Incrementally Identifying Objects from Referring Expressions using Spatial Object Models

Gaurav Manek

I. Introduction

The problem of parsing *referring expressions* is important, but hard. Referring expressions are phrases used to identify a particular object in a scene by describing it and its relative position to other objects in the same scene. 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 this paper, we present an incremental model for parsing referring expressions.

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) Since the robot is presented with the input one word at a time, batch-mode requires waiting for the complete utterance before processing and providing output, which takes time to complete. 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.

Figure I 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.

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.

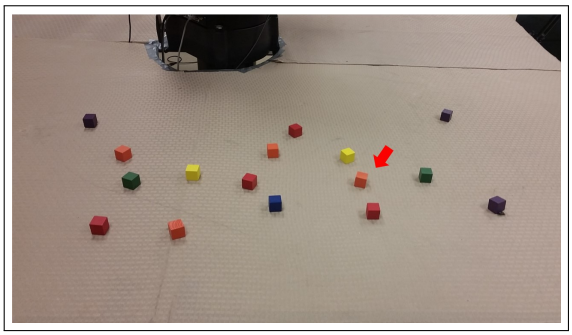


Fig. 1. Sample probability distribution over objects distributed on the table, updated incrementally.

II. Technical Approach

Given a sequence of words $\Lambda_\circ = \lambda_1, \lambda_2, \ldots, \lambda_t$, we estimate Γ, the distribution of the object the user is referring to. Γ is a distribution over ξ, the set of all objects on the table.

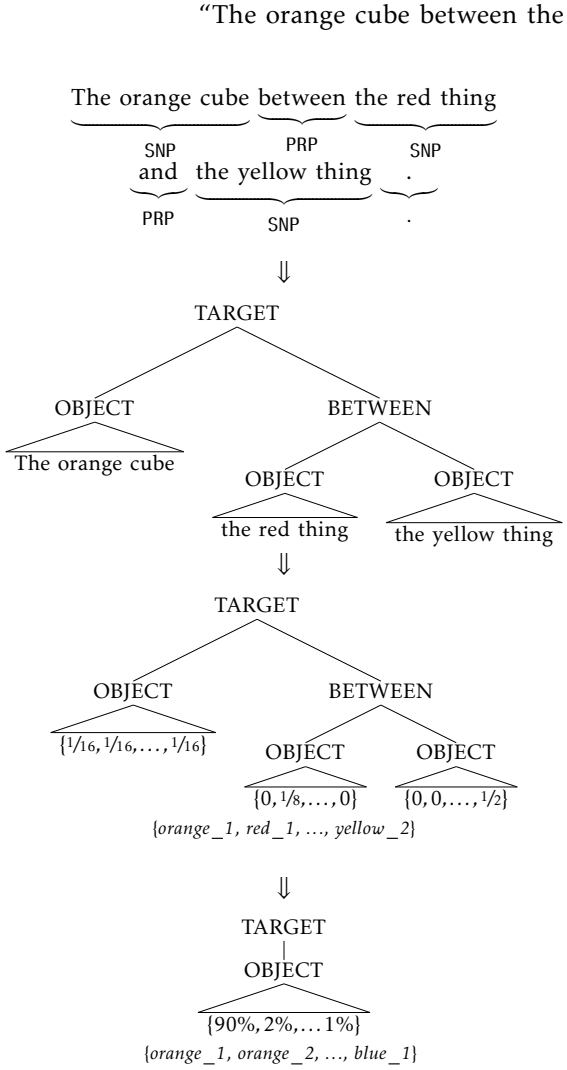
$$\Pr(\Gamma = \gamma | \lambda_1, \lambda_2, \ldots, \lambda_t) \tag{II.1}$$

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 description of the object (e.g. "the orange cube") or a prepositional phrase (e.g. "between the ..."). We assume these to be independent and factor the expression:

$$= \Pr(\Gamma = \gamma | \Lambda_1, \Lambda_2, \ldots, \Lambda_k) \tag{II.2}$$

$$= \Pr(\Gamma_1 = \gamma \wedge \Gamma_2 = \gamma \wedge \ldots | \Lambda_1, \Lambda_2, \ldots, \Lambda_k) \tag{II.3}$$

$$= \prod_{g=1}^{k} \Pr(\Gamma_g = \gamma | \Lambda_g) \tag{II.4}$$



Chunking
First, we use Mallet, (McCallum 2002) a sequence tagging library, to tag the referring expression with a set of tags. The tags were developed to suit our use case, and are a simplification of the typical English parts-of-speech tags.

Semantic Tree
The tagged sequence is then converted to a semantic tree. Because of the simplified tag set in the previous step, this transformation is deterministic.

Language Model
We use a language model to convert each of the simple noun phrases into individual distributions over possible objects.

