Jointly Learning to Parse and Perceive: Connecting Natural Language to the Physical World

Jayant Krishnamurthy
Computer Science Department
Carnegie Mellon University
jayantk@cs.cmu.edu

Thomas Kollar
Computer Science Department
Carnegie Mellon University
tkollar@andrew.cmu.edu

Abstract

This paper introduces Logical Semantics with Perception (LSP), a model for grounded language acquisition that learns to map natural language statements to their referents in a physical environment. For example, given an image, LSP can map the statement "blue mug on the table" to the set of image segments showing blue mugs on tables. LSP learns physical representations for both categorical ("blue," "mug") and relational ("on") language, and also learns to compose these representations to produce the referents of entire statements. We further introduce a weakly supervised training procedure that estimates LSP's parameters using annotated referents for entire statements, without annotated referents for individual words or the parse structure of the statement. We perform experiments on two applications: scene understanding and geographical question answering. We find that LSP outperforms existing, less expressive models that cannot represent relational language. We further find that weakly supervised training is competitive with fully supervised training while requiring significantly less annotation effort.

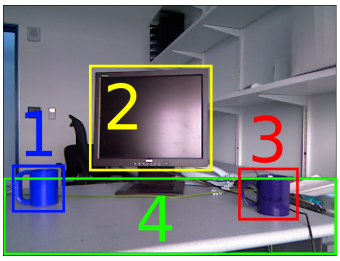
1 Introduction

Learning the mapping from natural language to physical environments is a central problem for natural language semantics. Understanding this mapping is necessary to enable natural language interactions with robots and other embodied systems. For example, for an autonomous robot to understand the sentence "The blue mug is on the table," it must be able to identify (1) the objects in its environment corresponding to "blue mug" and "table," and (2) the objects which participate in the spatial relation denoted by "on." If the robot can successfully identify these objects, it understands the meaning of the sentence.

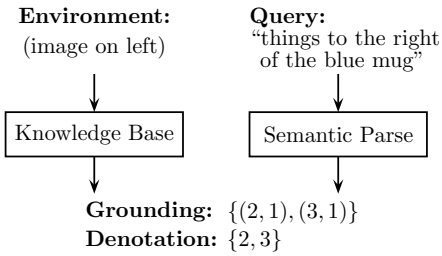
The problem of learning to map from natural language expressions to their referents in an environment is known as *grounded language acquisition*. In embodied settings, environments consist of raw sensor data – for example, an environment could be an image collected from a robot's camera. In such applications, grounded language acquisition has two subproblems: *parsing*, learning the compositional structure of natural language; and *perception*, learning the environmental referents of individual words. Acquiring both kinds of knowledge is necessary to understand novel language in novel environments.

Unfortunately, perception is often ignored in work on language acquisition. Other variants of grounded language acquisition eliminate the need for perception by assuming access to a logical representation of the environment (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Matuszek et al., 2010; Chen and Mooney, 2011; Liang et al., 2011). The existing work which has jointly addressed both parsing and perception has significant drawbacks, including: (1) fully supervised models requiring large amounts of manual annotation and (2) limited semantic representations (Kollar et al., 2010; Tellex et al., 2011; Matuszek et al., 2012).

This paper introduces Logical Semantics with Perception (LSP), a model for grounded language acquisition that jointly learns to semantically parse language and perceive the world. LSP models a mapping from natural language queries to sets of objects in a real-world environment. The input to LSP is an environment containing objects, such as a seg-



(a) An environment containing 4 objects (image segments).



(b) LSP predicting the environmental referents of a natural language query.

Language	Denotation
The mugs	{1, 3}
The objects on the table	{1, 2, 3}
There is an LCD monitor	{2}
Is the blue mug right of the monitor?	{}
The monitor is behind the blue cup.	{2}

(c) Training examples for weakly supervised training.

Figure 1: LSP applied to scene understanding. Given an environment containing a set of objects (left), and a natural language query, LSP produces a semantic parse, logical knowledge base, grounding and denotation (middle), using only language/denotation pairs (right) for training.

mented image (Figure 1a), and a natural language query, such as "the things to the right of the blue mug." Given these inputs, LSP produces (1) a logical knowledge base describing objects and relationships in the environment and (2) a semantic parse of the query capturing its compositional structure. LSP combines these two outputs to produce the query's *grounding*, which is the set of object referents of the query's noun phrases, and its *denotation*, which is the query's answer (Figure 1b).[1] Weakly supervised training estimates parameters for LSP using queries annotated with their denotations in an environment (Figure 1c).

This work has two contributions. The first contribution is LSP, which is more expressive than previous models, representing both one-argument categories and two-argument relations over sets of objects in the environment. The second contribution is a weakly supervised training procedure that estimates LSP's parameters without annotated semantic parses, noun phrase/object mappings, or manually-constructed knowledge bases.

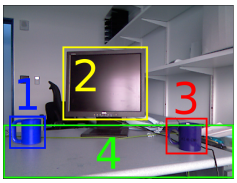
We perform experiments on two different applications. The first application is scene understanding, where LSP grounds descriptions of images in image segments. The second application is geographical question answering, where LSP learns to answer questions about locations, represented as polygons on a map. In geographical question answering, LSP correctly answers 34% more questions than the most comparable state-of-the-art model (Matuszek et al., 2012). In scene understanding, accuracy similarly improves by 16%. Furthermore, weakly supervised training achieves an accuracy within 6% of that achieved by fully supervised training, while requiring significantly less annotation effort.

[1] We treat declarative sentences as if they were queries about their subject, e.g., the denotation of "the mug is on the table" is the set of mugs on tables. Typically, the denotation of a sentence is either true or false; our treatment is strictly more general, as a sentence's denotation is nonempty if and only if the sentence is true.

