Appendix A: Detailed Score Computation for Constraint Detection

Given an input question q and an entity name e in KB, we denote the lengths of the question and the entity name as $|q|$ and $|n_e|$. For a mention m of the entity e which is an n-gram in q, we compute the longest consecutive common sub-sequence between m and e, and denote its length as $|m \cap e|$. All the lengths above are measured by the number of characters.

Based on the above numbers we compute the proportions of the length of the overlap between entity mention and entity name (in characters) in the entity name $\frac{|m \cap e|}{|e|}$ and in the question $\frac{|m \cap e|}{|q|}$; The final score for the question has a mention linking to e is

$$s_{linker}(e; q) = \max_m \frac{|m \cap e|}{|q|} + \frac{|m \cap e|}{|e|}$$

Appendix B: Special Rules for Constraint Detection

1. Special threshold for date constraints. The time stamps in KB usually follow the year-month-day format, while the time in WebQSP are usually years. This makes the overlap between the date entities in questions and the KB entity names smaller (length of overlap is usually 4). To deal with this, we only check whether the dates in questions could match the years in KB, thus have a special threshold of $\theta = 1$ for date constraints.

2. Filtering the constraints for answer nodes. Sometimes the answer node could connect to huge number of other nodes, e.g. when the question is asking for a country and we have an answer candidate *the U.S.*. From the observation on the WebQSP datasets, we found that for most of the time, the gold constraints on answers are their entity types (e.g., whether the question is asking for a country or a city). Based on this observation, in the constraint detection step, for the answer nodes we only keep the tuples with *type* relations (i.e. the relation name contains the word "*type*"), such as *common.topic.notable_types, education.educational_institution.school_type* etc.

Appendix C: Effects of Entity Re-Ranking on SimpleQuestions

Removing entity re-ranking step results in significant performance drop (see Table 3, the row of *w/o entity re-ranking*). Table 4 evaluates our re-ranker as an separate task. Our re-ranker results in large improvement, especially when the beam sizes are smaller than 10. This is indicating another important usage of our proposed improved relation detection model on entity linking re-ranking.

Top K	**FreeBase API**	**(Golub & He, 2016)**
1	40.9	52.9
10	64.3	74.0
20	69.7	77.8
50	75.7	82.0
Top K	**(Yin et al., 2016)**	**Our Re-Ranker**
1	72.7	**79.0**
10	86.9	**89.5**
20	88.4	**90.9**
50	90.2	**92.5**

Table 4: Entity re-ranking on SimpleQuestions (test set).