KREAG:基于实体三元组关联图的 RDF 数据关键词查询方法

李慧颖[1)] 瞿裕忠[2)]

[1)](东南大学计算机科学与工程学院 南京 210096)

[2)](南京大学计算机科学与技术系 南京 210093)

摘 要 语义网数据的大量增加使得 RDF 数据查询成为一个重要研究主题.关键词查询方式不需要掌握数据模式或查询语言,更适合普通用户使用.文中提出一种 RDF 数据关键词查询方法 KREAG(Keyword query over RDF data based on Entity-triple Association Graph).为了支持用户对属性或关系名进行查询,将 RDF 数据建模为顶点带标签的实体三元组关联图.该模型保证了 RDF 数据中实体间关联转化为关联图中顶点间的通路,且文本信息全部封装到关联图顶点标签上.在此基础上,将关键词查询问题转化为关联图上查找有向斯坦纳树问题.在保证近似比为 m 的前提下(m 为查询关键词的个数),利用近似算法实现快速查询响应.通过合理的评分方式衡量查询结果的相关性,支持 top-k 查询.算法的时间复杂度为 $O(m \cdot |V|)$,其中 $|V|$ 为实体三元组关联图中顶点个数.实验表明 KREAG 较其它方法具有更快的响应时间,同时能够有效地实现 RDF 数据的关键词查询.

关键词 关键词查询;RDF 数据;top-k;实体;关联

中图法分类号 TP311 **DOI 号**:10.3724/SP.J.1016.2011.00825

KREAG: Keyword Query Approach over RDF Data Based on Entity-Triple Association Graph

LI Hui-Ying[1)] QU Yu-Zhong[2)]

[1)](*School of Computer Science and Engineering, Southeast University, Nanjing* 210096)

[2)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

Abstract The increment in amount of Semantic Web data has made RDF data query an important research topic. Keyword query is regarded as an intuitive paradigm, especially for the users who are not familiar with the data and the RDF query language. In this paper, an approach named KREAG (Keyword query over RDF data based on Entity-triple Association Graph) is proposed, which enabling keyword-based query over RDF data. For supporting attribute and relation query, RDF data is modeled as a node-labeled Entity-Triple Association Graph, in which entity associations are translated into paths between nodes and all text information is encapsulated by nodes' label. Furthermore, the keyword query problem over RDF data is translated into a directed Steiner Tree problem in an Entity-Triple Association Graph. With an m-approximation ratio guarantee (m is the number of query keywords), KREAG employs the Steiner Tree approximation algorithm to support rapid query response. Its time complexity is $O(m \cdot |V|)$, where $|V|$ is the number of nodes of the Entity-triple Association Graph. Moreover, KREAG introduces a reasonable ranking function and supports top-k queries. The experimental results show that KREAG has a faster response time than other keyword query approaches and is more effective.

Keywords keyword-based query; RDF data; top-k; entity; association

收稿日期:2010-09-02;最终修改稿收到日期:2011-03-21.本课题得到国家自然科学基金(60973024)与江苏省自然科学基金(BK2008290)资助.**李慧颖**,女,1977 年生,博士研究生,讲师,研究方向为语义 Web、软件技术.E-mail:huiyingli@seu.edu.cn.**瞿裕忠**,男,1965 年生,博士,教授,博士生导师,主要研究领域包括语义 Web、软件方法与技术.

1 引 言

语义网[1](Semantic Web)的快速发展使得语义数据大量增加.据统计[2],万维网上已经发现超过1000万个包含语义数据的文档,其中大多数是以RDF、RDFS、OWL语言描述的.本文称这些由本体语言描述、以三元组形式组织起来的数据为语义网数据(也称RDF数据).

随着RDF数据的大量增加,普通用户对其进行查询的需求也在不断增加.目前已经有一些查询语言如SPARQL[3]、SeRQL[4]等支持RDF数据查询,但它们对普通用户而言过于复杂,原因在于其要求用户必须掌握查询语言的语法规则和待查询数据的模式信息.万维网搜索引擎中基于关键词的搜索技术得到广泛应用的事实表明普通用户更倾向于简便的查询方式.因此,提供关键词查询方式对RDF数据的检索和重用成为一个重要问题.