2 Prior Work

Logical Semantics with Perception (LSP) is related to work from planning, natural language processing, computer vision and robotics. Much of the related work focuses on interpreting natural language using a fixed formal representation. Some work constructs integrated systems which execute plans in response to natural language commands (Winograd, 1970; Hsiao et al., 2003; Roy et al., 2003; Skubic et al., 2004; MacMahon et al., 2006; Levit and Roy, 2007; Kruijff et al., 2007). These systems parse natural language to a formal representation which can be executed using a set of fixed control programs. Similarly, work on semantic parsing learns to map natural language to a given formal representation. Semantic parsers can be trained using sentences annotated with their formal representation (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Kate and Mooney, 2006; Kwiatkowski et al., 2010) or various less restrictive annotations (Clarke et al., 2010; Liang et al., 2011; Krishnamurthy and Mitchell, 2012). Finally, work on grounded language acquisition leverages semantic parsing to map from natural language to a formal representation of an environment (Kate and Mooney, 2007; Chen and Mooney, 2008; Shimizu and Haas, 2009; Matuszek

Environment d



Know. Base Γ

$\texttt{mug}(1)$
$\texttt{mug}(3)$
$\texttt{blue}(1)$
$\texttt{table}(4)$
$\texttt{on-rel}(1,4)$
$\texttt{on-rel}(3,4)$
...

(a) Perception f_{per} produces a logical knowledge base Γ from the environment d using an independent classifier for each category and relation.

Language z

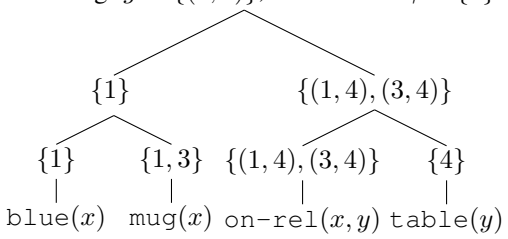
"blue mug on table"

Logical form ℓ

$\lambda x.\exists y.\texttt{blue}(x) \wedge$
$\texttt{mug}(x) \wedge$
$\texttt{on-rel}(x,y) \wedge$
$\texttt{table}(y)$

(b) Semantic parsing f_{prs} maps language z to a logical form ℓ.

Grounding: $g = \{(1,4)\}$, Denotation: $\gamma = \{1\}$



(c) Evaluation f_{eval} evaluates a logical form ℓ on a logical knowledge base Γ to produce a grounding g and denotation γ.

Figure 2: Overview of Logical Semantics with Perception (LSP).

et al., 2010; Dzifcak et al., 2009; Cantrell et al., 2010; Chen and Mooney, 2011). All of this work assumes that the formal environment representation is given, while LSP learns to produce this formal representation from raw sensor input.

Most similar to LSP is work on simultaneously understanding natural language and perceiving the environment. This problem has been addressed in the context of robot direction following (Kollar et al., 2010; Tellex et al., 2011) and visual attribute learning (Matuszek et al., 2012). However, this work is less semantically expressive than LSP and trained using more supervision. The G^3 model (Kollar et al., 2010; Tellex et al., 2011) assumes a one-to-one mapping from noun phrases to entities and is trained using full supervision, while LSP allows one-to-many mappings from noun phrases to entities and can be trained using minimal annotation. Matuszek et al. (2012) learns only one-argument categories ("attributes") and requires a fully supervised initial training phase. In contrast, LSP models two-argument relations and allows for weakly supervised supervised training throughout.

3 Logical Semantics with Perception

Logical Semantics with Perception (LSP) is a model for grounded language acquisition. LSP accepts as input a natural language statement and an environment and outputs the objects in the environment denoted by the statement. The LSP model has three components: perception, parsing and evaluation (see Figure 2). The perception component constructs logical knowledge bases from low-level feature-based representations of environments. The parsing component semantically parses natural language into lambda calculus queries against the constructed knowledge base. Finally, the evaluation component deterministically executes this query against the knowledge base to produce LSP's output.

The output of LSP can be either a *denotation* or a *grounding*. A denotation is the set of entity referents for the phrase as a whole, while a grounding is the set of entity referents for each component of the phrase. The distinction between these two outputs is shown in Figure 1b. In this example, the denotation is the set of "things to the right of the blue mug," which does not include the blue mug itself. On the other hand, the grounding includes both the referents of "things" and "blue mug." Only denotations are used during training, so we ignore groundings in the following model description. However, groundings are used in our evaluation, as they are a more complete description of the model's understanding.

Formally, LSP is a linear model f that predicts a denotation γ given a natural language statement z in an environment d. As shown in Figure 3, the structure of LSP factors into perception (f_{per}), semantic parsing (f_{prs}) and evaluation (f_{eval}) components using several latent variables:

$$f(\gamma, \Gamma, \ell, t, z, d; \theta) = f_{per}(\Gamma, d; \theta_{per}) + f_{prs}(\ell, t, z; \theta_{prs}) + f_{eval}(\gamma, \Gamma, \ell)$$

LSP assumes access to a set of predicates that take either one argument, called categories ($c \in C$) or two arguments, called relations ($r \in R$).[2] These predicates are the interface between LSP's perception and parsing components. The perception function f_{per} takes an environment d and produces a log-

[2]The set of predicates are derived from our training data. See Section 5.3.

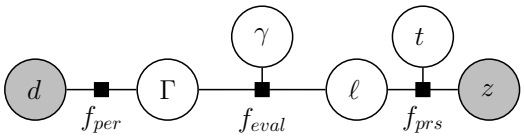


Figure 3: Factor graph of LSP. The environment d and language z are given as input, from which the model predicts a logical knowledge base Γ, logical form ℓ, syntactic tree t and denotation γ.

ical knowledge base Γ that assigns truth values to instances of these predicates using parameters θ_{per}. This function uses an independent classifier to predict the instances of each predicate. The semantic parser f_{prs} takes a natural language statement z and produces a logical form ℓ and syntactic parse t using parameters θ_{prs}. The logical form ℓ is a database query expressed in lambda calculus notation, constructed by logically combining the given predicates. Finally, the evaluation function f_{eval} deterministically evaluates the logical form ℓ on the knowledge base Γ to produce a denotation γ. These components are illustrated in Figure 2.

The following sections describe the perception function (Section 3.1), semantic parser (Section 3.2), evaluation function (Section 3.3), and inference (Section 3.4) in more detail.

