Bootstrapping Semantic Parsers from Conversations

Yoav Artzi and **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{yoav,lsz}@cs.washington.edu

Abstract

Conversations provide rich opportunities for interactive, continuous learning. When something goes wrong, a system can ask for clarification, rewording, or otherwise redirect the interaction to achieve its goals. In this paper, we present an approach for using conversational interactions of this type to induce semantic parsers. We demonstrate learning without any explicit annotation of the meanings of user utterances. Instead, we model meaning with latent variables, and introduce a loss function to measure how well potential meanings match the conversation. This loss drives the overall learning approach, which induces a weighted CCG grammar that could be used to automatically bootstrap the semantic analysis component in a complete dialog system. Experiments on DARPA Communicator conversational logs demonstrate effective learning, despite requiring no explicit meaning annotations.

1 Introduction

Conversational interactions provide significant opportunities for autonomous learning. A well-defined goal allows a system to engage in remediations when confused, such as asking for clarification, rewording, or additional explanation. The user's response to such requests provides a strong, if often indirect, signal that can be used to learn to avoid the original confusion in future conversations. In this paper, we show how to use this type of conversational feedback to learn to better recover the meaning of user utterances, by inducing semantic parsers from unannotated conversational logs. We believe that this style of learning will contribute to the long term goal of building self-improving dialog systems that continually learn from their mistakes, with little or no human intervention.

Many dialog systems use a semantic parsing component to analyze user utterances (e.g., Allen et al., 2007; Litman et al., 2009; Young et al., 2010). For example, in a flight booking system, the sentence

Sent: I want to go to Seattle on Friday
LF: $\lambda x.to(x, SEA) \wedge date(x, FRI)$

might be mapped to the logical form (LF) meaning representation above, a lambda-calculus expression defining the set of flights that match the user's desired constraints. This LF is a representation of the semantic content that comes from the sentence, and would be input to a context-dependent understanding component in a full dialog system, for example to find the date that the symbol FRI refers to.

To induce semantic parsers from interactions, we consider user statements in conversational logs and model their meaning with latent variables. We demonstrate that it is often possible to use the dialog that follows a statement (including remediations such as clarifications, simplifications, etc.) to learn the meaning of the original sentence. For example, consider the first user utterance in Figure 1, where the system failed to understand the user's request. To complete the task, the system must use a remediation strategy. Here, it takes the initiative by asking for and confirming each flight constraint in turn. This strategy produces an unnatural conversation but provides supervision for learning the meaning of the

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 421–432,
Edinburgh, Scotland, UK, July 27–31, 2011. ©2011 Association for Computational Linguistics

original utterance. We can easily record representations of the meanings the system intended to convey at each step, as seen in Figure 1, and use this indirect supervision for learning.

Learning from this weak signal is challenging. In any specific conversation, the system's remediations can fail to recover aspects of the original user meaning and can introduce spurious constraints, for example when users change their goals mid conversation. To learn effectively, the model must accumulate evidence from many interactions to best recover the meaning of each specific sentence.

We will learn semantic parsers defined by probabilistic Combinatory Categorial Grammars (PCCGs), which include both a lexicon and a weighted linear model for parse selection. The lexicon specifies the meanings of individual words and phrases, while the parameters of a parsing model define how to best combine word- and phrase-level meanings to analyze complete sentences. To learn without labeled meaning representations, we make use of a variant of the loss-sensitive Perceptron algorithm (Singh-Miller and Collins, 2007). We define loss functions to provide a rough measure of (1) how well a candidate meaning for a utterance matches the conversation that follows it and (2) how well the candidate matches our expectations about the types of things that are often said in the dialog's domain. These notions of loss drive not only the parameter estimation but also the grammar induction process that constructs the CCG lexicon. Experiments on conversation logs from the DARPA Communicator corpus (Walker et al., 2002) demonstrate the feasibility of our approach.

This paper makes the following contributions:

- A formalization of the problem of learning the meaning of user statements from conversational feedback, without requiring annotation.

- A new loss-sensitive learning algorithm for this problem that induces semantic parsers from conversation logs.

- Loss functions to measure the quality of hypothetical utterance meanings within the conversation in which they appear.

- An evaluation on logs from two dialog systems that demonstrate effective learning from conversations alone.

SYSTEM: how can i help you? (OPEN_TASK)
USER: i would like to fly from atlanta georgia to london england on september twenty fourth in the early evening i would like to return on october first departing from london in the late morning
SYSTEM: leaving what city? (ASK:$\lambda x.from(fl, x)$)
USER: atlanta georgia
SYSTEM: leaving atlanta. (CONFIRM:$from(fl, ATL)$) going to which city? (ASK:$\lambda x.to(fl, x)$)
USER: london
SYSTEM: arriving in london england. (CONFIRM:$to(fl, LON)$) what date would you like to depart atlanta? (ASK:$\lambda x.from(fl, ATL) \wedge depart\text{-}date(fl, x)$)
USER: september twenty fourth in the early evening
[conversation continues]

Figure 1: Conversational excerpt from a DARPA Communicator travel-planning dialog. Each system statement is labeled with representations of its speech act and logical meaning, in parentheses. The user utterances have no labels. Conversations of this type provide the training data to learn semantic parsers for user utterances.

2 Problem

Our goal is to learn a function that maps a sentence x to a lambda-calculus expression z. We assume access to logs of conversations with automatically generated annotation of system utterance meanings, but no explicit labeling of each user utterance meaning.

