Inferring the Structure of Probabilistic Graphical Models for Efficient Natural Language Understanding

Istvan Chung    Oron Propp
Mentor: Dr. Thomas Howard

1Robust Robotics Group
MIT CSAIL

Fourth Annual MIT PRIMES Conference
18 May 2014
Existing interfaces for controlling robots are specialized and difficult to use.

It would be much easier to control robots using natural language commands.

Existing natural language interfaces do not scale well with the complexity of the environment.
Example

\{WorldObject(0, 'robot'), WorldObject(1, 'crate'), WorldObject(2, 'box')\} + "approach the box" \rightarrow
Constraint(WorldObject(0), WorldObject(2), 'near')
Grammar

- It doesn’t make sense to view the input as a monolithic block of text
- It is more meaningful to understand the input with its grammatical structure
- A grammar is used to assign meaning to the words

\[
\begin{align*}
\text{VP} & \rightarrow \text{VB NP} \\
\text{VP} & \rightarrow \text{VB NP PP} \\
\text{VP} & \rightarrow \text{VB PP} \\
\text{NP} & \rightarrow \text{DT NN} \\
\text{NP} & \rightarrow \text{NP PP} \\
\text{PP} & \rightarrow \text{IN NP} \\
\text{VB} & \rightarrow \text{“approach”, “land”, “fly”} \\
\text{DT} & \rightarrow \text{“a”, “the”} \\
\text{NN} & \rightarrow \text{“box”, “chair”, “table”} \\
\text{IN} & \rightarrow \text{“near”, “far”, “to”}
\end{align*}
\]
Figure: Parse tree for “approach the box near the chair”
Some sentences are ambiguous

Figure: Alternate parse tree for “approach the box near the chair”
The CYK Parsing algorithm [4, 5, 6] accomplishes this task in $O(n^3)$ time.

All possible parses of an ambiguous sentence are returned.

```
<table>
<thead>
<tr>
<th>VP</th>
<th>NP/X0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VP</th>
<th>NP</th>
<th>NP</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VP</th>
<th>NP</th>
<th>NP</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VP</th>
<th>NP</th>
<th>NP</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>approach</th>
<th>the</th>
<th>box</th>
<th>near</th>
<th>the</th>
<th>chair</th>
</tr>
</thead>
</table>
```
Generalized Grounding Graph

- Comprised of “factors” which relate groundings, correspondences, and phrases, and are represented by log-linear models
- Grounding each phrase depends on the groundings of the child phrases

Figure: Generalized grounding graph for “approach the box near the chair”

Log-linear models [1] are used to assign a score to a grounding given some input. This is done using a set of features that evaluate aspects of the input and grounding.

**Scoring function**

\[ p(c \mid x, y; v) = \frac{\exp(v \cdot f(x, y, c))}{\sum_{c' \in C} \exp(v \cdot f(x, y, c'))} \]

Where \( x \) is the input, \( y \) is the grounding, \( c \) is a correspondence variable, \( f \) is the array of features, and \( v \) is the array of feature weights.

Log-linear Model – Training

- Feature weights $\mathbf{v}$ are trained according to data from a corpus of examples.
- The aim of training is to maximize the objective function:

$$L'(\mathbf{v}) = \sum_{i} \log p(c_i \mid x_i, y_i ; \mathbf{v}) - \frac{\lambda}{2} \sum_{k} v_k^2$$

$$(\nabla L')(\mathbf{v})_k = \sum_{i} f_k(x_i, y_i, c_i) - \sum_{i} \sum_{c \in \mathcal{C}} p(c \mid x_i, y_i; \mathbf{v}) f_k(x_i, y_i, c) - \lambda v_k$$

- The LBFGS optimization method [7] efficiently maximizes $L'$ while consuming little space.

The Problem

- Number of possible individual groundings is $O(n^2)$ in the number of objects
- Adding in sets of groundings makes it $2^{O(n^2)}$
The Problem

With 17 objects and 8 relations, the number of sets of constraints is

\[ 2^8 \times (17 + 8 \times 17)^2 = 3.08 \times 10^{56374} \]
Partitioning Grounding Spaces

- In many situations, most groundings are irrelevant
- Partition the grounding space to eliminate irrelevant objects from consideration
Aim of rules is to partition grounding spaces to only include pertinent groundings.

Example:

World: WorldObject(0, 'robot'), WorldObject(1, 'crate'), WorldObject(2, 'box')
“approach the box” → {Rule('box'), Rule('robot')}

Effectiveness of rules increases with complexity of environment and grounding spaces.
Hierarchical Grounding Graph

- Run inference on space of rules
- Apply result to grounding spaces in grounding graph model
- Run inference in graphical model on partitioned grounding spaces for efficient grounding
Hierarchical Grounding Graph

Figure: Hierarchical Grounding Graph for “approach the box near the chair”
Score Evaluations for $G^3$ Model and Hierarchical $G^3$ Model
Run-time for G³ Model and Hierarchical G³ Model

- G³ model
- Hierarchical G³ model
Holodeck Experiment
Future Work

- Expand space of rules to handle region and constraint types
- Implement spatial features with regards to physical world model
- Improve optimization routine (current runtime is impractical)
- Test on Distributed Correspondence Graph model [3]
- Handle parse ambiguity
- Support more sophisticated sentence structures
- Rigorous testing in more complex environments
- Compute bounds on the efficiency of the algorithm

Acknowledgements

Thank you to

- MIT PRIMES
- Dr. Thomas Howard
- Professor Nicholas Roy
- Dr. Marec Doniec
- Our parents