3.1 Perception Function

The perception function f_{per} constructs a logical knowledge base Γ given an environment d. The perception function assumes that an environment contains a collection of entities $e \in E_d$. The knowledge base produced by perception is a collection of ground predicate instances using these entities. For example, in Figure 2a, the entire image is the environment, and each image segment is an entity. The logical knowledge base Γ contains the shown predicate instances, where the categories include `blue`, `mug` and `table`, and the relations include `on-rel`.

The perception function scores logical knowledge bases using a set of per-predicate binary classifiers. These classifiers independently assign a score to whether each entity (entity pair) is an element of each category (relation). Let $\gamma^c \in \Gamma$ denote the set of entities which are elements of category c; similarly, let $\gamma^r \in \Gamma$ denote the set of entity pairs which are elements of the relation r. Given these sets, the score of a logical knowledge base Γ factors into per-relation and per-category scores h:

$$f_{per}(\Gamma, d; \theta_{per}) = \sum_{c \in C} h(\gamma^c, d; \theta^c_{per}) + \sum_{r \in R} h(\gamma^r, d; \theta^r_{per})$$

The per-predicate scores are in turn given by a sum of per-element classification scores:

$$h(\gamma^c, d; \theta^c_{per}) = \sum_{e \in E_d} \gamma^c(e)(\theta^c_{per})^T \phi_{cat}(e)$$

$$h(\gamma^r, d; \theta^r_{per}) = \sum_{(e_1,e_2) \in E_d} \gamma^r(e_1, e_2)(\theta^r_{per})^T \phi_{rel}(e_1, e_2)$$

Each term in the above sums represents a single binary classification, determining the score for a single entity (entity pair) belonging to a particular category (relation). We treat γ^c and γ^r as indicator functions for the sets they denote, i.e., $\gamma^c(e) = 1$ for entities e in the set, and 0 otherwise. Similarly, $\gamma^r(e_1, e_2) = 1$ for entity pairs (e_1, e_2) in the set, and 0 otherwise. The features of these classifiers are given by ϕ_{cat} and ϕ_{rel}, which are feature functions that map entities and entity pairs to feature vectors. The parameters of these classifiers are given by θ^c_{per} and θ^r_{per}. The perception parameters θ_{per} contain one such set of parameters for every category and relation, i.e., $\theta_{per} = \{\theta^c_{per} : c \in C\} \cup \{\theta^r_{per} : r \in R\}$.

3.2 Semantic Parser

The goal of semantic parsing is to identify which portions of the input natural language denote entities and relationships between entities in the environment. Semantic parsing accomplishes this goal by mapping from natural language to a logical form that explicitly describes the language's entity referents using one- and two-argument predicates. The logical form is combined with instances of these predicates to produce the statement's denotation.

LSP's semantic parser is defined using Combinatory Categorial Grammar (CCG) (Steedman, 1996). The grammar of the parser is given by a lexicon Λ which maps words to syntactic categories and logical forms. For example, "mug" may have the syntactic category N for noun, and the logical form $\lambda x.\texttt{mug}(x)$, denoting the set of all entities x such that `mug` is true. During parsing, the logical forms for adjacent phrases are combined to produce the logical form for the complete statement.

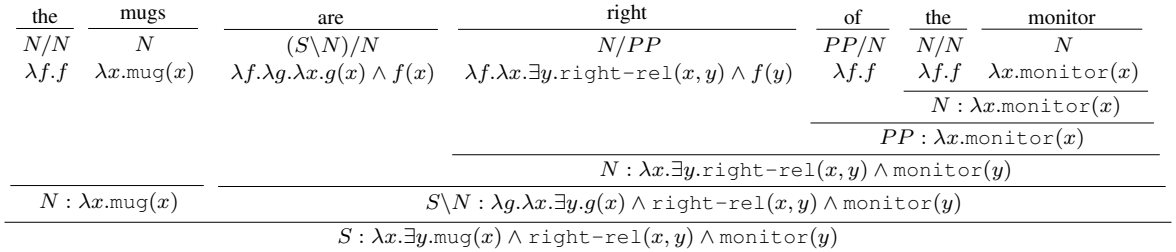


Figure 4: Example parse of "the mugs are right of the monitor." The first row of the derivation retrieves lexical categories from the lexicon, while the remaining rows represent applications of CCG combinators.

Figure 4 illustrates how CCG parsing produces a syntactic tree t and a logical form ℓ. The top row of the parse represents retrieving a lexicon entry for each word. Each successive row combines a pair of entries by applying a logical form to an adjacent argument. A given sentence may have multiple parses like the one shown, using a different set of lexicon entries or a different order of function applications. The semantic parser scores each such parse, learning to distinguish correct and incorrect parses.

The semantic parser in LSP is a linear model over CCG parses (ℓ, t) given language z:

$$f_{prs}(\ell, t, z; \theta_{prs}) = \theta_{prs}^T \phi_{prs}(\ell, t, z)$$

Here, $\phi_{prs}(\ell, t, z)$ represents a feature function mapping CCG parses to vectors of feature values. ϕ_{prs} factorizes according to the tree structure of the CCG parse; it contains features for local parsing operations which are summed to produce the feature values for a tree. If the parse tree is a terminal, then:

$$\phi_{prs}(\ell, t, z) = \mathbb{1}(\text{lexicon entry})$$

The notation $\mathbb{1}(x)$ denotes a vector with a single one entry whose position is determined by x. The terminal features are indicator features for each lexicon entry, as shown in the top row of Figure 4. These features allow the model to learn the correct syntactic and semantic function of each word. If the parse tree is a nonterminal, then:

$$\begin{aligned}\phi_{prs}(\ell, t, z) &= \phi_{prs}(\text{left}(\ell, t, z)) \\ &\quad + \phi_{prs}(\text{right}(\ell, t, z)) + \mathbb{1}(\text{combinator})\end{aligned}$$

These nonterminal features are defined over combinator rules in the parse tree, as in the remaining rows of Figure 4. These features allow the model to learn which adjacent parse trees are likely to combine. We refer the reader to Zettlemoyer and Collins (2005) for more information about CCG semantic parsing.

