Lightly Supervised Learning of Procedural Dialog Systems

Svitlana Volkova
CLSP
Johns Hopkins University
Baltimore, MD
svitlana@jhu.edu

Pallavi Choudhury, Chris Quirk, Bill Dolan
NLP Group
Microsoft Research
Redmond, WA
pallavic,chrisq,
billdol@microsoft.com

Luke Zettlemoyer
Computer Science and Engineering
University of Washington
Seattle, WA
lsz@cs.washington.edu

Abstract

Procedural dialog systems can help users achieve a wide range of goals. However, such systems are challenging to build, currently requiring manual engineering of substantial domain-specific task knowledge and dialog management strategies. In this paper, we demonstrate that it is possible to learn procedural dialog systems given only light supervision, of the type that can be provided by non-experts. We consider domains where the required task knowledge exists in textual form (*e.g.*, instructional web pages) and where system builders have access to statements of user intent (*e.g.*, search query logs or dialog interactions). To learn from such textual resources, we describe a novel approach that first automatically extracts task knowledge from instructions, then learns a dialog manager over this task knowledge to provide assistance. Evalutation in Microsoft Office domain demonstrates that the individual components are accurate enough and when integrated into a dialog system provide effective help to users.

1 Introduction

Procedural dialog systems aim to assist users with a wide range of goals. For example, they can guide visitors through a museum (Traum et al., 2012; Aggarwal et al., 2012), teach students physics (Steinhauser et al., 2011; Dzikovska et al., 2011), or enable interaction with a health care system (Morbini et al., 2012; Rizzo et al., 2011). However, such systems are challenging to build, currently requiring expensive, expert engineering of significant domain-specific task knowledge and dialog management strategies.

U: "I want to add page numbers and a title"
S: "Top or Bottom of the page?"
U: "Top"
S: "Please select page design from the templates" (*System shows drop down menu*)
U: *User selects from menu*
S: "Enter header or footer content"
U: "C.V."
S: "Task completed."

Figure 1: An example dialog interaction between a system (S) and user (U) that can be automatically achieved by learning from instructional web page and query click logs.

In this paper, we present a new approach for learning procedural dialog systems from task-oriented textual resources in combination with light, non-expert supervision. Specifically, we assume access to task knowledge in textual form (*e.g.*, instructional web pages) and examples of user intent statements (*e.g.*, search query logs or dialog interactions). Such instructional resources are available in many domains, ranging from recipes that describe how to cook meals to software help web pages that describe how to achieve goals by interacting with a user interface.[1]

[1] ehow.com, wikianswers.com

There are two key challenges: we must (1) learn to convert the textual knowledge into a usable form and (2) learn a dialog manager that provides robust assistance given such knowledge. For example, Figure 1 shows the type of task assistance that we are targeting in the Microsoft Office setting, where the system should learn from web pages and search query logs. Our central contribution is to show that such systems can be built without the help of knowledge engineers or domain experts. We present new approaches for both of our core problems. First, we introduce a method for learning to map instructions to tree representations of the procedures they describe. Nodes in the tree represent points of interaction with the questions the system can ask the user, while edges represent user responses. Next, we present an approach that uses example user intent statements to simulate dialog interactions, and learns how to best map user utterances to nodes in these induced dialog trees. When combined, these approaches produce a complete dialog system that can engage in conversations by automatically moving between the nodes of a large collection of induced dialog trees.

Experiments in the Windows Office help domain demonstrate that it is possible to build an effective end-to-end dialog system. We evaluate the dialog tree construction and dialog management components in isolation, demonstrating high accuracy (in the 80-90% range). We also conduct a small-scale user study which demonstrates that users can interact productively with the system, successfully completing over 80% of their tasks. Even when the system does fail, it often does so in a graceful way, for example by asking redundant questions but still reaching the goal within a few additional turns.

2 Overview of Approach

Our task-oriented dialog system understands user utterances by mapping them to nodes in dialog trees generated from instructional text. Figure 2 shows an example of a set of instructions and the corresponding dialog tree. This section describes the problems that we must solve to enable such interactions, and outlines our approach for each.

Knowledge Acquisition We extract task knowledge from instructional text (*e.g.*, Figure 2, left) that describes (1) actions to be performed, such as clicking a button, and (2) places where input is needed from the user, for example to enter the contents of the footer or header they are trying to create. We aim to convert this text into a form that will enable a dialog system to automatically assist with the described task. To this end, we construct *dialog trees* (*e.g.*, Figure 2, right) with nodes to represent entire documents (labeled as topics t), nodes to represent user goals or intents (g), and system action nodes (a) that enable execution of specific commands. Finally, each node has an associated system action a_s, which can prompt user input (*e.g.*, with the question "Top or bottom of the page?") and one or more user actions a_u that represent possible responses. All nodes connect to form a tree structure that follows the workflow described in the document. Section 3 presents a scalable approach for inducing dialog trees.

