The Impact of Task-Oriented Feature Sets on HMMs for Dialogue Modeling

Kristy Elizabeth Boyer **Eun Young Ha** **Robert Phillips*** **James Lester**

Department of Computer Science
North Carolina State University
*Dual affiliation with Applied Research Associates, Inc.
Raleigh, North Carolina, USA
{keboyer, eha, rphilli, lester}@ncsu.edu

Abstract

Human dialogue serves as a valuable model for learning the behavior of dialogue systems. Hidden Markov models' sequential structure is well suited to modeling human dialogue, and their theoretical underpinnings are consistent with the conception of dialogue as a stochastic process with a layer of implicit, highly influential structure. HMMs have been shown to be effective for a variety of descriptive and predictive dialogue tasks. For task-oriented dialogue, understanding the learning behavior of HMMs is an important step toward building unsupervised models of human dialogue. This paper examines the behavior of HMMs under six experimental conditions including different task-oriented feature sets and preprocessing approaches. The findings highlight the importance of providing HMM learning algorithms with rich task-based information. Additionally, the results suggest how specific metrics should be used depending on whether the models will be employed primarily in a descriptive or predictive manner.

1 Introduction

Human dialogue serves as a valuable model for learning the behavior of dialogue systems. For this reason, corpus-based approaches to dialogue management tasks have been an increasingly active area of research (Bangalore, Di Fabbrizio, & Stent, 2006; Di Eugenio, Xie, & Serafin, 2010; Georgila, Lemon, Henderson, & Moore, 2009; Rotaru & Litman, 2009). Modeling the dialogue policies that humans employ permits us to directly extract conversational and task-based expertise. These techniques hold great promise for scaling gracefully to large corpora, and for transferring well across domains.

The richness and flexibility of human dialogue introduce nondeterministic and complex patterns that present challenges for machine learning approaches. One approach that has been successfully employed in dialogue modeling is the hidden Markov model (HMM) (Rabiner, 1989). These models are well suited to the sequential nature of dialogue (Stolcke et al., 2000). Moreover, their theoretical underpinnings are consistent with the conception of dialogue as a stochastic process whose observations are influenced by a layer of implicit, yet highly relevant, structure (Boyer et al., 2009; Woszczyna & Waibel, 1994).

HMMs have been shown to perform well on important dialogue management tasks such as automatic dialogue act classification (Stolcke et al., 2000). Our work has employed HMMs for a different goal: learning dialogue policies, or strategies, from corpora (Boyer, Phillips, et al., 2010; Boyer, Phillips, Ingram, et al., in press). This work can be viewed from two perspectives. First, a *descriptive* goal of the work is to learn models that describe the nature of human dialogues in succinct probabilistic terms, in a way that facilitates important qualitative investigations. The second and complementary goal is *predictive*: learning models that accurately predict the dialogue moves of humans, in order to capture a dialogue policy that can be used within a system.

Proceedings of the SIGDIAL 2011: the 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 49–58,
Portland, Oregon, June 17-18, 2011.

Both of these goals are of paramount importance in *tutorial dialogue*, in which tutors and students engage in dialogue in support of a learning task (Boyer, Ha, et al., 2010; VanLehn et al., 2007). Descriptive modeling represents a critical step toward more fully understanding the phenomena that contribute to the high effectiveness of human tutoring, which has to date been unmatched by tutorial dialogue systems. Predictive models, on the other hand, may be used directly as dialogue policies within systems.

The HMMs considered here were learned from an annotated corpus of textual human-human tutorial dialogue. In this domain, HMMs have been shown to correspond qualitatively to widely held conceptions of tutorial dialogue strategies, and adjacency pair analysis before model learning has been shown to enhance this qualitative correspondence (Boyer et al., 2009). Moreover, HMMs can identify in an unsupervised fashion structural components that correlate with student knowledge gain (Boyer, Phillips, Ingram, et al., in press).

However, to date, several important questions have not been explored. The answers to these questions have implications for learning HMMs for task-oriented dialogues. The questions include the following: 1) How reliably does the HMM learning framework converge to the hyperparameter *N*, the best-fit number of hidden states? 2) What are the effects of preprocessing approaches, specifically, adjacency pair analysis, on the resulting HMMs? 3) How do different feature sets for task-oriented dialogue impact the descriptive fit and predictive power of learned HMMs? This paper addresses these questions. The findings suggest that model stability and predictive power benefit from the richest possible input sequences, which include not only dialogue acts but also information about the task state and the absence of particular tutor dialogue moves. Additionally, we find that traditional measures of HMM goodness-of-fit may not identify the most highly predictive models under some conditions.