3.3 Evaluation Function

The evaluation function f_{eval} deterministically scores denotations given a logical form ℓ and a logical knowledge base Γ. Intuitively, the evaluation function simply evaluates the query ℓ on the database Γ to produce a denotation. The evaluation function then assigns score 0 to this denotation, and score $-\infty$ to all other denotations.

We describe f_{eval} by giving a recurrence for computing the denotation γ of a logical form ℓ on a logical knowledge base Γ. This evaluation takes the form of a tree, as in Figure 2c. The base cases are:

- If $\ell = \lambda x.c(x)$ then $\gamma = \gamma^c$.
- If $\ell = \lambda x.\lambda y.r(x, y)$, then $\gamma = \gamma^r$.

The denotations for more complex logical forms are computed recursively by decomposing ℓ according to its logical structure. Our logical forms contain only conjunctions and existential quantifiers; the corresponding recursive computations are:

- If $\ell = \lambda x.\ell_1(x) \wedge \ell_2(x)$, then $\gamma(e) = 1$ iff $\gamma_1(e) = 1 \wedge \gamma_2(e) = 1$.
- If $\ell = \lambda x.\exists y.\ell_1(x, y)$, then $\gamma(e_1) = 1$ iff $\exists e_2.\gamma_1(e_1, e_2) = 1$.

Note that a similar recurrence can be used to compute groundings: simply retain the satisfying assignments to existentially-quantified variables.

3.4 Inference

The basic inference problem in LSP is to predict a denotation γ given language z and an environment d. This inference problem is straightforward due to the deterministic structure of f_{eval}. The highest-scoring γ can be found by independently maximizing f_{prs} and f_{per} to find the highest-scoring logical form ℓ and logical knowledge base Γ. Deterministically evaluating the recurrence for f_{eval} using these values yields the highest-scoring denotation.

Another inference problem occurs during training: identify the highest-scoring logical form and knowledge base which produce a particular denotation. Our approximate inference algorithm for this problem is described in Section 4.2.

4 Weakly Supervised Parameter Estimation

This section describes a weakly supervised training procedure for LSP, which estimates parameters using a corpus of sentences with annotated denotations. The algorithm jointly trains both the parsing and the perception components of LSP to best predict the denotations of the observed training sentences. Our approach trains LSP as a maximum margin Markov network using the stochastic subgradient method. The main difficulty is computing the subgradient, which requires computing values for the model's hidden variables, i.e., the logical knowledge base Γ and semantic parse ℓ that are responsible for the model's prediction.

4.1 Stochastic Subgradient Method

The training procedure trains LSP as a maximum margin Markov network (Taskar et al., 2004), a structured analog of a support vector machine. The training data for our weakly supervised algorithm is a collection $\{(z^i, \gamma^i, d^i)\}_{i=1}^n$, consisting of language z^i paired with its denotation γ^i in environment d^i. Given this data, the parameters $\theta = [\theta_{prs}, \theta_{per}]$ are estimated by minimizing the following objective function:

$$O(\theta) = \frac{\lambda}{2}||\theta||^2 + \frac{1}{n}\left[\sum_{i=1}^{n} \zeta_i\right] \quad (1)$$

where λ is a regularization parameter that controls the trade-off between model complexity and slack penalties. The slack variable ζ_i represents a margin violation penalty for the ith training example, defined as:

$$\zeta_i = \max_{\gamma,\Gamma,\ell,t} \left[f(\gamma, \Gamma, \ell, t, z^i, d^i; \theta) + \text{cost}(\gamma, \gamma^i)\right] - \max_{\Gamma,\ell,t} f(\gamma^i, \Gamma, \ell, t, z^i, d^i; \theta)$$

The above expression is the structured counterpart of the hinge loss, where $\text{cost}(\gamma, \gamma^i)$ is the margin by which γ^i's score must exceed γ's score. We let $\text{cost}(\gamma, \gamma^i)$ be the Hamming cost; it adds a cost of 1 for each entity e such that $\gamma^i(e) \neq \gamma(e)$.

We optimize this objective using the stochastic subgradient method (Ratliff et al., 2006). To compute the subgradient g^i, first compute the highest-scoring assignments to the model's hidden variables:

$$\hat{\gamma}, \hat{\Gamma}, \hat{\ell}, \hat{t} \leftarrow \arg\max_{\gamma,\Gamma,\ell,t} f(\gamma, \Gamma, \ell, t, z^i, d^i; \theta^j) + \text{cost}(\gamma, \gamma^i) \quad (2)$$

$$\Gamma^*, \ell^*, t^* \leftarrow \arg\max_{\Gamma,\ell,t} f(\gamma^i, \Gamma, \ell, t, z^i, d^i; \theta^j) \quad (3)$$

The first set of values (e.g., $\hat{\ell}$) are the best explanation for the denotation $\hat{\gamma}$ which most violates the margin constraint. The second set of values (e.g., ℓ^*) are the best explanation for the true denotation γ^i. The subgradient update increases the weights of features that explain the true denotation, while decreasing the weights of features that explain the denotation violating the margin. The subgradient factors into parsing and perception components: $g^i = [g^i_{prs}, g^i_{per}]$. The parsing subgradient is:

$$g^i_{prs} = \phi_{prs}(\hat{\ell}, \hat{t}, z^i) - \phi_{prs}(\ell^*, t^*, z^i)$$

The subgradient of the perception parameters θ_{per} factors into subgradients of the category and relation classifier parameters. Recall that $\theta_{per} = \{\theta^c_{per} : c \in C\} \cup \{\theta^r_{per} : r \in R\}$. Let $\hat{\gamma}^c \in \hat{\Gamma}$ be the best margin-violating set of entities for c, and $\gamma^{c*} \in \Gamma^*$ be the best truth-explaining set of entities. Similarly define $\hat{\gamma}^r$ and γ^{r*}. The subgradients of the category and relation classifier parameters are:

$$g^{i,c}_{per} = \sum_{e \in E_{d^i}} \left(\hat{\gamma}^c(e) - \gamma^{c*}(e)\right) \phi_{cat}(e)$$

$$g^{i,r}_{per} = \sum_{(e_1,e_2) \in E_{d^i}} \left(\hat{\gamma}^r(e_1, e_2) - \gamma^{r*}(e_1, e_2)\right) \phi_{rel}(e_1, e_2)$$

4.2 Inference: Computing the Subgradient

Solving the maximizations in Equations 2 and 3 is challenging because the weights placed on the denotation γ couple f_{prs} and f_{per}. Due to this coupling, exactly solving these problems requires (1) enumerating all possible logical forms ℓ, and (2) for each logical form, finding the highest-scoring logical knowledge base Γ by propagating the weights on γ back through f_{eval}.

