Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base Supplementary Notes

Wen-tau Yih Ming-Wei Chang Xiaodong He Jianfeng Gao
Microsoft Research
Redmond, WA 98052, USA
{scottyih,minchang,xiaohe,jfgao}@microsoft.com

Appendix A. Formal search algorithm

Given an input question, the query graph generation process can be formally defined as a search problem by a tuple $\langle \mathcal{S}, \mathcal{A}, \Pi, T, \gamma \rangle$. $\mathcal{S} = \bigcup \{\{\phi\}, \mathcal{S}_e, \mathcal{S}_p, \mathcal{S}_c\}$ is the set of states, where each state is an empty graph (ϕ), a single-node graph with the topic entity ($\mathcal{S}_e$), a core inferential chain ($\mathcal{S}_p$), or a more complex query graph with additional constraints ($\mathcal{S}_c$). $\mathcal{A} = \bigcup \{\mathcal{A}_e, \mathcal{A}_p, \mathcal{A}_c, \mathcal{A}_a\}$ is the set of actions. $\mathcal{A}_e$ picks an entity node; $\mathcal{A}_p$ determines the core inferential chain; $\mathcal{A}_c$ and $\mathcal{A}_a$ add constraints and aggregation nodes, respectively. Not all the actions are valid to a given state. We use $\Pi : \mathcal{S} \rightarrow 2^{\mathcal{A}}$ to denote the legitimate set of actions. Specifically, $\Pi(\phi) = \mathcal{A}_e$; $\Pi(s_e) = \mathcal{A}_p, \forall s_e \in \mathcal{S}_e$; $\Pi(s) = \mathcal{A}_c \cup \mathcal{A}_a, \forall s \in \mathcal{S}_p \cup \mathcal{S}_c$. $T(s,a) \rightarrow s'$ is the transition function that maps a state s and an action a to the next state s'. $\gamma : \mathcal{S} \rightarrow \mathbb{R}$ is the reward function that depends on the input question and is defined over the state space.

Staring from $s_0 = \phi$, valid actions are iteratively applied to the current best state in the priority queue. The procedure stops when there is no more state to examine. Algorithm 1 shows the pseudo code. The search procedure essentially keeps up to N candidate states in the priority queue ($N = 1000$ in this work), and explores as many legitimate states as possible to find the best query graph, according to the reward function γ. Obviously, whether the approach can work in practice depends on the quality of the reward function, as well as whether the search space can be effectively controlled.

Algorithm 1 Staged query graph generation

```
Require: Priority queue $H$ with limited size $N$
 1: $s_o \leftarrow \phi$; $r_o \leftarrow -\infty$
 2: $H$.add($s_o$,$r_o$)
 3: while $H$ is not empty do
 4:     $s, r \leftarrow H$.pop()
 5:     if $r > r_o$ then
 6:         $s_o \leftarrow s$; $r_o \leftarrow r$;
 7:     end if
 8:     for all $a \in \Pi(s)$ do
 9:         $s' \leftarrow T(s,a)$
10:         $H$.add($s', \gamma(s')$)
11:     end for
12: end while
13: return $s_o$
```

Appendix B. Implementation detail for constraints and aggregations

Based on the observation on the training set questions, we developed the following rules to determine whether we would consider adding a constraint or an aggregation node to a CVT node to extend a query graph.

- The constraint entity occurs in the question.
- The constraint predicate indicates the ending time of the event, but there is no value. (This indicates that it's a current event.)
- Some words of the constraint entity's name appear in the question.
- The predicate is `people.marriage.type_of_union`. (This indicates whether the relationship is domestic partnership, marriage or civil union.)
- The question contains "first" or "oldest" and the predicate is a "from" predicate (indicating the starting time of an event).
- The question contains "last", "latest" or "newest" and the predicate is a "to" predicate (indicating the ending time of an event).

For an answer node, we will only add a constraint node if the predicate is one of

- `people.person.gender`

- `common.topic.notable_types`
- `common.topic.notable_for`

Features of constraints and aggregations are indicator functions of whether some of the above rules are true, or simple statistics related to some rules. Again, these features were developed through the analysis on the training data. For a CVT node, the features are:

- Whether the constraint entity appears in the question.
- Whether it's a current event: the constraint predicate indicates the ending time of the event, but there is no value.
- Whether it's a current event and the question has one or more of the keywords "currently", "current", "now", "present" and "presently".
- The percentage of the words in the constraint entity that appear in the question.
- The type of the constraint predicate `people.marriage.type_of_union`.
- Whether the question contains "first" or "oldest" and the predicate is a "from" predicate, and the CVT node is the first when ordered by this "from" property.
- Whether the question contains "last", "latest" or "newest" and the predicate is a "to" predicate, and the CVT node is the last when ordered by this "to" property.

For the answer node, the features are:

- Gender consistency – male: whether the constraint predicate is `gender` and one of the male keywords {"dad", "father", "brother", "grandfather", "grandson", "son", "husband"} appears in the question.
- Gender consistency – female: whether the constraint predicate is `gender` and one of the female keywords {"mom", "mother", "sister", "grandmother", "granddaughter", "daughter", "wife"} appears in the question.
- The percentage of the words in the constraint entity that appear in the question when the constraint predicate is `notable_types` or `notable_for`.

Method	Prec.	Rec.	F_1
PatChain	46.0	54.5	45.6
+QuesEP	47.8	55.1	46.4
+ClueWeb	48.5	57.9	48.0

Table 4: The system results when only the inferential-chain query graphs are generated. The settings here are the same as those in Tab. 3, except that topic entities are identified using the Freebase API, instead of our entity linking component.

Appendix C. Freebase API for entity linking

We conducted a similar set of experiments by replacing our entity linking component with results from Freebase API, and summarized the results in Tab. 4. Comparing Tables 3 and 4, we can see that the negative impact of using an inferior entity linking component is consistent, leading to an approximately 4-point drop across different models for matching the core inferential chain.