2 Background

HMMs have been used for dialogue modeling tasks for many years. Early work utilized HMMs to model underlying linguistic structure for the purposes of identifying speech acts and reducing perplexity for speech recognition (Stolcke et al., 2000; Woszczyna & Waibel, 1994). These projects treated underlying dialogue structure as the hidden layer, and dialogue utterances as observations. This treatment is analogous to the work presented in this paper, except that our observations are dialogue act tags only, rather than being constituent words in each utterance. Our goals are also different: to create a qualitatively interpretable model of dialogue structure that corresponds to widely accepted notions of task-oriented dialogue, and to learn a highly predictive dialogue policy from a human-human dialogue corpus.

HMMs rely on treating dialogue as a sequential Markov process in which each observation depends only on a finite set of preceding observations. Some other approaches that rely on this assumption treat dialogue as a Markov decision process or partially observable Markov decision process, in which state changes are associated with actions and rewards (e.g., Young et al., 2010). Such work focuses on learning an optimal policy, typically utilizing a combination of human and simulated dialogue corpora. Reinforcement learning techniques can then be applied to learn the optimal policy based on the observed rewards. In contrast, we start with a rich corpus of human-human dialogue, which may have poor coverage in some areas (though the dialogue act tags were empirically derived and therefore mitigate this problem to some extent), and subsequently learn a model that explains the variance in that human corpus as well as possible.

Capturing the dialogue policy implicit within a corpus of human-human dialogue has been explored in other work in a catalogue-ordering domain (Bangalore, Di Fabbrizio, & Stent, 2006). That work utilized maximum entropy modeling to predict human agents' dialogue moves within a vector-based framework. Although a vector-based approach differs in many regards from the sequential HMM approach described here, both approaches assume a dependence only on a finite history. HMMs accomplish this through graphical dependencies, while vector-based approaches accomplish it by including features for a restricted window of left-hand context. The results of this catalogue-ordering project highlight how challenging it is to predict human agents' dialogue moves in a task-oriented domain.

3 Corpus

The corpus was collected during a human-human tutoring study. Students solved an introductory computer programming problem in the Java programming language. Tutors were located in a separate room and communicated with students through textual dialogue while viewing a synchronized view of the student's problem-solving workspace. Forty-eight students interacted for approximately one hour each with a tutor. Students exhibited statistically significant learning gains from pretest to posttest, indicating that the tutoring was effective (Boyer, Phillips, Ingram, et al., in press). The corpus contains 1,468 student moves and 3,338 tutor moves. Overlapping utterances, which are common in dialogue platforms such as instant messaging, were prevented by permitting only one user to construct a dialogue message at a time. Because the corpus is textual, utterances were segmented at textual message boundaries except when the lead dialogue annotator noted the presence of two separate dialogue acts within non-overlapping chunks of text. In these events the utterance was segmented by the primary annotator prior to being tagged by the second dialogue act annotator.

In addition to dialogue act annotation, the corpus was manually annotated for task structure and correctness (Section 3.2), and for delayed tutor feedback (Seciton 3.3). The appendix displays an excerpt from the annotated corpus.

3.1 Dialogue Act Annotation

As part of prior work, the corpus was annotated with dialogue acts for both tutor (Boyer, Phillips, Ingram, et al., in press) and student (Boyer, Ha, et al., 2010) utterances (Table 1). One annotator tagged the entire corpus, while a second annotator independently tagged a randomly selected 10% of tutoring sessions. The inter-annotator agreement Kappa score was 0.80.

3.2 Task Annotation

The corpus includes 97,509 keystroke-level task events (computer programming actions), all taken by the student. Tutors viewed synchronously, but could not edit, the computer program. The task actions were manually clustered and labeled for subtask structure (Boyer, Phillips, et al., 2010). The task structure annotation was hierarchical, with leaves corresponding to specific subtasks such as creating a temporary variable in order to swap two variables' values (subtask *3-c-iii-2*). Each problem-solving cluster, or subtask, was then labeled for correctness (Table 2). These correctness labels are utilized in the models presented in this paper. The Kappa agreement statistic for the correctness annotation on 20% of the corpus was 0.80.

