Incrementally Identifying Objects from Referring Expressions using Spatial Object Models

Gaurav Manek, Stefanie Tellex
Humans to Robots Laboratory, Brown University
gaurav_manek@brown.edu, stefie10@cs.brown.edu

I. Abstract

An important problem in human-robot interaction is that of referring expressions: phrases used to identify a particular object among others. Existing parsing models all operate on entire sentences. Incrementally parsing referring expressions is important for human-robot interaction and conversational feedback. We present a model for parsing real-word referring expressions, trained and tested on human-provided data. In our test corpus, when presented with the entire sentence, our model ranks the correct object as the most likely 60.3% of the time, and ranks the correct object in the top three 79.0% of the time. Given the entire sentence humans identify the correct object 79.0% of the time. With 50%, 80% and 90% of the sentence, the model ranks the correct object as the most likely 17.4%, 27.8%, and 36.2% of the time. Our parser is capable of keeping up with human speech, with 80% of all words processed on commodity hardware within 10ms, and 95% of all words in about 300ms.

II. Introduction

Referring expressions are phrases used to identify a particular object in a scene by describing it and its relative position to other objects. The integration of *social feedback*, where the robot shows its understanding of a human's utterances by generating small responses as it listens to the human, can prompt clarifications from the human and improve the accuracy of referring expression parsing in interactive contexts. However, this requires robots to be able to parse the human input *incrementally*: updating its understanding as each next word is uttered.

In current work, referring expression parsing is done in batch-mode, with the entire referring expression as input. (Tellex et al. 2011; Matuszek et al. 2012; Artzi and Zettlemoyer 2013; Fang, Doering, and Chai 2015) During interactive use, batch-mode requires waiting for the complete utterance before processing and providing output, which introduces unaccceptable latency in the robot's response. For example, practical implementations of the G^3 system, as created by Tellex et al. (2011), can take up to 30 seconds from the end of the input to the start of a response. Our incremental parsing system updates the distribution with each added word, substantially reducing the delay between input and response.

In this paper we present an incremental referring expression parser that can process prepositional phrases. The incremental parser works by using a conditional-random field chunker to add parts of speech tags to sentences. These tags are used to construct a parse tree, which is then evaluated using an object-word model to resolve references to objects and a preposition model to resolve prepositional phrases. A caching method avoids recomputation cost and gives good worst-case time guarantees.

We evaluate our model on novel real-world data and show that it assigns the correct object the highest probability 32.8% of the time and in the top-3 objects 63.5% of the time. In comparison, humans correctly identify the object 79.0% of the time, a unigram model 16.1% of the time and random selection is only correct 7.2% of the time. We also show how incremental performance converges to the above values as more words are available.

Our parser is capable of keeping up with human speech. When run on commodity hardware, 80% of all words in the test set are processed within 10ms, and 95% of all words in about 300ms. A cumulative density function of the 4052 words in the test set is in Figure 6.

In the future, we aim to use this parser with a social feedback model and examine how human-robot interaction is improved using a pick-and-place assay.

III. Related Work

Prepositional phrases have not been subject to as much computational analysis and study as noun- and verb-phrases. The current research on the topic uses referring expressions as parts of larger phrases, such as commands. There is an existing family of related work by Tellex et al. (2011), Matuszek et al. (2012), and Artzi and Zettlemoyer (2013), all of whom present modern models to process referring expressions. These models all operate on entire input sentences and are designed to parse general instructions and commands instead of only prepositional phrases. Tellex et al. (2011) present the G^3 framework. We use several key ideas from this paper: in particular we implicitly assume the binary correspondence variable that their model maximizes. Our algorithm is inspired in part by the algorithm they present. Matuszek et al. (2012) present a state-of-the-art process to learn models for a semantic parser and word-classifier alignment. Our approach is substantially different from Matuszek et al. since we do not separate perceptual features from the language model of each object. Also, in the learning phase of their algorithm, they calculate the marginal probability of a particular grounding and a particular word by performing a beam search over all possible parses. We assume instead that each node in the parse is independent of its sibling nodes, which allows us to use dynamic programming to incrementally build the distribution. Artzi and Zettlemoyer (2013) train Combinatory Categorial Grammars (CCG) with ambiguous validation functions to parse instructions, including spatial relations. While this approach is more flexible and likely performs better on entirely novel sentences, we deliberately choose a simpler model that lends itself to a dynamic programming approach.