We define a conversation $\mathcal{C} = (\vec{U}, O)$ to be a sequence of utterances $\vec{U} = [u_0, \ldots, u_m]$ and a set of conversational objects O. An object $o \in O$ is an entity that is being discussed, for example there would be a unique object for each flight leg discussed in a travel planning conversation. Each utterance $u_i = (s, x, a, z)$ represents the speaker $s \in \{User, System\}$ producing the natural language statement x which asserts a speech act $a \in \{ASK, CONFIRM, \ldots\}$ with meaning representation z. For example, from the second system utterance in Figure 1 the question $x =$"Leaving what city?" is an a=ASK speech act with lambda-calculus meaning $z = \lambda x.from(fl, x)$. This meaning represents the fact that the system asked for the departure city for the conversational object $o = fl$ representing the flight leg that is currently being discussed. We will learn from conversations where the speech acts a and logical forms z for user utterances are unlabeled. Such data can be generated by recording interactions, along with each system's internal representation of its own utterances.

Finally, since we will be analyzing sentences at a specific point in a complete conversation, we define our training data as a set $\{(j_i, \mathcal{C}_i)|i = 1 \ldots n\}$. Each pair is a conversation $\mathcal{C}_i$ and the index j_i of the user utterance x in $\mathcal{C}_i$ whose meaning we will attempt to learn to recover. In general, the same conversation $\mathcal{C}$ can be used in multiple examples, each with a different sentence index. Section 8 provides the details of how the data was gathered for our experiments.

3 Overview of Approach

We will present an algorithm for learning a weighted CCG parser, as defined in Section 5, that can be used to map sentences to logical forms. The approach induces a lexicon to represent the meanings of words and phrases while also estimating the parameters of a weighted linear model for selecting the best parse given the lexicon.

Learning As defined in Section 2, the algorithm takes a set of n training examples, $\{(j_i, \mathcal{C}_i) : i = 1, \ldots, n\}$. For each example, our goal is to learn to parse the user utterance x at position j_i in $\mathcal{C}_i$. The training data contains no direct evidence about the logical form z that should be paired with x, or the CCG analysis that would be used to construct z. We model all of these choices as latent variables.

To learn effectively in this complex, latent space, we introduce a loss function $\mathcal{L}(z, j, \mathcal{C}) \in \mathbb{R}$ that measures how well a logical form z models the meaning for the user utterance at position j in $\mathcal{C}$. In Section 6, we will present the details of the loss we use, which is designed to be sensitive to remediations in $\mathcal{C}$ (system requests for clarification, etc.) but also be robust to the fact that conversations often do not uniquely determine which z should be selected, for example when the user prematurely ends the discussion. Then, in Section 7, we present an approach for incorporating this loss function into a complete algorithm that induces a CCG lexicon and estimates the parameters of the parsing model.

This learning setup focuses on a subproblem in dialog; semantic interpretation. We do not yet learn to recover user speech acts or integrate the logical form into the context of the conversation. These are important areas for future work.

Evaluation We will evaluate performance on a test set $\{(x_i, z_i)|i = 1, \ldots, m\}$ of m sentences x_i that have been explicitly labeled with logical forms z_i. This data will allow us to directly evaluate the quality of the learned model. Each sentence is analyzed with the learned model alone; the loss function and any conversational context are not used during evaluation. Parsers that perform well in this setting will be strong candidates for inclusion in a more complete dialog system, as motivated in Section 1.

4 Related Work

Most previous work on learning from conversational interactions has focused on the dialog sub-problems of response planning (e.g., Levin et al., 2000; Singh et al., 2002) and natural language generation (e.g., Lemon, 2011). We are not aware of previous work on inducing semantic parsers from conversations.

There has been significant work on supervised learning for inducing semantic parsers. Various techniques were applied to the problem including machine translation (Papineni et al., 1997; Ramaswamy and Kleindienst, 2000; Wong and Mooney, 2006; 2007; Matuszek et al., 2010), higher-order unification (Kwiatkowski et al., 2010), parsing (Ruifang and Mooney, 2006; Lu et al., 2008), inductive logic programming (Zelle and Mooney, 1996; Thompson and Mooney, 2003; Tang and Mooney, 2000), probabilistic push-down automata (He and Young, 2005; 2006) and ideas from support vector machines and string kernels (Kate and Mooney, 2006; Nguyen et al., 2006). The algorithms we develop in this paper build on previous work on supervised learning of CCG parsers (Zettlemoyer and Collins, 2005; 2007), as we describe in Section 5.3.

There is also work on learning to do semantic analysis with alternate forms of supervision. Clarke et al. (2010) and Liang et al. (2011) describe approaches for learning semantic parsers from questions paired with database answers, while Goldwasser et al. (2011) presents work on unsupervised learning. Our approach provides an alternative method of supervision that could complement these approaches. Additionally, there has been significant recent work on learning to do other, re-