Table 1. Dialogue act tags

Dialogue Act	Tutor Example
ASSESSING Q.	*Which type should that be?*
EXTRA-DOMAIN	*A coordinator will be there soon.*
GROUNDING	*Ok.*
LUKEWARM FDBK	*That's close.*
LUKEWARM CONTENT FDBK	*Almost there, but the second parameter isn't quite right.*
NEGATIVE FDBK	*That's not right.*
NEGATIVE CONTENT FDBK	*No, the counter has to be an int.*
POSITIVE FDBK	*Perfect.*
POSITIVE CONTENT FDBK	*Right, the array is a local variable.*
QUESTION	*Which approach do you prefer?*
RESPONSE	*It will be an int.*
STATEMENT	*They start at 0.*

Table 2. Task correctness tags

Correctness Tag	Description
CORRECT	*Fully conforming to the requirements of the task.*
BUGGY	*Violating the requirements of the task. These task events typically require tutorial remediation.*
INCOMPLETE	*Not violating, but not yet fulfilling, the requirements of the task.*
DISPREFERRED	*Technically fulfilling requirements but not utilizing the target concepts being tutored. These events typically require tutorial remediation.*

3.3 Annotation for Delayed Tutor Feedback

The dialogue act and task annotations reflect positive evidence regarding what *did* occur in the dialogues. An additional annotation was introduced for what *did not* occur—specifically, instances in which tutors did not to make a dialogue move in response to students' relevant task actions. The task in our corpus is computer programming, so bugs in the task correspond to errors either in syntax or se-

mantics of the computer program compared to the desired outcome. The human tutors were working with only one student at a time and were carefully monitoring student task actions during the dialogue, so we take the absence of a dialogue move at a relevant point to be an intentional choice by the tutor to delay feedback as part of the tutorial strategy. The automatic annotation for delayed feedback introduced two new event tags: NO-MENTION of correctly completed subtasks, and NO-REMEDIATION of existing bugs within the task.

The intuition behind these tags is that within a learned dialogue policy, specifically modeling when *not* to intervene is crucial. Typically human tutors mention correctly completed subtasks, but at times other tutorial goals eclipse the importance of doing so. The NO-MENTION tag captures these instances. On the other hand, typically when working with novices, human tutors remediate an existing bug quickly. However, tutors may choose to delay this remediation for a variety of reasons such as remediating a different bug instead or asking a conceptual question to encourage the student to reflect on the issue. The NO-REMEDIATION tag captures these instances of the absence of remediation given that a bug was present. These two annotations for delayed feedback were performed automatically (Boyer, Phillips, Ha, et al., in press).

3.4 Adjacency Pair Modeling

Prior work has demonstrated that adjacency pairs can be identified in an unsupervised fashion from a corpus (Midgley, Harrison, & MacNish, 2006). This technique relies on statistical analysis to determine the significant dependencies that exist between pairs of dialogue acts, or in our task-oriented corpus, pairs of dialogue acts or task actions. After the pairs of dependent events are identified, they are joined within the corpus algorithmically (Boyer et al., 2009). Joining a pair of dependent moves in this way is equivalent to introducing a deterministic (probability=1) succession between observation symbols. This type of dependency cannot be learned in the traditional first-order HMM framework, but is desirable when two observations are strongly linked.[1]

[1] Enhanced HMM structures, such as autoregressive HMMs, which allow for direct graphical links between observation symbols, can learn such a dependency but only in stochastic terms.

The experiment that is described in Section 4 utilizes different feature sets to learn and compare HMMs. Table 3 shows these feature sets and their most highly statistically significant adjacency pairs.

Table 3. Experimental conditions and top three adjacency pairs (subscripts denote speaker, **S**tudent or **T**utor)

Condition	Description	Significant Adjacency Pairs
DAONLY	*Dialogue acts only*	Q_S~Rsp_T $Ground_S$~$Ground_T$ $AssessQ_T$~$PosFdbk_S$
DATASK	*Dialogue acts & task correctness events*	Q_S~Rsp_T $CorrectTask_S$~$CorrectTask_S$ $Ground_S$~$Ground_T$
DATASK-DELAY	*Dialogue acts, task correctness, & delayed feedback*	Q_S~Rsp_T $NoRemediate_T$~$BuggyTask_S$ $CorrectTask_S$~$CorrectTask_S$

4 Models

HMMs were selected as the modeling framework for this work because their sequential nature is well suited to the structure of human dialogue, and their "hidden" variable corresponds to widely held conceptions of dialogue as having an unobservable, but influential, layer of stochastic structure. For example, in tutoring, an "explanation" mode is common, in which the tutor presents new information and the student provides acknowledgments or takes task actions accordingly. Although the presence of the "explanation" goal is not directly observable in most dialogues, it may be inferred from the observations. These sequences correspond to the input observations for learning an HMM.

4.1 Hidden Markov Models

HMMs explicitly model hidden states within a doubly stochastic structure (Rabiner, 1989). A first-order HMM, in which each hidden state depends only on the immediately preceding hidden state, is defined by the following components:

- $\sum = \{\sigma_1, \sigma_2, \ldots, \sigma_M\}$, the observation symbol alphabet
- $S = \{s_1, s_2, \ldots, s_N\}$, the set of hidden states

- $\Pi=[\pi_i]$, $i=1,...,N$, the initial probability distribution, where π_i is the probability of the model beginning in hidden state s_i in S
- $A=[a_{ij}]$, a transition probability distribution, where a_{ij} is the probability of the model transitioning from hidden state i to hidden state j for $i,j=1,...,N$
- $B=[b_{ik}]$, an emission probability distribution where b_{ik} is the probability of state i ($i=1,...,N$) emitting (or generating) observation symbol k ($k=1,...,M$).