Fang, Doering, and Chai (2015) present a model to collaboratively generate a referring expression, incorporating feedback from the human subject to generate additional terms. The paper focuses on the generation of referring expressions and the use of gestural feedback, and so is of limited use in the context of this paper.

IV. Technical Approach

Given a sequence of words $\lambda_1, \lambda_2, \dots, \lambda_t$, we estimate the distribution over the objects, of the objects that the user is referring to as Γ, which is a distribution over ξ (the set of all objects on the table). γ refers to an arbitrary object in that distribution. We evaluate:

$$\operatorname*{arg\,max}_{\gamma \in \xi} \Pr(\Gamma = \gamma | \lambda_1, \lambda_2, \dots, \lambda_t) \tag{IV.1}$$

We assume that we can factor the sequence of words into separate independent constituents $(\Lambda_1, \Lambda_2, \dots, \Lambda_k)$, according to the compositional structure of language (Heim and Kratzer 1998), each of which corresponds to either a grounding (a description of the object, such as "the orange cube") or a prepositional phrase (e.g. "between the ..."). We assume these to be independent, apply Bayes' rule, assume uniform priors, and obtain:

$$\propto \prod_{i=1}^{k} \Pr(\Lambda_i | \Gamma = \gamma) \tag{IV.2}$$

This factorization is done using a chunking algorithm, (McCallum 2002). We assume that the chunkings given here are certain, and so eliminate the probability term associated with that. For speed, we approximate with a chunking which has been shown to give good results in practice.

A. Bottom-up Evaluation

Once we have a semantic tree, we can simplify the tree in a bottom-up manner to obtain the final distribution. Figure 1 illustrates this process and the three possible cases we apply to convert the tree into a distribution over objects.

Case 1 Estimating the distribution of a grounding.

Each grounding in the tree is modeled by a distribution that is obtained from the language model.

$$\Pr(\Lambda_i | \Gamma = \gamma) = \prod_{\lambda \in \Lambda_i} \left(\frac{Q(\gamma, \lambda) + \alpha}{\sum_{\omega} Q(\omega, \lambda) + \alpha \cdot |\xi|} \right) \tag{IV.3}$$

The language model used is a unigram language model, where each word λ refers to object γ with some joint score $Q(\gamma, \lambda)$. In our model, we set $Q(\gamma, \lambda)$ to be the number of times that λ was used to describe γ in our training set. The distribution of simple noun phrase Λ_j referring to object γ is given by Equation IV.3. We use add-alpha smoothing arbitrarily setting $\alpha \approx 5\%$. ξ is the set of all objects, and $|\xi|$ is the number of objects. Our implementation will be able to perform a lookup in $\mathcal{O}(|\Lambda_j|)$ time, where $|\Lambda_j|$ is the number of words in Λ_j.

Case 2 Simplifying a preposition and associated noun-phrases.

We have preposition $p \in P = \{$'near', 'left', 'right', 'front', 'behind', 'between'$\}$. We discuss the selection of these in Section IV-D.

We begin with $\Pr(\Lambda_j | \Gamma = \gamma)$. We use the structure of prepositional phrases to factor Λ_j into a preposition and groundings. $\Lambda_{j,P}$ refers to the subset of Λ_j which describes the preposition (e.g. the word "between" or "near"), and corresponds to P. Each $\Lambda_{j,i}$ refers to the subset of Λ_j which describes the i^{th} object that the preposition references (i.e. the object to which the target is "near") and corresponds to the distribution over all objects $\Gamma_{j,i}$. Let n be the number of such groundings.