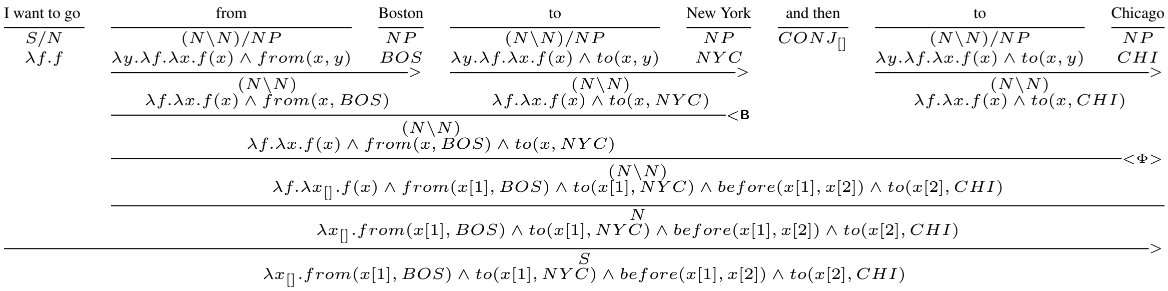
I want to go
from
Boston
to
New York
and then
to
Chicago
S/N
λf.f
(N\N)/NP
λy.λf.λx.f(x) ∧ from(x, y)
NP
BOS
(N\N)/NP
λy.λf.λx.f(x) ∧ to(x, y)
NP
NYC
CONJ[]
(N\N)/NP
λy.λf.λx.f(x) ∧ to(x, y)
NP
CHI
>
(N\N)
λf.λx.f(x) ∧ from(x, BOS)
(N\N)
λf.λx.f(x) ∧ to(x, NYC)
(N\N)
λf.λx.f(x) ∧ to(x, CHI)
<B
(N\N)
λf.λx.f(x) ∧ from(x, BOS) ∧ to(x, NYC)
<Φ>
(N\N)
λf.λx[].f(x) ∧ from(x[1], BOS) ∧ to(x[1], NYC) ∧ before(x[1], x[2]) ∧ to(x[2], CHI)
N
λx[].from(x[1], BOS) ∧ to(x[1], NYC) ∧ before(x[1], x[2]) ∧ to(x[2], CHI)
S
λx[].from(x[1], BOS) ∧ to(x[1], NYC) ∧ before(x[1], x[2]) ∧ to(x[2], CHI)

Figure 2: An example CCG parse. This parse shows the construction of a logical form with an array-typed variable $x_{[]}$ that specifies a list of flight legs, indexed by $x[1]$ and $x[2]$. The top-most parse steps introduce lexical items while the lower ones create new nonterminals according the CCG combinators ($>$, $<$, etc.), see Steedman (2000) for details.

lated, natural language semantic analysis tasks from context-dependent database queries (Miller et al., 1996; Zettlemoyer and Collins, 2009), grounded event streams (Chen et al., 2010; Liang et al., 2009), environment interactions (Branavan et al., 2009; 2010; Vogel and Jurafsky, 2010), and even unannotated text (Poon and Domingos, 2009; 2010).

Finally, the DARPA Communicator data (Walker et al., 2002) has been previously studied. Walker and Passonneau (2001) introduced a schema of speech acts for evaluation of the DARPA Communicator system performance. Georgila et al. (2009) extended this annotation schema to user utterances using an automatic process. Our speech acts extend this work to additionally include full meaning representations.

5 Mapping Sentences to Logical Form

We will use a weighted linear CCG grammar for semantic parsing, as briefly reviewed in this section.

5.1 Combinatory Categorial Grammars

Combinatory categorial grammars (CCGs) are a linguistically-motivated model for a wide range of language phenomena (Steedman, 1996; 2000). A CCG is defined by a lexicon and a set of combinators. The grammar defines a set of possible parse trees, where each tree includes syntactic and semantic information that can be used to construct logical forms for sentences.

The lexicon contains entries that define categories for words or phrases. For example, the second lexical entry in the parse in Figure 2 is:

$$\text{from} \ := \ (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge from(x, y)$$

Each category includes both syntactic and semantic information. For example, the phrase "from" is assigned the category with syntax $(N\backslash N)/NP$ and semantics $\lambda y.\lambda f.\lambda x.f(x) \wedge from(x, y)$. The outermost syntactic forward slash specifies that the entry must first be combined with an NP to the right (the departure city), while the inner back slash specifies that it will later modify a noun N to the left (to add a constraint to a set of flights). The lambda-calculus semantic expression is designed to build the appropriate meaning representation at each of these steps, as seen in the parse in Figure 2.

In general, we make use of typed lambda calculus to represent meaning (Carpenter, 1997), both in the lexicon and in intermediate parse tree nodes. We also introduce an extension for modeling array-typed variables to represent lists of individual entries. These constructions are used, for example, to model sentences describing a sequence of segments while specifying flight preferences.

Figure 2 shows how a CCG parse builds a logical form for a complete sentence with an array-typed variable. Each intermediate node in the tree is constructed with one of a small set of CCG combinator rules, see the explanation from Steedman (1996; 2000). We make use of the standard application, composition and coordination combinators, as well as type-shifting rules introduced by Zettlemoyer and Collins (2007) to model spontaneous, unedited text.

5.2 Weighted Linear CCGs

A weighted linear CCG (Clark and Curran, 2007) provides a ranking on the space of possible parses under the grammar, which can be used to select the best logical form for a sentence. This type of model is closely related to several other approaches (Ratnaparkhi et al., 1994; Johnson et al., 1999;

Lafferty et al., 2001; Collins, 2004; Taskar et al., 2004). Let x be a sentence, y be a CCG parse, and GEN$(x;\Lambda)$ be the set of all possible CCG parses for x given the lexicon Λ. Define $\phi(x, y) \in \mathbb{R}^d$ to be a d-dimensional *feature–vector* representation and $\theta \in \mathbb{R}^d$ to be a parameter vector. The optimal parse for sentence x is

$$y^*(x) = \arg \max_{y \in \text{GEN}(x;\Lambda)} \theta \cdot \phi(x, y)$$

and the final output logical form z is the lambda-calculus expression at the root of $y^*(x)$.

We compute $y^*(x)$ with a CKY-style chart parsing algorithm. Since each chart entry contains a full lambda-calculus meaning expression, we use N-best pruning to control the number of options we consider at each span. Learning a model of this form involves learning the parameters θ and the lexicon Λ. We will show that this is possible from conversational logs that do not contain any explicit labeling of the logical forms for user utterances.

5.3 Supervised learning with GENLEX

Previous work on lexical induction, including the GENLEX approach which we briefly review here, has required labeled logical meaning representations. In Section 7, we will introduce a new way of using GENLEX to learn from unannotated conversation logs.

