Incrementally Identifying Objects from Referring Expressions using Spatial Object Models

Gaurav Manek
(Advisor: Prof. Stefanie Tellex)

I. Abstract

An important problem in human-robot interaction is that of referring expressions: phrases used to identify a particular object among others. There are existing models to parse these, but all operate on entire sentences; incrementally parsing referring expressions has applications in 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 identifies the correct object 60.3% of the time, and ranks the correct object in the top three 79.0% of the time. In comparison, humans identify the correct object 79.0% of the time. Incremental performance is also characterized.

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.

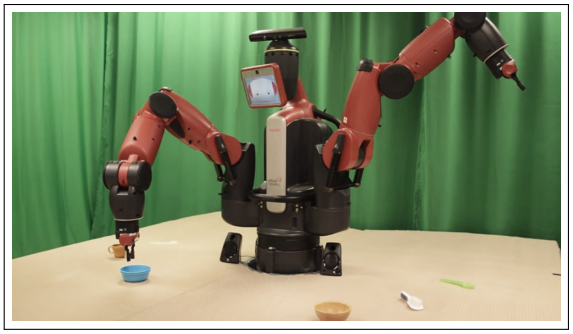
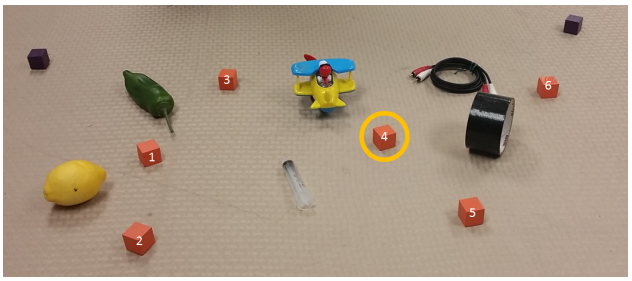


Fig. 1. Baxter Robot in H2R lab, Brown University.

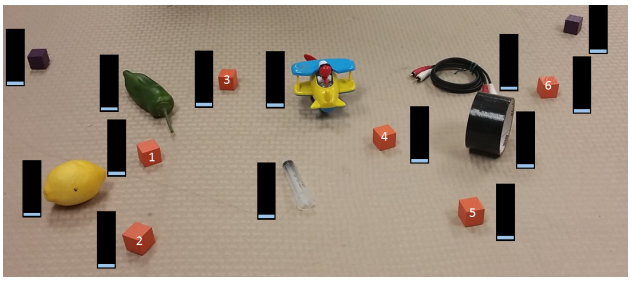
In this paper we present an incremental referring expression parser that can process prepositional phrases. The incremental nature of the parser is the key contribution: state-of-the-art parsers all operate on complete referring expressions.

The 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

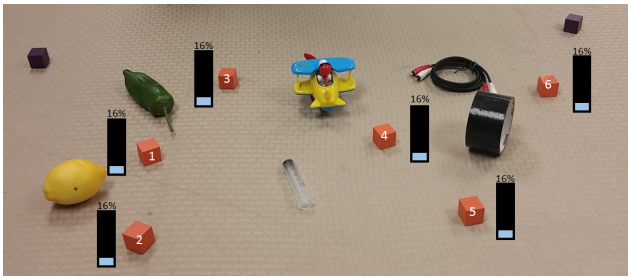


Cube 4 is the target of the referring expression.



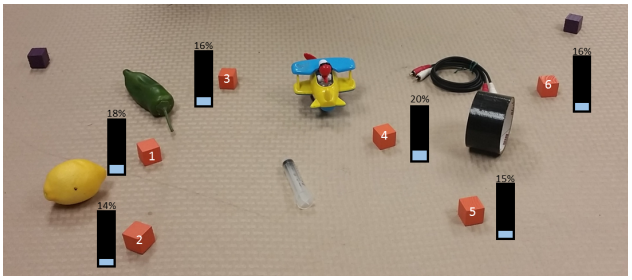
"_"

Without any information, distribution is uniform over all objects.



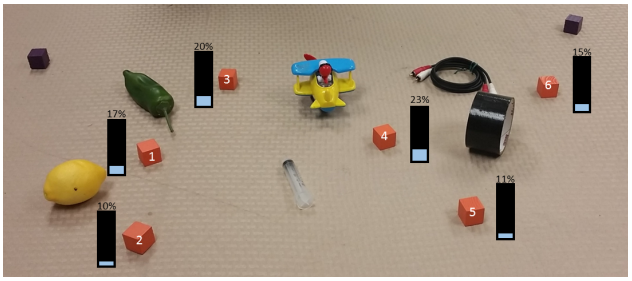
"Orange–"

The first word increases the probability of all cubes and reduces the probability of all non-cubes.