4.2 Dialogue Modeling with HMMs

In this work, the observation symbol alphabet $\sum$ is given. For each experimental condition, $\sum$ is either 1) all dialogue act tags, 2) all dialogue acts plus task correctness tags, or 3) dialogue act, task correctness, and delayed feedback tags. The transition probability distribution A, emission probability distribution B, and initial probability distribution Π are learned by the standard Baum-Welch algorithm for optimizing HMM parameters (Rabiner, 1989). This algorithm is susceptible to becoming trapped in local optima, so our approach uses ten-time random restart with new initial parameters for each model to reduce the probability of selecting a model that represents only a local optimum.

The hyperparameter N, which is the best number of hidden states, is also learned rather than fixed. This process involves running the full HMM training algorithm, including random restarts in ten-fold cross-validation, across the data and selecting the N that corresponds to the best mean goodness-of-fit measure. For HMMs, a typical goodness-of-fit measure is log-likelihood, which captures how likely the observations would be under the current model. The log is taken for practical reasons, to avoid numerical underflow. Higher log-likelihood corresponds to improved model fit. However, typically it is desirable to penalize a higher number of hidden states, since increasing the model complexity results in tradeoffs that may not be fully warranted by the improvement in model fit. In this work, we utilize the Akaike Information Criterion (AIC), a standard penalized log-likelihood metric (Akaike, 1976).

$$AIC = 2*N - 2*ln(likelihood)$$

Lower values of AIC indicate better model fit.

4.3 Experimental Conditions

HMMs were learned using three separate feature sets, each providing a progressively more complete picture of the task-oriented dialogues: dialogue acts only (DAONLY), dialogue acts and task events (DATASK), and dialogue acts with both task correctness events and tags for delayed tutor feedback (DATASKDELAY).

In addition to the three different feature sets, each condition included one of two types of preprocessing. Each type of model was trained on unaltered sequences of the annotated tags, which we refer to as the UNIGRAM condition. Additionally, each type of model was trained on sequences with statistically dependent adjacency pairs joined in a preprocessing step as described in Section 3.4. The UNIGRAM and ADJPAIR conditions were explored for each of the three feature sets, resulting in six experimental conditions. These conditions were chosen in order to explore the convergence behavior of HMMs under the different feature sets and preprocessing, and to compare measures of descriptive fit with measures of predictive power.

4.4 Learned HMMs

Figures 1 and 2 show a subset of the DAONLY UNIGRAM model and the DATASKDELAY ADJPAIR model. These figures depict the structure of our HMMs: each hidden state is associated with an emission probability distribution over the possible observation symbols.

5 Goodness-of-Fit Curves

The learning algorithm described in Section 4.2 was applied to input sequences under the six experimental conditions to learn the best-fit HMM parameters. Figure 3 displays these AIC results, which are discussed in detail in the remainder of this section.