We use the factorization from Tellex et al. (2011), and obtain:

$$= \sum_{\gamma_1 \in \xi} \cdots \sum_{\gamma_n \in \xi} \left(\prod_{i=1}^{n} \Pr(\Lambda_{j,i} | \Gamma_{j,i} = \gamma_i) \right) \mathrm{S}\left(\frac{f(\gamma, \gamma_1, \ldots) \cdot \theta_p}{z} \right) \tag{IV.4}$$

The last term relates the groundings, the target object, and the preposition used to the correspondence variable. We parametrize this relationship with feature-vector function f, weights θ_p, logistic function S, and some normalization factor z. We assume we can directly obtain p from $\Lambda_{j,P}$ using a lookup table. θ_p is calculated by performing logistic regression, with the feature-vector functions f detailed in IV-E. We observe that our naïve implementation takes time to the order of $\mathcal{O}(|\xi|^{n+1})$, where $|\xi|$ is the number of objects and n is the number of groundings that each preposition has.

CASE 3 Combining multiple distributions.

In Figure 1, we simplify groundings $Pr(\Lambda_1|\Gamma = \gamma)$ and derived distribution $Pr(\Lambda_2|\Gamma = \gamma)$ to get Γ, our estimated distribution. As established in the initial factorization, we simply take the inner product of all distributions to find the overall distribution. Equation IV.2 is reproduced here:

$$\Pr(\Gamma = \gamma | \lambda_1, \lambda_2, \ldots, \lambda_t) = \prod_{i=1}^{k} \Pr(\Lambda_i | \Gamma = \gamma)$$

The naïve implementation also takes time to the order of $\mathcal{O}(|\xi|^{n+1})$, where $|\xi|$ is the number of objects and n is the number of groundings for which the marginal distribution must be taken.

B. Incremental Parsing Algorithm

The algorithm we present uses the factorization presented above but operates in a bottom-up manner. We represent the referring expression as a tree, updating it each time we receive the next word from the user. This representation allows us to perform computationally-intensive tasks only once and cache intermediate results, allowing us to produce intermediate results without having to recompute them.

More precisely, we construct a semantic tree such that each leaf node in the tree is a noun-phrase that refers to some object. This tree is constructed by chunking the input using conditional random fields and then using deterministic transformations to turn the chunked input into a tree. We convert each of the leaf nodes into distributions over objects using a language model, and then finally evaluate this structure to obtain the final distribution.

C. Chunking and Semantic Tree construction

The algorithm's input is a sequence of words which needs to be transformed to a semantic tree for use with later stages. The first stage in this transformation is to assign a tag to each word that is similar to parts-of-speech tags. Instead of using the full set of English parts of speech, we use a reduced set developed for this application.

We model the relationship between words and tags as a conditional random field, where the tag for any particular word depends on the neighboring words. We directly estimate the distribution of tags using the existing library Mallet, developed by McCallum (2002). The transformation from the tagged sequence to the semantic tree is entirely deterministic, as the tags are tailored to the specific form of the queries in the corpus.

D. Preposition Choice

To select the final set of prepositions $P = \{$'near', 'left', 'right', 'front', 'behind', 'between'$\}$, we performed an online test with humans to identify the prepositions people naturally use when performing such tasks. We gathered 68 prepositions over 58 sentences, and grouped them by meaning. Figure 3 contains the exhaustive list.

