Modeling Relational Vocabulary for Grounded Language Acquisition

Caitlin Harding
University of Washington
cehardi@cs.washington.edu

Abstract

Previous work has presented a joint model of language and perception for grounded language acquisition (Matuszek et. al.). This paper will extend that work to incorporate the domain of relational vocabulary. It is reasonable to assume that in many real world settings, it will not be enough for a robotic system to merely learn how to identify objects, but to be able to reason about the relationships between different objects in a scene. This paper outlines the process of collecting data specially designed for this task, annotating that data, and subsequently training a semantic parser for this domain. Experiments will be run to determine how precision and recall of the semantic parser change as more training data is adduced. Finally, future work in this area will be discussed that will allow the relational domain model to be expanded and incorporated with the end-to-end visual perception system described in Matuszek et. al.

1 Introduction

The viability of robotic utilization and assistance in a nonacademic setting is rapidly becoming a reality. Many people are considering the issue of how robots might be best enabled to coexist and work in an environment with people not necessarily trained to work with them. When discussing ways to enable natural interaction with robots, a major topic is grounded language acquisition. Grounded language acquisition describes the process by which a robotic system learns to relate natural language to real world referants that it can perceive with a visual sensor. Grounded language acquisition also allows robotic systems to learn new vocabulary by observing real world objects. In theory a perfect system of grounded language acquisition would allow robotic learning to always be online, making integration with untrained users much easier.

Through a grounded language acquisition approach, a system will learn to connect words or phrases to real world data. For example, given a
picture of a red bowl and the phrase “red bowl”, such a system will connect the word “red” to the object color that it can extract from visual data and the word “bowl” to the shape of the object that it can discern.

Previous work in this area has been limited to learning basic attributes such as color and shape. While this work is good, it cannot handle relational vocabulary. Relational vocabulary makes for a valuable extension to the existing system because it functions as a closer approximation of real speech. However, it also introduces a number of new problems, which will be explored in this paper. The following is an example of the type of extensions modeled in this work:

![Image of objects with a circle indicating a selection]

Figure 1

Prior work:

“Which are the yellow objects?”

Current work:

“Which are the objects at the top?”

2 Previous Work

Much of the work done in this project is designed to extend the joint model of language and perception for grounded language acquisition presented by Matuszek et. al. (2012). In their work, Matuszek et. al. used a data set comprised of simple objects (spheres, cubes, etc.) to set up various scenes. In the end the model they constructed was able to reasonably learn and respond to requests utilizing color and shape attributes of objects. Recently, further work has also been done in modeling relational vocabulary within the joint model of language and perception by Krishnamurthy & Kollar (2013). Work has also been done on grounded language acquisition with unaligned parallel data (Tellex et. al., 2013) and with ambiguous real world data using PCFG (Kim & Mooney, 2012).

Semantic parsing also plays a key role in the production and viability of this system. Artzi & Zettlemoyer (2013) have presented work on semantic parsing for mapping instructions to actions in the real world. Kwiatkowski et. al. have also made significant contributions in lexical generation for CCG parsers (2011).
3 Data Collection and Annotation

Existing data sets related to this work proved insufficient for modeling relational vocabulary. New data was collected that would be better suited for this task.

3.1 Data

The previous work done by Matuszek et. al. (2013) used a dataset composed of simple colored objects such as cubes and spheres. For this step in the process, real world objects that would naturally appear in the contexts where a robotic system might be integrated with people, such as a home, seemed much more appropriate. With that in mind, a set of different colored kitchen objects such as bowls and mugs were acquired. With these objects, a number of basic scenes were set up, swapping out different colored bowls and mugs to get several different versions for each scene. The design of the scenes was done carefully in order to target relational vocabulary. For example, objects were specifically arranged near each other in different formations, allowing for the possibility of relational references for objects.

After setting up and capturing scenes, various important objects or sets of objects were selected and circled for each scene image. The selection of object(s) was also done very carefully in an attempt to elicit relational words from labelers. In the following image, for example, the majority of the objects are uniform in color and shape (red bowls). However, there is a yellow mug on the left edge and a green bowl set between two of the red bowls. If one of the two consecutive red bowls is circled, it is nearly impossible to succinctly label it without referring to the yellow mug or green bowl. This forces labellers to use relational expressions.

![Figure 2](image)

291 images were created from the scene design and object selection process.