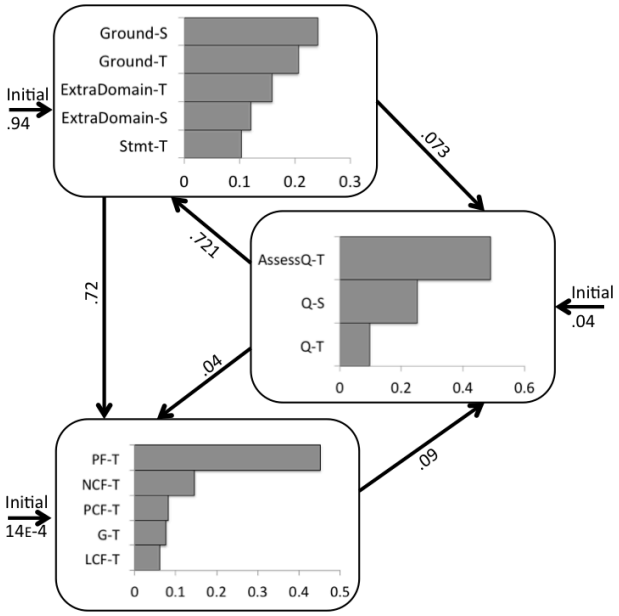

Figure 1. Subset of learned HMM (*N*=13) for DAONLY UNIGRAM condition

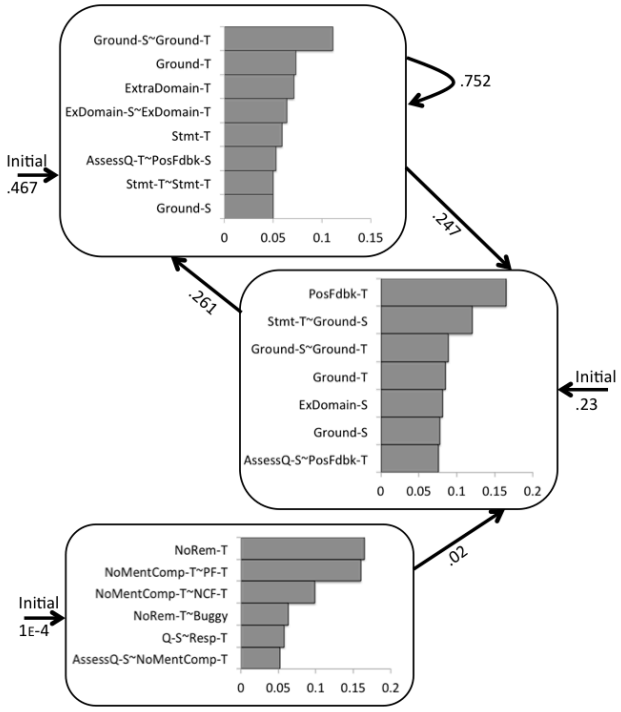

Figure 2. Subset of learned HMM (N=9) for DATASKDELAY ADJPAIR condition

5.1 Impact of Experimental Conditions

For the DAONLY condition, both the UNIGRAM and ADJPAIR models generally improve until *N*=12 or 13, after which the fit generally worsens. A different pattern emerges for the DATASK condition, in which the UNIGRAM sequences are optimally fit to a model with 16 states, while the ADJPAIR sequences are fit to a model with 8 states. Finally, for the DATASKDELAY condition, the UNIGRAM sequences are best fit by a model with 10 hidden states, while the ADJPAIR sequences are fit best by 9. Typically, we see that ADJPAIR sequences are fit to slightly simpler models in terms of the hyperparameter *N*, number of hidden states.

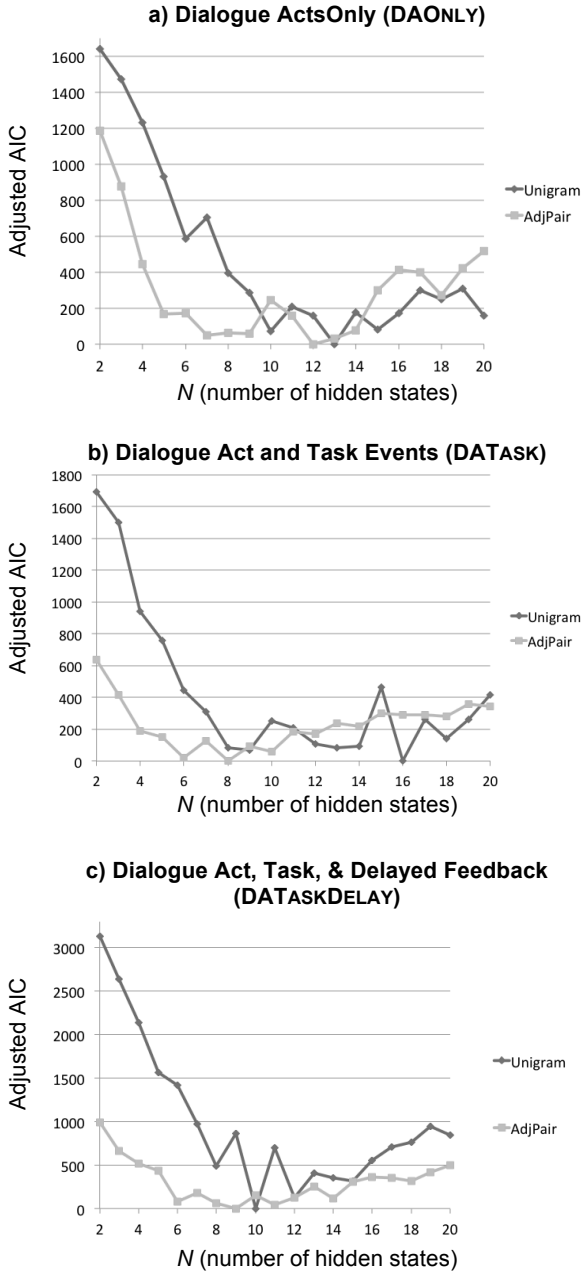

Figure 3. Number of hidden states and corresponding adjusted AIC, shifted to a minimum score of zero indicating the best-fit *N*

Stability in the hyperparameter N is an important consideration because an underlying assumption of our work is that the hidden states correspond to unobserved stochastic structures of the *real world process*—that is, we hypothesize that a "true" value for N exists. We would like models to exhibit decreasing variation in goodness of fit measures around this true N. To examine this stability we consider the three best AIC values for each condition and their corresponding Ns: the set $\{N_{k\text{-}best} \mid k=1,2,3\}$. The range of this set indicates how "far apart" the best three Ns are: for example, in the DAONLY UNIGRAM condition, the top three models have Ns of *{13,10,15}*, yielding a range of *5*. Intuitively, a small value for this metric indicates that the model has converged tightly on N.