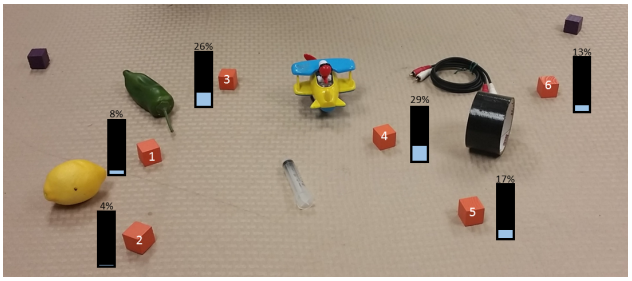
"Orange cube between–"

Symmetry of the distribution is broken by the preposition because some cubes are between more pairs of objects than other cubes. This occurs because we marginalize over all possible pairs of objects that between can be grounded on.



"Orange cube between the toy–"

The direct provision of one of between's groundings shifts the distribution towards cubes that are between the toy plane and other cubes.



"Orange cube between the toy and the tape roll."

At the end of the input, cube 4 has the highest assigned probability.

Fig. 2. Example incremental evaluation – Example #368 from the test set, proceeds in reading order.

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.

Figure 2 shows an example of the incremental parsing of a referring expression. As more words are available to the parser, the parser identifies the target object by assigning it the highest probability.

III. Technical Approach

Given a sequence of words $\lambda_1, \lambda_2, \ldots, \lambda_t$, we estimate the distribution of the objects the user is referring to as Γ. Γ is a distribution over ξ, the set of all objects on the table. In essence, we evaluate:

$$\arg\max_{\gamma \in \xi} \Pr(\Gamma = \gamma | \lambda_1, \lambda_2, \ldots, \lambda_t) \tag{III.1}$$

We borrow the application of correspondence variables from work by Tellex et al. (2011). We reformulate the problem with correspondence variable Φ, which is true if and only if the object γ is correctly identified:

$$\arg\max_{\gamma \in \xi} \Pr(\Phi = \top | \Gamma = \gamma, \lambda_1, \lambda_2, \ldots, \lambda_t) \tag{III.2}$$

We then assume that we can factor the sequence of words into separate independent constituents $(\Lambda_1, \Lambda_2, \ldots, \Lambda_k)$, 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 and factor the expression:

$$\Pr(\Phi = \top | \Gamma = \gamma, \lambda_1, \lambda_2, \ldots, \lambda_t) \tag{III.3}$$

$$= \Pr(\Phi = \top | \Gamma = \gamma, \Lambda_1, \Lambda_2, \ldots, \Lambda_k) \tag{III.4}$$

Applying Bayes' rule:

$$= \frac{\Pr(\Gamma = \gamma, \Lambda_1, \Lambda_2, \ldots, \Lambda_k | \Phi = \top) \cdot \Pr(\Phi = \top)}{\Pr(\Gamma = \gamma, \Lambda_1, \Lambda_2, \ldots, \Lambda_k)} \tag{III.5}$$

We assume a uniform prior on both language and target object, and so eliminate the other terms. The algorithm will perform this calculation for all possible groundings γ, and account for the prior probabilities by normalizing the distribution.

$$\propto \Pr(\Gamma = \gamma, \Lambda_1, \Lambda_2, \ldots, \Lambda_k | \Phi = \top) \tag{III.6}$$

We have assumed that the constitutents are independent, so we separate the terms. Each constituent has its own correspondence variable ϕ_i that is true if and only if that constituent correctly identifies the target object γ.

$$= \prod_{i=1}^{k} \Pr(\Gamma = \gamma, \Lambda_i | \phi_i) \tag{III.7}$$

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. 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. Here is a concrete example:

We attempt to locate the *orange cube* using the following referring expression: "The orange cube between the red thing and the yellow thing." A selection of *red*, *blue*, *orange*, *purple*, *green*, and *yellow* cubes are arranged on a table, as shown in Figure 2. The orange cube referred to in the above statement is pointed to by a red arrow in the figure. The process of estimating the object from this information is in Figure **??**.

B. 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. An example of this transformation is in Figure **??**.

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.

