Knowledge Graph Reliability

July 23, 2021

Algorithm 1 EMBEDRULE-KG

Input: $KB = \{(e_1, r, e_2)\}$, relation set R, length $k > 0$ of rule's body, support threshold $\eta > 0$
Output: Candidate rules Q
$Q \leftarrow \emptyset$
for $h \in R$ **do**
  Select the set of start relations $S = \{s \in R \mid \mathcal{X}_s \cap \mathcal{X}_r \neq \emptyset\}$
  Select the set of end relations $T = \{t \in R \mid \mathcal{Y}_t \cap \mathcal{Y}_r \neq \emptyset\}$
  $SEQ_h \leftarrow$ find all possible sequences of k distinct relations other than h and complying with S and T
  {*Visit* KB *to compute the support of a relation sequence in* SEQ_h}
  $subjects(h) \leftarrow \{e_1 | (e_1, h, e_2) \in KB\}$
  for $(r_1, r_2, \ldots, r_k) \in SEQ_h$ **do**
    $visited \leftarrow \{e_1 \mid (e_1, r_1, e_2) \in KB\}$
    $paths \leftarrow \{\langle e_1, 1\rangle \mid (e_1, r_1, e_2) \in KB\}$
    for $i = 1, \ldots, k$, **until** $|paths| > 0$ **do**
      for $\langle e, p\rangle \in paths$ **do**
        $N(e, r_i) \leftarrow \{e' \mid (e, r_i, e') \in KB, e' \notin visited\}$
        {*all non-visited entities connected to e via* r_i}
      end for
      $N_i \leftarrow \bigcup_{\langle e,p\rangle \in paths} N(e, r_i)$
      $p(e) \leftarrow 0, \ \forall e \in N_i$
      for $\langle e, p\rangle \in paths, \ e' \in N(e, r_i)$ **do**
        $p(e') \leftarrow p(e') + p$
      end for
      $paths \leftarrow \{\langle e, p(e)\rangle \mid e \in N_i\}; \ \ visited \leftarrow visited \cup N_i$
    end for
    if $|paths| > 0$ **then**
      $pos \leftarrow$ sum $p(e)$ $\forall \langle e, p(e)\rangle \in paths$ s.t. $\exists e' \in subjects(h) : (e, h, e') \in KB$
      $neg \leftarrow$ sum $p(e)$ $\forall \langle e, p(e)\rangle \in paths$ s.t. $\nexists e' \in subjects(h) : (e, h, e') \in KB$
      $sup \leftarrow pos/(pos + neg)$
      if $sup \geq \eta$ **then**
        $Q \leftarrow Q \cup \{((r_1, \ldots, r_k) \Rightarrow h)\}$
      end if
    end if
  end for
end for

☞ **NOTE:** Optimization: while processing a relation sequence $(r_1, r_2, \ldots, r_k) \in SEQ_h$, one can cache $paths$ computed for all prefixes $\{(r_1, \ldots, r_i)\}_{i=1}^{k-1}$, and, then, for a next sequence $(r'_1, r'_2, \ldots, r'_k) \in SEQ_h$, the process can start from the longest prefix of $(r'_1, r'_2, \ldots, r'_k)$ that is stored in cache, and not from scratch.

1 Reliability

Let as assume to build a complete, weighted, entity-relation-entity graph, containing all possible relations and with arc weights equivalent to the scoring function of a considered embedding model $f_r(h,t)$.

We define **reliability** $R(\mathcal{K})$ as follows:

$$R(\mathcal{K}) = \sum_{(h,r,t)\in\mathcal{K}} (1 - f_r(h,t)) + \sum_{(h,r,t)\notin\mathcal{K}} f_r(h,t)$$

Where $\mathcal{K}$ is a knowledge graph (provided as a set of triplets).

In this experiment, we are going to use a link-prediction strategy to *compare* the **reliability** of a subgraph with the **accuracy** of a classifier on the subgraph triplets.

The pseudo-code of the algorithm that you should implement is reported below.

- Pick integers
  - k: subgraph size
  - h: number of subgraphs to be sampled
- Split the input KG into $TRAIN_E$ (triplets to be used for training the embeddings) and $LP^+ = KG \setminus TRAIN_E$ (positive examples for the link-prediction classifier)
- Define train set and test set for the ultimate link-prediction task
  - Define LP^- by sampling $|LP^+|$ non-existing triplets from KG (in some way; check the literature)
  - Split LP^+ into $TRAIN_{LP^+}$ and $TEST_{LP^+}$
  - Split LP^- into $TRAIN_{LP^-}$ and $TEST_{LP^-}$
  - $TRAIN_{LP} = TRAIN_{LP^+} \cup TRAIN_{LP^-}$
  - $TEST_{LP} = TEST_{LP^+} \cup TEST_{LP^-}$
- Train embeddings on $TRAIN_E$
- Train a classifier for link prediction on $TRAIN_{LP}$ (with embeddings used as feature vectors, e.g., letting the feature vector of triple t correspond to the concatenation of the embeddings of t's subject, t's predicate, and t's object)
- Test the link-prediction classifier. In particular:
  - Sample h subgraphs $\{S_1, \ldots, S_h\}$ of size (#nodes) k from KG
  - For every subgraph S_i, $i = 1, \ldots, h$:
    * Compute accuracy ACC_i of the link-prediction classifier on $TEST_{LP} \cap S_i$

* Compute the reliability score REL_i of subgraph S_i
* Show correlation among the set $\{ACC_i\}i = 1^h$ of accuracy scores and the set $\{REL_i\}i = 1^h$ of reliability scores
* Ideally, such a correlation should show *high accuracy* $\Leftrightarrow$ *high reliability*