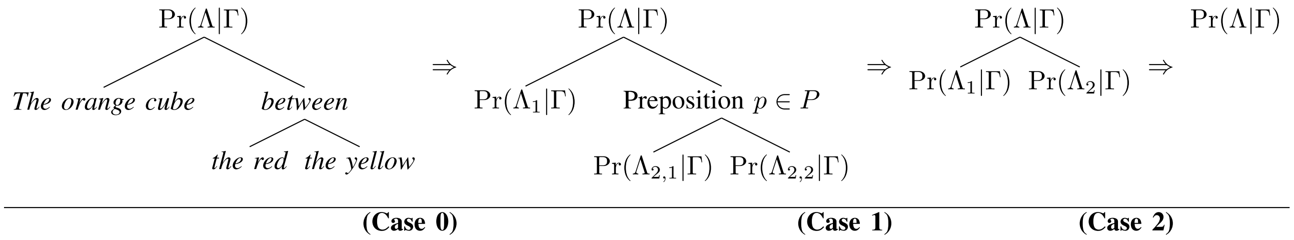


Fig. 1. The three cases of bottom-up evaluation.

33	**between**	between				
18	**near**	closest (to)	next (to)	near	nearer	closer
7	**in front**	in front (of)	below	bottom		
4	**left**	left				
3	**behind**	top (of)	behind			
3	**far**	far (from)	far away	farthest		

Fig. 3. The prepositions people naturally use, characterized by an initial test. The prepositions are grouped by meaning, with the frequency of each group on the left. The exact usage is shown decreasing in frequency from left to right.

From this test, the initial set of prepositions was selected to be {'near', 'far', 'left', 'right', 'front', 'behind', 'between'}, including the preposition "right" as the counterpart to "left". After gathering more data, the lack of data forced us to drop 'far'.

E. Feature-Vector Functions

Before features are computed, all scenes are scaled so that longer axis lies from 0 to 1. The aspect ratio of axes is maintained.

For all one-place prepositions ('near', 'left', 'right', 'front', 'behind'), we use only three features. The target object is located at (x_γ, y_γ), and the object referred to by the preposition is at (x_1, y_1).

1) The difference in the x-coordinate:
$f_1((x_\gamma, y_\gamma), (x_1, y_1)) = x_\gamma - x_1$
2) The difference in the y-coordinate:
$f_2((x_\gamma, y_\gamma), (x_1, y_1)) = y_\gamma - y_1$
3) The Cartesian distance:
$f_3((x_\gamma, y_\gamma), (x_1, y_1)) = \sqrt{(y_\gamma - y_1)^2 (x_\gamma - x_1)^2}$

For the only two-place preposition ('between'), we first draw a line connecting the two grounding objects. We project the point corresponding to the target object onto that line, and compute the distance from the midpoint along the line and perpendicular to the line, scaled by the distance between the two objects. We have three different features, all in terms of the parallel distance p and the perpendicular distance q:

1) The parallel distance: $f_1(p, q) = |p|$
2) The perpendicular distance: $f_2(p, q) = |q|$
3) The smoothed product of the distances:
$f_1(p, q) = (|p| + \epsilon) \cdot (|q| + \epsilon)$, with arbitrary $\epsilon = .1$.

F. Incremental Parsing

In the previous section we factored the simplification of the semantic tree into three separate cases. We cache the result of each simplifying step, as illustrated in Figure 2.

1) Runtime Analysis: We use the runtime analysis of each separate case to draw conclusions about the worst-case time taken for the algorithm to produce an updated distribution given one additional word.

Given the addition of one word, we need to make at most one more recursive simplification than the depth of the tree. In all our training data, the maximum tree depth observed is never more than two, so an upper bound of three simplifications per word input means that this algorithm can easily meet the runtime requirements of online algorithms.

More formally, given the runtimes and caching behavior discussed earlier, the worst-case time to update the distribution Γ to include the next word from the user is $\mathcal{O}(|\Lambda|) \prod_i^{k-1} \mathcal{O}(|\xi|^{n+1}) = \mathcal{O}(|\Lambda| * |\xi|^{kn})$, where k is the number of layers in the tree.

When $k = 3$ and $n \leq 2$, as in real-world examples, this instead evaluates to the very manageable $\mathcal{O}(|\Lambda| * |\xi|^4)$, where $|\Lambda|$ is the number of words in the simple noun phrase, and $|\xi|$ is the number of objects in the scene.