The supervised CCG learning algorithms of Zettlemoyer and Collins (2005; 2007) induce a weighted CCG from training examples (x_i, z_i) for $i = 1 \ldots n$, where x_i is a sentence and z_i is the corresponding lambda-calculus meaning representation. The output from the algorithm is a pair (θ, Λ) containing the learned parameters and CCG lexicon.

They defined the function GENLEX(x, z) to map a sentence x and its meaning z to a large set of potential lexical entries. These entries are generated with rules that consider the logical form z and guess potential CCG categories. For example, the rule $p \rightarrow (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge p(x, y)$ introduces categories commonly used to model certain types of prepositions. This rule would, for example, introduce the category $(N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x, y)$ for any logical form z that contains the constant to. GENLEX uses a small set of rules of this type to generate categories that are paired with all possible substrings in x to form an overly general lexicon. The complete learning algorithm then simultaneously selects a small subset of all entries generated by GENLEX and estimates parameter values θ. Zettlemoyer and Collins (2005) present a more detailed explanation.

Example Conversation:

(1) SYSTEM: how can i help you? (OPEN_TASK)

(2) USER: i would like to fly from boston arriving to new york and back from new york to boston

(3) SYSTEM: leaving boston (CONFIRM:$from(fl1, BOS)$) on what date? (ASK:$\lambda x.departdate(fl1, x)$)

(4) USER: september twenty four

(5) SYSTEM: leaving boston (CONFIRM:$from(fl1, BOS)$) on september twenty four (CONFIRM:$departdate(fl1, sept\_24)$) to new york (CONFIRM:$to(fl1, NYC)$) is new york your final destination? (ASK:$\neg\exists x.fr(x, NYC)$)

(6) USER: yes i want to return to boston

(7) SYSTEM: sorry, i didn't understand. (APOLOGY) where are you flying to from new york? (ASK:$\lambda x.fr(fl2, NYC) \wedge to(fl2, x)$)

[conversation ends]

Candidate Logical Expressions for Utterance #2:

(a) $\lambda x.to(x, BOS) \wedge from(x, NYC)$

(b) $\lambda x.from(x, BOS) \wedge to(x, NYC)$

(c) $\lambda x.to(x, BOS) \wedge to(x, NYC)$

(d) $\lambda x[].from(x[1], BOS) \wedge to(x[1], NYC)$
$\wedge\ before(x[1], x[2]) \wedge return(x[2])$
$\wedge\ from(x[2], NYC) \wedge to(x[2], BOS))$

(e) $\lambda x[].from(x[1], BOS) \wedge to(x[1], NYC)$
$\wedge\ before(x[1], x[2]) \wedge return(x[2])$
$\wedge\ from(x[2], BOS) \wedge to(x[2], NYC)$

Figure 3: Conversation reflecting an interaction as seen in the DARPA Communicator travel-planning dialogs.

6 Measuring Loss

In Section 7, we will present a loss-sensitive learning algorithm that models the meaning of user utterances as latent variables to be estimated from conversational interactions.

We first introduce a loss function to measure the quality of potential meaning representations. This loss function $\mathcal{L}(z, j, \mathcal{C}) \in \mathbb{R}$ indicates how well a logical expression z represents the meaning of the j-th user utterance in conversation $\mathcal{C}$. For example, consider the first user utterance ($j = 2$) in Figure 3, which is a request for a return trip from Boston to New York. We would like to assign the lowest loss to the meaning representation (d) in Figure 3 that correctly encodes all of the stated constraints.

We make use of a loss function with two parts: $\mathcal{L}(z, j, \mathcal{C}) = \mathcal{L}_c(z, j, \mathcal{C}) + \mathcal{L}_d(z)$. The conversation loss $\mathcal{L}_c$ (defined in Section 6.1) measures how well the candidate meaning representation fits the conversation, for example incorporating information recovered through conversational remediations as motivated in Section 1. The domain loss $\mathcal{L}_d$ (described in Section 6.2) measures how well a logical form z matches domain expectations, such as the fact that flights can only have a single origin. These functions guide the types of meaning representations we expect to see, but in many cases will fail to specify a unique best option, for example in conversations where the user prematurely terminates the interaction. In Section 7, we will present a complete, loss-driven learning algorithm that is robust to these types of ambiguities while inducing a weighted CCG parser from conversations.

6.1 Conversation Loss

We will use a conversation loss function $\mathcal{L}_c(z, j, \mathcal{C})$ that provides a rough indication of how well the logical expression z represents a potential meaning for the user utterance at position j in $\mathcal{C}$. For example, the first user utterance ($j = 2$) in Figure 3 is a request for a return trip from Boston to New York where the user has explicitly mentioned both legs. The figure also shows five options (a-e) for the logical form z. We want to assign the lowest loss to option (d), which includes all of the stated constraints.

The loss is computed in four steps for a user utterance x at position j by (1) selecting a subset of system utterances in the conversation $\mathcal{C}$, (2) extracting and computing loss for semantic content from selected system utterances, (3) aligning the subexpressions in z to the extracted semantic content, and (4) computing the minimal loss value from the best alignment. In Figure 3, the loss for the candidate logical forms is computed by considering the segment of system utterances up until the conversation end. Within this segment, the matching for expression (d) involves mapping the origin and departure constraints for the first leg (Boston - New York) onto the earlier system confirmations while also aligning the ones for the second leg to system utterances later in the selected portion of the conversation. Finally, the overall score depends on the quality of the alignment, for example how many of the constraints match to confirmations. This section presents the full approach.