We use a two-step approximate inference algorithm for both maximizations. The first step performs a beam search over CCG parses, producing

k possible logical forms $\ell_1, ..., \ell_k$. The second step uses an integer linear program (ILP) to find the best logical knowledge base Γ given each parse ℓ_i. In our experiments, we parse with a beam size of 1000, then solve an ILP for each of the 10 highest-scoring logical forms. The highest-scoring parse/logical knowledge base pair is the approximate maximizer.

Given a logical form ℓ output by beam search, the second step of inference computes the best values for the logical knowledge base Γ and denotation γ:

$$\max_{\Gamma,\gamma} f_{per}(\Gamma, d; \theta_{per}) + f_{eval}(\gamma, \ell, \Gamma) + \psi(\gamma) \quad (4)$$

Here, $\psi(\gamma) = \sum_{e \in E_d} \psi_e \gamma(e)$ represents a set of weights on the entities in the predicted denotation γ. For Equation 2, ψ represents $\text{cost}(\gamma, \gamma^i)$. For Equation 3, ψ is a hard constraint encoding $\gamma = \gamma^i$ (i.e., $\psi(\gamma) = -\infty$ when $\gamma \neq \gamma^i$ and 0 otherwise).

We encode the maximization in Equation 4 as an ILP. For each category c and relation r, we create binary variables $\gamma^c(e_1)$ and $\gamma^r(e_1, e_2)$ for each entity in the environment, $e_1, e_2 \in E_d$. We similarly create binary variables $\gamma(e)$ for the denotation γ. Using the fact that f_{per} is a linear function of these variables, we write the ILP objective as:

$$f_{per}(\Gamma, d; \theta_{per}) + \psi(\gamma) = \sum_{e_1 \in E_d} \sum_{c \in C} w^c_{e_1} \gamma^c(e_1)$$
$$+ \sum_{e_1, e_2 \in E_d} \sum_{r \in R} w^r_{e_1, e_2} \gamma^r(e_1, e_2) + \sum_{e_1 \in E_d} \psi_{e_1} \gamma(e_1)$$

where the weights $w^c_{e_1}$ and $w^r_{e_1,e_2}$ determine how likely it is that each entity or entity pair belongs to the predicates c and r:

$$w^c_{e_1} = (\theta^c_{per})^T \phi_{cat}(e_1)$$
$$w^r_{e_1,e_2} = (\theta^r_{per})^T \phi_{rel}(e_1, e_2)$$

The ILP also includes constraints and additional auxiliary variables that represent f_{eval}. These constraints couple the denotation γ and the logical knowledge base Γ such that γ is the result of evaluating ℓ on Γ. ℓ is recursively decomposed as in Section 3.3, and each intermediate set of entities in the recurrence is given its own set of $|E_d|$ (or $|E_d|^2$) variables. These variables are then logically constrained to enforce ℓ's structure.

5 Evaluation

Our evaluation performs three major comparisons. First, we examine the performance impact of weakly supervised training by comparing weakly and fully supervised variants of LSP. Second, we examine the performance impact of modelling relations by comparing against a category-only baseline, which is an ablated version of LSP similar to the model of Matuszek et al. (2012). Finally, we examine the causes of errors by performing an error analysis of LSP's semantic parser and perceptual classifiers.

Before describing our results, we first describe some necessary set-up for the experiments. These sections describe the data sets, features, construction of the CCG lexicon, and details of the models. Our data sets and additional evaluation resources are available online from `http://rtw.ml.cmu.edu/tacl2013_lsp/`.

5.1 Data Sets

We evaluate LSP on two applications: scene understanding (SCENE) and geographical question answering (GEOQA). These data sets are collections $\{(z^i, \gamma^i, d^i, \ell^i, \Gamma^i)\}_{i=1}^n$, consisting of a number of natural language statements z^i with annotated denotations γ^i in environments d^i. For fully supervised training, each statement is annotated with a gold standard logical form ℓ^i, and each environment with a gold standard logical knowledge base Γ^i. Statistics of these data sets are shown in Table 1, and example environments and statements are shown in Figure 5.

The SCENE data set consists of segmented images of indoor environments containing a number of ordinary objects such as mugs and monitors. Each image is an environment, and each image segment (bounding box) is an entity. We collected natural language descriptions of each scene from members of our lab and Amazon Mechanical Turk, asking subjects to describe the objects in each scene. The authors then manually annotated the collected statements with their denotations and logical forms. In this data set, each image contains the same set of objects; note that this does not trivialize the task, as the model only observes visual features of the objects, which are not consistent across environments.

The GEOQA data set consists of several maps containing entities such as cities, states, national parks, lakes and oceans. Each map is an environment, and its component entities are given by polygons of latitude/longitude coordinates marking

Data Set Statistics	SCENE	GEOQA
# of environments	15	10
Mean entities / environment d	4.1	6.9
Mean # of entities in denotation γ	1.5	1.2
# of statements	284	263
Mean words / statement	6.3	6.3
Mean predicates / log. form	2.6	2.8
# of preds. in annotated worlds	46	38
Lexicon Statistics		
# of words in lexicon	169	288
# of lexicon entries	712	876
Mean parses / statement	15.0	8.9

Table 1: Statistics of the two data sets used in our evaluation, and of the generated lexicons.

their boundaries.[3] Furthermore, each entity has one known name (e.g., "Greenville"). In this data set, distinct entities occur on average in 1.25 environments; repeated entities are mostly large locations, such as states and oceans. The language for this data set was contributed by members of our research group, who were instructed to provide a mixture of simple and complex geographical questions. The authors then manually annotated each question with a denotation (its answer) and a logical form.