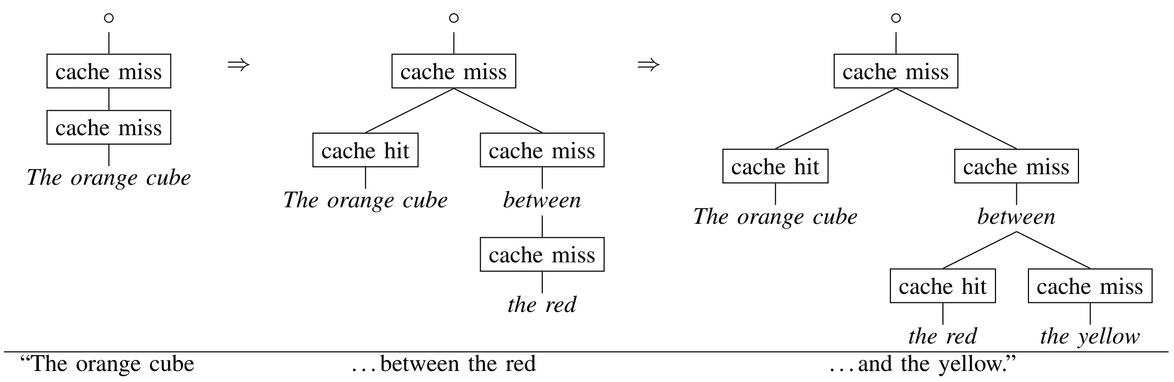


Fig. 2. Cache behavior as words are added.

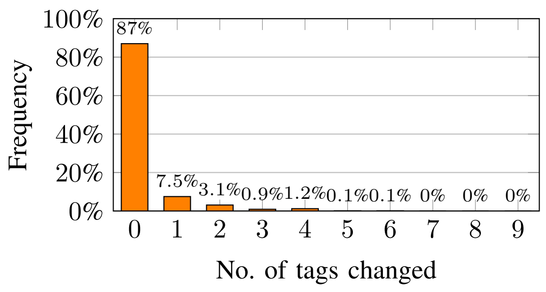


Fig. 4. Distribution of number of tags changed each time the chunker is run with one additional word.

2) Chunking with Conditional Random Fields: The use of conditional random fields complicates the analysis somewhat because adding a word can affect the tags of words already processed, which causes the tree structure to change. In this case, we recompute all nodes in the tree corresponding to words with changed tags, paying the recomputation penalty. Other models such as Hidden Markov Models also have this property.

We found that, in our test set, each time a word is added an average of 0.23 earlier tags are changed. The observed distribution is presented in Figure IV-F2, and is small enough to not significantly increase our runtime.

V. Evaluation

We collected a corpus of human-generated data using Human Intelligence Tasks (HITs) on the Amazon Mechanical Turk (AMT) platform and hand-generated scenes. We additionally use AMT to have humans evaluate our test set to obtain a baseline. We trained the language model and spatial-prepositional model on a training set of 11 scenes, each with an average of 14 objects, for a total of 417 input sentences. We trained the chunker on hand-annotated parses of these input sentences.

A. Data Collection

A set of HITs were created to elicit referring expressions. A total of 19 scenes were constructed, each with a set of about 12 to 15 objects scattered on a table and at least 6 identical orange cubes. For each orange cube in each scene, 9 different workers were told to ask a robot across the table for the indicated orange cube.

For each referring expression in the test set, we get three separate human raters to identify the target and provide feedback on the ease of understanding of the referring expression. Refer to Figure 5 for how often humans correctly identify the target, and Figure 7 for interrater agreement.

B. Results

After training our model on the training set, we tested it using a test set of 10 scenes, each with an average of 14 objects, for a total of 381 input sentences. The result of running the parser on complete referring expressions is in Figure 5.

