., Grafsgaard, J. F., Ha, E. Y., Phillips, R., & Lester, J. C. (2011). An affect-enriched dialogue act classification model for task-oriented dialogue. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 1190–1199). Portland, OR.

Boyer, K. E., Ha, E. Y., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2010). Dialogue Act Modeling in a Complex Task-Oriented Domain. *Proceedings of the 11th Annual SIGDIAL Meeting on Discourse and Dialogue* (pp. 297–305). Tokyo, Japan.

Carberry, S. (1991). *Plan Recognition in Natural Language Dialogue.* MIT Press.

Cavicchio, F. (2009). The modulation of cooperation and emotion in dialogue: The REC corpus. *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop* (pp. 81–87). Suntec, Singapore.

Chi, M., VanLehn, K., Litman, D., & Jordan, P. (2010). Inducing Effective Pedagogical Strategies Using Learning Context Features. *Proceedings of the Eighteenth International Conference on User Modeling, Adaptation, and Personalization* (pp 147-158). Big Island, HI.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37 – 46.

Di Eugenio, B., Xie, Z., & Serafin, R. (2010). Dialogue act classification, instance-based learning, and higher order dialogue structure. *Dialogue and Discourse*, *1*(2), 81 – 104.

Dzikovska, M.O., Farrow, E, & Moore, J.D. (2013). Combining deep parsing and classification for improved explanation processing in a tutorial dialogue system. *Proceedings of the 16th International Conference on Artificial Intelligence in Education* (pp. 279 - 288). Memphis, TN.

Forbes-Riley, K. & Litman, D. (2011). Designing and evaluating a wizarded uncertainty-adaptive spoken dialogue tutoring system. *Computer Speech and Language*, 25(1), 105-126.

Ha, E. Y., Grafsgaard, J. F., & Mitchell, C. M. (2012). Combining Verbal and Nonverbal Features to Overcome the "Information Gap" in Task-Oriented Dialogue. *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 247–256). Seoul, South Korea.

Hardy, H., Biermann, A., Inouye, R. B., McKenzie, A., Strzalkowski, T., Ursu, C., Webb, N., et al. (2006). The Amitiés system: Data-driven techniques for automated dialogue. *Speech Communication*, *48*(3-4), 354–373.

Henderson, J., Lemon, O., & Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, *34*(4), 487–511.

Ivanovic, E. (2008). *Automatic instant messaging dialogue using statistical models and dialogue acts*. The University of Melbourne.

Kim, S. N., Cavedon, L., & Baldwin, T. (2010). Classifying dialogue acts in one-on-one live chats. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 862–871). Cambridge, MA, USA: Association for Computational Linguistics.

Levin, E., Pieraccini, R., & Eckert, W. (2000). A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Transactions on Speech and Audio Processing*, *8*(1), 11–23.

Lewis, C., & Di Fabbrizio, G. (2006). Prompt selection with reinforcement learning in an AT&T call routing application. *Proceedings of the Ninth International Conference on Spoken Language Processing* (pp. 96–103).

Mitchell, C.M., Boyer, K.E., & Lester, J.C. (2013). A Markov Decision Process Model of Tutorial Intervention in Task-Oriented Dialogue. *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 828-831), Memphis, TN.

Mitchell, C. M., Ha, E. Y., Boyer, K. E., & Lester, J. C. (2012). Recognizing effective and student-adaptive tutor moves in task-oriented tutorial dialogue. *Proceedings of the Intelligent Tutoring Systems Track of the 25th International Conference of the Florida Artificial Intelligence Research Society* (pp. 450–455).

Rich, C., & Sidner, C. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Inter-action*, 8(3-4), 315–350.

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics* (pp. 93–100). Wanchai, Hong Kong.

Scheffler, K., & Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning.

Proceedings of the second international conference on Human Language Technology Research (pp. 12–19). San Diego, CA.

Singh, S., Litman, D. J., Kearns, M., & Walker, M. (2002). Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, *16*, 105–133.

Sridhar, R., Bangalore, S., & Narayanan, S. (2009). Combining lexical, syntactic and prosodic cues for improved online dialog act tagging. *Computer Speech and Language*, *23*(4), 407 – 422.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., et al. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, *26*(3), 339–373.

Walker, M., Litman, D., Kamm, C., & Abella, A. (1997). Paradise: A framework for evaluating spoken dialogue agents. *Proceedings of ACL* (pp. 271–280). Madrid, Spain.

Williams, J., & Young, S. (2007). Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, *21*(2), 393–422.

Young, S. (2002). Talking to machines (statistically speaking). *Proceedings of ICSLP* (pp. 32–41). Denver, CO.

Young, S., Gasic, M., Thomson, B., & Williams, J. (2013). POMDP-Based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, *101*(5), 1160–1179.

Appendix A. An Excerpt from the Task-Oriented Dialogue Corpus

Lesson ID	Task ID	Role	Type	Text	Timestamp
1	4	STUDENT	CODING	`System.out.printIn("Hello World"`	2011-09-21 08:17:17.737
1	4	STUDENT	CODING	`System.out.printIn("Hello World")`	2011-09-21 08:17:19.407
1	4	STUDENT	CODING	`System.out.printIn("Hello World");`	2011-09-21 08:17:19.812
1	4	TUTOR	MESSAGE	good.	2011-09-21 08:17:24.913
1	4	TUTOR	MESSAGE	also you can try to compile at anytime.	2011-09-21 08:17:33.805
1	4	STUDENT	COMPILE_ BEGIN	studentCode\jt101\JavaTutor3.java	2011-09-21 08:17:38.080
1	4	STUDENT	COMPILE_ ERROR	line 1 : cannot find symbol symbol : method printIn(java.lang.String) location: class java.io.PrintStream System.out.printIn("Hello World"); ^ 1 error	2011-09-21 08:17:38.220
1	4	TUTOR	MESSAGE	carefully compare your line with the example	2011-09-21 08:17:57.330

Appendix B. Types of Activity Logs in Corpus

Log Type	Description	Action Initiator
MESSAGE	Either student or tutor has sent a chat message.	Student, Tutor
SESSION_PROGRESS	Tutor has allowed student to progress to next task.	Tutor
CODING	Student has written programming code.	Student
COMPILE_BEGIN	Student has begun compiling code.	Student
COMPILE_SUCCESS	Recent code compilation has ended successfully.	N/A
COMPILE_ERROR	Recent code compilation has failed with errors.	N/A
RUN_BEGIN	Student has begun running code.	Student
INPUT_SENT	Student has sent an input to a running code.	Student
RUN_SUCCESS	Recent code running has ended successfully.	N/A
RUN_STOP	Tutor has stopped running student's code because of errors in the code.	Tutor