Dialog Management To understand user intent and provide task assistance, we need a dialog management approach that specifies what the system should do and say. We adopt a simple approach that at all times maintains an index into a node in a dialog tree. Each system utterance is then simply the action a_s for that node. However, the key challenge comes in interpreting user utterances. After each user statement, we must automatically update our node index. At any point, the user can state a general goal (*e.g.*, "I want to add page numbers"), refine their goal (*e.g.*, "in a footer"), or both (*e.g.*,"I want to add page numbers in the footer"). Users can also change their goals in the process of completing the tasks.

We develop a simple classification approach that is robust to these different types of user behavior. Specifically, we learn classifiers that, given the dialog interaction history, predict how to pick the next tree node from the space of all nodes in the dialog trees that define the task knowledge. We isolate two specific cases, classifying initial user utterances (Section 4) and classifying all subsequent utterances (Section 5). This approach allows us to isolate the difference in language for the two cases, and bias the second case to prefer tree nodes near the current one. The resulting approach allows for significant flexibility in traversing the dialog trees.

Data and Evaluation We collected a large set of such naturally-occurring web search queries that resulted in a user click on a URL in the Microsoft Office help domain.[2] We found that queries longer that 4-5 words often resembled natural language utterances that could be used for dialog interac-

[2] `http://office.microsoft.com`

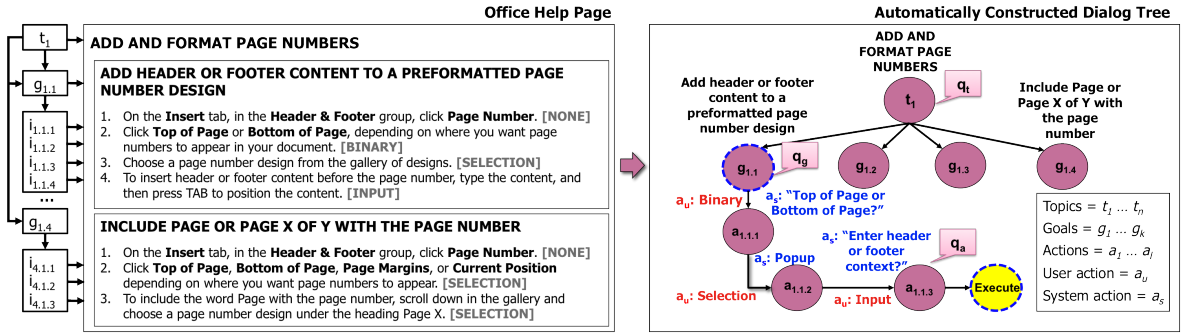


Figure 2: An example instructional text paired with a section of the corresponding dialog tree.

tions, for example *how do you add borders, how can I add a footer, how to insert continuous page numbers, and where is the header and footer.*

We also collected instructional texts from the web pages that describe how to solve 76 of the most pressing user goals, as indicated by query click log statistics. On average 1,000 user queries were associated with each goal. To some extent clickthroughs can be treated as a proxy for user frustration; popular search targets probably represent user pain points.

3 Building Dialog Trees from Instructions

Our first problem is to convert sets of instructions for user goals to dialog trees, as shown in Figure 2. These goals are broadly grouped into topics (instructional pages). In addition, we manually associate each node in a dialog tree with a training set of 10 queries. For the 76 goals (246 instructions) in our data, this annotation effort took a single annotator a total of 41 hours. Scaling this approach to the entire Office help domain would require a focused annotation effort. Crucially, though, this annotation work can be carried out by non-specialists, and could even be crowdsourced (Bernstein et al., 2010).

Problem Definition As input, we are given instructional text $(p_1 \dots p_n)$, comprised of topics $(t_1 \dots t_n)$ describing:

(1) high-level user intents (*e.g.,* t_1 – "add and format page numbers")

(2) goals $(g_1, \dots, g_k)$ that represent more specific user intents (*e.g.,* g_1 – "add header or footer content to a preformatted page number design", g_2 – "place the page number in the side margin of the page").

Given instructional text $p_1 \dots p_n$ and queries $q_1 \dots q_m$ per topic t_i, our goals are as follows:

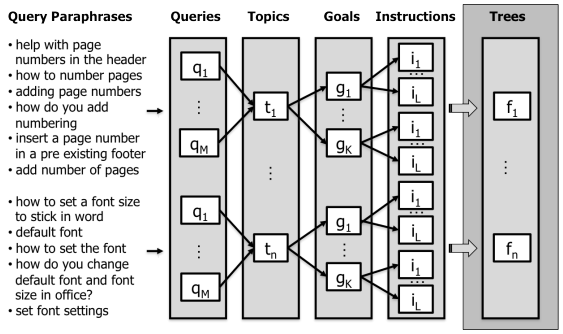


Figure 3: Relationships between user queries and OHP with goals, instructions and dialog trees.

- for every instructional page p_i extract a topic t_i and a set of goals $g_1 \dots g_k$;
- for every goal g_j for a topic t_i, extract a set of instructions $i_1 \dots i_l$;
- from topics, goals and instructions, construct dialog trees $f_1 \dots f_n$ (one dialog tree per topic). Classify instructions to user interaction types thereby identifying system action nodes $a_s^1 \dots a_s^l$. Transitions between these nodes are the user actions $a_u^1 \dots a_u^l$.

Figure 2 (left) presents an example of a topic extracted from the help page, and a set of goals and instructions annotated with user action types.