Figure 5 shows the correctness rate of our algorithm and of three baselines for comparison. The percentage next to each preposition is the fraction of sentences in the test set that contain this preposition, and so will not add up to 100%. The baselines are:

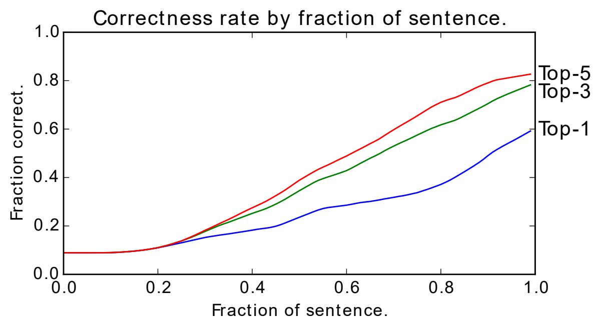
1) The *Human* baseline, which was established by having humans select the object best identified by the referring expression, and scoring them against our corpus.
2) The *Unigram* baseline, which is the expectation of selecting the correct object using a simple unigram object model across the entire input sentence.
3) The *Random* baseline, which is the expectation of selecting the correct object by selecting one object uniformly at random.

The *Results* column lists the rate of correct identification using the entire sentence as input, the *Top-1* column lists the rate at which the correct object is rated the most likely by the algorithm, and the *Top-3* column lists the rate at which the correct object is in the top 3 items. For evaluation, the rate of correct identification is the number of trials in which the algorithm assigns a higher probability to the correct item than to any other item. Should there be a tie, the rate is divided by the number of items of equal probability.

The percentage on the left of each preposition is the fraction of the test set that contains that preposition.

To evaluate incremental performance, we report performance on the test set as a fraction of each sentence provided to the algorithm. Figure 6 reports the evaluation of the test set when our algorithm is run on it word-by-word. Note that, due to the variation in lengths of sentences, the charts are drawn by interpolating fraction of each sentence into bins from 0 to 1. Because of the structure of the algorithm, incremental evaluation converge to the batch-mode results after the final word in the sentence is presented.

Three separate lines are drawn to show the distribution of the rank of the correct option. Each Top-k line includes an example if the target object has at least as much probability as the k^{th}-highest probability in the distribution. Should there be a tie, the rate is divided by the number of items of equal probability.



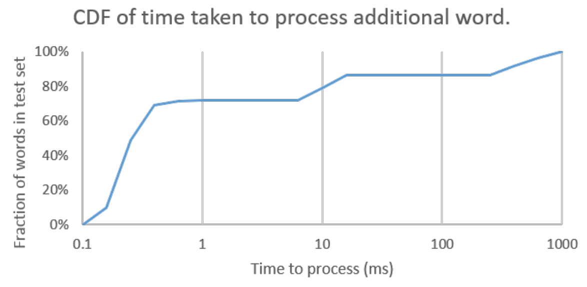


Fig. 6. Performance and time taken on test set.

C. Imprecise and Incorrect Data

A major source of error is the presence of imprecise or incorrect data in our training and test set. An imprecise referring expression is one that does not uniquely identify a target object, and an incorrect referring expression is one that does not identify the correct object. Here we estimate how much of our error is due to imprecise and incorrect referring expressions by evaluating our algorithm on referring expressions with r humans correctly identifying the target object.

Figure 7 shows the outcome of this when we restrict the test set to referring expressions that have been correctly identified by an increasing number of humans. The first column is the control (using all data), the second includes only referring expressions correctly solved by at least one human, then at least two humans, and finally only those by all three humans.

	Rate of correct identification, Test (%)			
r	≥ 0	≥ 1	≥ 2	$= 3$
Top-1	60.3	63.2	65.2	69.6
Top-3	79.0	81.0	82.7	85.1
(%)	100	93.1	84.5	59.3

Fig. 7. Correctness rate for different number of correct raters, with size of data.

From this we see that genuine confusion accounts for between 5-10 percentage points of the total error, which is substantial.