本文提出一种基于实体三元组关联图的RDF数据关键词查询方法KREAG(Keyword query over RDF data based on Entity-triple Association Graph).为支持用户对RDF数据的关系和属性进行查询,提出实体三元组关联图模型.该模型封装RDF数据文本信息到关联图顶点标签上,通过实体顶点到三元组顶点再到实体顶点的通路表示实体间关联.给出关键词查询问题的定义,将其转化为实体三元组关联图上查找有向斯坦纳树问题.利用近似算法实现快速查询响应且保证查询结果具有较好的近似比,支持top-k查询.

本文第2节介绍相关研究工作;第3节给出实体三元组关联图模型及相关定义;第4节描述关键词查询算法;第5节给出实验结果和分析;第6节对本研究工作进行总结.

2 相关工作

目前,在RDF数据中进行关键词查询以获取信息的方法主要分为两类,一类由关键词查询构造形式化查询语句再得到查询结果,称为查询转换方法[5-10];另一类由关键词查询直接从RDF数据中构造查询结果,称为直接查询方法[11-15].

查询转换方法主要关注将关键词查询转换为形式化查询的过程.首先,匹配查询关键词到RDF图的顶点或边.然后,在模板或模式信息的辅助下,找到查询关键词之间的关联,确定用户的查询对象.最后,构造符合语法规则的形式化查询语句,将其排序返回.用户通过选择查询语句向RDF数据库发起查询并获得最终查询结果.这类方法不需要建立大规模的结构索引而是依赖RDF模式信息(RDF schema)确定查询关键词之间的关联,但目前万维网上RDF数据大部分没有或缺少模式信息.

IQTQA[10]是查询转换方法中比较特殊的一个工作.它可以将关键词查询转换为形式化查询语句返回,也可以在此基础上,直接构造查询结果返回.该方法从RDF数据中抽取结构信息,构造查询搜索图.从中搜索符合要求的子图(称为查询图)生成形式化查询,再进一步结合实体索引和关系索引直接得到查询结果.但是,IQTQA抽取结构信息的时间开销大.并且,由于其响应时间等于查询转换时间加查询结果生成时间,实时响应速度并不理想.

直接查询方法以RDF图为基础,定义查询结果(确定满足何种条件的子图为查询结果),建立相关索引以支持快速查询响应,利用查询算法找到候选查询结果,对候选结果进行评分并将前k个返回给用户.直接查询方法的优势为直接在RDF图中构造查询结果返回,查询的粒度细,不需要背景知识也不需要RDF模式信息的支持.

BLINKS[13]是直接查询方法中的一个代表性工作.通过建立关键词索引和路径索引支持快速实时响应,利用启发式规则寻找候选查询结果,以候选结果的路径长度和之倒数为其评分,返回top-k个结果.该方法同时支持对数据的分块索引.但是,BLINKS不支持对边标签的图进行关键词查询,无法处理用户将属性或关系名作为关键词进行查询的情况.

本文根据RDF数据的特点,提出实体三元组关联图模型并给出其性质,封装RDF数据文本信息到关联图顶点标签.在结构索引基础上,利用斯坦纳树近似算法实现了快速查询响应,解决了RDF数据的关键词查询问题.支持用户以属性或关系名为关键词进行查询.另外,可以通过改变阈值,灵活地控制索引时间和空间,提高索引效率.

目前已经存在一些标准查询语言如SPARQL支持RDF数据查询,但其对普通用户而言仍然十分复杂.本文提出的关键词查询方式虽然不支持形式化查询方法,但恰好与该查询方式形成互补,使得普通用户能够更方便地检索和重用RDF数据.

3　实体三元组关联图模型及相关定义

3.1　实体三元组关联图模型

RDF 数据指以 RDF 三元组形式组织起来的数据，通常由 RDF、RDFS 或 OWL 语言描述，可以通过有向图模型表示. 以 RDF 三元组的主体(*subject*)和客体(*object*)作为顶点，以三元组的谓词(*predicate*)作为从主体指向客体的有向边. 依据 Klyne 的方法①，简单给出 RDF 图的定义.

定义 1(RDF 图).　设 U,B,L 分别表示 URIref 集合，空白顶点集合和字面量集合. 一个 $(subject, predicate, object)\in(U\cup B)\times U\times(U\cup B\cup L)$ 称为 RDF 三元组. RDF 图为 RDF 三元组的集合(记为 T)，其顶点集合为 T 中三元组主体和客体构成的集合.