C. 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 3 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.

$$\Pr(\Gamma = \gamma, \Lambda_i | \phi_i) = \prod_{\lambda \in \Lambda_j} \left(\frac{Q(\gamma, \lambda) + \alpha}{\sum_\omega Q(\omega, \lambda) + \alpha \cdot |\xi|} \right) \quad \text{(III.8)}$$

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 III.8. 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'$\}$. The i^{th} grounding that the preposition relies on is modeled by the distributions $\Pr(\Lambda_{j,i} | \Gamma_{j,i})$, obtained from the language model (See Case 0). In this case, we model We find $\Pr(\Lambda_j | \Gamma = \gamma)$ with:

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.

$$\begin{aligned} &\Pr(\Gamma = \gamma, \Lambda_i | \phi_i) \\ &= \Pr(\Gamma = \gamma, \Lambda_{j,P}, \Lambda_{j,1}, \ldots, \Lambda_{j,n} | \phi_i) \qquad \text{(III.9)} \\ &= \sum_{\gamma_1 \in \xi} \cdots \sum_{\gamma_n \in \xi} \\ &\quad \Pr(\Lambda_{j,P}, \Lambda_{j,1}, \ldots, \Lambda_{j,n} | \Gamma = \gamma, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n) \\ &\quad \cdot \Pr(\Gamma = \gamma, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n | \phi_i) \qquad \text{(III.10)} \end{aligned}$$

We assume that each grounding is independent, which allows us to separate each grounding in the first inner term:

$$\begin{aligned} &= \sum_{p \in P} \sum_{\gamma_1 \in \xi} \cdots \sum_{\gamma_n \in \xi} \\ &\quad \Pr(\Lambda_{j,P} | P = p) \cdot \left(\prod_{i=1}^{n} \Pr(\Lambda_{j,i} | \Gamma_{j,i} = \gamma_i) \right) \qquad \text{(III.11)} \\ &\quad \cdot \Pr(P = p, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n | \Gamma = \gamma) \end{aligned}$$

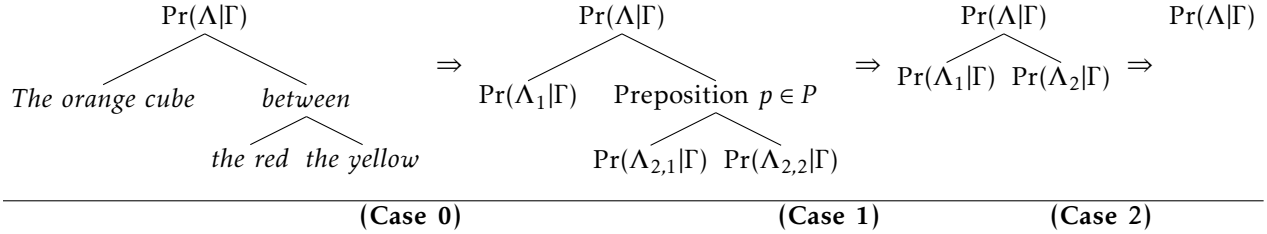


Fig. 3. The three cases of bottom-up evaluation.

We assume that prepositions are certain, and so eliminate the associated term and summation.

$$= \sum_{\gamma_1 \in \xi} \cdots \sum_{\gamma_n \in \xi} \left(\prod_{i=1}^{n} \Pr(\Lambda_{j,i} | \Gamma_{j,i} = \gamma_i) \right) \cdot \Pr(P = p, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n | \Gamma = \gamma) \quad \text{(III.12)}$$

We now consider this remaining inner term, modeling it with the prepositional spatial model:

$$\Pr(P = p, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n | \Gamma = \gamma) \quad \text{(III.13)}$$

Applying Bayes' theorem:

$$= \Pr(\Gamma = \gamma | P = p, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n) \cdot \frac{\Pr(P = p, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n)}{\Pr(\Gamma = \gamma)} \quad \text{(III.14)}$$

Since we have already assumed that prepositions are certain and that the prior probability of all groundings is uniform over all objects, we can simply this to:

$$\propto \Pr(\Gamma = \gamma | P = p, \Gamma_{j,1} = \gamma_1, \ldots, \Gamma_{j,n} = \gamma_n) \quad \text{(III.15)}$$

We parametrize this with feature-vector function f, weights θ_p, logistic function S, and some normalization factor z:

$$= \mathrm{S}\left(\frac{f(\gamma, \gamma_1, \ldots) \cdot \theta_p}{z} \right) \quad \text{(III.16)}$$

We estimate θ_p for each p using logistic regression, with the feature-vector functions detailed in III-D.

