基于语义度量的RDF图近似查询

章登义,吴文李,欧阳黜霏
(武汉大学计算机学院,湖北武汉 430072)

摘 要: 近似查询是图数据库资源管理的操作之一.已有工作主要基于距离来度量查询语句与图的近似值,忽略了两者之间的语义近似性.对于语义图的近似查询,忽略图与查询的语义近似将难以有效完成查询.针对该问题,本文在考虑语义近似的基础上为RDF(Resource Description Framework,资源描述框架)图的近似查询提出基于语义距离的度量方法.同时,为提高查询效率,本文提出语义结构剪枝策略.最后,我们构造查询框架以实现查询的响应过程,并在该框架下设计实验以评价本文方法.实验表明,本文方法可高效执行RDF近似查询并有效返回top-k结果集.

关键词: RDF图; 近似查询; 图数据库; 查询处理
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2015)07-1320-09
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.07.011

Approximating Query with Semantic-Based Measure on RDF Graphs

ZHANG Deng-yi, WU Wen-li, OUYANG Chu-fei
(*School of Computer, Wuhan University, Wuhan, Hubei* 430072, *China*)

Abstract: Approximating query is one of the operations for resource management in graph database. Existing works mainly based on the distance similarity to measure the query and graphs without considering the semantic similarity. For the approximating query on semantic graphs, disregarding the semantic similarity may fail the query. In this paper, we propose a semantic-based measure for approximating query on RDF graphs, considering the semantic similarity. In the meanwhile, we specify a semantic structural pruning strategy to ameliorate the efficiency of query process. Finally, we construct the query framework to answer approximating query, and design experiments to test our methods under this framework. Results show that the approaches in this paper can efficiently execute approximating query on RDF graphs and effectively return top-k results.

Key words: RDF graph; approximating query; graph database; query processing

1 引言

资源描述框架[1]是为了实现机器的自行交流和理解交互方意图所提出的一个标准,主要应用于语义网.随着语义网的发展,越来越多的数据以RDF三元组的形式存储,即<资源,属性,描述>(<subject, property, object>).从图的角度上看,资源(subject)作为图的顶点,属性(property)是连接各个顶点的边,而三元组的最后一个对象(object),既可以表示资源的值同时也能作为图中的顶点.这种图的表示方式简单灵活,日益成为描述大数据集的重要手段.

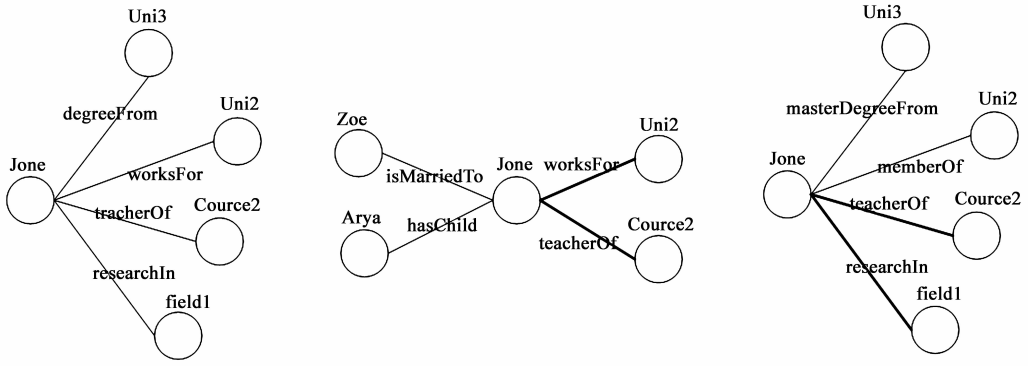
图查询分为两种:精确查询和近似查询.文献[2~5]表明,精确查询是资源数据管理的基本操作之一.随着图数据的不断增长,近似查询日益成为数据分析与复杂关系挖掘的重要手段.目前,对图近似查询的研究有文献[6~10].其中,文献[7,10]、文献[8]和文献[9]分别对子图、超图和整图做近似查询.同时,三者均用距离来度量查询语句和图的近似值.若将基于最大共同子图MCS(Maximum Common Subgraph)的距离计算方法记为$mcs(q,g)$,文献[7]的度量公式为$dist(q,g)=|E(q)|-|E(mcs(q,g))|$,文献[8]的度量公式为$dist(q,g)=|E(q)|-|E(mcs(q,s))|$,文献[9]对两公式进行改进并取得较好的查询效果.但是,以上方法中的距离均指的是查询与图之间的不同边数,即这些方法仅从结构差异的角度计算近似值.然而,对于语义图-RDF图的近似查询,仅考虑查询与图的结构近似,则忽略其本身具有的语义近似性,因此,难以有效完成查询.

图1给定查询q和数据库$D=\{g_1,g_2\}$,并做top-1近似查询,用公式$dist(q,g)=|E(q)|-|E(mcs(q,g))|$或$dist(q,g)=|E(mcs(q,g))|-|E(q)|$作为距离的

收稿日期:2014-12-01;修回日期:2015-01-29;责任编辑:孙瑶

度量,有 $dist(q,g_1)=4-2=2$ 和 $dist(q,g_2)=4-3=1$. 由于 $dist(q,g_2)$小于 $dist(q,g_1)$,因此,返回 g_1. 但是,根据图的语义近似性,g_2的 mastersDegreeFrom 和 memberOf 分别是 q 中 degreeFrom 和 worksFor 的子类,同时,g_2的 contributionOf 是 q 中 researchIn 的父类. 而 g_1的 isMarriedTo 和 hasChild 与 q 中的 teachOf 和 researchIn 基本无关. 因此,从语义上看,g_2比 g_1更接近 q,最佳结果为 g_2.

针对于此,本文在考虑语义近似的情况下为 RDF 图近似查询提出基于本体的度量方法. 为提高查询效率并获取结构相近的候选图集,本文利用语义结构剪枝策略,有效避免查询语句对数据库的每个图都进行一次相似性计算. 同时,为高效执行该策略,本文设计了数据编码方法并提取相应的摘要. 最后,我们构造查询框架以实现 RDF 图近似查询,并在该框架下设计实验以评价本文方法. 实验表明,本文方法可高效执行查询并有效返回结果.