In the next few sections of the paper, we outline an overall system component design demonstrating how queries and topics are mapped to the dialog trees in Figure 3. The figure shows *many-to-one* relations between queries and topics, *one-to-many* relations between topics and goals, goals and instructions, and *one-to-one* relations between topics and dialog trees.

User Action Classification We aim to classify instructional text $(i_1 \dots i_l)$ for every goal g_j in the decision tree into four categories: *binary, selection, input* or *none*.

Given a single instruction i with category a_u, we use a log-linear model to represent the distri-

bution over the space of possible user actions. Under this representation, the user action distribution is defined as:

$$p(a_u|i,\theta) = \frac{e^{\theta \cdot \phi(a_u,i)}}{\sum_{a'_u} e^{\theta \cdot \phi(a_u,i)}}, \quad (1)$$

where $\phi(a_u, i) \in \mathbb{R}^n$ is an n-dimensional feature representation and $\vec{\theta}$ is a parameter vector we aim to learn. Features are indicator functions of properties of the instructions and a particular class. For smoothing we use a zero mean, unit variance Gaussian prior $(0, 1)$ that penalizes $\vec{\theta}$ for drifting too far from the mean, along with the following optimization function:

$$\log p(A_u, \theta|I) = \log p(A_u|I,\theta) - \log p(\theta) =$$
$$= \sum_{a_u, i \in (A_u, I)} p(a_u|i,\theta) - \sum_i \frac{(\theta - \mu_i)^2}{2\sigma_i^2} + k \quad (2)$$

We use L-BFGS (Nocedal and Wright, 2000) as an optimizer.

Experimental Setup As described in Section 2, our dataset consists of 76 goals grouped into 30 topics (average 2-3 goals per topic) for a total of 246 instructions (average 3 instructions per goal). We manually label all instructions with user action a_u categories. The distribution over categories is *binary=14*, *input=23*, *selection=80* and *none=129*. The data is skewed towards the categories *none* and *selection*. Many instruction do not require any user input and can be done automatically, e.g., "On the Insert tab, in the Header and Footer group, click Page Number". The example instructions with corresponding user action labels are shown in Figure 2 (left) . Finally, we divide the 246 instructions into 2 sets: 80% training and 20% test, 199 and 47 instructions respectively.

Results We apply the user action type classification model described in the Eq.1 and Eq.2 to classify instructions from the test set into 4 categories. In Table 1 we report classification results for 2 baselines: a majority class and heuristic-based approach, and 2 models with different feature types: *ngrams* and *ngrams + stems*. For a heuristic baseline, we use simple lexical clues to classify instructions (*e.g., X or Y* for *binary*, *select Y* for *selection* and *type X, insert Y* for *input*). Table 1 summarizes the results of mapping instructional text to user actions.

Features	# Features	Accuracy
Baseline 1: Majority	–	0.53
Baseline 2: Heuristic	–	0.64
Ngrams	10,556	0.89
Ngrams + Stems	12,196	0.89

Table 1: Instruction classification results.

Building the Dialog Trees Based on the classified user action types, we identify system actions $a_s^1 \dots a_s^l$ which correspond to 3 types of user actions $a_s^1 \dots a_s^l$ (excluding *none* type) for every goal in a topic t_i. This involved associating all words from an instruction i_l with a system action a_s^l. Finally, for every topic we automatically construct a dialog tree as shown in Figure 2 (right). The dialog tree includes a topic t_1 with goals $g_1 \dots g_4$, and actions (user actions a_u and system actions a_s).

Definition 1. *A dialog tree encodes a user-system dialog flow about a topic t_i represented as a directed unweighted graph $f_i = (V, E)$ where topics, goals and actions are nodes of corresponding types $\{t_1 \dots t_n\}, \{g_1 \dots g_k\}, \{a_1 \dots a_l\} \in V$. There is a hierarchical dependency between topic, goal and action nodes. User interactions are represented by edges $t_i \rightarrow \{g_1 \dots g_k\}, a_u^1 = (g_j, a_1) \dots a_u^l = (a_{k-1}, a_k) \in E$.*

For example, in the dialog tree in Figure 2 there is a relation $t_1 \rightarrow g_4$ between the topic t_1 "add and format page numbers" and the goal g_4 "include page of page X of Y with the page number". Moreover, in the dialog tree, the topic level node has one index $i \in [1..n]$, where n is the number of topics. Every goal node includes information about its parent (topic) node and has double index $i.j$, where $j \in [1..k]$. Finally, action nodes include information about their parent (goal) and grandparent (topic) nodes and have triple index $i.j.z$, where $z \in [1..l]$.

4 Understanding Initial Queries

This section presents a model for classifying initial user queries to nodes in a dialog tree, which allows for a variety of different types of queries. They can be under-specified, including information about a topic only (*e.g.*, "add or delete page numbers"); partially specified, including information about a goal (*e.g.*, "insert page number"); or over-specified, including information about an action (*e.g.*, "page numbering at bottom page".)

