Human-in-the-Loop parsing

Luheng He, Julian Michael, Mike Lewis and Luke Zettlemoyer
Problem

- Large size of datasets is a bottleneck for natural language processing systems
- Proposed solution - Human-in-the-loop parsing.
- Non-experts improve parsing my answering questions automatically generated from the parser’s output

Temple also said Sea Containers’ plan raises numerous legal, regulatory, financial and fairness issues, but didn’t *elaborate*.

| Q: What didn’t *elaborate*? |
|-----------------------------|
| [1]**** Temple              |
| [2] * Sea Containers’ plan  |
| [3] None of the above.      |
CCG Categories

Syntax
- Primitive symbols: NP, ADJ etc
- Syntactic combination operators (/,\)

Semantics
- \( ADJ : \lambda x. \text{fun}(x) \)
- \( \lambda \)-calculus expression

- Basic building block
- Capture syntactic and semantic information jointly
CCG Lexical Entries

- Pair words and phrases with meaning
- Meaning captured by a CCG category
CCG Lexicons

fun ⊢ ADJ : \lambda x.\text{fun}(x)

is ⊢ (S\backslash NP)/ADJ : \lambda f.\lambda x.f(x)

CCG ⊢ NP : CCG

- Pair words and phrases with meaning
- Meaning captured by a CCG category
CCG Operations

- Small set of operators
- Input: 1-2 CCG categories
- Output: A single CCG category
- Operate on syntax semantics together
- Mirror natural logic operations
CCG Operations

\[
\begin{align*}
B &: \ g \\
A \backslash B &: \ f \\
\Rightarrow & \quad A : f(g) \\
\text{(less than)}
\end{align*}
\]

\[
\begin{align*}
A / B &: \ f \\
B &: \ g \\
\Rightarrow & \quad A : f(g) \\
\text{(greater than)}
\end{align*}
\]

- Equivalent to function application
- Two directions: forward and backward
  - Determined by slash direction
CCG Parsing

\[
\frac{CCG \quad NP \quad CCG}{is \quad S\setminus NP/ADJ \quad \lambda f.\lambda x.f(x)} \quad fun \quad ADJ \quad \lambda x.fun(x)
\]

\[
S\setminus NP \quad \lambda x.fun(x)
\]

Combine categories using operators
Weighted Linear CCGs

- Given a weighted linear model:
  - CCG lexicon $\Lambda$
  - Feature function $f : X \times Y \rightarrow \mathbb{R}^m$
  - Weights $w \in \mathbb{R}^m$

- The best parse is:
  $$y^* = \arg \max_y w \cdot f(x, y)$$

- We consider all possible parses $y$ for sentence $x$ given the lexicon $\Lambda$
Mapping CCG parses to queries

- Parse sentence using Combinatory Categorial Grammar (CCG) parser.
- Determine verb’s set of arguments by the CCG supertag assigned to it
- Obtain dependencies for each argument position
- Replace noun phrases by something

\[
\text{put - CCG supertag } (\text{S\textbackslash NP)/PP})/\text{NP} \\
\text{CCG supertag to dependency - “simple heuristic”}
\]
Mapping CCG parses to queries

- Generate Q for every parse in 100-best outputs of the parser
- Pool Q by the head of the dependency, it’s CCG category and question string
- Each pool becomes a query
- Compute marginalized score for each QA phrase by summing over scores of all parses that generated them
- For each unique dependency, add candidate answer to the query by choosing the answer phrase that has the highest marginalized score for that dependency
- Remove queries and answers with marginalized score below certain threshold, and queries with one answer (only keep confident questions with uncertain answers)
Table 2: Example annotations from the CCGbank development set. Answers that agree with the gold parse are in bold. The answer choice *None of the above* was present for all examples, but we only show it when it was chosen by annotators.
Re-parsing with QA annotation

- For question $q$, with answer $a$, denote by $v(a)$ the fraction/number of annotators that chose $a$.
- Add re-parsing constraints as follows

  - If $v(\text{None of the above}) \geq T^+$, penalize parses that agree with $q$'s supertag on the verb by $w^t$
  - If $v(a) \leq T^-$, penalize parses containing $d$ by $w^-$
  - If $v(a) \geq T^+$, penalize parses that do not contain $d$ by $w^+$

| Data    | L16 | HITL |
|---------|-----|------|
| CCG-Dev | 87.9| 88.4 |
| CCG-Test| 88.1| 88.3 |
| Bioinfer| 82.2| 82.8 |

Table 6: CCG parsing accuracy with human in the loop (HITL) versus the state-of-the-art baseline (L16) in terms of labeled F1 score. For both in-domain and out-domain, we have a modest gain over the entire corpus.

| Data    | L16 | HITL | Pct. |
|---------|-----|------|------|
| CCG-Dev | 83.9| 87.1 | 12%  |
| CCG-Test| 84.2| 85.9 | 10%  |

Table 7: Improvements of CCG parsing accuracy on changed sentences for in-domain data. We achieved significant improvement over the 10%–12% (Pct.) sentences that were changed by re-parsing.