(*a*) 查询图q　　(*b*) 图g_1　　(*c*) 图g_2

图1　示例

2　预备知识

RDF 图[11]数据库 $D=\{g_1,g_2,\cdots,g_n\}$,且 $g_i=(t_1,t_2,\cdots,t_n)$,$t_i=<s,p,o>\in(I\cup B)\times(I\cup B)\times(I\cup B\cup L)$. 其中,$g_i$是单图,$t_i$是三元组,$I$ 是 IRI 的集合,B 是空顶点集,L 是字符串. 具体的 RDF 三元组数据如图 2 所示. 图 2 的三元组数据与图 1 的 g_2对应,后者是前者的图描述.

三元组		
Subject	Property	Object
Jone	mastersDegreeFrom	Uni3
Jone	memberOf	Uni2
Jone	teacherOf	Course2
Jone	contributionOf	Field1

图2　图g_2RDF三元组

SPARQL[12]的查询模式 $q=(tp_1,tp_2,\cdots,tp_n)$且 $tp_i=<s,p,o>\in(I\cup V)\times(I\cup V)\times(I\cup V\cup L)$. 其中,$tp_i$是元组模式,$V$ 是变量集. 具体的 SPARQL 查询语句如图 3 所示. 该查询的查询图如图 1(*a*)所示. 其中,以'?'开头的词为变量.

```
查询 q:
Prefix ub:www.whu,edu#
SELECT ? P
WHERE{
    ? P ub:mastersDegreeFrom ub:Univ3.
    ? P ub:worksFor ub:Uniy2.
    ? P ub:teacherOf ub:Course2.
    ? P ub:researchIn ub:Field1.
    }
```

图 3　SPARQL 查询

RDFs(RDF Schema)是 RDF 图数据的本体,它涵盖了资源和属性的分类及关联. 通过本体可推导任意两个资源或属性的关联关系. 本文所用的本体如图 4 所示. 给定属性 mastersDegreeFrom 和 DegreeFrom,由图 4 可见,后者是前者的父类. 同时,由于属性 DegreeFrom 和 Book 无任何共同祖先,因此,本文称之为为无关联属性. 同理,无共同祖先的顶点被称为无关联顶点.

问题描述　本文解决的问题为:给定 SPARQL 查询图 q 及 RDF 图数据库 $D=\{g_1,g_2,\cdots,g_n\}$,返回 top-k $Sdist(q,g)$最小图. 其中,$Sdist(q,g)$表示 q 和 g 的语义距离.

3　语义近似查询

3.1　语义距离计算

本节基于本体,利用最少共同祖先(Least Common Ancestor,LCA)的概念完成语义距离的度量. 计算过程涉

及查询 q 与 RDF 图 g,顶点与顶点、边与边,以及三元组(元组模式)的语义距离.其中,$Sdist\_v(v_i, v_j)$,$Sdist\_e(e_i, e_j)$,$Sdist_t(t_i, t_j)$分别表示顶点间、边之间以及元组间的语义距离,两顶点的 LCA 指的是它们在 IS-A 层次关系中的共同祖先.某一顶点(属性)的深度指的是从本体的根节点到该顶点的最大长度.另外,假设 q 的候选图集包含在有限集 SC(Set of Class)和 SP(Set of Property)中.

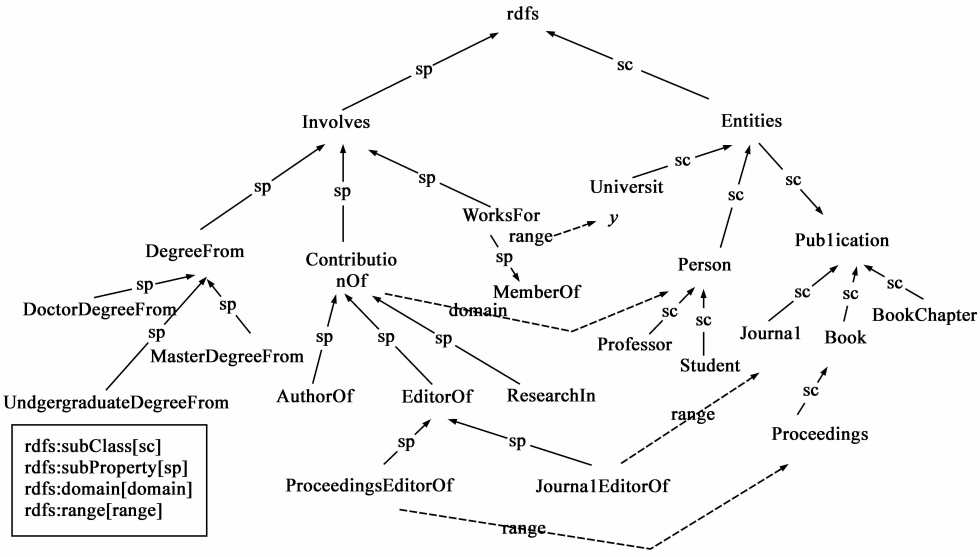


图4 图g_1和g_2对应的本体

顶点的语义距离 元组模式的顶点(s 或 o)是查询的最小单元之一.给定类 c_1,已知 c_1在 RDFs 中有定义,它与父类顶点 $c_2 \in SC$ 之间的语义距离计算公式如下:

$$Sdist\_v(c_1, c_2) = depth(c_1) + depth(c_2) - 2depth(LCA(c_1, c_2)) \quad (1)$$

由式(1)可得,$Sdist\_v$('Book','Proceeding') = 2 + 3 − 2 × 2 = 1,即类 Book 和 Proceeding 的语义距离为 1. 此外,本文将变量与图顶点的语义距离定义为零.同时,无关联顶点的深度被定义为最大深度,即无关联顶点等同于本体中深度取得最大值的顶点.因此,图 1 中有 $Sdist\_v$('? P','Jone') = 0,$Sdist\_v$('Uni3','Zoe') = 7 + 7 − 0 = 14.