Bottom-up Evaluation
We evaluate the tree from the bottom upwards, using spatial features and trained weights to convert each input distribution and the type of preposition into a set of output scores. The final result is the object with the highest score.

Fig. 2. The processing steps in our algorithm.

In general, this factorization is done using a chunking algorithm, which we discuss in Section II-B. 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.

Each of these expressions can either be describing the target object or describing the location of the target object using a preposition. In the former case, we use a unigram model (detailed in Section II-C) to estimate this probability.

$$\Pr(\Gamma_i = \gamma | \Lambda_j) \tag{II.5}$$
$$= \frac{\Pr(\Gamma_i = \gamma \cap \Lambda_j)}{\Pr(\Lambda_j)} \tag{II.6}$$
$$\propto \Pr(X_i = o \cap \Lambda_j) \tag{II.7}$$
$$= \prod_{\lambda \in \Lambda_j} \left((1-\alpha) * \frac{Q(\gamma, \lambda)}{\sum_\omega Q(\omega, \lambda)} + \frac{\alpha}{k} \right) \tag{II.8}$$

In the latter case, we convert the expression to a combination of a preposition and at least one other distribution over objects. We parameterize the distribution using a feature-vector generator $f : \Gamma \to \mathbb{R}^{|\theta_p|}$, weights θ_p, normalizing factor z, and sigmoid functon S. The probabilities $\Pr(\Gamma_{i_h} = \gamma_h | \Lambda_j)$ are factored as above. Details of this process are in Section II-D.

$$\Pr(\Gamma_i = \gamma | \Lambda_j) \tag{II.9}$$
$$= \Pr(\Gamma_i = \gamma | P = p, \Gamma_{i_1}, \Gamma_{i_2}, \ldots) \cdot \Pr(P = p, \Gamma_{i_1}, \Gamma_{i_2}, \ldots | \Lambda_j) \tag{II.10}$$
$$\propto S\left(\frac{f(\gamma, \gamma_1, \ldots, \gamma_h, \ldots) \cdot \theta_p}{z} \right) \cdot \prod_h \Pr(\Gamma_{i_h} = \gamma_h | \Lambda_j) \tag{II.11}$$

A. 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 I. 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 2

B. Chunking and Semantic Tree construction

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. Language Model for Objects

The language model used is a unigram language model, where each word λ refers to object γ with some joint score $Q(\gamma, \lambda)$ that we can estimate statistically. The distribution of simple noun phrase Λ_j referring to object γ is given by Equation II.8, reproduced here.

$$\Pr(\Gamma_i = \gamma | \Lambda_j) \propto \prod_{\lambda \in \Lambda_j} \left((1-\alpha) * \frac{Q(\gamma, \lambda)}{\sum_\omega Q(\omega, \lambda)} + \frac{\alpha}{k} \right)$$

For smoothing, we assume that there is some small probability α that the object is uniformly at random chosen without regard to the word. We arbitrarily set $\alpha \approx 5\%$.

D. 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 0 Simplifying noun-phrases into a distribution.

Each grounding in the tree is modeled by a distribution that is obtained from the language model. We have described how this is done in Section II-C. A simple 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 1 Simplifying a preposition and associated noun-phrases.

We have preposition $p \in P = \{$'near', 'left', 'right', 'front', 'behind', 'between'$\}$. Each grounding that the preposition relies on is modeled by the distributions $\Gamma_1, \Gamma_2, \cdots \Gamma_n$, obtained from the language model (See Case 0). Γ is the distribution of the object that the user is referring to. We find Γ with:

$$\Pr(\Gamma = \gamma | \Lambda_j) = \Pr(\Gamma = \gamma | \Gamma_1, \cdots \Gamma_n, P = p, \Lambda_j) \quad \text{(II.12)}$$

Assume that each grounding is independent and factor the expression:

$$\propto \sum_{\gamma_1,\ldots,\gamma_n} \Pr(\Gamma = \gamma \cap P = p \cap \Gamma_1 = \gamma_1, \ldots, \Gamma_n = \gamma_n, \Lambda_j) \quad \text{(II.13)}$$

$$= \sum_{\gamma_1,\ldots,\gamma_n} \Pr(\Gamma = \gamma \cap P = p | \Gamma_1 = \gamma_1, \ldots, \Gamma_n = \gamma_n) \cdot \Pr(\Gamma_1 = \gamma_1, \ldots, \Gamma_n = \gamma_n | \Lambda_j) \quad \text{(II.14)}$$

Parametrize the first term with feature-vector function f, weights θ_p, logistic function S, and some normalization factor z:

$$= \mathrm{S}\left(\frac{f(\gamma, \gamma_1, \ldots, \gamma_n) \cdot \theta_p}{z}\right) \cdot \prod_h \Pr(\Gamma_{i_h} = \gamma_h | \Lambda_j) \quad \text{(II.15)}$$

We can estimate θ_p for each p using logistic regression, and select feature-vector function f separately.