Figure 4 shows the stability results for the six different experimental conditions. As shown in the figure, for the DATASK and DATASKDELAY conditions, the ADJPAIR models achieve the smallest range among the top three values of N; these models converge most tightly to the "best" value.

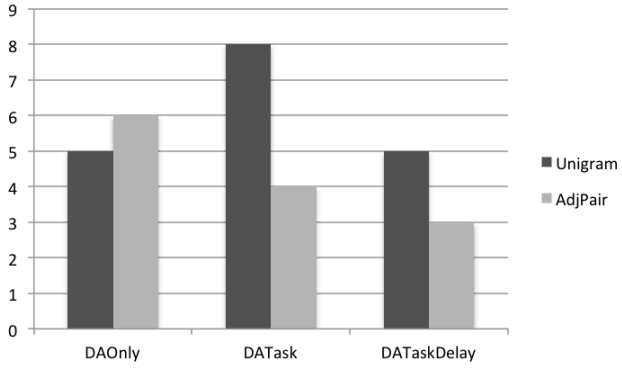

Figure 4. Stability of N (range of $\{N_{1best,} N_{2best,} N_{3best}\}$) – smaller implies tighter convergence to "best" N

6 Predictive Analysis

Section 5 presented an analysis of the goodness-of-fit curves of HMMs learned from the corpus. The measures of stability and discrimination for N capture important aspects of the behavior of HMMs toward this parameter, which is conceived of as representing "true" real-world stochastic behavior. In this way, Section 5 has presented a descriptive view of HMM dialogue models.

This section presents a predictive view of the models. Specifically, we consider *prediction accuracy*, defined as the percent of tutor dialogue moves that the model is able to correctly predict given the dialogue history sequence up to that point.

6.1 Impact of Dependent Adjacency Pairs

We first explore whether the preprocessing step of joining dependent adjacency pairs impacted prediction accuracy. The prediction accuracy of the best-fit model in each condition is displayed in Figure 5. This figure includes prediction accuracy on training data, which were used to learn model parameters, as well as prediction accuracy on testing data, which were withheld from model training.

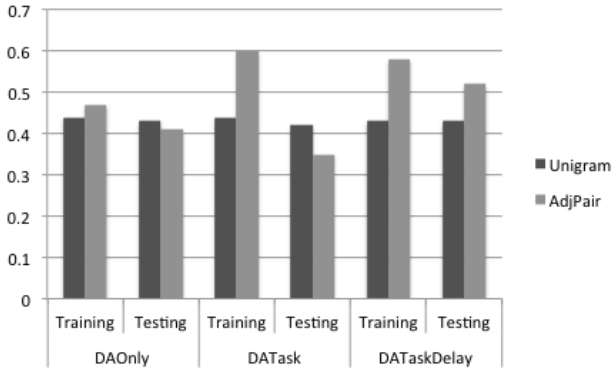

Figure 5. Prediction accuracy for tutor moves

As shown in Figure 5, joining the adjacency pairs improved model performance on the training sets of all three conditions, indicating that the variation within the training data was better explained by ADJPAIR models. (This measure of predictive power is different from a goodness-of-fit criterion as described in the previous section, a relationship that will be discussed further in Section 7.) In contrast to the training set performance, the ADJPAIR models performed better than UNIGRAM models for the testing set only in the DATASKDELAY condition.

6.2 Impact of Task-Oriented Feature Sets

As illustrated in Figure 5, the three feature sets perform similarly under the UNIGRAM condition. This performance is slightly above baseline (DAONLY and DATASK baselines = 0.38; DATASKDELAY baseline = 0.30), and diminishes little between the training and testing sets. In contrast, under the ADJPAIR condition, performance between conditions and across training and testing sets varies. The DATask model performs far better on predicting observations in the training than the testing set,

suggesting possible overfitting to the training set. This relationship is discussed further in Section 7. The DATASKDELAY model performs well during both training and testing, though with a slight decrease in accuracy on the testing set.