Next, all images were posted to Amazon Mechanical Turk as a sentence completion annotation task. Turkers were asked to finish the following sentence in a way that would pick out the circled object(s) in the picture and only that object(s):

“Please pick up the _____”.

Below is an example of Turker responses to an image:
As evidenced above, most pictures were labelled with a variety of different relation types. This image received sentences utilizing relations such as right, next to, in between, in the middle, and adjacent.

10 sentences were collected per image, yielding a total of 2910 natural language labels, a monolith of data that can be used not only for this project, but for various other projects and future work.

3.2 Annotation

In order to scope the amount of work done for this project, a subset of the canonical relations represented in the data were chosen: next to, left, right, front, and behind. Many other canonical relations exist in the data, such as middle, diagonal, and touching. This leaves room for more work to be done in the future expanding the types of relations that this system is able to handle.

For the subset of canonical relations selected above, a total of almost 300 sentences were sampled randomly without replacement (10% of the corpus). From there each sentence was annotated with logical forms expressed in lambda calculus.

3.2.1 Lambda Calculus

Lambda calculus is a formalism that allows for expressing computation in a logical form. It is based on the concepts of function abstraction and variable binding. This work uses a typed version of lambda calculus, where the types consist of e for entities and t for truth values, and each function has an associated type. For example, the expression $\lambda x.\text{bowl}(x)$ takes an entity $e$ and returns a truth value $t$, thus having the function type $<e,t>$.

3.2.2 Complexities

Several complexities came up throughout the task of data annotation. The majority of these are summed up below.

- Synonymity, subsumption, and constructional homonymity

Some phrases may contain different words, but ultimately map to the same logical form. Conversely, some phrases many contain
the same words, but be represented with different logical forms. The decision of what words or phrases should be considered equivalent is not always clear.

Synonymity:
Synonymity refers to two different words mapping to the same logical form/function:

**Figure 4**
“the two yellow bowls next to the red bowl”
“the two yellow bowls adjacent to the red bowl”

While many synonyms are easy to distinguish, determining exact logical equivalence must still be done with care. With the above example, thought must be put into all possible uses of “next to” and “adjacent”, and whether or not any of those occurrences might yield noticeable semantic variation.

Subsumption:
Subsumption comes up often in determining which words or phrases should be treated as synonyms. Consider the following two natural language sentences for **Figure 5**.

“the red bowl and the yellow bowl next to it”
“the red bowl and the yellow bowl to the right of it”

In this example “next to” and “to the right of” denote the same type of relationship and both are valid labels for the given picture. However, “next to” and “to the right of” do not always apply to the same arrangements of objects; there may be, for example, another scene where “next to” and “to the left of” appear synonymous. This is because “next to” subsumes both “right” and “left” relationships. As a result, all three must be modeled with separate
Constructional homonymity:
There is also a high occurrence of phrases that use the same words on the surface, but will ultimately map to different logical forms. This is an example of constructional homonymity. The following two natural language sentences correspond to Figure 5 above.

“the red bowl and the yellow bowl to the right of it”
→ \( \lambda x.\lambda y.\text{right}(x, y) \)

“the red bowl and the yellow bowl on the right”
→ \( \lambda x.\lambda y.\text{right}(x, \text{plane}) \land \text{right}(y, \text{plane}) \)

While both appear to represent a right relation, the first example expresses the relationship between two objects, while the second one expresses the relationship between two objects and the rest of the objects in the scene. As a result, the logical forms for both will differ.

* Semantically empty words

Some words in a natural language sentence might not contribute anything semantically, and as such do not need to be mapped to any function.

In this example, “directly” does not convey any additional information; that is, the label “the red bowl next to the green one” would select the exact same object. To model this, the word “directly” is left out of the logical form and will not have a function mapped to it for semantic parsing. The logical form is represented below:

\( \lambda x.\text{red}(x) \land \text{bowl}(x) \land \text{next_to}(x, (\lambda y.\text{green}(y) \land \text{misc}(y))) \)

However, determining what words are semantically empty is by no means a clear cut task. People may disagree, for example, on whether or not “directly” in the following label for the same image above is also semantically empty.

“the red bowl directly to the right of the green one”
As a result, the selection of semantically empty words must be done with care.