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 2 Combining multiple distributions.

In Figure 3, we simplify groundings Γ_1 and derived distribution Γ_2 to get Γ, our estimated distribution. This case describes the simplification process. In cases where we have multiple distributions, we estimate the true distribution by assuming each of $\Gamma_1, \Gamma_2, \cdots \Gamma_n$ are independent and taking the joint probability, assuming a uniform prior over all objects:

$$\begin{aligned}\Pr(\Gamma = \gamma) &= \Pr(\Gamma = \gamma | \Gamma_1, \Gamma_2, \cdots \Gamma_n) \\ &= \Pr(\Gamma_1 = \gamma, \Gamma_2 = \gamma, \cdots \Gamma_n = \gamma) \\ &= \prod_{i=1}^{n} \Pr(\Gamma_i = \gamma)\end{aligned}$$

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.

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

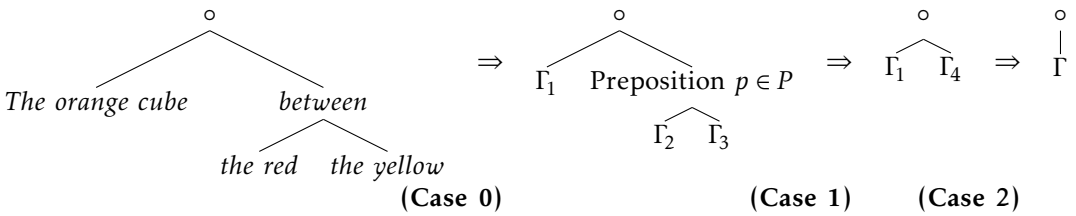


Fig. 3. The three cases of bottom-up evaluation.

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.

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...

Todo Characterize the penalty associated with CRF-retagging leading to recomputation. Some basic probing shows that it isn't much after the first three or four words, but I would like to characterize this better.

F. Technical Approach Conclusion

In this section, we presented a top-down factorization of the problem and mathematical steps for bottom-up evaluation of arbitrary referring expressions. We have also shown that our strategy gives good time guarantees in the worst case scenario, and so is suitable for realtime use.

Now we present the mechanism by which we train various model parameters.

III. Learning Model Parameters

Training each model that we use requires a lot of data. We collected many examples to train and test our system on. We also had humans evaluate our test set to benchmark the performance of our system. 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.

We have 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.

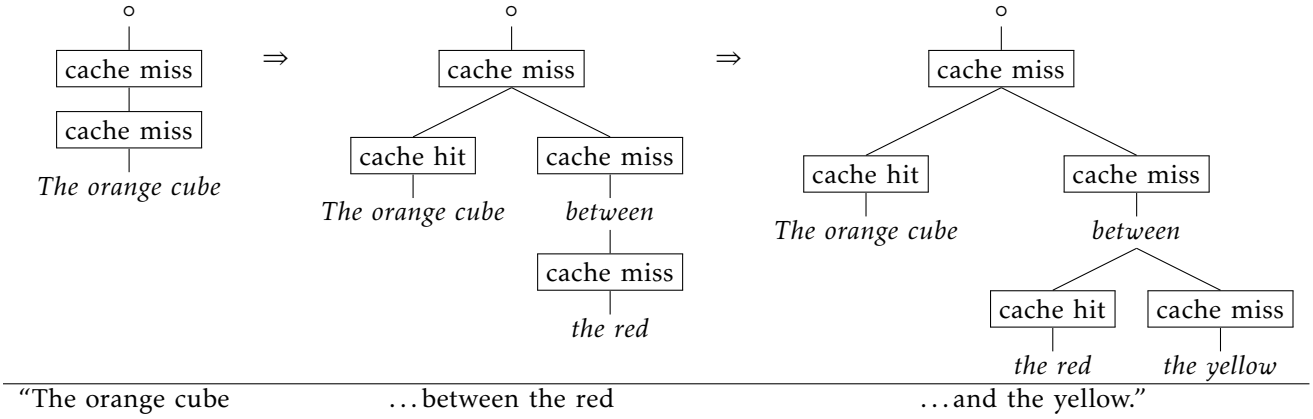


Fig. 4. Cache behavior as words are added.

A. Referring Expression Generation

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 5 provides an example of such a labeled scene.

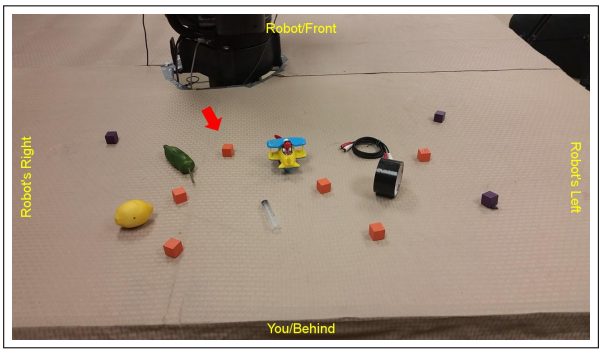


Fig. 5. Example picture provided to AMT workers to elicit referring expressions.

