Appendix: Semantic Parsing to Probabilistic Programs for Situated Question Answering

Jayant Krishnamurthy and **Oyvind Tafjord** and **Aniruddha Kembhavi**
Allen Institute for Artificial Intelligence
jayantk,oyvindt,anik@allenai.org

1 Semantic Parser

The CCG semantic parser used in P^3 has a rich set of features based on those for syntactic CCG parsing. Its lexicon entries have four fields:

1. **Word sequence** – The words for which the entry can be used.
2. **Syntactic category** – The CCG syntactic category with head-passing markup.
3. **Logical form**
4. **Predicate** – a representation of the entry's semantics used to populate dependency structures. We create a unique predicate per logical form.

All of the fields of these lexicon entries are automatically produced by running PAL (Krishnamurthy, 2016). The parser also dynamically creates lexicon entries using the OCR text labels in the environment in order to identify references to organisms in the diagram.

During parsing, each chart entry consists of a syntactic category, a predicate X_p and a word index X_i. The predicate and word index represent the head word of the entry. Which word is the head is determined from the head passing markup on syntactic categories. Chart entries also include logical forms; however, the parser's features do not apply to these directly, rather they depend on logical forms via predicates.

The CCG parser's features are:

1. **Lexicon entry counts** – A feature per lexicon entry that counts the number of times it is used in the parse.
2. **Word skip counts** – A feature per word that counts the number of times it is skipped in the parse.
3. **Dependency features** – The parser creates dependency structures during parsing whenever an argument to a syntactic category is filled by applying a combinator. There is a feature per dependency structure that counts its number of occurrences in the parse. The dependency structure consists of the syntactic category, its associated predicate, the argument number that was filled, and the predicate of the argument category.
4. **Dependency distance features** – These features capture the number of intervening words between the two words connected by a dependency structure. There is a dependency distance feature per syntactic category, associated predicate, argument number, and the number of words between the two dependency structure elements. The number of words is histogrammed into 4 bins $0, 1, 2, \geq 3$.
5. **Combinator features** – Combinators are applied to a left and right syntactic category to produce a parent syntactic category. There is a combinator feature that counts occurrences of every left/right/parent triple.
6. **Headed combinator features** – Same as

above, but additionally including the predicate of the parent.

7. **Root features** - A feature per syntactic category and predicate pair indicating whether the pair appears at the root of the parse.

2 Execution Model

The execution model's features rely on the predictions of a vision system. This system outputs a collection of scored diagram elements and linkages:

1. **Blobs** – Objects detected in the image, e.g., the drawing of a mouse.
2. **Text** – Text detected in the image using OCR. Each text element has a text value as well as a bounding box.
3. **Arrows** – Arrows in the image, including simple lines and more complex bitmap arrows.
4. **Arrowheads** – The heads of arrows with an accompanying bounding box.
5. **Intraobject Labels** – A linkage associating a text element as the label of a blob.
6. **Interobject Linkage** – A three-way linkage between two elements mediated by an arrow. The two elements can be either blobs or text elements.

The execution features are functions of text elements and pairs of text elements. As a preprocessing step, we associate each text element with a blob by computing a maximum matching using the intraobject label scores. We also associate a best arrow with each pair of text labels using the interobject linkage scores.

The instance features for ORGANISM are:

1. **Text Score** – OCR score from the vision system for the detection.
2. **Bias** – Always 1
3. **Sub-bounding Box** – 1 if is the text element is contained in a larger text element's bounding box.
4. **Number of words** – Two indicator features if the number of words is ≥ 4 or ≥ 6.

The instance features for EATS are:

1. **Self Link** – 1 if the instance is of the form $\text{EATS}(x, x)$
2. **No Link** – indicator feature that is 1 if no interobject linkage connects the two text elements.
3. **Link Exists** – indicator feature that is 1 if an interobject linkage connects the two text elements, either directly or through their associated blobs.
4. **Link Score** – score of the interobject linkage connecting the two text labels. This score is both histogrammed into bins $[0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1]$ and included as a real-valued feature.
5. **Link Path Score** – product of the interobject linkage score and any intraobject label scores. This score differs from the above score if the text labels are connected through blobs, as it also incorporates the intraobject label scores connecting the text labels and blobs. This score is both histogrammed into bins $[0, 0.01, 0.05, 0.1, 0.3, 0.7, 1.0]$ and included as a real-valued feature.

The predicate features for EATS are:

1. **Cycle Features** – These features approximately count the number of cycles of various lengths in the graph of true EATS relation instances. There is a separate feature for each cycle length $k = 2, 3, 4, \geq 5$. Let n_k be the number of edges (i, j) for which the shortest path from $j \rightarrow i$ has length $k - 1$. The kth cycle feature's value is n_k/k. Note that if the graph has only one cycle and that cycle has length k, the value of the kth cycle detection feature is 1. In the ≥ 5 case, each edge completing a cycle of length m adds $1/m$ to the feature's value.
2. **Arrow Reuse Features** – We count for each arrow a in the diagram the number of true eats relation instances for which a is the best arrow. The reuse features are count features for the number of arrows used $2, 3, \geq 4$ times.

The denotation features are:

1. **Denotation Size** by denotation type. For each type of logical form, a size feature for whether the denotation contains $0, 1, 2, 3, \geq 4$ elements.

2. **Count Value Features**. If the denotation is an integer, an indicator feature for each value in $0, 1, 2, 3, \geq 4$.

3. **Role Features**. An indicator feature for each role and animal that occurs ≥ 5 times in the training set. These features fire if either the logical form returns the role of an animal or returns all animals with a specific role.

4. **Direction Features**. An indicator feature for each change direction. These features fire if the denotation is a set of change directions.

References

Jayant Krishnamurthy. 2016. Probabilistic models for learning a semantic parser lexicon. In *NAACL*.