6.3 Relationship Between Predictive and Descriptive Metrics

Measures of fit such as log-likelihood and AIC capture the likelihood of observing the data given a model. Predictive accuracy, on the other hand, measures the probability that the model can predict the next observation given a partial sequence. In general, we would expect these measures to correlate well; however, there is not perfect correlation between these metrics because the mechanism by which log-likelihood (and thereby AIC) is derived involves maximizing likelihood over complete sequences, while prediction is performed over partial sequences.

To examine how well AIC and prediction accuracy correlate, Figure 6 displays these values for a subset of the models in the DAONLY UNIGRAM condition and the DATASKDELAY ADJPAIR condition. These two conditions represent the extremes of the experimental conditions, with DAONLY containing the least information about the task-oriented dialogue while DATASKDELAY contains the most information.

As shown in Figure 6, the correlation for DAONLY UNIGRAM roughly conforms to what would be expected: lower AIC, indicating better model fit, is associated with the highest prediction accuracies. The relationship is less clear for the DATASKDELAY ADJPAIR condition. While its worst AIC is associated with the lowest prediction accuracy as expected, the best AIC is not associated with the highest prediction accuracy. This phenomenon may be due to the lack of spread among AIC values overall for this condition; as seen in Figure 3, the DATASKDELAY ADJPAIR condition has the flattest AIC curve of all conditions, indicating that for this condition the difference between best-fit and worst-fit models is smaller than for any other condition. The inconsistent relationship between AIC and prediction accuracy, therefore, may be the product of noise surrounding a large set of "good" models, whereas for the DAONLY UNIGRAM condition, the set of good models is smaller.

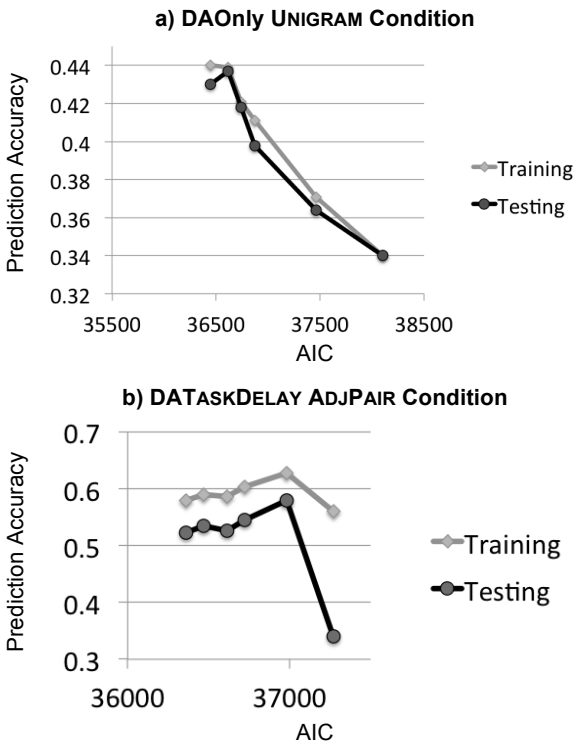

Figure 6. Prediction accuracy vs. AIC

7 Discussion

The results suggest several important findings regarding feature sets and preprocessing for learning HMMs of task-oriented dialogue. First, the models' convergence patterns to a best-fit *N*, number of hidden states, indicate that more information embedded within the sequences may correspond with a flatter goodness-of-fit curve. Adding more information to the input sequences may introduce some regularities that partly mitigate the limitations of even a poorly fit HMM. This additional information may come in the form of adjacency pairs discovered in an unsupervised fashion, which improved the stability of convergence on the best-fit *N* under the DATASK and DATASKDELAY conditions. This increased stability is likely due to the fact that under these conditions, leveraging adjacency pair information augments the HMM's structure with contextual dependencies that could otherwise not be learned under the traditional HMM framework.

For predictive accuracy, the benefits of richer input sequences are also highlighted. The most highly predictive models included all three sources

of information: dialogue acts, task events, and delayed feedback tags. However, with the addition of this rich information to the input sequences and the accompanying flatter goodness-of-fit curve as discussed above, we noted an irregular pattern of correlation between goodness-of-fit and predictive accuracy that is worthy of future exploration. Specifically, it appears that the most highly predictive DATASKDELAY ADJPAIR model, which is the most highly predictive of all models in all conditions, does *not* correspond to the best (lowest) AIC for that condition (Figure 3). This finding suggests that when a predictive task is the primary goal, a predictive metric should be used to select the best-fit model. Additional support for such an approach is provided by the close correspondence between training and testing set prediction accuracy.