边的语义距离 已知 p_1是查询图的一条边,$p_2 \in SP$ 且在本体中有定义,$Sdist_e(p_1, p_2)$的计算公式与式(1)类似,因此有:

$$Sdist\_e(p_1, p_2) = depth(p_1) + depth(p_2) - 2depth(LCA(p_1, p_2)) \quad (2)$$

式(2)中,$LCA(p_1, p_2)$表示属性 p_1和 p_2的共同祖先.同时,属性的深度指的是根节点的子属性到该属性的最大长度.因此,图 1 示例中 $Sdist\_e$('mastersDegreeFrom','degreeFrom') = 1 + 0 − 2 × 0 = 1. 此外,本节将变量与具体属性的语义距离定义为零.无关联边的属性深度被定义为最大深度,即无关联边等同于本体中深度取得最大值的属性.

元组的语义距离 给定三元组 $t(s, p, o)$和元组模式 $tp'(s', p', o')$,综合式(1)和(2),本文将 t 和 tp 的语义距离定义为:

$$Sdist\_t(t, tp') = Sdist\_v(s, s') + Sdist\_e(p, p') + Sdist\_v(o, o') \quad (3)$$

查询与图的语义距离 给定查询图模式 $q(tp_1, tp_2, \cdots, tp_n)$以及 RDF 图 $g(t_1', t_2', \cdots, t_n')$,本文将 q 和 g 的语义距离定义为:

$$Sdist(q, g') = \sum_{i=1}^{n} Sdist\_t(tp_i, t_i') \quad (4)$$

综上所述,图 1 示例的语义距离计算过程如下:

$Sdist(q, g_1)$
= $Sdist\_e$(<'degreeFrom','isMarriedTo'>)
+ $Sdist\_v$(<'Zoe','Uni3'>)
+ $Sdist\_e$(<'hasChild','worksFor'>)
+ $Sdist\_v$(<'Arya','Uni2'>)
= 6 + 6 + 7 + 7 + 6 + 6 + 7 + 7
= 52

$Sdist(q, g_2)$
= $Sdist\_v$('degreeFrom','mastersDegreeFrom')
+ $Sdist\_v$('worksFor','memberOf')
+ $Sdist\_v$('researchIn','contributionOf')
= $depth$('degreeFrom') + $depth$('mastersDegreeFrom')
− 2 * $depth$(LCA('degreeFrom','mastersDegreeFrom'))
+ $depth$('worksFor') + $depth$('memberOf')
− 2 * $depth$(LCA('worksFor','memberOf'))

$+ depth(\text{'researchIn'}) + depth(\text{'contributionOf'})$
$- 2 * depth(LCA(\text{'researchIn'}, \text{'contributionOf'}))$
$= 0 + 1 - 2 * 0 + 0 + 1 - 2 * 0 + 0 + 1 - 2 * 0$
$= 3$

其中,属性的最大深度是 6,顶点的最大深度是 7. 上述 $Sdist(q, g_1)$与 $Sdist(q, g_2)$的计算过程省略了结果为零的部分. 由计算结果可见,$Sdist(q, g_1) \geqslant Sdist(q, g_2)$,因此,返回结果为 g_2.

3.2　语义结构剪枝策略

为避免将整个数据库中的图作为候选集,以提高查询效率,本节对数据库做语义结构剪枝,以获取尽可能小的 *SC* 和 *SP*. 目前,图数据库结构剪枝的研究有文献[13,14]. 两者主要为精确查询提供剪枝方案,不能直接应用于近似查询. 因此,本节为近似查询提供"近似精确化"编码方式,即把语义距离小于 ε 的两条边或两个顶点予以相同的编码. 本节还为近似查询提取 RDF 数据的摘要信息. 同时,将近似的编码和摘要应用于文献[14]的剪枝过程中. 下文将数据库中的顶点 s 或 o 表示成 v,属性 p 表示为 e. SPARQL 查询 q 的顶点和边分别表示为 q_v和 q_e. 顶点和边的字符串值用 l 表示.

3.2.1　摘要提取

根据 SPARQL 查询包含的已知顶点和边,本节在 RDF 数据库中提取相应的摘要信息. 其中,编码是摘要提取部分的核心内容. 在预处理中对所有顶点和边进行编码,不仅大大减少 RDF 长字符串的重复存储带来的额外开销,更重要的是编码为剪枝提供了有力的判断依据.

编码的基本方法　本过程首先对本体进行类型划分,并按分类编码. 然后,顶点或边根据自身所属的类型获取相应的编号.

本体类型划分时,为每个非叶子节点获取与之相邻的 n-hop 子节点,并形成一个分类,记为 M. 其中,$n > 0$ 并由具体要求决定它的取值. 假设 $n = 1$,图 4 中的 masterDegreeFrom、undergraduateDegreeFrom 和 DegreeFrom 均属于类型 $M_{\text{DegreeFrom}}$. 同时,该类型的编码为 u.

接下来,对顶点编码时,若 $\exists\, l(v_i) \in M_i$,且 $0 < i < n$,那么 v_i的编号等于 M_i的编码. 同理,对边的编码规则为:给定边 e_j,若 $l(e_j) \in M_j$且 $0 < j < n$,e_j的编码与 M_j的一致. 假设每个节点只属于一个分类,利用两规则对图 4(a)和图 4(c)进行编码,编码结果分别如图 5(a)和图 5(b)所示. 图中可见,除变量外,具体的顶点和边均被简单的编码标识.

带分类表的编码方法　基本编码方法的每个分类中,任意两顶点间的语义距离均小于 $\varepsilon(\varepsilon \leqslant n-1)$,即属于同一分类的两个顶点或者边可判断为近似. 但上述方法人为做了一个假设"每个节点只属于一个分类",而实际情况是一个顶点或者边可能同时属于多个分类. 为此,本节为每个类添加一个分类列表. 每个分类对应一个 $\boldsymbol{LV}(v)$或 $\boldsymbol{LV}(e)$向量,记录该类存在的所有编码. 当 n 取 1,利用带分类表的编码规则对图 4(a)和图 4(c)重新编码,编码结果如图 6 所示.