VI. Conclusion

In this paper we have presented an incremental referring expression parser that can process prepositional phrases. The incremental nature of the parser is the key contribution: state-of-the-art parsers all operate on complete referring expressions.

The primary future task is to integrate this with the social feedback framework on Baxter in the H2R lab and conduct user studies to investigate if this provides a measurable improvement to user interaction. Other future work includes developing a model of prepositions that better match human sensibilities. This may even extend to learning the spatial shift between humans and our system and adjusting spatial models to account for that.

From an algorithmic development perspective, a natural extension of this algorithm is to replace the chunking and semantic tree construction with a chart parser. This may even allow the spatial model to inform the parsing, such as by increasing the probability of parses that correspond to narrower distributions. The increased power of the model would be offset by the greater computational overhead: further analysis and experimentation is required to see if this is suitable for use in interactive scenarios.

		Rate of correct identification, Test (%)				
		Baselines			Results	
Preposition		Human	Unigram	Random	Top-1	Top-3
26.8%	*between*	88.2	14.3	7.1	77.5	97.1
21.3%	*near*	82.5	17.2	7.3	64.7	88.1
14.2%	*behind*	76.9	15.7	6.9	81.5	94.4
11.0%	*in front of*	69.9	17.9	7.5	57.1	88.1
9.4%	*left of*	89.8	16.1	7.3	69.4	100.0
5.8%	*right of*	58.0	15.4	7.3	55.2	91.6
	Total	79.0	16.1	7.2	60.3	79.0

Fig. 5. Performance on test set.

REFERENCES

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

Barbu, Andrei, Siddharth Narayanaswamy, and Jeffrey Mark Siskind. 2013. "Saying What You're Looking For: Linguistics Meets Video Search." *CoRR* abs/1309.5174. http://arxiv.org/abs/1309.5174.

Brill, E., and P. Resnik. 1994. "A Rule-Based Approach To Prepositional Phrase Attachment Disambiguation." In *eprint arXiv:cmp-lg/9410026,* 10026. October.

Collins, Michael, and James Brooks. 1995. "Prepositional Phrase Attachment through a Backed-Off Model." *CoRR* abs/cmp-lg/9506021. http://arxiv.org/abs/cmp-lg/9506021.

Fang, Rui, Malcolm Doering, and Joyce Y Chai. 2015. "Embodied Collaborative Referring Expression Generation in Situated Human-Robot Interaction." In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction,* 271–278. ACM.

Heim, Irene, and Angelika Kratzer. 1998. *Semantics in generative grammar.* Vol. 13. Blackwell Oxford.

Liang, Percy, Michael I Jordan, and Dan Klein. 2013. "Learning dependency-based compositional semantics." *Computational Linguistics* 39 (2): 389–446.

Matuszek, Cynthia, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. "A Joint Model of Language and Perception for Grounded Attribute Learning." In *Proc. of the 2012 International Conference on Machine Learning.* Edinburgh, Scotland, June.

McCallum, Andrew Kachites. 2002. "MALLET: A Machine Learning for Language Toolkit." Http://mallet.cs.umass.edu.

Merlo, Paola, Matthew W. Crocker, and Cathy Berthouzoz. 1997. "Attaching Multiple Prepositional Phrases: Generalized Backed-off Estimation." *CoRR* cmp-lg/9710005. http://arxiv.org/abs/cmp-lg/9710005.

Ratnaparkhi, Adwait. 1998. "Statistical Models for Unsupervised Prepositional Phrase Attachment." *CoRR* cmp-lg/9807011. http://arxiv.org/abs/cmp-lg/9807011.

Rudzicz, Frank, and Serguei A. Mokhov. 2003. "Towards a Heuristic Categorization of Prepositional Phrases in English with WordNet." *CoRR* abs/1002.1095. http://arxiv.org/abs/1002.1095.

Tellex, Stefanie, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. "Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation." In *AAAI.*