5.2 Features

The features of both applications are intended to capture properties of entities and relations between them. As such, both applications share a set of physical features which are functions of the bounding polygons of each entity. Example category features (ϕ_{cat}) are the area and perimeter of the entity, and an example relation feature (ϕ_{rel}) is the distance between entity centroids.

The SCENE data set additionally includes visual appearance features in ϕ_{cat} to capture visual properties of objects. These features include a Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) and an RGB color histogram.

The GEOQA data set additionally includes distributional semantic features to distinguish between different types of entities (e.g., states vs. lakes) and to capture non-spatial relations (e.g., capitals of states). These features are derived from phrase co-occurrences with entity names in the Clueweb09 web corpus.[4] The category features ϕ_{cat} include indicators for the 20 contexts which most frequently occur with an entity's name (e.g., "X is a city"). Similarly, the relation features ϕ_{rel} include indicators for the 20 most frequent contexts between two entities' names (e.g., "X in eastern Y").

5.3 Lexicon Induction

One of the inputs to the semantic parser (Section 3.2) is a lexicon that lists possible syntactic and semantic functions for each word. Together, these per-word entries define the set of possible logical forms for every statement. Each word may have multiple lexicon entries to capture uncertainty about its meaning. For example, the word "right" may have entries $N : \lambda x.\texttt{right}(x)$ and $N/PP : \lambda f.\lambda x.\exists y.\texttt{right-rel}(x, y) \wedge f(y)$. The semantic parser learns to distinguish among these interpretations to produce good logical forms.

We automatically generated lexicons for both applications using part-of-speech tag heuristics.[5] These heuristics map words to lexicon entries containing category and relation predicates derived from the word's lemma. Nouns and adjectives produce lexicon entries containing either categories or relations (as shown above for "right"). Mapping these parts-of-speech to relations is necessary for phrases like "to the right of," where the noun "right" denotes a relation. Verbs and prepositions produce lexicon entries containing relations. Additional heuristics generate semantically-empty lexicon entries, allowing words like determiners to have no physical interpretation. Finally, there are special heuristics for forms of "to be" and, in GEOQA, to handle known entity names. The complete set of lexicon induction heuristics is available online.

The automatically generated lexicon makes it difficult to compare semantic parses across models, since the correctness of a semantic parse depends on the learned perceptual classifiers. To facilitate such a comparison (Section 5.6), we filtered out lexicon entries containing predicates which were not used in any of the annotated logical forms. Statistics of the filtered lexicons are shown in Table 1.

[3] Polygons were extracted from OpenStreetMap, `http://www.openstreetmap.org/`.

[4] `http://www.lemurproject.org/clueweb09/`

[5] We used the Stanford POS tagger (Toutanova et al., 2003).

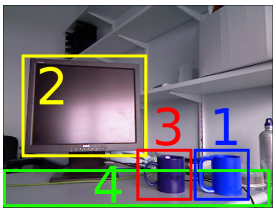
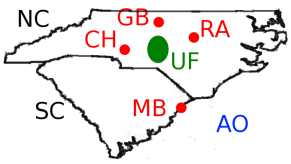
Environment d	Language z and predicted logical form ℓ	Predicted grounding	True grounding
	monitor to the left of the mugs $\lambda x.\exists y.\texttt{monitor}(x) \wedge \texttt{left-rel}(x,y) \wedge \texttt{mug}(y)$	$\{(2,1),(2,3)\}$	$\{(2,1),(2,3)\}$
	mug to the left of the other mug $\lambda x.\exists y.\texttt{mug}(x) \wedge \texttt{left-rel}(x,y) \wedge \texttt{mug}(y)$	$\{(3,1)\}$	$\{(3,1)\}$
	objects on the table $\lambda x.\exists y.\texttt{object}(x) \wedge \texttt{on-rel}(x,y) \wedge \texttt{table}(y)$	$\{(1,4),(2,4)$ $(3,4)\}$	$\{(1,4),(2,4),$ $(3,4)\}$
	two blue cups are placed near to the computer screen $\lambda x.\texttt{blue}(x) \wedge \texttt{cup}(x) \wedge \texttt{comp.}(x) \wedge \texttt{screen}(x)$	$\{(1)\}$	$\{(1,2),(3,2)\}$
	What cities are in North Carolina? $\lambda x.\exists y.\texttt{city}(x) \wedge \texttt{in-rel}(x,y) \wedge y = \texttt{NC}$	$\{(\texttt{CH},\texttt{NC}),(\texttt{GB},\texttt{NC})$ $(\texttt{RA},\texttt{NC})\}$	$\{(\texttt{CH},\texttt{NC}),(\texttt{GB},\texttt{NC})$ $(\texttt{RA},\texttt{NC})\}$
	What city is east of Greensboro in North Carolina? $\lambda x.\exists y,z.\texttt{city}(x) \wedge \texttt{east-rel}(x,y)$ $\wedge\, y = \texttt{GB} \wedge \texttt{in-rel}(y,z) \wedge z = \texttt{NC}$	$\{(\texttt{RA},\texttt{GB},\texttt{NC}),$ $(\texttt{MB},\texttt{GB},\texttt{NC})\}$	$\{(\texttt{RA},\texttt{GB},\texttt{NC})\}$
	What cities are on the ocean? $\lambda x.\exists y.\texttt{city}(x) \wedge \texttt{on-rel}(x,y) \wedge \texttt{ocean}(y)$	$\{(\texttt{CH},\texttt{AO}),(\texttt{GB},\texttt{AO}),$ $(\texttt{MB},\texttt{AO}),(\texttt{RA},\texttt{AO})\}$	$\{(\texttt{MB},\texttt{AO})\}$

Figure 5: Example environments, statements, and model predictions from the SCENE and GEOQA data sets.