Segmentation For a user utterance at position j, we select all system utterances from $j - 1$ until the system believes it has completed the current subtask, as indicated by a reset action or final offer. We call this selected segment $\bar{\mathcal{C}}$. In Figure 3, $\bar{\mathcal{C}}$ ends with a reset, but in a successful interaction it would have ended with the offer of a specific flight.

Extracting Properties A property is a predicate-entity-value triplet, where the entity can be a variable from z or a conversational object. For example, $\langle from, fl, BOS\rangle$ is a property where fl is a object from $\bar{\mathcal{C}}$ and $\langle from, x, BOS\rangle$ is a property from $z = \lambda x.from(x, BOS)$. We define $P_{\bar{\mathcal{C}}}$ to be the set of properties from logical forms for system utterances in $\bar{\mathcal{C}}$. Similarly, we define P_z to be the set of properties in z.

Scoring System Properties For each system property $p \in P_{\bar{\mathcal{C}}}$ we compute its position value $pos(p)$, which is a normalized weighted average over all the positions where it appears in a logical form. For each mention the weight is obtained from its speech act. For example, properties that are explicitly confirmed contribute more to the average than those that were merely offered to the user in a select statement.

We use $pos(p)$ to compute a loss $loss(p)$ for each property $p \in P_{\bar{\mathcal{C}}}$. We first define $P^e_{\bar{\mathcal{C}}}$ to be all properties in $P_{\bar{\mathcal{C}}}$ with entity e. For entity e and position d, we define the entity-normalization function:

$$n_e(d) = \frac{d - \min_{p \in P^e_{\bar{\mathcal{C}}}} pos(p)}{\max_{p \in P^e_{\bar{\mathcal{C}}}} pos(p) - \min_{p \in P^e_{\bar{\mathcal{C}}}} pos(p)} \ .$$

For a given property $p \in P_{\bar{\mathcal{C}}}$ with an entity e we compute the loss value:

$$loss(p) = n_e^{-1}(1 - n_e(pos(p))) - 1 \ .$$

Where n_e^{-1} is the inverse of n_e. This loss value is designed to, first, provide less loss for later properties so that it, for example, favors the last property in a series of statements that finally resolves a confusion in the conversation. Second, the loss value is lower for objects mentioned closer to the user utterance x, thereby preferring objects discussed sooner.

Matching Properties An alignment $\mathcal{A}$ maps variables in z to conversational objects in $\bar{\mathcal{C}}$, for example the flight legs $fl1$ and $fl2$ being discussed in Figure 3. We will use alignments to match properties of z and $\bar{\mathcal{C}}$. To do this we extend the alignment function $\mathcal{A}$ to apply to properties, for example $\mathcal{A}(\langle from, x, BOS\rangle) = \langle from, \mathcal{A}(x), BOS\rangle$.

Scoring Alignments Finally, we compute the conversation loss $\mathcal{L}_c(z, j, \mathcal{C})$ as follows:

$$\mathcal{L}_c(z, j, \mathcal{C}) = \min_{\mathcal{A}} \sum_{p_u \in P_z} \sum_{p_s \in P_{\bar{\mathcal{C}}}} s(\mathcal{A}(p_u), p_s) \ .$$

The function $s(\mathcal{A}(p_u), p_s) \in \mathbb{R}$ computes the compatibility of the two input properties. It is zero if $\mathcal{A}(p_u) \neq p_s$. Otherwise, it returns $loss(p_s)$.

We approximate the $\min$ computation in $\mathcal{L}_c$ over alignments $\mathcal{A}$ as follows. For a logical form z at position j, we align the outer-most variable to the conversational object in $\bar{\mathcal{C}}$ that is being discussed at j. The remaining variables are aligned greedily to minimize the loss, by selecting a single conversational object for each in turn.

Finally, for each aligned variable, we increase the loss by one for each unmatched property from P_z. This increases the loss of logical forms that include spurious information. However, since a conversation might stop prematurely and therefore won't discuss the entire user request, we only increase the loss for variables that are already aligned. For this purpose, we define an aligned variable to be one that has at least one property matched successfully.

6.2 Domain Loss

We also make use of a domain loss function $\mathcal{L}_d(z) \in \mathbb{R}$. The function takes a logical form z and returns the number of violations there are in z to a set of constraints on logical forms that occur commonly in the dialog domain. For example, in a travel domain, a violation might occur if a flight leg has two different destination cities. The set of possible violations must be specified for each dialog system, but can often be compiled from existing resources, such as a database of valid flight ticketing options.