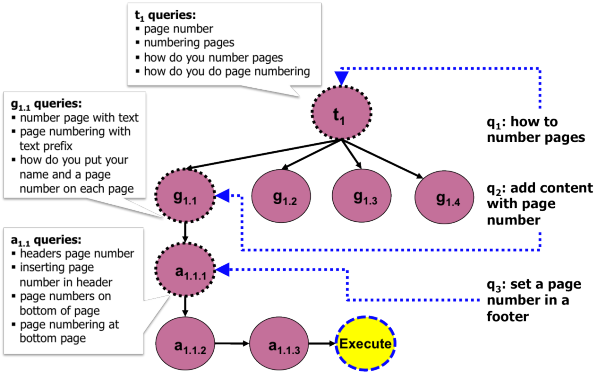


Figure 4: Mapping initial user queries to the nodes on different depth in a dialog tree.

Problem Definition Given an initial query, the dialog system initializes to a state s_0, searches for the deepest relevant node given a query, and maps the query to a node on a topic t_i, goal g_j or action a_k level in the dialog tree f_i, as shown in Figure 4.

More formally, as input, we are given automatically constructed dialog trees $f_1 \dots f_n$ for instructional text (help pages) annotated with topic, goal and action nodes and associated with system actions as shown in Figure 2 (right). From the query logs, we associate queries with each node type: topic q_t, goal q_g and action q_a. This is shown in Figure 2 and 4. We join these dialog trees representing different topics into a *dialog network* by introducing a global root. Within the network, we aim to find (1) an initial dialog state s_0 that maximizes the probability of state given a query $p(s_0|q,\theta)$; and (2) the deepest relevant node $v \in V$ on topic t_i, goal g_j or action a_k depth in the tree.

Initial Dialog State Model We aim to predict the best node in a dialog tree $t_i, g_j, a_l \in V$ based on a user query q. A query-to-node mapping is encoded as an initial dialog state s_0 represented by a binary vector over all nodes in the dialog network:

$s_0 = [t_1, g_{1.1}, g_{1.2}, g_{1.2.1} \dots, t_n, g_{n.1}, g_{n.1.1}]$.

We employ a log-linear model and try to maximize initial dialog state distribution over the space of all nodes in a dialog network:

$$p(s_0|q,\theta) = \frac{e^{\sum_i \theta_i \phi_i(s_0,q)}}{\sum_{s'_0} e^{\sum_i \theta_i \phi_i(s'_0,q)}}, \qquad (3)$$

Optimization follows Eq. 2.

We experimented with a variety of features. Lexical features included query *ngrams* (up to 3-grams) associated with every node in a dialog tree with removed stopwords and stemming query unigrams. We also used network structural features:

	Accuracy		
Features	Topic	Goal	Action
Random	0.10	0.04	0.04
TFIDF 1Best	0.81	0.21	0.45
Lexical (L)	0.92	0.66	0.63
L + 10TFIDF	0.94	0.66	0.64
L + 10TFIDF + PO	0.94	0.65	0.65
L + 10TFIDF + QO	0.95	0.72	0.69
All above + QHistO	0.96	0.73	0.71

Table 2: Initial dialog state classification results where L stands for lexical features, 10TFIDF - 10 best tf-idf scores, PO - prompt overlap, QO - query overlap, and QHistO - query history overlap.

tf-idf scores, query *ngram* overlap with the topic and goal descriptions, as well as system action prompts, and query *ngram* overlap with a history including queries from parent nodes.

Experimental Setup For each dialog tree, nodes corresponding to single instructions were hand-annotated with a small set of user queries, as described in Section 3. Approximately 60% of all action nodes have no associated queries[3] For the 76 goals, the resulting dataset consists of 972 node-query pairs, 80% training and 20% test.

Results The initial dialog state classification model of finding a single node given an initial query is described in Eq. 3.

We chose two simple baselines: (1) randomly select a node in a dialog network and (2) use a tf-idf 1-best model.[4] Stemming, stopword removal and including top 10 tf-idf results as features led to a 19% increase in accuracy on an action node level over baseline (2). Adding the following features led to an overall 26% improvement: query overlap with a system prompt (PO), query overlap with other node queries (QO), and query overlap with its parent queries (QHistO) .

We present more detailed results for topic, goal and action nodes in Table 2. For nodes deeper in the network, the task of mapping a user query to an action becomes more challenging. Note, however, that the action node accuracy numbers actually un-

[3]There are multiple possible reasons for this: the software user interface may already make it clear how to accomplish this intent, the user may not understand that the software makes this fine-grained option available to them, or their experience with search engines may lead them to state their intent in a more coarse-grained way.

[4]We use cosine similarity to rank all nodes in a dialog network and select the node with the highest rank.

derstate the utility of the resulting dialog system. The reason is that even incorrect node assignments can lead to useful system performance. As long as a misclassification results being assigned to a too-high node within the correct dialog tree, the user will experience a graceful failure: they may be forced to answer some redundant questions, but they will still be able to accomplish the task.

5 Understanding Query Refinements

We also developed a classifier model for mapping followup queries to the nodes in a dialog network, while maintaining a dialog state that summarizes the history of the current interaction.