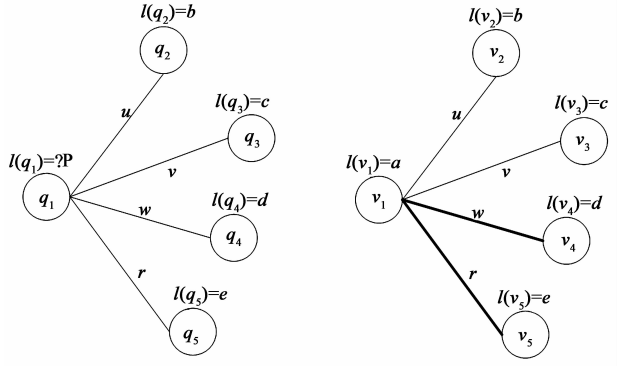


图5　查询q和图g_2编码

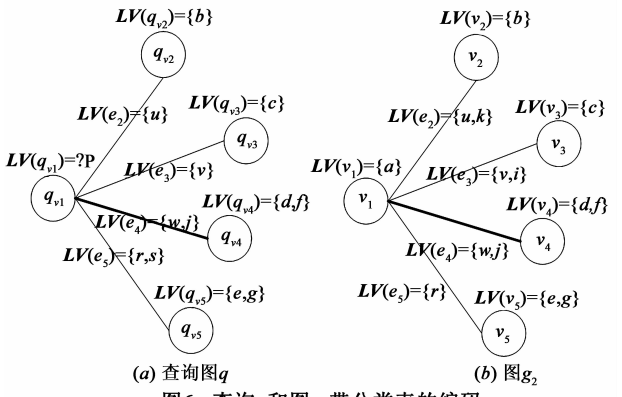


图6　查询q和图g_2带分类表的编码

图 6 显示,$\boldsymbol{LV}(v)$或 $\boldsymbol{LV}(e)$向量全面记录了语义距离小于 ε 时,某一顶点或边所属的所有分类,有效解决因字符串和编码一一对应的局限所带来的问题. 同时,因为本文查询是近似查询,近似程度不宜太小,所以编码阶段的 n 总是小于等于 5,即编码向量的长度小于等于 4. 因此,本文中,带分类表的编码方法不会导致编码向量的长度所占空间比原有字符串的大.

相交取其一的最终编码　本文对字符串的一个重要目的在于将近似的顶点对或边对以相同的编码显示,以更好展开下一步的剪枝工作,即一对顶点是否同类,只要两者的编码列表存在公共编码即可,因此,一个顶点或边包含多个编码,不仅冗余,而且给后续工作带来不必要的麻烦. 本文将"带分类表的编码方法"的整个过程作为编码中的第一个步骤,第二个步骤对顶点对和边对的编码向量做交集,若交集不为空,则随机

取交集的一个编码作为顶点对的编码,否则,分别为该顶点或边选择列表中的某一个编码作为它最终的编码.

查询模式 q 和图 g_2 的编码如图 7 所示.对比图 6 和图 7,包含两个编码的顶点 q_{v_4} 和 v_4,$\boldsymbol{LV}(q_{v_4}) \cap \boldsymbol{LV}(v_4) = \{d, f\}$,随机取 d 作为 q_{v_4} 和 v_4 的编码.

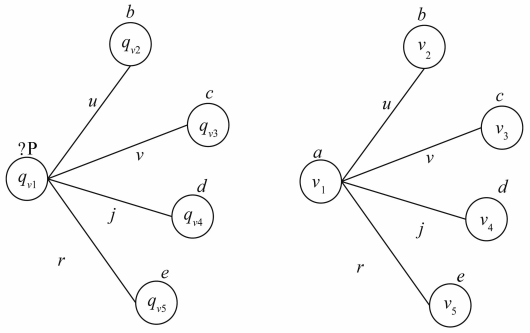


图7 查询q和图g_2的编码

顶点的摘要提取 摘要提取过程对 RDF 数据库的每个顶点 v_i 进行宽度优先扫描,并取每个方向的 k-hop 相邻顶点作为 v_i 的摘要.其中,$0 \leqslant k \leqslant d_{\max}$,$d_{\max}$ 是查询图 q 的最大宽度.顶点 v_i 的摘要中,每个 k 值对应一个 $\boldsymbol{CV}_{\leqslant k}(v_i)$ 向量,以记录 v_i 在 $[0,k]$ 范围内所包含的顶点信息.其中,$\boldsymbol{CV}_{\leqslant k}(v_i)$ 是一个比特向量.

图 7(b)中,整个编码的码表为 $\{a,b,c,d,e,u,v,j,r\}$.$\boldsymbol{CV}_{\leqslant 1}(v_1)$ 总共包含了 $a \sim e$ 五个顶点,通过直接映射,$a \sim e$ 分别在 $\boldsymbol{CV}_{\leqslant 1}(v_1)$ 相应的位置上以 1 标注,其余位置为 0,结果为 $\boldsymbol{CV}_{\leqslant 1}(v_1) = \{1,1,1,1,1,0,0,0,0\}$.同时,图 7($a$)可见,顶点 q_{v_1} 的 $d_{\max}$ 等于 1,即 k 的取值只有 1.

边的摘要提取 边摘要的提取过程与顶点摘要的类似.通过对每个顶点 v_i 进行宽度优先遍历并获取跳数为 k 的边作为边的摘要.其中,k 满足 $0 \leqslant k \leqslant d_{\max}$.同时,向量 $\boldsymbol{CV}^{e}_{\leqslant k}(v_i)$ 记录了顶点 v_i 在 $[0,k]$ 范围内所包含的边.

图 5(b)中,$\boldsymbol{CV}^{e}_{\leqslant 1}(v_1)$ 包含的编码有 u、v、w、r.由于 v_1 的 $d_{\max}$ 值为 1,因此,v_1 包含的边摘要只有 $\boldsymbol{CV}^{e}_{\leqslant 1}(v_1)$.

3.2.2 基于摘要剪枝

利用摘要信息,比较查询 q 与数据库中图 g 之间的差异,初步移除无法成为结果的图,以获取有限集 SC 和 SP.同时,下文的 $l(v_i)$ ($l(e_i)$)均指的是顶点(边)的编码.