In our experiments, we will use a set of eight simple constraints to check for violations in flight itineraries, which can have multiple legs. These include, for example, checking that the legs have unique origins and destinations that match across the entire itinerary. For example, in Figure 3 the logical forms (a), (b) and (d) will have no violations; they describe valid flights. Example (c) has a single violation: a flight has two origins. Example (e) violates a more complex constraint: the second flight's origin is different from the first flight's destination.

```
Inputs: Training set $\{(j_i, \mathcal{C}_i) : i = 1 \dots n\}$ where each example includes the index $j_i$ of a sentence $x_i$ in the conversation $\mathcal{C}_i$. Initial lexicon $\Lambda_0$. Number of iterations $T$. Margin $\gamma$. Beam size $k$ for lexicon generation. Loss function $\mathcal{L}(x, j, \mathcal{C})$, as described in Section 6.

Definitions: $GENLEX(x, \mathcal{C})$ takes as input a sentence and a conversation and returns a set of lexical items as described in Section 7. GEN($x; \Lambda$) is the set of all possible CCG parses for $x$ given the lexicon $\Lambda$. $LF(y)$ returns the logical form $z$ at the root of the parse tree $y$. Let $\Phi_i(y)$ be shorthand for the feature function $\Phi(x_i, y)$ defined in Section 5. Define $LEX(y)$ to be the set of lexical entries used in parse $y$. Finally, let $MIN\mathcal{L}_i(Y)$ be $\{y | \forall y' \in Y, \mathcal{L}(LF(y), j_i, \mathcal{C}_i) \leq \mathcal{L}(LF(y'), j_i, \mathcal{C}_i)\}$, the set of minimal loss parses in $Y$.

Algorithm:
  $\theta = \bar{0}$ , $\Lambda = \Lambda_0$
  For $t = 1 \dots T, i = 1 \dots n$ :
    Step 1: (Lexical generation)
      a. Set $\lambda = \Lambda \cup GENLEX(x_i, \mathcal{C}_i)$
      b. Let $Y$ be the $k$ highest scoring parses of $x_i$ using $\lambda$
      c. Select new lexical entries from the lowest loss parses
         $\lambda_i = \bigcup_{y \in MIN\mathcal{L}_i(Y)} \{l | l \in LEX(y)\}$
      d. Set lexicon to $\Lambda = \Lambda \cup \lambda_i$
    Step 2: (Update parameters)
      a. Define $G_i = MIN\mathcal{L}_i(GEN(x_i, \Lambda, \theta))$ and
         $\mathcal{L}_{min}$ to be the minimal loss
      b. Set $B_i = GEN(x_i, \Lambda, \theta) - G_i$
      c. Set the relative loss function: $\Delta_i(y) = \mathcal{L}(y, \mathcal{C}_i) - \mathcal{L}_{min}$
      d. Construct sets of margin violating good and bad parses:
         $R_i = \{r | r \in G_i \wedge$
                $\exists y' \in B_i \ s.t. \ \langle \theta, \Phi_i(r) - \Phi_i(y') \rangle < \gamma \Delta_i(r)\}$
         $E_i = \{e | e \in B_i \wedge$
                $\exists y' \in G_i \ s.t. \ \langle \theta, \Phi_i(y') - \Phi_i(e) \rangle < \gamma \Delta_i(e)\}$
      e. Apply the additive update:
         $\theta = \theta + \sum_{r \in R_i} \frac{1}{|R_i|} \Phi_i(r) - \sum_{e \in E_i} \frac{1}{|E_i|} \Phi_i(e)$

Output: Parameters $\theta$ and lexicon $\Lambda$
```

Figure 4: The learning algorithm.

7 Learning

Figure 4 presents the complete learning algorithm. We assume access to training examples, $\{(j_i, \mathcal{C}_i) : i = 1, \dots, n\}$, where each example includes the index j_i of a sentence x_i in the conversation $\mathcal{C}_i$. The algorithm learns a weighted CCG parser, described in Section 5, including both a lexicon Λ and parameters θ. The approach is online, considering each example in turn and performing two steps: (1) expanding the lexicon and (2) updating the parameters.

Step 1: Lexical Induction We introduce new lexical items by selecting candidates from the function $GENLEX$, following previous work (Zettlemoyer and Collins, 2005; 2007) as reviewed in Section 5.3. However, we face the new challenge that there is no labeled logical-form meaning z. Instead, let $Z_{\bar{\mathcal{C}}}$ be set of all logical forms that appear in system utterances in the relevant conversation segment $\bar{\mathcal{C}}$. We will now define the conversational lexicon set:

$$GENLEX(x, \bar{\mathcal{C}}) = \bigcup_{z \in Z_{\bar{\mathcal{C}}}} GENLEX(x, z)$$

where we use logical forms from system utterances to guess possible CCG categories for analyzing the user utterance. This approach will overgeneralize, when the system talks about things that are unrelated to what the user said, and will also often be incomplete, for example when the system does not repeat parts of the original content. However, it provides a way of guessing lexical items that can be combined with previously learned ones, which can fill in any gaps and help select the best analysis.

Step 1(a) in Figure 4 uses $GENLEX$ to temporarily create a large set of potential categories based on the conversation. Steps (b-d) select a small subset of these entries to add to the current lexicon Λ: we find the k-best parses under the model, rerank them according to loss, find the lexical items used in the best trees, and add them to Λ. This approach favors lexical items that are used in high-scoring but low-loss analyses, as computed given the current model.

Step 2: Parameter Updates Given the loss function $\mathcal{L}(x, i, \mathcal{C})$, we use a variant of a loss-sensitive perceptron to update the parameters (Singh-Miller and Collins, 2007). In Steps (a-c), for the current example i, we compute the relative loss function Δ_i that scales with the loss achieved by the best and worst possible parses under the model. In contrast to previous work, we do not only compute the loss over a fixed n-best list of possible outputs, but instead use the current model score to recompute the options at each update. Then, Steps (d-e) find the set R_i of least loss analyses and E_i of higher-loss candidates whose models scores are not separated by at least $\gamma\Delta_i$, where γ is a margin scale constant. The final update (Step f) is additive and increases the parameters for features indicative of the analyses with less loss while down weighting those for parses that were not sufficiently separated.

Discussion This algorithm uses the conversation to drive learning in two ways: it guides the lexical items that are proposed while also providing the conversational feedback that defines the loss used to update the parameters. The resulting approach is, at every step, using information about how the conversation progressed after a user utterance to reconstruct the meaning of the original statement.

8 Data Sets

For evaluation, we used conversation logs from the Lucent and BBN dialog systems in the DARPA Communicator corpus (Walker et al., 2002). We selected these systems since they provide significant opportunities for learning. They asked relatively open ended questions, allowing for more complex user responses, while also using a number of simple remediating strategies to recover from misunderstandings. The original conversational logs included unannotated transcripts of system and user utterances. Inspired by the speech act labeling approach of Walker and Passonneau (2001), we wrote a set of scripts to label the speech acts and logical forms for system statements. This could be done with high accuracy since the original text was generated with templates. These labels represent what the system explicitly said and do not require complex, potentially error-prone annotation of the full state of the original dialog system. The set of speech acts includes confirmations, information requests, selects, offers, instructions, and a miscellaneous category.