- Coreference Resolution

Coreference resolution is the process by which referring expressions such as *it* are semantically linked to their antecedents. Coreference resolution can be somewhat straightforward when a sentence contains only one possible referent. However, more complex sentences with multiple possible referents become more difficult to parse. Given the following sentence, it is difficult even for a native speaker to determine which noun “*it*” is referring to:

“the close yellow bowl, the cup next to it and the green bowl next to it

The addition of the following image helps a human observer, but the disambiguation task remains difficult for an automated system.

Figure 7

The problem of coreference resolution is not specific to this work, but rather is well studied in the field of computational linguistics. Due to the fact that it is difficult to implement and beyond the scope of this project, it is currently not included in the model.

- Superlatives

Superlatives are well represented in the collected data set. However, they prove difficult to translate into something that can be reasonably learned by a semantic parser. Superlatives typically involve comparison of one or more objects against all other objects in a scene, usually looking for an *argmax* or *argmin*. The following is a simple example of a superlative label and what sort of logic is represented therein:
Figure 8

“the two bowls furthest on the right”
red bowl: ∃p.∀q.¬right(q,p)
yellow bowl: ∃x.∀y.∀z.right(y,x) ∧ ¬right(z,y)

Exactly how this should be modeled in the system remains a bit unclear, and so superlatives are currently omitted.

4 Semantic Parsing

The next step after data collection and annotation is semantic parsing. Semantic parsing is done utilizing Combinatory Categorial Grammar (CCG), Artzi’s Semantic Parsing Framework (SPF), and a specific modification to SPF that leverages the GENLEX system for learning factored lexicons given labelled data and templates. Each of these components will be discussed below.

4.0.3 CCG

Combinatory Categorial Grammar, or CCG, is a grammar formalism used for parsing sentences to logical forms. The space of valid parses given a natural language sentence is constrained by a lexicon and a set of combinators. The lexicon consists of words paired with categories, where categories express both syntactic and semantic information. A lexical entry for bowl would look like the following, where bowl is syntactically a noun (N) having the semantics λx.bowl(x):

\[
\text{bowl} \vdash N : \lambda x.\text{bowl}(x)
\]

Combinators define how lexical entries in a phrase can be combined to produce a final parse, where a final parse consists of a semantic label for the entire phrase. Specifically, a combinator specifies which syntactic categories can combine, and what the resulting category is:

\[
\text{red} \vdash N/N : \lambda x.\text{red}(x)
\]

The / specifies that red combines with a function appearing on its right side. The N on the right is the syntactic category that red can combine with, while the N on the left states that a combination will produce a N. Essentially the entry above states that red looks to the right for a N to combine with, producing in turn another N. red might
combine with the lexical entry for bowl given above, producing the phrase red bowl, which is a N with semantics $\lambda x.\text{red}(x) \land \text{bowl}(x)$.

CCG also allows for a set of type raising rules, which can apply to transform the syntactic category of a function. For example, the syntactic type $ADJ$ can be type raised to the type $N/N$. Thus the lexical entry for red given above would actually look like the following:

\[
\text{red} \vdash ADJ : \lambda x.\text{red}(x)
\]

At parse time, red would be raised from $ADJ$ to $N/N$, allowing it to combine with the lexical entry for bowl given above to yield $N : \lambda x.\text{red}(x) \land \text{bowl}(x)$.

Additionally, this work uses Probabilistic CCG (PCCG) to model inherent ambiguity in natural language. PCCG enables parse scoring for selecting between multiple possible parses for ambiguous sentences or phrases.

### 4.1 SPF

Semantic parsing is done by training a parser on the annotated data discussed above using Artzi’s Semantic Parsing Framework (SPF). SPF bundles together several different algorithms and approaches to semantic parsing. SPF requires hand constructed metadata in order to train a semantic parser. This metadata is discussed below.

- Training and test data: the training and test data files contain a number of randomly selected natural language sentences, along with their logical forms expressed in lambda calculus. Logical forms are collapsed in a LISP-style syntax for consumption. An example of this is given here:

  the green bowl next to the yellow bowl
  (the :<< e,t >,e > (lambda $0 : e
  (and :< t*,t > (color :< color,< e,t >>
  green : color $0) (shape :< shape,< e,t >>
  bowl : shape $0) (next_to :< e,< e,t >> $0 (the :<< e,t >,e >
  (lambda $1 : e (and :< t*,t >
  (color :< color,< e,t >> yellow : color $1)
  (shape :< shape,< e,t >> bowl : shape $1))))))))