We substitute the spatial model in Equation III.16 into Equation III.12, and obtain:

$$\Pr(\Lambda_j | \Gamma = \gamma) \propto \sum_{\gamma_1 \in \xi} \cdots \sum_{\gamma_n \in \xi} \left(\prod_{i=1}^{n} \Pr(\Lambda_{j,i} | \Gamma_{j,i}) \right) \cdot \mathrm{S}\left(\frac{f(\gamma, \gamma_1, \ldots) \cdot \theta_p}{z} \right) \quad \text{(III.17)}$$

We observe that the 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 3, 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 III.7 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.

These cases are sufficient to recursively reduce the tree into a single distribution.

D. 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 product of the distances: $f_1(p, q) = (|p| + \epsilon) \cdot (|q| + \epsilon)$, with arbitrary $\epsilon = .1$.

E. 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 4.

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.

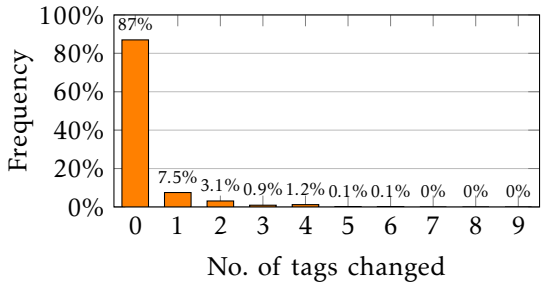


Fig. 5. Distribution of number of tags changed each time the chunker is run with one additional word.

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 choose to use Conditional Random Fields in part because we can explicitly set the look-ahead limit, and so pick a tradeoff between tagging accuracy and potential worst-case performance. For our algorithm, we arbitrarily set it to 5.

The worst-case runtime scales linearly with the look-ahead limit. To characterize the typical performance penalty caused by our choice of look-ahead parameter, we gathered statistics on the likelihood of this happening in our training set.

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 III-E2. The 95$^{\text{th}}$ percentile case is still within the same order of magnitude as the case without any changes.

IV. 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,

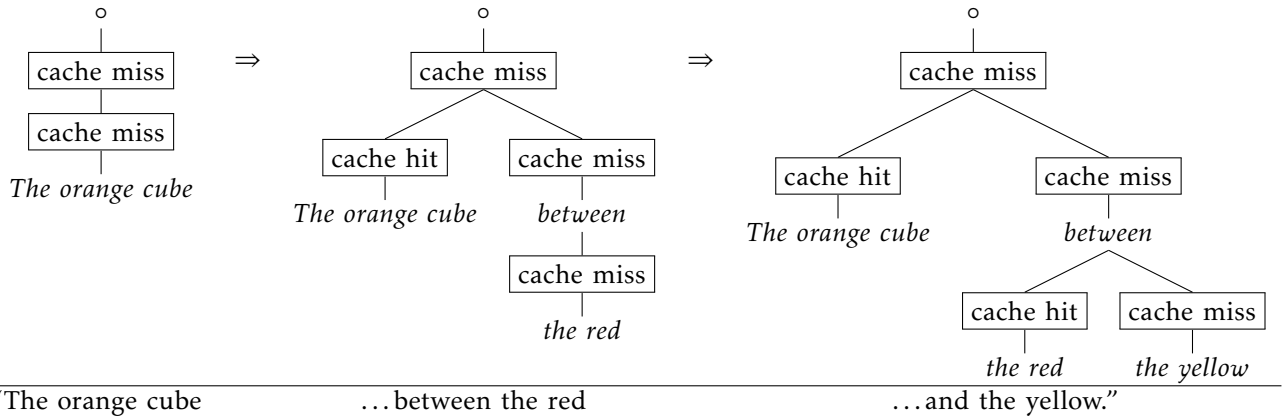
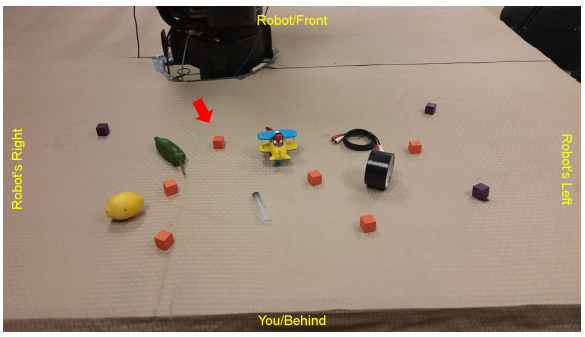


Fig. 4. Cache behavior as words are added.