本文称三元组主体和客体上出现的 U 中元素为具名实体，B 中元素为匿名实体，两者统称实体，用 $entity(T)$ 表示 T 的实体集合. 如果存在从实体 e_1 到 e_n 的实体关系序列 $e_1\,p_1\,e_2\,p_2\cdots p_{n-1}\,e_n$，称 e_1 关联 e_n. 另外，我们用 $subj(t)$，$pred(t)$，$obj(t)$ 分别表示三元组 t 的主体、谓词和客体，用 T_R 和 T_P 分别表示 T 中关系三元组集合和属性三元组集合.

图 1 给出一个真实的 RDF 数据片段，简洁起见，图中仅以 URI 的本地名称(local name)标识边.

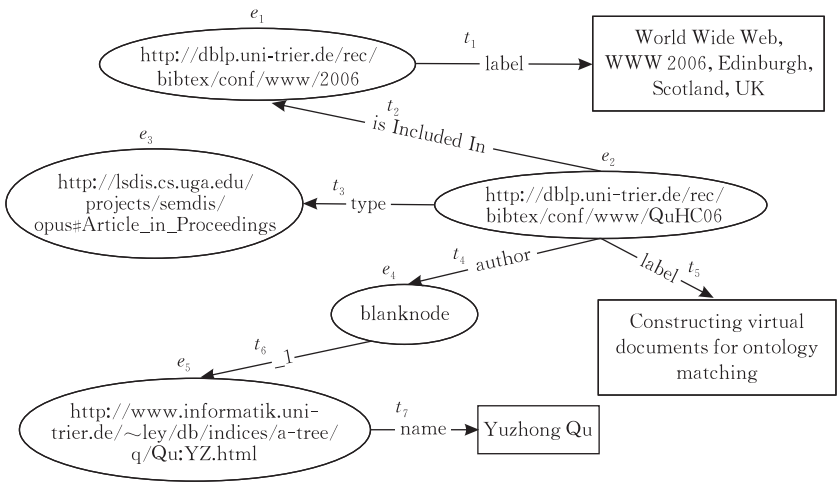

图 1　RDF 图示例

目前支持图结构数据的关键词查询方法[13-14]只能处理顶点标签图. 但在 RDF 图中，边标签具有重要的信息，通常表示实体间关系或实体的属性. 在关键词查询场景下，用户自然地会以关系名或属性名作为查询关键词之一. 以图 1 为例，用户会以关键词“WWW article author Yuzhong”查询作者为 Yuzhong 且发表在 WWW 上的论文. 然而，上面提到的方法无法处理这类查询.

为封装 RDF 数据文本信息到图顶点标签上，且不破坏实体之间的关联，本文提出 RDF 数据的实体三元组关联图模型，定义如下.

定义 2(实体三元组关联图).　设 RDF 图为 T，其对应的实体三元组关联图记为 $D(T)=\langle V_E\cup V_T, E, l_{V_T}\rangle$，其中 $V_E=\{v_e\mid e\in entity(T)\}$，$V_T=\{v_t\mid t\in T\}$，边集合

$$E=\{\langle v_t,v_e\rangle\mid(v_t\in V_T\wedge v_e\in V_E\wedge obj(t)=e)\}\cup\{\langle v_e,v_t\rangle\mid(v_t\in V_T\wedge v_e\in V_E\wedge subj(t)=e)\}$$

称 V_E 中元素为实体顶点，V_T 中元素为三元组顶点. 设 T 中所有关键词的集合为 W，定义函数 $l_{V_T}:V_T\to\rho(W)$，对任意 $v_t\in V_T$，$l_{V_T}(v_t)$ 等于三元组 t 的主体、客体、谓词的 URI 本地名称或字面量中包含的关键词集合.

直观地说，实体三元组关联图是以 RDF 图的实体集合和三元组集合作为两个不相交的顶点集合，如果一个实体是某三元组的主体，则实体三元组关联图中存在从该实体顶点指向该三元组顶点的一条有向边，如果一个实体是某三元组的客体，则存在从该三元组顶点到该实体顶点的一条有向边. 因此，实体三元组关联图是一个有向二部图，顶点集合划分为 V_E，V_T 两个子集，任意边的两个端点分别属于 V_E 和 V_T.