Problem Definition Similar to the problem definition in Section 4, we are given a network of dialog trees $f_1 \ldots f_n$ and a query q', but in addition we are given the previous dialog state s, which contains the previous user utterance q and the last system action a_s. We aim to find a new dialog state s' that pairs a node from the dialog tree with updated history information, thereby undergoing a dialog state update.

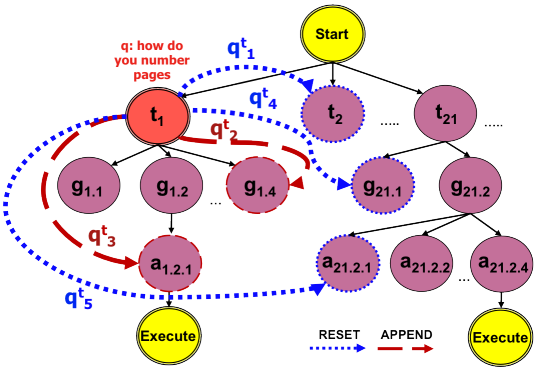
We learn a linear classifier that models $p(s'|q', q, a_s, \theta)$, the dialog state update distribution, where we constrain the new state s' to contain the new utterance q' we are interpreting. This distribution models 3 transition types: *append*, *override* and *reset*.

Definition 2. *An append action defines a dialog state update when transitioning from a node to its children at any depth in the same dialog tree e.g., $t_i \rightarrow g_{i.j}$ (from a topic to a goal node), $g_{i.j} \rightarrow a_{i.j.z}$ (from a goal to an action node) etc.*

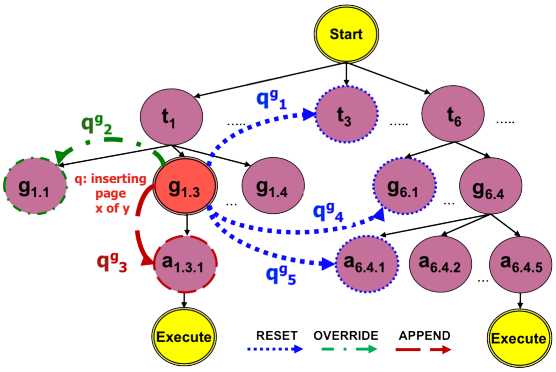
Definition 3. *An override action defines a dialog state update when transitioning from a goal to its sibling node. It could also be from an action node[5] to another in its parent sibling node in the same dialog tree e.g., $g_{i.j-1} \rightarrow g_{i.j}$ (from one goal to another goal in the same topic tree), $a_{i.j.z} \rightarrow a_{i.\neg j.z}$ (from an action node to another action node in a different goal in the same dialog tree) etc.*

Definition 4. *A reset action defines a dialog state update when transitioning from a node in a current dialog tree to any other node at any depth in a dialog tree other than the current dialog tree e.g., $t_i \rightarrow t_{\neg i}$, (from one topic node to another topic node) $t_i \rightarrow g_{\neg i.j}$ (from a topic node to a goal node in a different topic subtree), etc.*

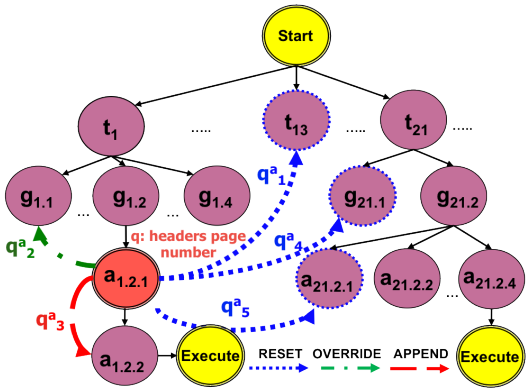
[5] A transition from $a_{i.j.z}$ must be to a different goal or an action node in a different goal but in the same dialog tree.



(a) Updates from topic node t_i



(b) Updates from goal node g_j



(c) Updates from action node a_l

Figure 5: Information state updates: append, reset and override updates based on Definition 2, 3 and 4, respectively, from topic, goal and action nodes.

The *append action* should be selected when the user's intent is to clarify a previous query (*e.g.*, "insert page numbers" → "page numbers in the footer"). An *override action* is appropriate when the user's intent is to change a goal within the same topic (*e.g.*, "insert page number → "change page number"). Finally, a *reset action* should be used when the user's intent is to restart the dialog (*e.g.*, "insert page x of y" → "set default font"). We present more examples for *append*, *override* and *reset* dialog state update actions in Table 3.