顶点与边剪枝 本过程包含两部分,一是顶点的剪枝,二是边的剪枝.对于顶点,给定图 g 的顶点 v_i 与查询 q 的顶点 q_{v_i},当且仅当 $l(q_{v_i}) = l(v_i)$,两顶点匹配.同时,两匹配顶点的 $\boldsymbol{CV}_{\leqslant k}(q_{v_i})[j] = 1$ 和 $\boldsymbol{CV}_{\leqslant k}(v_i)[j] = 1$ 在摘要中成对出现.因此,如果图 g 与查询 q 不匹配,当且仅当顶点 v_i 与 q_{v_i} 满足以下条件:

$$\boldsymbol{CV}_{\leqslant k}(v_i) \wedge \boldsymbol{CV}_{\leqslant k}(q_{v_i}) \neq \boldsymbol{CV}_{\leqslant k}(q_{v_i}) \tag{5}$$

式(5)的证明如下:$\boldsymbol{CV}_{\leqslant k}(v_i)$ 表示图 g 以 v_i 为根节点在 $\leqslant k$-hop 内所包含的编码,$\boldsymbol{CV}_{\leqslant k}(q_{v_i})$ 表示查询 q 以 q_{v_i} 为根节点在 $\leqslant k$-hop 内所包含的编码.如果图 g 与查询模式 q 匹配,那么图 g 包含了 q 中所有字符串,即包含的 q 所有的编码.若式(5)成立,说明 $\boldsymbol{CV}_{\leqslant k}(v_i)$ 与 $\boldsymbol{CV}_{\leqslant k}(q_{v_i})$ 不相同或者 $\boldsymbol{CV}_{\leqslant k}(q_{v_i})$ 不是 $\boldsymbol{CV}_{\leqslant k}(v_i)$ 的子集时,均可推导出图 g 与查询 q 不匹配.

图 7 可见,$\boldsymbol{CV}_{\leqslant 1}(q_{v_1})$ 包含编码 $\{b,c,d,e,u,v,j,r\}$,$\boldsymbol{CV}_{\leqslant 1}(v_1)$ 包含编码为 $\{a,b,c,d,e,u,v,j,r\}$,因此,$\boldsymbol{CV}_{\leqslant 1}(q_{v_1})$ 是 $\boldsymbol{CV}_{\leqslant 1}(v_1)$ 的子集,q 与 g_2 匹配.因此,式(5)不成立.

针对于此,式(5)推导出的顶点剪枝规则为:给定顶点 $v_i \in g$ 和 $q_{v_i} \in q$,且 k 分别取区间 $[0, d_{\max}]$ 的值,若 k 的部分值使式(5)成立,则 g 被安全剪枝.

此外,对于边的剪枝,剪枝规则与顶点的类似.同时,用于推导剪枝规则的公式如下:

$$\boldsymbol{CV}^{e}_{\leqslant k}(e_i) \wedge \boldsymbol{CV}^{e}_{\leqslant k}(q_{e_i}) = \boldsymbol{CV}^{e}_{\leqslant k}(q_{e_i}) \tag{6}$$

最短距离剪枝 该过程利用已有的摘要信息完成顶点对的最短距离剪枝.给定顶点对 $(q_{v_i}, q_{v_j}) \in QPlan$,($i \neq j$,$QPlan$ 是查询 q 经过语法分析后在程序中的表示),且 q_{v_i} 与 q_{v_j} 的最短距离为 $d_{\min}$.对于图的顶点对 v_i 和 v_j,若存在 $k \in [1, d_{\min})$ 的部分取值满足式(7),候选顶点对 (v_i, v_j) 被安全剪去.

$$\boldsymbol{CV}_{\leqslant k}(v_i) \wedge \boldsymbol{CV}_{\leqslant d_{\min}-k}(v_j) = \boldsymbol{O} \tag{7}$$

其中,$\boldsymbol{O}$ 为零向量.

最短距离剪枝证明:v_i 和 v_j 可能与 q_{v_i} 和 q_{v_j} 匹配,假设两顶点集不一定存在交集,那么至少存在一个 $k \in [1, d_{\min})$ 取值使得 v_i 的 k-hop 顶点集 $\boldsymbol{CV}_{\leqslant k}(v_i)$ 与 v_j 的 $(d_{\min}-k)$-hop 顶点集 $\boldsymbol{CV}_{\leqslant d_{\min}-k}(v_j)$ 交集为空,此时,v_i 和 v_j 的最短路径上至少存在一个顶点 $v_t \notin \boldsymbol{CV}_{\leqslant k}(v_i)$ 且 $v_t \notin \boldsymbol{CV}_{\leqslant d_{\min}-k}(v_j)$,因此,$v_i$ 到 v_j 的最短距离应该是 k,$(d_{\min}-k)$,到 v_i''(v_i'' 属于 $\boldsymbol{CV}_{\leqslant k}(v_i)$,在 v_i 到 v_j 的最短路径上,且距离 v_i 为 k-hop),v_t 到 v_j''(v_j'' 属于 $\boldsymbol{CV}_{\leqslant d_{\min}-k}(v_j)$,在 v_i 到 v_j 的最短路径上,且距离 v_j 为 $(d_{\min}-k)$-hop)之和.该结果大于 $d_{\min}$,与原命题 v_i 到 v_j 的最短距离为 $d_{\min}$ 矛盾,因此,$\boldsymbol{CV}_{\leqslant k}(v_i)$ 与 $\boldsymbol{CV}_{\leqslant d_{\min}-k}(v_j)$ 存在交集时,v_i 和 v_j 与 q_{v_i} 和 q_{v_j} 不匹配,即 $\boldsymbol{CV}_{\leqslant k}(v_i)$ 与 $\boldsymbol{CV}_{\leqslant d_{\min}-k}(v_j)$ 的向量交集为零,可进行剪枝.

该剪枝策略的具体实例见图 7. 图中,v_2到 v_5的最短距离不大于 $d_{\min}$值为 2,即 q_{v_2}到 q_{v_5}的最短距离,且 $k=1$. 由于 v_2的($\leqslant 1$) - hop 与($\leqslant 2-1$) - hop 包含共同顶点集,因此,相应向量的位与结果不为零. 根据式(7),保留顶点对(v_2,v_5).