图 2 给出图 1 对应的实体三元组关联图，图中三元组顶点下方是其 l_{V_T} 函数值. 可以观察，与图 1

① Klyne G, Carroll J. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

相比，实体三元组关联图是一个将文本信息封装到三元组顶点上的有向图.虽然实体顶点上没有文本标签，但由于任何实体必是某三元组的主体或客体，根据定义 2，其文本信息已封装到对应的三元组顶点上.因此，从 RDF 图转化为实体三元组关联图的过程中没有丢失文本信息.另外，实体三元组关联图通过实体顶点到三元组顶点再到实体顶点的通路保持了实体之间的关联.

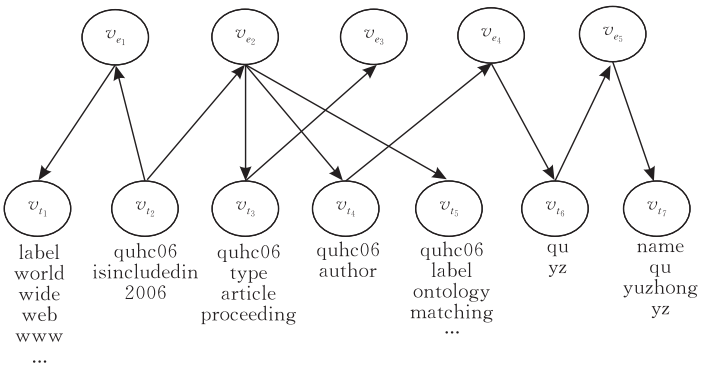

图 2 实体三元组关联图示例

实体三元组关联图模型具有如下性质.

性质 1. 设 RDF 图 T 对应的实体三元组关联图为 $D(T)=\langle V_E \cup V_T, E, l_{V_T}\rangle$，则有 $|E|=2|T_R|+|T_P|$.

证明.

对于每个 $t=(s,p,o)\in T$，如果 $t\in T_R$，则 t 表达了实体 s 和 o 的关系，根据定义 2，$\langle v_s, v_t\rangle \in E$，$\langle v_t, v_o\rangle \in E$.即 RDF 图中一个表示实体间关系的三元组转换为实体三元组关联图中的两条边.如果 $t\in T_P$，则 t 表达了实体 s 和字面量 o 的属性值对，有 $\langle v_s, v_t\rangle \in E$，$\langle v_t, v_o\rangle \notin E$.即 RDF 图中一个表示属性的三元组转换为实体三元组关联图中的一条边.所以，$|E|=2|T_R|+|T_P|$. 证毕.

性质 2. 设 RDF 图 T 对应的实体三元组关联图为 $D(T)=\langle V_E \cup V_T, E, l_{V_T}\rangle$，$D(T)$ 中从实体顶点 v_{e_1} 到 v_{e_n} 存在有向通路当且仅当 T 中实体 e_1 关联 e_n.

证明. 必要性证明.设从 v_{e_1} 到 v_{e_n} 的有向通路经过顶点序列记为 $v_{e_1} v_{t_1} v_{e_2} v_{t_2} \cdots v_{e_{n-1}} v_{t_{n-1}} v_{e_n}$，根据定义 2 可知，对于任意 $1\leq i\leq n-1$ 有，$subj(t_i)=e_i$，$obj(t_i)=e_{i+1}$.设 $pred(t_i)=p_i$，则 T 中存在从 e_1 到 e_n 的实体关系序列 $e_1 p_1 e_2 p_2 \cdots p_{n-1} e_n$，实体 e_1 关联 e_n.

充分性证明.因为实体 e_1 关联 e_n，设 RDF 图中从 e_1 到 e_n 的实体关系序列为 $e_1 p_1 e_2 p_2 \cdots p_{n-1} e_n$，也可以用三元组表示为 $t_1 t_2 \cdots t_{n-1}$，其中 $t_i=(e_i, p_i, e_{i+1})$.因为 $sub(t_i)=e_i$，$obj(t_i)=e_{i+1}$，根据定义 2 可知，在 $D(T)$ 中，从 v_{e_i} 到 v_{t_i} 有边，从 v_{t_i} 到 $v_{e_{i+1}}$ 有边.因此，从 v_{e_1} 到 v_{e_n} 存在有向通路，经过的顶点序列为 $v_{e_1} v_{t_1} v_{e_2} v_{t_2} \cdots v_{e_{n-1}} v_{t_{n-1}} v_{e_n}$. 证毕.