Previous Utterance, q	**User Utterance,** q'	**Transition**	**Update Action,** a
inserting page numbers	q_1^t add a background	$t_i \rightarrow t_{\neg i}$	2, reset-T, reset
how to number pages	q_2^t insert numbers on pages in margin	$t_i \rightarrow s_{i.j}$	1.4, append-G, append
page numbers	q_3^t set a page number in a footer	$t_i \rightarrow a_{i.j.z}$	1.2.1, append-A, append
page number a document	q_4^t insert a comment	$t_i \rightarrow g_{\neg i.j}$	21.1, reset-G, reset
page number	q_5^t add a comment "redo"	$t_i \rightarrow a_{\neg i.j.z}$	21.2.1, reset-A, reset
page x of y	q_1^g add a border	$g_{i.j} \rightarrow t_{\neg i}$	6, reset-T, reset
format page x of x	q_2^g enter text and page numbers	$g_{i.j} \rightarrow g_{i.\neg j}$	1.1, override-G, override
enter page x of y	q_3^g page x of y in footer	$g_{i.j} \rightarrow a_{i.j.z}$	1.3.1, append-A, append
inserting page x of y	q_4^g setting a default font	$g_{i.j} \rightarrow g_{\neg i.j}$	6.1, reset-G, reset
showing page x of x	q_5^g set default font and style	$g_{i.j} \rightarrow a_{\neg i.j.z}$	6.4.1, reset-A, reset
page numbers bottom	q_1^a make a degree symbol	$a_{i.j.z} \rightarrow t_{\neg i}$	13, reset-T, reset
numbering at bottom page	q_2^a insert page numbers	$a_{i.j.z} \rightarrow g_{i.\neg j}$	1.1, override-G, override
insert footer page numbers	q_3^a page number design	$a_{i.j.z-1} \rightarrow a_{i.j.z}$	1.2.2, append-A, append
headers page number	q_4^a comments in document	$a_{i.j.z} \rightarrow g_{\neg i.j}$	21.1, reset-G, reset
page number in a footer	q_5^a changing initials in a comment	$a_{i.j.z} \rightarrow a_{\neg i.j.z}$	21.2.1, reset-A, reset

Table 3: Example q and q' queries for append, override and reset dialog state updates.

Figure 5 illustrates examples of *append*, *override* and *reset* dialog state updates. All transitions presented in Figure 5 are aligned with the example q and q' queries in Table 3.

Dialog State Update Model We use a log-linear model to maximize a dialog state distribution over the space of all nodes in a dialog network:

$$p(s'|q', q, a_s\theta) = \frac{e^{\sum_i \theta_i \phi_i(s', q', a_s, q)}}{\sum_{s''} e^{\sum_i \theta_i \phi_i(s'', q', a_s, q)}}, \quad (4)$$

Optimization is done as described in Section 3.

Experimental Setup Ideally, dialog systems should be evaluated relative to large volumes of real user interaction data. Our query log data, however, does not include dialog turns, and so we turn to simulated user behavior to test our system.

Our approach, inspired by recent work (Schatzmann et al., 2006; Scheffler and Young, 2002; Georgila et al., 2005), involves simulating dialog turns as follows. To define a state s we sample a query q from a set of queries per node v and get a corresponding system action a_s for this node; to define a state s', we sample a new query q' from another node $v' \in V, v \neq v'$ which is sampled using a prior probability biased towards append: *p(append)=0.7, p(override)=0.2, p(reset)=0.1*. This prior distribution defines a dialog strategy where the user primarily continues the current goal and rarely resets.

We simulate 1100 previous state and new query pairs for training and 440 pairs for testing. The features were lexical, including word ngrams, stems with no stopwords; we also tested network structure, such as:

- old q and new q' query overlap (QO);
- q' overlap with a system prompt a_s (PO);
- q' *ngram* overlap with all queries from the old state s (SQO);
- q' *ngram* overlap with all queries from the new state s' (S$'$QO);
- q' *ngram* overlap with all queries from the new state parents (S$'$ParQO).

Results Table 4 reports results for dialog state updates for topic, goal and action nodes. We also report performance for two types of dialog updates such as: *append* (App.) and *override* (Over.).

We found that the combination of lexical and query overlap with the previous and new state queries yielded the best accuracies: 0.95, 0.84 and 0.83 for topic, goal and action node level, respectively. As in Section 4, the accuracy on the topic level node was highest. Perhaps surprisingly, the reset action was perfectly predicted (accuracy is 100% for all feature combinations, not included in figure). The accuracies for append and override actions are also high (append 95%, override 90%).

Features	Topic	Goal	Action	App.	Over.
L	0.92	0.76	0.78	0.90	0.89
L+Q	0.93	0.80	0.80	0.92	0.83
L+P	0.93	0.80	0.79	0.91	0.85
L+Q+P	0.94	0.80	0.80	0.93	0.85
L+SQ	0.94	0.82	0.81	0.93	0.85
L+S$'$Q	0.93	0.80	0.80	0.91	**0.90**
L+S$'$+ParQ	0.94	0.80	0.80	0.91	0.86
L+Q+S$'$Q	0.94	0.81	0.81	0.91	0.88
L+SQ+S$'$Q	**0.95**	**0.84**	**0.83**	**0.94**	0.88

Table 4: Dialog state updates classification accuracies where L stands for lexical features, Q - query overlap, P - prompt overlap, SQ - previous state query overlap, S$'$Q - new state query overlap, S$'$ParQ - new state parent query overlap.

6 The Complete Dialog System

Following the overall setup described in Section 2, we integrate the learned models into a complete dialog system. To evaluate the quality of the interactions with this system, we performed a small scale user study, as described here.

Experimental Setup We randomly chose 8 goals, across topics, from the 76 used to train our models. For each goal, six users were shown a Microsoft Word document before and after the goal was executed. Users were allowed to interact with the system in an unconstrained manner providing a total of 48 interactions. If at any point they thought the system response was irrelevant to their intent, they could either rephrase or start over.