统计信息剪枝　通过统计信息剪枝可有效找出结构相似的候选图. 本剪枝过程在摘要的基础上进行信息统计,统计的信息有顶点 q_{v_i}和 v_i的度、($\leqslant k$ - hop)所包含的顶点数、($\leqslant k$ - hop)所包含的边数,分别记为 $deg(v_i)$ ($deg(q_{v_i})$)、$Num(v_i)$ ($Num(q_{v_i})$)、$Num^e(v_i)$ ($Num^e(q_{v_i})$).

$$deg(v_i) < deg(q_{v_i}) \tag{8}$$

$$Num(v_i) < Num(q_{v_i}) \tag{9}$$

$$Num^e(v_i) < Num^e(q_{v_i}) \tag{10}$$

若 $q_{v_i} \in q$,$v_i \in g$,当且仅当 q 和 g 满足条件(8)(9)(10)的其中之一,则 g 被安全剪除.

统计信息剪枝证明如下:给定顶点 q_{v_i}和 v_i,若存在 $deg(v_i) < deg(q_{v_i})$,则说明至少存在一条与 q_{v_i}相连的边未能与 v_i直接相连边匹配,因此,q 和 g 不可能匹配,安全剪去 g. 同理可证式(9)和式(10).

4　查询响应

4.1　查询框架

本节实现了 RDF 图数据上的 top-k 近似查询. 该查询框架的伪代码如算法 1 所示.

算法 1　*fm*-top-k(D_cand, $QPlan$, k)
输入:候选图集 D_cand,图查询的查询计划 $QPlan$ 和整型值 k
输出:$QPlan$ 的 top-k 近似图
开始
1.　$A \leftarrow D\_cand$ 前 k 个图 //大小为 k 的大堆,堆中的语义距离值
　　//下界记为 $Sdist(q,g)$
2.　$H \leftarrow \varnothing$　//小堆
3.　for all $g' \in D\_cand$ do//不包括 A 中的 k 个图
4.　$H.push(g', Sdist(q,g'))$
5.　end for all
6.　while head(H).$Sdist$ < head (A).$Sdist$&$H \neq \varnothing$do
7.　temp $\leftarrow H.pop()$
8.　$g' \leftarrow$temp.graph
9.　$A.pop()$
10.　$A.push(g', Sdist(q,g'))$
11.　end while
12.　return A 中的 top-k 结果
结束

算法 1 构造了两个数据结构,分别是大堆 A 和小堆 H. 大堆 A 存储当前的 top-k 结果,小堆存储需要进行语义距离计算的候选图集. 其中,堆的入口是一个由图及其 $Sdist$ 形成的键值对. 算法开始时,A 被初始化为数据库的前 k 个图,同时,H 的元素 g 均满足条件:$g \in D\_cand \cap g \notin A \cap g.Sdist(q,g) < \underline{Sdist}(q,g)$(第 1 行到第 5 行). 算法逐渐推进,$H$ 堆顶的图被取出并计算 $Sdist$,若该距离小于 A 当前堆顶的 $Sdist$,A 移除堆顶并将 H 的堆顶入堆,同时,H 移除堆顶并重新排序(第 6 - 10 行). 当 H 为空或 H 堆顶的 $Sdist$ 大于 A 堆顶的 $Sdist$ 时,算法终止并返回 A 的结果集(第 5 行和第 12 行).

4.2　查询响应

本节主要在 4.1 节的框架上实现 3.2 节的剪枝策略,以高效响应 RDF 图的近似查询. 具体响应过程如算法 2 所示.

算法 2　Query-Processing(D, q, k)
输入:数据库 D,图查询的查询 q 和整型值 k
输出:q 的 top-k 近似图
开始
1. $QPlan \leftarrow q$ 的查询计划
2. $S \leftarrow q$ 的已知顶点
3. $E \leftarrow q$ 的已知边
4. 获取每个 $q_{v_i} \in S$ 和 $q_{e_i} \in E$ 的摘要
5. $cand(q_{v_i}) \leftarrow \varnothing$//每个 q_{v_i}的候选图列表
6. $cand(q_{v_i}) \leftarrow$在摘要中为每个 $q_{v_i} \in S$ 以及关联边 $E \leftarrow q$ 进行顶点和边剪枝、统计信息剪枝
7. $cand(q_{v_i}, q_{v_j}) \leftarrow$为顶点对$(q_{v_i}, q_{v_j}) \in QPlan$ 做最短距离剪枝,其中 $q_{v_i} \in cand(q_{v_i}) \cap q_{v_j} \in cand(q_{v_j})$
8. 根据 $QPlan \leftarrow q$ 将顶点对(q_{v_i}, q_{v_j})的候选集 $cand(q_{v_i}, q_{v_j})$进行连接并形成 $QPlan$ 的候选图集合 D_cand. 其中,D_cand 包含 SC 和 SP
9. return *fm*-top-k(D_cand, $QPlan$, k)
结束

给定查询 q、数据库 D 以及返回的结果数 k,算法 2 首先获取 q 的查询计划(第 1 行)并为查询的已知顶点和边提取摘要信息(第 2 ~ 4 行). 然后根据摘要进行剪枝(第 6、7 行). 为提高查询效率,算法 2 执行剪枝策略的顺序为顶点和边剪枝、统计信息剪枝、最短距离剪枝,即剪枝策略的复杂度与之处理的数据量成反比. 最后,算法 2 根据 $QPlan$ 连接候选顶点对,形成候选图集(第 8 行). 同时,该图集被作为查询框架的数据输入(第 9 行),由查询框架完成语义距离的计算并返回最终结果.

5　实验

实验评价主要包括三个方面—近似性度量、查询时间以及 k 值的影响. 本节所有实验结果均来自于多次(>5)单独执行的平均值.

5.1 实验设置

实验设置的第一个参数是实验平台.我们实现了近似查询的框架以及查询的响应过程,并利用 Jena[15] 对 RDF 三元组做图建模.实验环境运行在 64bit 的 Ubuntu 上.其中,该机器拥有一个 2.4GHz 第三代英特尔酷睿四核 CPU、16GB 的 RAM 内存、一个 64GB SSD 固态硬盘和一块 750GB 硬盘.

第二个参数是实验数据.本节实验所用数据为著名的数据基准—LUBM(Lehigh University Benchmark).该数据集的统计信息如表 1 所示.

表 1 数据集的统计信息

数据集	LUBM
元组数	25,511,312
实例数	4,666,423
类型数	43
属性数	32

第三个参数是查询语句.本节实验所用的查询语句如表 2 所示.其中,查询的 k 值根据具体实验而定.