8 Conclusion

Understanding how HMMs behave under different feature sets is an important step toward learning effective models of task-oriented dialogue. This paper has examined how HMMs converge to a best number of hidden states under different experimental conditions. We have also considered how well HMMs under these conditions predict tutor dialogue acts within a corpus of task-oriented tutoring, a crucial step toward learning dialogue policies from human corpora. The findings highlight the importance of adding rich task-based features to the input sequences in order to learn HMMs that converge tightly on the best-fit number of hidden states. The results also indicate that caution should be used when utilizing traditional goodness-of-fit metrics, which are appropriate for descriptive applications, if the goal is to learn a highly predictive model.

This line of research is part of a larger research program of learning unsupervised models of human task-oriented dialogue that can be used to define the behavior of dialogue systems. Developing a framework for learning a dialogue policy from human corpora, as discussed here, is a critical step toward that goal. Future work should focus on unsupervised dialogue act classification, and address the challenges of user plan recognition.

Acknowledgments. This work is supported in part by National Science Foundation through Grants REC-0632450, IIS-0812291, DRL-1007962 and the STARS Alliance Grant CNS-0739216. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

References

Akaike, H. (1976). An information criterion (AIC). *Math. Sci.*, *14*(153), 5-9.

Bangalore, S., Di Fabbrizio, G., & Stents, A. (2006). Learning the structure of task-driven human-human dialogs. *Proceedings of ACL '06*, 201-208.

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 SIGDIAL* (pp. 297-305).

Boyer, K. E., Phillips, R., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (in press). Learning a Tutorial Dialogue Policy for Delayed Feedback. *Proceedings of the 24th International FLAIRS Conference*.

Boyer, K. E., Phillips, R., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2009). Modeling dialogue structure with adjacency pair analysis and hidden Markov models. *Proceedings of NAACL HLT, Companion Volume*, 49-52.

Boyer, K. E., Phillips, R., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2010). Leveraging Hidden Dialogue State to Select Tutorial Moves. *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 66-73).

Boyer, K. E., Phillips, R., Ingram, A., Young, E., Wallis, M., Vouk, M., et al. (in press). Investigating the Relationship Between Dialogue Structure and Tutoring Effectiveness: A Hidden Markov Modeling Approach. *International Journal of Artificial Intelligence in Education*.

Di Eugenio, B., Xie, Z., & Serafin, R. (2010). Dialogue Act Classification, Higher Order Dialogue Structure, and Instance-Based Learning. *Dialogue & Discourse*, *1*(2), 1-24.

Georgila, K., Lemon, O., Henderson, J., & Moore, J. D. (2009). Automatic annotation of context and speech acts for dialogue corpora. *Natural Language Engineering*, *15*(3), 315-353.

Midgley, T. D., Harrison, S., & MacNish, C. (2006). Empirical verification of adjacency pairs using dialogue segmentation. *Proceedings of SIGDIAL*, 104-108.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257-286.

Rotaru, M., & Litman, D. J. (2009). Discourse Structure and Performance Analysis : Beyond the Correlation. *Proceedings of SIGDIAL* (pp. 178-187).

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., et al. (2000). Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, *26*(3), 339-373.

VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rose, C. P. (2007). When Are Tutorial Dialogues More Effective Than Reading? *Cognitive Science: A Multidisciplinary Journal*, *30*(1), 3-62.

Woszczyna, M., & Waibel, A. (1994). Inferring linguistic structure in spoken language. *Proceedings of the International Conference on Spoken Language Processing* (pp. 847-850).

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., et al. (2010). The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, *24*(2), 150-174.

Appendix. Excerpt from task-oriented textual human-human tutoring corpus.

Speaker	Utterance or Event	Tag
Student:	*[Task action on subtask 3-c-i-4]*	BUGGY
Student:	*[Task action on subtask 3-c-ii-5]*	CORRECT
Tutor:	*[Does not provide remediation for existing bug]*	NOREMEDIATION
Student:	*[Task action on subtask 3-c-iii-1]*	BUGGY
Student:	i don't remember off the top of my head how the swap function worked. most of the time i just copied and pasted it from some of my older code	NEGATIVECONTENTFDBK
Tutor:	The easiest way to swap x and y is to make a temporary variable	
Student:	Ok	ACK
Student:	do i need to pass the entire array and the indecies of the items to swap?	ASSESSQ
Tutor:	if you want to use a seperate method to swap, then yes, you'll have to pass those things	POSCONTENTFDBK
Tutor:	*[Does not mention a correctly completed subtask]*	NOMENTIONCOMP
Student:	oh. i guess i could just swap it in the same method. it is problably easier that way, and less code. we were showed in class how to do it separately, but i had never thought of doing it the other way.	STMT
Student:	*[Task action on subtask 3-c-iii-2]*	DISPREFERRED
Tutor:	Both ways work, but it's definitely less code to just do it inside this method.	STMT
Student:	Ok	ACK