User Study We objectively measured task completion and the quality of individual system utterances. We also asked the users to rate the system (on a scale of 1-5, with 5 being best) on various measures, including whether the system asked relevant questions. Table 5 summarizes the results. In general, users were able to complete the tasks and were reasonably satisfied.

Discussion Figure 6 shows four example dialogs. User utterances ranged from single keywords to rich, sentence-like goal descriptions. For the shorter inputs, the system was generally able to identify a small cohort of high-probability intents. For longer inputs, especially with explicitly specified parameter settings, the system was usually able to elicit whatever additional information was required to complete the task. The majority of errors fell into two categories when the system:

I. missed key information in the input and asked redundant questions – Figure 6 (C);
II. failed to correctly identify the intent, and asked an irrelevant question – Figure 6 (D).

Table 5 shows the frequency of each error type.

Objective Measures		Observed Rates
User achieved task		83%
System asked irrelevant questions		19%
System asked redundant questions		18%
User Satisfaction	**Score**	**Std.Dev.**
Overall satisfaction	3.42	0.49
Questions were relevant	4.17	0.41
Ambiguity was managed	4.00	0.89

Table 5: User study results.

U: "I want a page number and title at the bottom of the page"
S: "Choose a page number design from the gallery of designs"
U: *User selects design*
S: "To insert header or footer content before the page number, type the content, and then press TAB to position the content"
U: "Professional Resume"
S: "Task Completed"

(A)

U: "Alphabetize bullets"
S: "Do you want to sort a list alphabetically"
U: "Sort a list"
S: "Select the text in a bulleted or numbered list"
U: *User selection*
S: "Ascending or Descending?"
U: "Ascending"
S: "Task Completed"

(B)

U: "I want the background color to be yellow"
S: "Choose a color for the document"
U: "Yellow "
S: "Task completed"

(C)

U: "I want to change the font face and size"
S: "Choose the default font"
U: "No, reduce the size of the text"
S: *System Restarts*

(D)

Figure 6: Four example dialogs from the user study, including cases that (A and B) complete successfully, (C) have a redundant question, and (D) fail to recognize the user intent.

7 Related work

To the best of our knowledge, this paper presents the first effort to induce full procedural dialog systems from instructional text and query click logs.

Grounded Language Learning There has been significant interest in grounded language learning. Perhaps the most closely related work learns to understand instructions and automati-

cally complete the tasks they describe (Branavan et al., 2009; Vogel and Jurafsky, 2010; Kushman et al., 2009; Branavan et al., 2010; Artzi and Zettlemoyer, 2013). However, these approaches did not model user interaction. There are also many related approaches for other grounded language problems, including understanding game strategy guides (Branavan et al., 2011), modeling users goals in a Windows domain (Horvitz et al., 1998), learning from conversational interaction (Artzi and Zettlemoyer, 2011), learning to sportscast (Chen and Mooney, 2011), learning from event streams (Liang et al., 2009), and learning paraphrases from crowdsourced captions of video snippets (Chen and Dolan, 2011).

Dialog Generation from Text Similarly to Piwek's work (2007; 2010; 2011), we study extracting dialog knowledge from documents (monologues or instructions). However, Piwek's approach generates static dialogs, for example to generate animations of virtual characters having a conversation. There is no model of dialog management or user interaction, and the approach does not use any machine learning. In contrast, to the best of our knowledge, we are the first to demonstrate it is possible to learn complete, interactive dialog systems using instructional texts (and non-expert annotation).

Learning from Web Query Logs Web query logs have been extensively studied. For example, they are widely used to represent user intents in spoken language dialogs (Tür et al., 2011; Celikyilmaz et al., 2011; Celikyilmaz and Hakkani-Tur, 2012). Web query logs are also used in many other NLP tasks, including entity linking (Pantel et al., 2012) and training product and job intent classifiers (Li et al., 2008).

Dialog Modeling and User Simulation Many existing dialog systems learn dialog strategies from user interactions (Young, 2010; Rieser and Lemon, 2008). Moreover, dialog data is often limited and, therefore, user simulation is commonly used (Scheffler and Young, 2002; Schatzmann et al., 2006; Georgila et al., 2005).

Our overall approach is also related to many other dialog management approaches, including those that construct dialog graphs from dialog data via clustering (Lee et al., 2009), learn information state updates using discriminative classification models (Hakkani-Tur et al., 2012; Mairesse et al., 2009), optimize dialog strategy using reinforcement learning (RL) (Scheffler and Young, 2002; Rieser and Lemon, 2008), or combine RL with information state update rules (Heeman, 2007). However, our approach is unique in the use of inducing task and domain knowledge with light supervision to assist the user with many goals.

8 Conclusions and Future Work

This paper presented a novel approach for automatically constructing procedural dialog systems with light supervision, given only textual resources such as instructional text and search query click logs. Evaluations demonstrated highly accurate performance, on automatic benchmarks and through a user study.

Although we showed it is possible to build complete systems, more work will be required to scale the approach to new domains, scale the complexity of the dialog manager, and explore the range of possible textual knowledge sources that could be incorporated. We are particularly interested in scenarios that would enable end users to author new goals by writing procedural instructions in natural language.