5.4 Models and Training

The evaluation compares three models. The first model is LSP-W, which is LSP trained using the weakly supervised algorithm described in Section 4. The second model, LSP-CAT, replicates the model of Matuszek et al. (2012) by restricting LSP to use category predicates. LSP-CAT is constructed by removing all relation predicates in lexicon entries, mapping entries like $\lambda f.\lambda g.\lambda x.\exists y.r(x,y) \wedge g(x) \wedge f(y)$ to $\lambda f.\lambda g.\lambda x.\exists y.g(x) \wedge f(y)$. This model is also trained using our weakly supervised algorithm. The third model, LSP-F, is LSP trained with full supervision, using the manually annotated semantic parses and logical knowledge bases in our data sets. Given these annotations, training LSP amounts to independently training a semantic parser (using sentences with annotated logical forms, $\{(z^i,\ell^i)\}$) and a set of perceptual classifiers (using environments with annotated logical knowledge bases, $\{(d^i,\Gamma^i)\}$). This model measures the performance achievable with LSP given significantly more supervision.

All three variants of LSP were trained using the same hyperparameters. For SCENE, we computed subgradients in 5 example minibatches and performed 100 passes over the data using $\lambda = 0.03$. For GEOQA, we computed subgradients in 8 example minibatches, again performing 100 passes over the data using $\lambda = 0.02$. We tried varying the regularization parameter, but found that performance was relatively stable under $\lambda \leq 0.05$. All experiments use leave-one-environment-out cross-validation to estimate model performance. We hold out each environment in turn, train each model on the remaining environments, then test on the held-out environment.

5.5 Results

We consider two prediction problems in the evaluation. The first problem is to predict the correct denotation γ^i for a statement z^i in an environment d^i. A correct prediction on this task corresponds to a correctly answered question. A weakness of this task is that it is possible to guess the right denotation without fully understanding the language. For example, given a query like "mugs on the table," it might be possible to guess the denotation based solely on "mugs," ignoring "table" altogether. The grounding prediction task corrects for this problem. Here, each model predicts a grounding, which is the set of all satisfying assignments to the variables in a logical form. For example, for the logical form $\lambda x.\exists y.\texttt{left-rel}(x,y) \wedge \texttt{mug}(y)$, the grounding is the set of (x,y) tuples for which both $\texttt{left-rel}(x,y)$ and $\texttt{mug}(y)$ return true. Note that, if the predicted semantic parse is incorrect, the predicted grounding for a statement may contain a different number of variables than the true grounding; such groundings are incorrect. Figure 5 shows model predictions for the grounding task.

Performance on both tasks is measured using exact match accuracy. This metric is the fraction of examples for which the predicted set of entities (be it the denotation or grounding) exactly equals the annotated set. This is a challenging metric, as the

Denotation γ	0 rel.	1 rel.	other	total
LSP-CAT	**0.94**	0.45	**0.20**	0.51
LSP-F	0.89	**0.81**	**0.20**	**0.70**
LSP-W	0.89	0.77	0.16	0.67
Grounding g	0 rel.	1 rel.	other	total
LSP-CAT	**0.94**	0.37	0.00	0.42
LSP-F	0.89	**0.80**	0.00	**0.65**
LSP-W	0.89	0.70	0.00	0.59
% of data	23	56	21	100

(a) Results on the SCENE data set.

Denotation γ	0 rel.	1 rel.	other	total
LSP-CAT	0.22	0.19	0.07	0.17
LSP-F	**0.64**	0.53	**0.21**	0.48
LSP-W	**0.64**	**0.58**	**0.21**	**0.51**
Grounding g	0 rel.	1 rel.	other	total
LSP-CAT	0.22	0.19	0.00	0.16
LSP-F	0.64	0.53	**0.17**	0.47
LSP-W	**0.64**	**0.58**	0.15	**0.50**
% of data	8	72	20	100

(b) Results on the GEOQA data set.

Table 2: Model performance on the SCENE and GEOQA datasets. LSP-CAT is an ablated version of LSP that only learns categories (similar to Matuszek et al. (2012)), LSP-F is LSP trained with annotated semantic parses and logical knowledge bases, and LSP-W is LSP trained on sentences with annotated denotations. Results are separated by the number of relations in each test natural language statement.

number of possible sets grows exponentially in the number of entities in the environment. Say an environment has 5 entities and a logical form has two variables; then there are 2^5 possible denotations and 2^{25} possible groundings. To quantify this difficulty, note that selecting a denotation uniformly at random achieves 6% accuracy on SCENE, and 1% accuracy on GEOQA; selecting a random grounding achieves 1% and 0% accuracy, respectively.

Table 2 shows results for both applications using exact match accuracy. To better understand the performance of each model, we break down performance according to linguistic complexity. We compute the number of relations in the annotated logical form for each statement, and show separate results for 0 and 1 relations. We also include an "other" category to capture sentences with more than one relation (very infrequent), or that include quantifiers, comparatives, or other linguistic phenomena not captured by LSP.

The results from these experiments suggest three conclusions. First, we find that modelling relations is important for both applications, as (1) the majority of examples contain relational language, and (2) LSP-W and LSP-F dramatically outperform LSP-CAT on these examples. The low performance of LSP-CAT suggests that many denotations cannot be predicted from only the first noun phrase in a statement, demonstrating that both applications require an understanding of relations. Second, we find that weakly supervised training and fully supervised training perform similarly, with accuracy differences in the range of 3%-6%. Finally, we find that LSP-W performs similarly on both the denotation and complete grounding tasks; this result suggests that when LSP-W predicts a correct denotation, it does so because it has identified the correct entity referents of each portion of the statement.

5.6 Component Error Analysis

We performed an error analysis of each model component to better understand the causes of errors. Table 3 shows the accuracy of the semantic parser from each trained model. Each held-out sentence z^i is parsed to produce a logical form ℓ, which is marked correct if it exactly matches our manual annotation ℓ^i. A correct logical form implies a correct grounding for the statement when the parse is evaluated in the gold standard logical knowledge base. These results show that both LSP-W and LSP-F have reasonably accurate semantic parsers, given the restrictions on possible logical forms. Common mistakes include missing lexicon entries (e.g., "borders" is POS-tagged as a noun, so the GEOQA lexicon does not include a verb for it) and prepositional phrase attachments (e.g., 6th example in Figure 5).

