A  SQG construction algorithm

Algorithm 1 Build Semantic Query Graph

**Input:** Node set $V$, Relation Extraction model $RE()$, Reward Function $\gamma()$

**Output:** The final Semantic Query Graph

1: for each pair $(u, v) \in V \times V$ do
2: \hspace{1em} $RE(u, v)$
3: end for
4: Initialize priority queue $H$
5: $SQG_{s_0} = \{V, E = \emptyset\}$
6: $H.add(s_0, \gamma(s_0))$
7: while $H$ is not empty do
8: \hspace{1em} $s, r = H.pop()$
9: \hspace{1em} if $isValidSQG(s)$ then
10: \hspace{2em} return $s$
11: \hspace{1em} end if
12: \hspace{1em} for operation $op$ do
13: \hspace{2em} for operate node $u \in S.V$ do
14: \hspace{3em} if $checkConstraint(op, u)$ then
15: \hspace{4em} $s' = TS(s, op, u)$
16: \hspace{4em} if $s'$ is a new state then
17: \hspace{5em} $H.add(s', \gamma(s'))$
18: \hspace{4em} end if
19: \hspace{3em} end if
20: \hspace{2em} end for
21: \hspace{1em} end for
22: end while

Algorithm 1 shows the pseudo code of the SQG construction procedure. As shown in Line 1-3, we first extract relations between each pair of nodes by the relation extraction model. Each potential relation has a confidence probability which can be used in the reward function $\gamma()$. The initial state $s_0$ is a semantic query graph contains all isolated nodes with no edges. We put $s_0$ and its score $\gamma(s_0)$ to the priority queue $H$ (Line 4-6). During the search procedure, in each epoch we get the current best state and check whether it is a valid SQG. A valid SQG should be a connected graph with at least two nodes. It should has matches in the knowledge graph and has no subsequent SQGs with higher scores. The first valid SQG is considered as the final semantic query graph (Line 7-11). Line 12-21 are the enumeration of state transition. Specifically, for each operation $op$ we enumerate each possible operate node $u$. The function checkConstraint check whether $op$ and $u$ satisfy the corresponding condition. Although this is a greedy search algorithm, the final SQG we generated is usually the best one, especially when the small node set.