Acknowledgments

The authors would like to thank Jason Williams and the anonymous reviewers for their helpful comments and suggestions.

References

Priti Aggarwal, Ron Artstein, Jillian Gerten, Anthanasios Katsamanis, Shrikanth Narayanan, Angela Nazarian, and David R. Traum. 2012. The twins corpus of museum visitor questions. In *Proceedings of LREC*.

Yoav Artzi and Luke Zettlemoyer. 2011. Learning to recover meaning from unannotated conversational interactions. In *NIPS Workshop In Learning Semantics*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of ACM Symposium on User Interface Software and Technology*.

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of ACL*.

S. R. K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: learning to map high-level instructions to commands. In *Proceedings of ACL*.

S. R. K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of ACL*.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2012. A joint model for discovery of aspects in utterances. In *Proceedings of ACL*.

Asli Celikyilmaz, Dilek Hakkani-Tür, and Gokhan Tür. 2011. Mining search query logs for spoken language understanding. In *Proceedings of ICML*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of ACL*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of AAAI*.

Myroslava Dzikovska, Amy Isard, Peter Bell, Johanna D. Moore, Natalie B. Steinhauser, Gwendolyn E. Campbell, Leanne S. Taylor, Simon Caine, and Charlie Scott. 2011. Adaptive intelligent tutorial dialogue in the beetle ii system. In *Proceedings of AIED*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2005. Learning user simulations for information state update dialogue systems. In *Proceedings of Eurospeech*.

Dilek Hakkani-Tur, Gokhan Tur, Larry Heck, Ashley Fidler, and Asli Celikyilmaz. 2012. A discriminative classification-based approach to information state updates for a multi-domain dialog system. In *Proceedings of Interspeech*.

Peter Heeman. 2007. Combining Reinforcement Learning with Information-State Update Rules. In *Proceedings of ACL*.

Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of Uncertainty in Artificial Intelligence*.

Nate Kushman, Micah Brodsky, S. R. K. Branavan, Dina Katabi, Regina Barzilay, and Martin Rinard. 2009. WikiDo. In *ACM HotNets*.

Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, and Gary Geunbae Lee. 2009. Automatic agenda graph construction from human-human dialogs using clustering method. In *Proceedings of NAACL*.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of SIGIR*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of ACL-IJCNLP*.

F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young. 2009. Spoken language understanding from unaligned data using discriminative classification models. In *Proceedings of Acoustics, Speech and Signal Processing*.

Fabrizio Morbini, Eric Forbell, David DeVault, Kenji Sagae, David R. Traum, and Albert A. Rizzo. 2012. A mixed-initiative conversational dialogue system for healthcare. In *Proceedings of SIGDIAL*.

Jorge Nocedal and Stephen J. Wright. 2000. *Numerical Optimization*. Springer.

Patric Pantel, Thomas Lin, and Michael Gamon. 2012. Mining entity types from query logs via user intent. In *Proceedings of ACL*.

Paul Piwek and Svetlana Stoyanchev. 2010. Generating expository dialogue from monologue: Motivation, corpus and preliminary rules. In *Proceedings of NAACL*.

Paul Piwek and Svetlana Stoyanchev. 2011. Data-oriented monologue-to-dialogue generation. In *Proceedings of ACL*, pages 242–247.

Paul Piwek, Hugo Hernault, Helmut Prendinger, and Mitsuru Ishizuka. 2007. T2d: Generating dialogues between virtual agents automatically from text. In *Proceedings of Intelligent Virtual Agents*.

Verena Rieser and Oliver Lemon. 2008. Learning effective multimodal dialogue strategies from wizard-of-oz data: Bootstrapping and evaluation. In *Proceedings of ACL*.

A. Rizzo, Kenji Sagae, E. Forbell, J. Kim, B. Lange, J. Buckwalter, J. Williams, T. Parsons, P. Kenny, David R. Traum, J. Difede, and B. Rothbaum. 2011. Simcoach: An intelligent virtual human system for providing healthcare information and support. In *Proceedings of ITSEC*.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2).

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of Human Language Technology Research*.

Natalie B. Steinhauser, Gwendolyn E. Campbell, Leanne S. Taylor, Simon Caine, Charlie Scott, Myroslava Dzikovska, and Johanna D. Moore. 2011.

Talk like an electrician: Student dialogue mimicking behavior in an intelligent tutoring system. In *Proceedings of AIED*.

David R. Traum, Priti Aggarwal, Ron Artstein, Susan Foutz, Jillian Gerten, Athanasios Katsamanis, Anton Leuski, Dan Noren, and William R. Swartout. 2012. Ada and grace: Direct interaction with museum visitors. In *Proceedings of Intelligent Virtual Agents*.

Gökhan Tür, Dilek Z. Hakkani-Tür, Dustin Hillard, and Asli Çelikyilmaz. 2011. Towards unsupervised spoken language understanding: Exploiting query click logs for slot filling. In *Proceedings of Interspeech*.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of ACL*.

Steve Young. 2010. Cognitive user interfaces. In *IEEE Signal Processing Magazine*.