根据性质 2 可知从 RDF 图转化为实体三元组关联图的过程中没有破坏实体间的关联.

设 RDF 图为 T，将其转化为实体三元组关联图只需扫描一遍 T 中所有三元组，其时间复杂度为 $O(|T|)$.根据性质 1，存储实体三元组关联图所需要的空间开销为 $2\cdot|T_R|+|T_P|$，其中 $|T|=|T_R|+|T_P|$.

3.2 相关定义

本节在实体三元组关联图模型基础上，形式化地给出 RDF 数据关键词查询的相关定义.

给定实体三元组关联图 $D(T)=\langle V_E \cup V_T, E, l_{V_T}\rangle$ 和关键词查询 $Q=\{q_1, q_2, \cdots, q_m\}$.首先定位关键词击中的三元组顶点，如果 $q_i \in l_{V_T}(v_t)$，称 q_i 击中三元组顶点 v_t.在此基础上，Q 的查询结果定义如下.

定义 3(查询结果). 设查询关键词 q_i 击中的三元组顶点集合为 KT_i，其中元素称为关键顶点.Q 的查询结果定义为一棵满足如下条件的有向树 T_R(设树根为 r)：

(1) r 为具名实体顶点；

(2) 每片树叶都是关键顶点；

(3) 对每个 KT_i，r 可达其中一个顶点.

如果查询关键词 $q_i \in l_{V_T}(v_t)$，则可以确定 q_i 击中了三元组顶点 v_t，但孤立地将这些不连通的三元组顶点作为结果返回显然不能满足用户的查询要求.通常需要将这些顶点关联起来作为查询结果.本文定义查询结果为以某个具名实体顶点为根的有向树，树根可达每个关键顶点集合中的一个元素，且树叶必须为关键顶点.只要查询结果的树根不同，则认为查询结果不相同.

对于规模较大的实体三元组关联图，一个查询会有多个满足要求的查询结果，需要定义评分函数对查询结果进行评分. 目前公认的方法[7,13]是利用结果的紧凑性进行评分. 本文定义查询结果的评分方式如下.

定义 4(查询结果评分). 设 $Q=\{q_1,q_2,\cdots,q_m\}$的查询结果为 T_R，定义其评分为

$$score(T_R,Q)=size(T_R),$$

其中，$size(T_R)$为树中顶点的个数.

这样的评分方式使得越紧凑的查询结果评分越低，最终将查询结果按照评分的升序排列，返回前 k 个(top-k).

实际上，给定关键词查询 Q，求评分最低的查询结果可以转化为图论经典问题——组斯坦纳树(Group Steiner Tree)问题. 依据文献[16]，给出组斯坦纳树问题的定义.

定义 5(有向组斯坦纳树). 设有向带权图 $D_G=\langle V_G,E_G\rangle$，给定树根 r 和顶点集合族 $X=\{g_1,g_2,\cdots,g_m\}$，$g_i\subseteq V_G$，有向组斯坦纳树 T_G 定义为权值最小的有向树，且满足

(1) T_G的树根为 r；

(2) 对每个集合 g_i 中的某个顶点 v_i，存在从 r 到 v_i的唯一有向通路.

直观地说，组斯坦纳树问题是在带权有向图中指定一个树根和一个顶点集合族，要求找到一棵权值最小的树，且对每个顶点集合，树根可达其中某个点. 由于要求找到权值最小的树，就决定了树叶必须是 g_i中元素.

如果将关键词查询问题中的$\{KT_1,KT_2,\cdots,KT_m\}$认为是顶点集合族，设实体三元组关联图中边的权值均为 1，求 top-k 查询结果可以转化为：将实体三元组关联图中每个具名实体顶点 r 指定为根，求给定树根 r 和顶点集合族$\{KT_1,KT_2,\cdots,KT_m\}$的组斯坦纳树，将其按权值升序排列返回前 k 个.

虽然组斯坦纳树是 NP 完全问题，但可以规约为斯坦纳树问题利用近似算法[16-17]解决.