Examples of responses from AMT workers:
"I want the orange cube in front of you, between the chili and the toy."
"hand me the orange cube that is in between jalapeño and airplane"
"Directly in front of you next to the green pepper and airplane."
"Take the orange block between the toy airplane and green chili pepper"
"bring the orange cube between the pepper and the plane."

Fig. 6. Example picture provided to AMT workers to elicit referring expressions and sample responses.

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. Figure 6 provides an example of such a labeled scene.

We provide the following instructions:

> You are standing across the table from a robot. Write down what you would say to the robot if you wanted the indicated orange cube on the table.
>
> - Use phrases like "between", "near", "left of", "right of", "in front of" and "behind".
> - Use front/behind/left/right from the robot's perspective, as labeled in the image.
> - All orange cubes look the same.
> - You must ask for the item indicated by the red arrow
> - The instruction must be a single sentence.

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 7 for how often humans correctly identify the target, and Figure 9 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 7.

Figure 7 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 8 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.

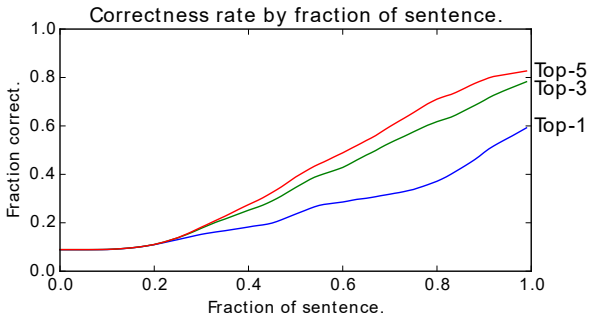


Fig. 8. Performance on test set.

Five 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.

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 9 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. 9. 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.

V. Related Work

Prepositional phrases have not been subject to as much computational analysis and study as noun- and verb-phrases. However, there are still a number of papers related to the topic.

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

Preposition		Rate of correct identification, Test (%)				
		Baselines			Results	
		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. 7. Performance on test set.

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.

In addition to these, we rely on previous work about prepositional phrases:

Collins and Brooks (1995) present a model for disambiguating prepositional phrase attachments. They deal with Noun-Phrases and Verb-Phrases, but their statistical technique may be useful. This concept is also explored by Ratnaparkhi (1998), and Brill and Resnik (1994), each of whom suggest alternative models. Additionally, Merlo, Crocker, and Berthouzoz (1997) explore disambiguating multiple prepositional phrases, rather than a single phrase between multiple targets, both of which are relevant to future work. However, our current model uses sentences focused around a single noun phrase, unlike the general English corpus used in this paper. Additionally, the incremental nature of our parsing requires an alternative framework, and as such not all techniques suggested in these papers can be used.

Another issue we face is the challenge of identifying objects and mapping them to probability distributions via language model. To deal with this, Barbu, Narayanaswamy, and Siskind (2013) present a model for mapping language to object models in video data and Matuszek et al. (2012) describe an approach to the problem of simultaneously observing and extracting representations of a perceived world. These approaches can be adapted to help design features, choose objects, and select language models for prepositional phrase training, rather than using pre-defined objects and locations as we currently do, similar to the work of Tellex et al. (2011) which presents a method to dynamically generate a probabilistic model of a natural language input and perform inferences relating to semantic meaning. This is similar to the work we will preform in semantic parsing, though we will do so incrementally rather than with an entire sentence and thus our work will need to modify these models.

There are still a number of improvements to be made to our machine learning techniques. Rudzicz and Mokhov (2003) give us a framework for parsing and understanding prepositional phrases. Similarly, Liang, Jordan, and Klein (2013) describe a method to create a semantic parser using question-answer pairs as data, rather than requiring annotated sentences, solving the same issue we approach of transforming natural language into semantic meaning. However, while these may be useful for generating a semantic model over entire sentences our technique will be different as we are currently only working with noun phrases, and apply additional information from the spatial model.

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 the preposition 'near' that better matches 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. The chart parser will provide an incremental model of parsing that our spatial model can be directly integrated into. 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.

VII. Acknowledgments

This thesis project is the culmination of work done from Fall 2014 to present. Some work was previously completed in collaboration with undergraduate student Zachary Loery.

This project was completed in the H2R laboratory in the Brown computer science department, with Prof. Stefanie Tellex as advisor.

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*.