Table 4 shows the precision and recall of the individual perceptual classifiers. We computed these metrics by comparing each annotated predicate in the held-out environment with the model's predictions for the same predicate, treating each entity (or entity pair) as an independent example for classifi-

	SCENE	GEOQA
LSP-CAT	0.21	0.17
LSP-W	0.72	0.71
LSP-F	**0.73**	**0.75**
Upper Bound	0.79	0.87

Table 3: Accuracy of the semantic parser from each trained model. Upper bound is the highest accuracy achievable without modelling comparatives and other linguistic phenomena not captured by LSP.

cation. Fully supervised training appears to produce better perceptual classifiers than weakly supervised training; however, this result conflicts with the full system evaluation in Table 2, where both systems perform equally well. There are two causes for this phenomenon: uninformative adjectives and unimportant relation instances.

Uninformative adjective predicates are responsible for the low performance of the category classifiers in SCENE. Phrases like "LCD screen" in this domain are annotated with logical forms such as $\lambda x.\texttt{lcd}(x) \wedge \texttt{screen}(x)$. Here, `lcd` is uninformative, since `screen` already denotes a unique object in the environment. Therefore, it is not important to learn an accurate classifier for `lcd`. Weakly supervised training learns that `lcd` is meaningless, yet predicts the correct denotation for $\lambda x.\texttt{lcd}(x) \wedge \texttt{screen}(x)$ using its `screen` classifier.

The discrepancy in relation performance occurs because the relation evaluation weights every relation equally, whereas in reality some relations are more frequent. Furthermore, even within a single relation, each entity pair is not equally important – for example, people tend to ask what is in a state, but not what is in an ocean. To account for these factors, we define a reweighted relation metric using the annotated logical forms containing only one relation, of the form $\lambda x.\exists y.c_1(x) \wedge r(x,y) \wedge c_2(y)$. Using these logical forms, we measure the performance of r on the set of x, y pairs such that $c_1(x) \wedge c_2(y)$, then average this over all examples. Table 4 shows that, using this metric, both training regimes have similar performance. This result suggests that weakly supervised training adapts LSP's relation classifiers to the relation instances which are empirically important for grounding natural language.

	SCENE			GEOQA		
Categories	P	R	F1	P	R	F1
LSP-CAT	0.40	0.86	0.55	0.78	0.25	0.38
LSP-W	0.40	0.84	0.54	0.85	**0.63**	0.73
LSP-F	**0.98**	**0.96**	**0.97**	**0.89**	**0.63**	**0.74**
Relations	P	R	F1	P	R	F1
LSP-W	0.40	0.42	0.41	0.34	**0.51**	0.41
LSP-F	**0.99**	**0.87**	**0.92**	**0.70**	0.46	**0.55**
Relations (rw)	P	R	F1	P	R	F1
LSP-W	**0.98**	**0.98**	**0.98**	0.86	**0.72**	**0.79**
LSP-F	**0.98**	0.95	0.96	**0.89**	0.66	0.76

Table 4: Perceptual classifier performance, measured against the gold standard logical knowledge bases. LSP-CAT is excluded from the relation evaluations, since it does not learn relations. Relations (rw) is the reweighted metric (see text for details).

6 Conclusions

This paper introduces Logical Semantics with Perception (LSP), a model for mapping natural language statements to their referents in a physical environment. LSP jointly models perception and language understanding, simultaneously learning (1) to map from environments to logical knowledge bases containing instances of both one-argument categories and two-argument relations, and (2) to semantically parse natural language. Furthermore, we introduce a weakly supervised training procedure that trains LSP using only sentences with annotated denotations, without annotated semantic parses or noun phrase/entity mappings. An experimental evaluation reveals that this procedure performs nearly as well fully supervised training, while requiring significantly less annotation effort. Our experiments also find that LSP's ability to learn relations improves performance over comparable prior work (Matuszek et al., 2012).

Acknowledgments

This research has been supported in part by DARPA under award FA8750-13-2-0005, and in part by a gift from Google. We also gratefully acknowledge the CMU Read the Web group for assistance with data set construction, and Tom Mitchell, Manuela Veloso, Brendan O'Connor, Felix Duvallet, Robert Fisher and the anonymous reviewers for helpful discussions and comments on the paper.

References

Rehj Cantrell, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu. 2010. Robust spoken instruction understanding for HRI. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction*.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine learning*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. 2009. What to do and how to do it: translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*.

Kai-yuh Hsiao, Nikolaos Mavridis, and Deb Roy. 2003. Coupling perception and simulation: Steps towards conversational robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Proceedings of the 22nd Conference on Artificial Intelligence*.

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction*.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Geert-Jan M. Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. 2007. Situated dialogue and spatial organization: What, where... and why. *International Journal of Advanced Robotic Systems*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Michael Levit and Deb Roy. 2007. Interpretation of spatial language in a map navigation task. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics*.

Matthew MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence*.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. 2010. Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning*.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. 2006. (online) subgradient methods for structured prediction. *Artificial Intelligence and Statistics*.

Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis. 2003. Conversational robots: building blocks for grounding word meaning. In *Proceedings of the HLT-NAACL 2003 Workshop on Learning Word Meaning from Non-linguistic Data*.

Nobuyuki Shimizu and Andrew Haas. 2009. Learning to follow navigational route instructions. In *Proceedings of the 21st international joint conference on artifical intelligence*.

Marjorie Skubic, Dennis Perzanowski, Sam Blisard, Alan Schultz, William Adams, Magda Bugajska, and Derek Brock. 2004. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*.

Mark Steedman. 1996. *Surface Structure and Interpretation*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *Advances in Neural Information Processing Systems*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Terry Winograd. 1970. *Procedures as a representation for data in a computer program for understanding natural language.* Ph.D. thesis, Massachusetts Institute of Technology.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of NAACL.*

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

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.