- Constants: the constants file lists all logical constants for the given domain. Logical constants appear with the constant name and their type:

  red : color
  bowl : shape

- Types: the types file introduces an additional type system for the logical constant types. Logical constant types are grouped together under metatypes:

  attr // attribute type
• Predicates: the predicates file specifies all functions for the given domain:
  \[ \text{shape} : <\text{shape}, <e,t>> \]
  \[ \text{next_to} : <e, <e,t>> \]

• Lexicon

  Lexemes: the lexemes file contains a list of all lexemes appearing in the given domain, along with their semantics. Manually creating this file can be cumbersome, as it is likely to be quite large. Lexical entries are specified in the following way:
  \[ \text{[red]} = [\text{color} : <\text{color}, <e,t>>, \text{red} : \text{color}] \]

  Lexeme refers to the word itself, whereas lexical item refers to the coupling of lexeme with logical information. One lexeme could have multiple entries with differing logical information in the lexemes file. This corresponds to the fact that in natural language, one word can function in multiple different ways, both syntactically and semantically.

  Templates: the templates file is required for generating the lexemes file using GENLEX, which will be discussed in the next subsection. Templates are parameterized, where variable slots will be filled by words occurring in a given natural language sentence. The following template could be used to generate the lexical entry for \text{red} that was given above:

  \[ [\#0 < \text{attr}, <e,t>> : <\text{attr}, <e,t>>, \#1\text{attr} : \text{attr}] \rightarrow A.D.J : (\lambda \text{e} (\#0 < \text{attr}, <e,t>> : <\text{attr}, <e,t>>, \#1\text{attr} : \text{attr} \text{e})) \]

  Parameters to the template are designated with \#?, where variables in the actual logical form are specified with the same $ used in the annotations for sentences.

  The constants, predicates, types, and lexicon are used for learning lexemes and generating parses of sentences in SPF.

4.2 GENLEX

Instead of manually creating a lexemes file, lexemes can be learned from labeled data given a set of templates. This is the principle behind the GENLEX version of SPF. At training time, GENLEX will take the cross product of all words in a given sentence with all templates in the templates file to produce candidate lexical items. If any of those lexical items can be used to generate valid parses for the sentence, those lexical items are added to the lexicon. In this way learning the lexicon and training the parser occur at the same time and eliminate the need for a manually specified list of lexemes; however, preci-
sion and recall may decrease as a result of incorrect lexeme learning.

4.3 Semantic Parser Training

The FUBL algorithm is applied to the annotated data described above to train a semantic parser. FUBL, as opposed to UBL, refers to the introduction of GENLEX to the semantic parser training. Parsing is done with a CKY-style parser with parameters adjusted according to valid parses. A walkthrough of semantic parser training is given below.

1. GENLEX - generate candidate lexical entries
   - Cross product of templates and logical constants
2. Parse with the current lexicon, including the candidate lexical entries from Step 1
3. Select top k parses
4. Add any candidate lexical entries that appear in top k parses to lexicon
5. Update parameters based on any correct parses

After training, the semantic parser is evaluated against the held-out test data that was supplied at the beginning. Various performance metrics such as precision and recall are then provided.

5 Experimental Setup

When exploring a new domain, it is important to evaluate if a learning task is possible and how difficult that learning task may be. In particular, it is valuable to look at how precision and recall change as additional training data is adduced. Such information can give insights into migrating away from a fully supervised approach and towards a semi-supervised one.

For these experiments, the data was filtered to a small set of important relations: next to, right, left, front, and behind. From among these, annotated data was sampled randomly without replacement. The resulting data subset was divided into training and test. While the number of sentences for testing remained constant, the amount of sentences allotted for training was altered to create several variations of the same data set. The semantic parser was trained on each of these variations using 10-fold cross validation. Each fold was also run 5 times, yielding 50 distinct trials per training data variation. Precision and recall values were averaged across all 50 trials for each variation.

6 Results

At first precision and recall values behave as expected when more training data is adduced to the
semantic parser. However, at around 150 annotated sentences precision and recall begin to slowly drop off. This is likely a product of overfitting, when the model begins to learn incorrect lexical entries. In the following incorrect parse on test data, an incorrect lexical entry was learned during training that generates an invalid parse.

```
the right red bowl
(lambda: e (color: red) (shape: bowl))(lambda: e (color: red))(lambda: e (shape: bowl))
```

**Figure 10**

Here the word “right” was matched to the function *two*. This is no doubt due to the high frequency of “right” and “two” appearing together in natural language sentences (“the right two bowls”).