The data sets include a total of 376 conversations, divided into training and testing sets. Table 1 provides details about the training and testing sets, as well as general data set statistics. We developed our system using 4-fold cross validation on the training sets. Although there are approximately 12,000 user

	Lucent	BBN
# Conversations	214	162
Total # of utterances	11,974	12,579
Avg. utterances per conversation	55.95	77.65
Avg. tokens per user utterance	3.24	2.39
Total # of training utterances	208	67
Total # of testing utterances	96	67
Avg. tokens per selected utterance	11.72	9.53

Table 1: Data set statistics for Lucent and BBN systems.

utterances in the data sets, the vast majority are simple, short phrases (such as "yes" or "no") which are not useful for learning a semantic parser. We select user utterances with a small set of heuristics, including a threshold (6 for Lucent, 4 for BBN) on the number of words and requiring that at least one noun phrase is present from our initial lexicon. This approach was manually developed to perform well on the training sets, but is not perfect and does introduce a small amount of noise into the data.

9 Experimental Setup

This section describes our experimental setup and comparisons. We follow the setup of Zettlemoyer and Collins (2007) where possible, including feature design, initialization of the semantic parser, and evaluation metrics, as reviewed below.

Features and Parser The features include indicators for lexical item use, properties of the logical form that is being constructed, and indicators for parsing operators used to build the tree. The parser attempts to boost recall with a two-pass strategy that allows for word skipping if the initial parse fails.

Initialization and Parameters We use an initial lexicon that includes a list of domain-specific noun phrases, such as city and airport names, and a list of domain-independent categories for closed-class words such as "the" and "and". We also used a time and number parser to expand this lexicon for each input sentence with the BIU Number Normalizer.[1] The learning parameters were tuned using the development sets: the margin constant γ is set to 0.5, we use 6 iterations and take the top 30 parses for lexical generation (step 1, figure 4). The parser used for parameter update (step 2, figure 4) has a beam of 250. The parameter vector is initialized to $\bar{0}$.

Evaluation Metrics For evaluation, we measure performance against gold standard labels. We report both the number of exact matches, fully correct logical forms, and a partial-credit number. We measure partial-credit accuracy by mapping logical forms to attribute-value pairs (for example, the expression $from(x, LA)$ will be mapped to $from = LA$) and report precision and recall on attribute sets. This more lenient measure does not test the overall structure of the logical expression, only its components.

Systems We compare performance with the following systems:

Full Supervision: We measured how a fully supervised approach would perform on our data by hand-labeling the training data and using a 0-1 loss function that tests if the output logical form matches the labeled one. For lexicon generation, the labels were used instead of the conversation.

No Conversation Baseline: We also report results for a no conversation baseline. This baseline system is constructed by making two modifications to the full approach. We remove the conversation loss function and apply the GENLEX templates to every possible logical constant, instead of only those in the conversation. This baseline allows us to measure the importance of having access to the conversations by completely ignoring the context for each sentence.

Ablations: In addition to the baseline above, we also do ablation tests by turning off various individual components of the complete algorithm.

10 Results

Table 2 shows exact match results for the development sets, including different system configurations. We report mean results across four folds. To verify their contributions, we include results where we ablate the conversational loss and domain loss functions. Both are essential.

The test results are listed in Table 3. The full method significantly outperforms the baseline, indicating that we are making effective use of the conversational feedback, although we do not yet match the fully supervised result. The poor baseline performance is not surprising, given the difficulty of the task and lack of guidance when the conversations are removed. The partial-credit numbers also demonstrate an empirical trend that we observed; in many

[1] http://www.cs.biu.ac.il/˜nlp/downloads/

Exact Match Metric	Lucent			BBN		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Without conversational loss	0.35	0.34	0.35	0.66	0.54	0.59
Without domain loss	0.42	0.42	0.42	0.69	0.56	0.61
Our Approach	0.63	0.61	0.62	0.77	0.64	0.69
Supervised method	0.76	0.75	0.75	0.81	0.67	0.73

Table 2: Mean exact-match results for cross fold evaluation on the development sets.

Exact Match Metric	Lucent			BBN		
	Prec.	Rec.	F1	Prec.	Rec.	F1
No Conversations Baseline	0	0	0	0.16	0.15	0.15
Our Approach	0.58	0.55	0.56	0.85	0.75	0.79
Supervised method	0.7	0.68	0.69	0.87	0.78	0.82

Partial Credit Metric	Lucent			BBN		
	Prec.	Rec.	F1	Prec.	Rec.	F1
No Conversations Baseline	0.26	0.35	0.29	0.26	0.33	0.29
Our Approach	0.68	0.63	0.65	0.97	0.57	0.72
Supervised method	0.75	0.68	0.72	0.96	0.68	0.79

Table 3: Exact- and partial-match results on the test sets.

cases where we do not produce the correct logical form, the output is often close to correct, with only one or two missed flight constraints.

The difference between the two systems is evident. The BBN system presents a simpler approach to the dialog problem by creating a more constrained conversation. This is done by handling one flight at a time, in the case of flight planing, and posing simple and close ended questions to the user. Such an approach encourages the user to make simpler requests, with relatively few constraints in each request. In contrast, the Lucent system presents a less-constrained approach: interactions start with an open ended prompt and the conversations flow in a more natural, less constrained fashion. BBN's simplified approach makes it easier for learning, giving us superior performance when compared to the Lucent system, despite the smaller training set. This is true for both our approach and supervised learning.