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.

B. Referring Expression Evaluation

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. Figure 6 shows an example scene.

C. Data Gathered

We received referring expressions from human AMT workers. These referring expressions are some

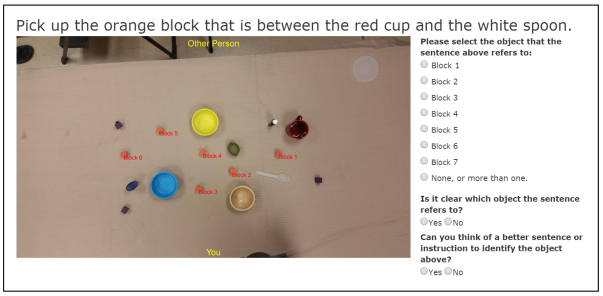


Fig. 6. Example picture provided to AMT workers to evaluate referring expressions.

representative samples of responses to the scene in Figure 5:

- "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."

IV. Current 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

Preposition		Rate of correct identification, Test (%) Baselines Human	Unigram	Random	Results Top-1	Top-3
26.8%	*between*	88.2	14.3	7.1	71.6	94.1
21.3%	*near*	82.5	17.2	7.3	7.9	51.1
14.2%	*behind*	76.9	15.7	6.9	25.9	75.9
11.0%	*in front of*	69.9	17.9	7.5	21.4	64.3
9.4%	*left of*	89.8	16.1	7.3	25.0	83.3
5.8%	*right of*	58.0	15.4	7.3	32.5	55.2
	Total	79.0	16.1	7.2	32.8	63.5

Fig. 7. Performance on test set.

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.

A. Incremental Performance

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.

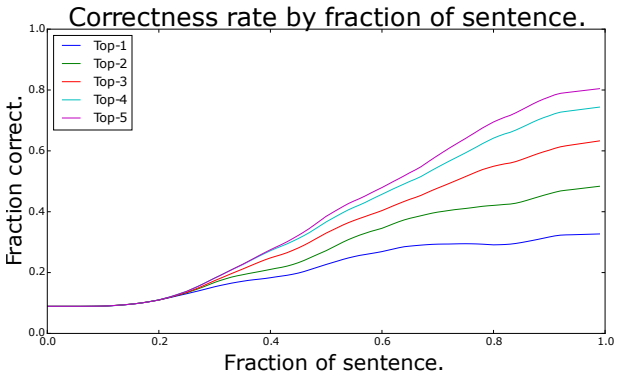


Fig. 8. Performance on test set.

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.

B. Failure Analysis

After testing was complete, we examined individual cases in the test set to characterize failures.

1) Spatial Model of 'near': We observe that the rate of correct identification of our model is particularly low in comparison to that of humans when dealing with the preposition 'near'. For each phrase "[Figure] near [Landmark]" in the training and test sets, we plot the location of the "Landmark" object in comparison to the target "Figure" object in in the training and test sets in Figure 9. It is clear that this simple model does not fully capture the nuances of the use of 'near'.

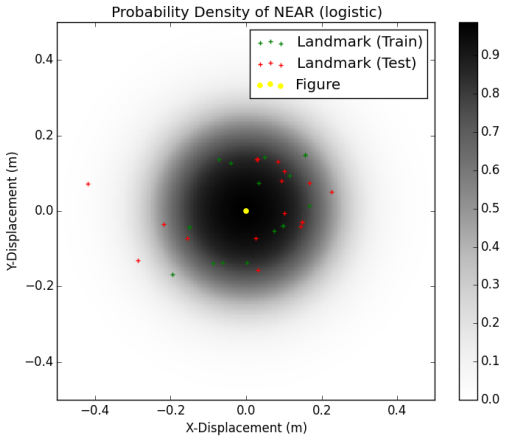


Fig. 9. Relative position of "[Figure] near [Landmark]" in the training and the test sets, with the probability density of the model as background.

Perhaps this model could be extended to include the relative positions of other objects, or to reflect

the local geometry of the scene. We suspect that a major source of error is the difference in perspective of the robot and the human. The human perspective is in front of the table and so is subject to foreshortening, while the robot's perspective does not.

2) Dealing with Imprecise and Incorrect Data: Another 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 10 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	32.8	33.7	34.2	36.9
Top-3	63.5	65.3	65.9	69.6
(%)	100	93.1	84.5	59.3

Fig. 10. Correctness rate for different number of correct raters, with size of data.

From this we see that genuine confusion accounts for between 4-6 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 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, Matuszek et. al. 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.

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*.