So far there is still a relatively small amount of training data. More labelled training data with a wider spread of relations will likely increase the overall values of precision and recall and help to prevent overfitting with larger amounts of data.

Much of the errors in training and test can also be attributed to the adopting of incorrect lexical items. Manually specifying the lexemes file would fix this problem, but require a lot of time and work. A hybrid approach that will learn lexical items and utilize a seed lexicon for bootstrapping, or that will learn lexical items and allow for manual corrections, would be an interesting approach to this problem.

## 7 Future Work

There is a lot of opportunity for future work in modeling relational vocabulary for grounded language acquisition. Among these, integrating the current work into the joint model of language and perception pipeline, adding previous excluded functionality, expanding relational coverage, and introducing the current work to an interactive setting will be discussed here.

### 7.1 Joint Language/Perception Model

Matuszek et al. presented a joint model of language and perception for grounded language acquisition. This system couples visual classifiers that identify properties of objects and terms appearing in sentences, grounding language to objects in given scenes. In addition to combining visual classifiers and representations of words, entire sentences must also be analysed semantically. The end goal is to select a subset of objects $G$ from a given scene $O$ and a natural language sentence $x$. The model contains the following components:

- **Semantic parsing model**: The semantic parsing model defines a probability distribution $P(z | \theta)$.
Set of visual attribute classifiers: The set of visual classifiers defines a probability distribution $P(c = true \mid o)$ over certain properties for each object $o$ in a given scene. Essentially there is a separate visual classifier for each property an object could have, such as color and shape.

The work described in this paper has covered the semantic parsing component of the system. The next step would be to determine which visual features are relevant for representing relational vocabulary. A good place to start for this would be to look at distances between objects as well as depths and positions of objects between each other and with reference to the plane they rest on. The relational dataset contains RGB and depth information for each image. In order to extract values for visual features, these images will need to go through various forms of vision processing, including plane and point cloud information extraction. From here, one visual classifier can be trained per relation using the visual features. Combined with the semantic parsing module, this would produce a complete end-to-end system.

7.2 Adding Excluded Functionality

In the Complexities section above, several difficult problems were discussed that are currently not modeled in the system, namely coreference resolution and superlatives. Future work in this area could seek to add this presently absent functionality.

Coreference resolution could be introduced in basic form to disambiguate sentences containing only one antecedent. However, enabling coreference resolution for sentences with multiple possible antecedents will be a more challenging and interesting task. It is likely that recent literature in this area would lend some assistance on the best approach for this.

It would also be valuable to explore the correct representation for superlatives, as they appear quite often in the relational dataset.

7.3 Extensions

Besides the work needed to incorporate relational vocabulary into the joint language/perception system, more work can be done to improve the amount of the relational vocabulary that can be modeled. From the canonical relations used for data annotation and experimentation above, there are several other similar relations that can be modeled in almost the exact same way. Among these are middle, diagonal, and in the corner.

However, there are additional cases which will be slightly more difficult to model, specifically those...
that have complicated visual information. This includes words such as touching and inside, the latter of which currently has no images in the dataset supporting it; this is a result of difficulties in extracting two separate objects from the vision data. Below is an example of the touching relationship, which is also difficult to process visually.

![Figure 11](image)

7.4 Interactive Setting

Lastly, it would be interesting to introduce the system to an interactive setting. Feedback from interactions with users serves as an ultimate form of evaluation, as one of the main goals in designing this system was to allow integration with untrained users.

8 Discussion

It has been shown that the data set collected supports the learning task and current work is demonstrably learning over relational vocabulary. Precision and recall values remain a little low with overfitting occurring for higher amounts of training data supplied. Improvements can be expected as more annotated data with a wider scope is introduced to the system. There is a wealth of opportunity for future work in this area; the work presented here is valuable for future situated language understanding.

9 Acknowledgments

Many thanks to Nicholas Fitzgerald for guiding me through this process, Cynthia Matuszek for mentoring me and always giving helpful feedback, Luke Zettlemoyer for advising me and participating in helpful discussions, and any others who stood in as my rubberducks.

References


Tellex, S.; Thaker, P.; Joseph, J.; and Rot, N. Learning perceptually grounded word meanings from unaligned parallel data. Machine Learning Journal (Spe-