We compared the logical forms recovered by the best conversational model to the labeled ones in the training set. Many of the errors came from cases where the dialog system never fully recovered from confusions in the conversation. For example, the Lucent system almost never understood user utterances that specified flight arrival times. Since it was unable to consistently recover and introduce this constraint, the user would often just recalculate and specify a departure time that would achieve the original goal. This type of failure provides no signal for our learning algorithm, whereas the fully supervised algorithm would use labeled logical forms to resolve the confusion. Interestingly, the test set had more sentences that suffered such failures than the development set, which contributed to the performance gap.

11 Discussion

We presented a loss-driven learning approach that induces the lexicon and parameters of a CCG parser for mapping sentences to logical forms. The loss was defined over the conversational context, without requiring annotation of user utterances meaning.

The overall approach assumes that, in aggregate, the conversations contain sufficient signal (remediations such as clarification, etc.) to learn effectively. In this paper, we satisfied this requirement by using logs from automated systems that deployed reasonably effective recovery strategies. An important area for future work is to consider how this learning can be best integrated into a complete dialog system. This would include designing remediation strategies that allow for the most effective learning and considering how similar techniques could be used simultaneously for other dialog subproblems.

Acknowledgments

The research was supported by funding from the DARPA Computer Science Study Group. Thanks to Dan Weld, Raphael Hoffmann, Jonathan Berant, Hoifung Poon and Mark Yatskar for their suggestions and comments. We also thank Shachar Mirkin for providing access to the BIU Normalizer.

References

Allen, J., M. Manshadi, M. Dzikovska, and M. Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the Workshop on Deep Linguistic Processing*.

Branavan, SRK, H. Chen, L.S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Branavan, SRK, L.S. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: learning to map high-level instructions to commands. In *Proceedings of the Association for Computational Linguistics*.

Carpenter, B. 1997. *Type-Logical Semantics*. The MIT Press.

Chen, D.L., J. Kim, and R.J. Mooney. 2010. Training a multilingual sportscaster: using perceptual context to learn language. *Journal of Artificial Intelligence Research* 37(1):397–436.

Clark, S. and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4):493–552.

Clarke, J., D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Collins, M. 2004. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *New Developments in Parsing Technology*.

Georgila, K., O. Lemon, J. Henderson, and J.D. Moore. 2009. Automatic annotation of context and speech acts for dialogue corpora. *Natural Language Engineering* 15(03):315–353.

Goldwasser, D., R. Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings. of the Association of Computational Linguistics*.

He, Y. and S. Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech and Language* 19:85–106.

He, Y. and S. Young. 2006. Spoken language understanding using the hidden vector state model. *Speech Communication* 48(3-4).

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the Association for Computational Linguistics*.

Kate, R.J. and R.J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the Association for Computational Linguistics*.

Kwiatkowski, T., L.S. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

Lemon, O. 2011. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. *Computer Speech & Language* 25(2):210–221.

Levin, E., R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing* 8(1):11–23.

Liang, P., M.I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Association for Computational Linguistics the International Joint Conference on Natural Language Processing*.

Liang, P., M.I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics*.

Litman, D., J. Moore, M.O. Dzikovska, and E. Farrow. 2009. Using Natural Language Processing to Analyze Tutorial Dialogue Corpora Across Domains Modalities. In *Proceeding of the Conference on Artificial Intelligence in Education*.

Lu, W., H.T. Ng, W.S. Lee, and L.S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matuszek, C., D. Fox, and K. Koscher. 2010. Following directions using statistical machine translation. In *Proceeding of the international conference on Human-robot interaction*.

Miller, S., D. Stallard, R.J. Bobrow, and R.L. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the Association for Computational Linguistics*.

Nguyen, L., A. Shimazu, and X. Phan. 2006. Semantic parsing with structured SVM ensemble classification models. In *Proceedings of the joint conference* *of the International Committee on Computational Linguistics and the Association for Computational Linguistics*.

Papineni, K.A., S. Roukos, and T.R. Ward. 1997. Feature-based language understanding. In *Proceedings of the European Conference on Speech Communication and Technology*.

Poon, H. and P. Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Poon, H. and P. Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the Association for Computational Linguistics*.

Ramaswamy, G.N. and J. Kleindienst. 2000. Hierarchical feature-based translation for scalable natural language understanding. In *Proceedings of the International Conference on Spoken Language Processing*.

Ratnaparkhi, A., S. Roukos, and R.T. Ward. 1994. A maximum entropy model for parsing. In *Proceedings of the International Conference on Spoken Language Processing*.

Ruifang, G. and R.J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Porceedings of the Association for Computational Linguistics*.

Singh, S.P., D.J. Litman, M.J. Kearns, and M.A. Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research* 16(1):105–133.

Singh-Miller, N. and M. Collins. 2007. Trigger-based language modeling using a loss-sensitive perceptron algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Steedman, M. 1996. *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. 2000. *The Syntactic Process*. The MIT Press.

Tang, L.R. and R.J. Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Thompson, C.A. and R.J. Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research* 18:1–44.

Vogel, A. and D. Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Association for Computational Linguistics*.

Walker, M. and R. Passonneau. 2001. DATE: a dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of the First International Conference on Human Language Technology Research*.

Walker, M., A. Rudnicky, R. Prasad, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, et al. 2002. DARPA Communicator: Cross-system results for the 2001 evaluation. In *Proceedings of the International Conference on Spoken Language Processing*.

Wong, Y.W. and R.J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Association for Computational Linguistics*.

Wong, Y.W. and R.J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*.

Young, S., M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language* 24(2):150–174.

Zelle, J.M. and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L.S. and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L.S. and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, L.S. and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.