表 2 查询语句

查询语句	查询条件
Q_1	SELECT ? P WHERE{ ? P ub:mastersDegreeFrom ub:University3. ? P ub:worksFor ub:University0. ? P ub:teacherOf ? C.}
Q_2	SELECT ? P ? N WHERE{ ? P ub:mastersDegreeFrom ub:University3. ? P ub:researchInterest 'Research9'. ? P rdf:type ub:Professor. ? P ub:name ? N.}
Q_3	SELECT ? P WHERE{ ? P rdf:type ub:FullProfessor. ? P ub:name 'FullProfessor0'. ? P ub:teacherOf ? C. ? P ub:telephone ? N. ? P ub:doctoralDegreeFrom ub:University5.}
Q_4	SELECT ? P ? C WHERE{ ? P ub:teacherOf ? C. ? P ub:rdf:type ub:FullProfessor ? C rdf:type ub:GraduateCourse.}
Q_5	SELECT ? P ? S WHERE{ ? P ub:advisor ? S. ? S ub:worksFor ub:University0.}

第四个参数是比较对象.本文的比较对象是[9],记为 MCS.本节将本文的方法记为 SMCS.

5.2 近似性度量的评价

本节将查询 Q_1 – Q_5分别在 MCS 和 SMCS 中执行,并对结果集的无关联边和语义相同边进行边数统计,即对于图 g 与查询 q 相对应的两条边,若两条边编码一致,语义相同边计数器加一,否则无关联边计数器加一.该实验的统计结果如图 8 所示.图中,X 轴表示当前结果集所对应的查询语句,Y 轴表示边数.

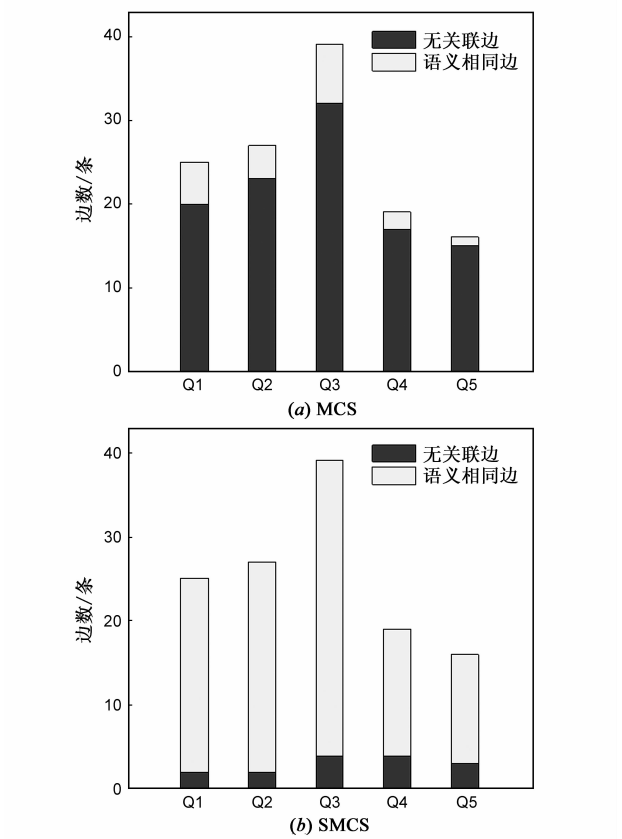


图8 结果集的统计

图 8(a)中,每个结果集的无关联边数远远大于语义相同边的边数.同一查询在 SMCS 框架下所获取的结果集中,语义相同边的边数比例明显增大,同时,无关联边的边数比例下降.该现象的主要原因在于 MCS 把图与查询之间的不同边数作为两者的距离,笼统将语义相同边与无关联边归类为不同边.而 SMCS 为 RDF 语义图设计,在候选图集中,优先选择语义距离较小的图并加入大堆,即对语义相同边和无关联边加以区分,在两者总数恒定的情况下,以包含语义相同边较多的图为最佳.因此,SMCS 比 MCS 更能保证结果图与查询语句的高度近似.

5.3 查询性能的评价

本组实验取 k = 50 并在相同环境下测试 MCS 和 SMCS 的查询时间.测试结果如图 9 所示,图中 X 轴表示查询语句,Y 轴是响应时间.

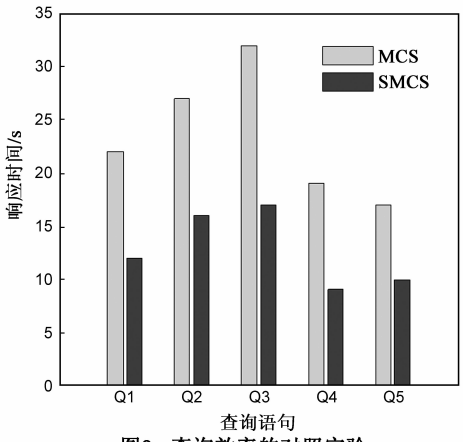


图9　查询效率的对照实验

图 9 可见,从 Q_1到 Q_5,SMCS 的查询效率一直高于 MCS 的.回顾近似查询的响应过程,该过程主要包括候选图集的获取以及近似值的计算.第一阶段,MCS 通过采用多种方法不断估计 *dist* 的上界和下界以获取较小的中间结果.而 SMCS 利用不同的策略对数据库进行剪枝并获取结构相近的候选图集.通过观察两者的实现算法可知,MCS 的复杂度比 SMCS 的高.复杂度过高的估计带来过多的额外开销并延长响应时间.同时,计算近似值时,由于 SMCS 的候选图集比 MCS 的小,所以 SMCS 计算耗时较 MCS 短.因此,相比 MCS,SMCS 的查询响应效率更高.

5.4　*k* 取值对关联边数及查询性能的影响

本节主要讨论 k 取值对关联边数和查询效率的影响.实验结果分别如图 10(a)和图 10(b)所示.第一组实验结果中,语义关联边数随 k 取值增长,五个查询结果形成的折线均非线性增长,但与相应 k 值的差异范围较小.本文方法有效保证查询框架返回与查询语句语义最近的图.第二组实验中,随着 k 的直线上涨,查询响应时间依次递增,但波动范围较小.分析查询的框架可知,k 取值主要影响大堆的大小,即影响大堆的插入和删除耗时.然而,查询的主要耗时部分是图的近似计算和候选图集的获取,计算的总图数(小堆的大小)和候选图集均不受 k 影响.因此,查询时间随 k 取值的变化而变化,但变动范围不明显.

6　总结

本文在考虑语义近似的情况下,为 RDF 图近似查询提出基于本体的度量方法并在此基础上设计查询.同时,为提高查询效率,本文提出语义结构剪枝策略.该策略根据查询语句提取 RDF 图的摘要信息,并利用这些摘要完成本文查询的结构剪枝.最后,我们构造查询框架以实现 RDF 图的 top-k 近似查并在该框架下设计测试实验以评价本文的方法.实验表明,本文方法可高效执行查询并有效返回结果.

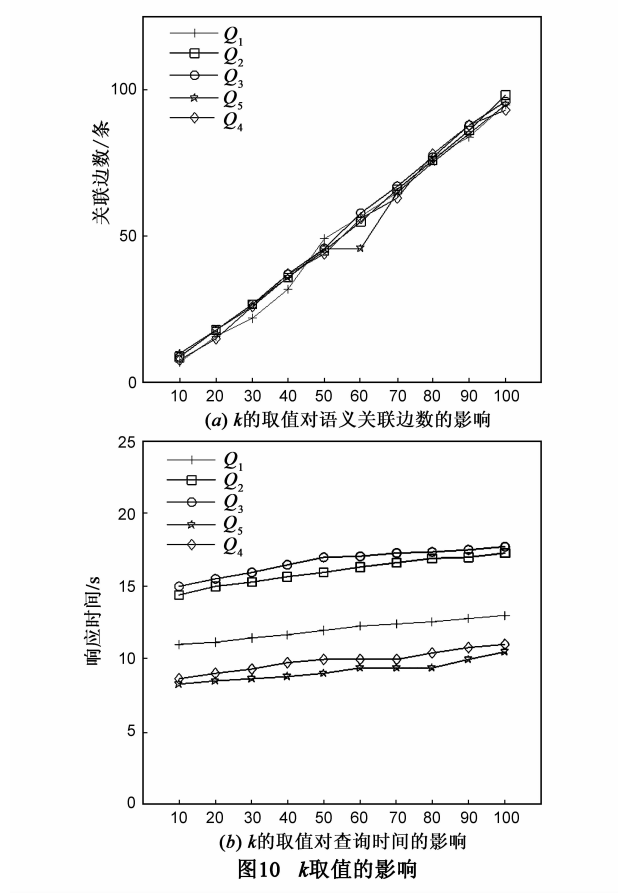


图10　*k*取值的影响

参考文献

[1] RDF primer. W3C Recommendation[EB/OL]. http://www.w3.org/TR/rdf-primer.

[2] Thomash N, Gerhard W. The RDF-3X engine for scalable management of RDF data[J]. VLDB Journal, 2010, 19(1): 91 – 149.

[3] Mihaela A B, et al. Building an efficient RDF store over a relational database[A]. ACM Conference on Management of Data[C]. New York: ACM, 2013. 121 – 132.

[4] Jiewen H, et al. Query optimization of distributed pattern matching[A]. IEEE International Conference on Data Engineering[C]. Washington: IEEE Press, 2014. 64 – 75.

[5] Jiewen H, et al. Scalable SPARQL querying of large RDF graphs[J]. The Proceedings of the VLDB Endowment, 2011, 4(11): 1123 – 1134.

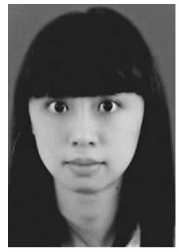
[6] Weiguo Z, et al. Graph similarity search with edit distance constraint in large graph databases[A]. ACM International Conference on Information and Knowledge Management[C]. New York: ACM, 2013. 1595 – 1600.

[7] Haichuan S, et al. Connected substructure similarity search[A].

ACM Conference on Management of Data[C]. New York: ACM,2010.903 – 914.
[8] Haichuan S, et al. Similarity search on supergraph containment [A]. IEEE International Conference on Data Engineering[C]. Washington: IEEE Press,2010.637 – 648.
[9] Yuanyuan Z, et al. Finding top-k similar graphs in graph databases[A]. International Conference on Extending DB Technology[C]. London: Springer,2012.456 – 467.
[10] Yuan Y, et al. Efficient subgraph similarity search on large Probabilistic Graph Databases[J]. Proceedings of the VLDB Endowment,2012,5(9):800 – 811.
[11] 姚绍文,余江,周明天.面向语义 Web 的逻辑描述原语扩展[J].电子学报,2002,30(S1):2115 – 2118.
Yao Shaowen, Yu Jiang, Zhou Mingtian. Semantic web-oriented specification of logic descriptive primitives[J]. Acta Electronica Sinica,2002,30(S1):2115 – 2118. (in Chinese)
[12] 王海,高岭,范琳,李增智.基于 SPARQL-DL 的语义 Web 服务查询[J].电子学报,2011,39(3A):52 – 56.
Wang Hai, Gao Ling, Fan Lin, Li Zengzhi. Query for semantic Web service based on SPARQL-DL[J]. Acta Electronica Sinica,2011,39(3A):52 – 56. (in Chinese)
[13] Xiang L, Lei. Efficient query answering in probabilistic RDF graphs[A]. ACM Conference on Management of Data[C]. New York: ACM,2011.157 – 168.
[14] Peixiang Z, Jiawei H. On graph query optimization in large networks[J]. The Proceedings of the VLDB Endowment, 2010,3(1):340 – 351.
[15] ARQ[EB/OL]. http://incubator.apache.org/jena/ documentation/ query/index.html.

作者简介

章登义 男,1955 年 2 月出生于湖北省荆州市.现为武汉大学计算机学院教授、博士生导师.主要研究方向为安防、嵌入式和数据库.
E-mail: dyzhang@whu.edu.cn



吴文李 女,1986 年 12 月出生于广东省湛江市.现为武汉大学博士研究生.主要研究方向为语义网数据挖掘.
E-mail: huihuigou@whu.edu.cn