文献[17]中首先假设如果图中顶点 u 和 v 之间存在长度为 d 的最短通路，则在图中加入一条从 u 到 v 的权值为 d 的有向边. 接着，定义有向树中树叶的层数为从树根到树叶的通路中边的条数. 如果所有树叶的层数最大值为 l，则定义该树为 l-层树. 最后指出，对任意 $l\geq 1$，可以求 l-层树作为组斯坦纳树问题的近似解，其近似比为 $m^{\frac{1}{l}}$. 其中，m 为定义 5 中顶点集合族的基数. 因此，组斯坦纳树问题可以通过求 l-层树作为近似解，l 值越大越逼近最优解. 同时，文献[17]中也阐明随着 l 值的增大，求解 l-层树的算法响应时间也随之增大，其时间复杂度为 $O(\alpha+|V|^{l-1}\cdot m^l)$，其中 α 为计算所有顶点对的最短通路时间，V 为图中顶点集合.

对于关键词查询问题，通常用户输入的关键词个数有限，但要求快速的查询响应时间. 为了提高实时响应速度，本文事先在索引过程中计算出图中所有顶点对的最短通路. 因此，实时求解 l-层树的算法时间复杂度为 $O(|V|^{l-1}\cdot m^l)$. 可以观察，当 l 取 1 时，可以在常数时间内求出指定树根的 1-层树，即使在实体三元组关联图中每个具名实体顶点都指定为根的情况下，算法也能快速响应. 同时，可以保证 1-层树的近似比为 m(m 值通常较小). 当 l 取 2 时，求解指定树根的 2-层树的时间复杂度为 $O(|V|\cdot m^2)$，如果要求出所有的以具名实体顶点为根的 2-层树，响应时间明显增加. 当 l 取值大于 2 时，更加无法快速实时响应. 本文权衡了近似比和算法时间复杂度后，选择求 1-层树作为近似解. 1-层树的含义就是从树根到关键顶点最多经过 1 条边可达，前提是事先计算出任意两点 u,v 之间的最短通路长度，并在图中加入一条以其为权值的从 u 到 v 的有向边. 因此，只要 u 到 v 可达，必有从 u 到 v 的一条边. 直观地理解，近似解 1-层树就是对每个 KT_i，树根通过最短通路到达其中一个顶点而构成的树. 求解 1-层树的算法时间复杂度分析见 4.2 节，可以保证近似比为 m.

4　关键词查询算法

4.1　索引建立过程

为提高关键词查询效率，本文对实体三元组关联图进行了预处理，主要指建立关键词索引和最短通路索引.

关键词索引用于帮助快速定位被击中的关键顶点，该技术在信息检索领域已经被广泛使用，本文不再赘述.

建立最短通路索引是因为 3.2 节中提到，求近似解前需计算图中任意两点 u,v 之间的最短通路长度(设为 d)，并向图中加入一条从 u 到 v 以 d 为权值的有向边. 本文不改变实体关系图的结构，而是通过建立索引达到相同目的.

设关键词 q 击中顶点 v，称顶点 u 可达关键词

q,如果 u 可达 v. 另外,如果 q 击中多个顶点 v_1, v_2,…,v_n,顶点 u 到每个 v_i 都有最短通路,我们只索引其中最短的一条,称 u 到关键词 q 的最短通路. 原因是查询结果只要求树根可达关键顶点集合中的一个元素即可. 因此,本文将关键词索引和最短通路索引结合起来建立的索引结构如图 3 所示. 对每个关键词,首先索引被该关键词击中的顶点,然后索引通过最短通路可达该关键词的顶点,每个顶点构成一个索引记录,记录结构为(*distance*,*lastnode*,*node*),其中,*node* 是当前顶点,*distance* 记录当前顶点到关键词的最短通路长度,*lastnode* 记录该通路中当前顶点的下一个点. 关键词的索引记录按照 *distance* 的值升序排列. 例如关键词"yuzhong"的第 1 个索引记录是(0,v_{t_7},v_{t_7})表示"yuzhong"击中 v_{t_7},v_{t_7} 是关键顶点. 第 2 个记录是(1,v_{t_7},v_{e_5})表示 v_{e_5} 到关键词的最短通路长度为 1,该通路中 v_{e_5} 的下一个点是 v_{t_7}.

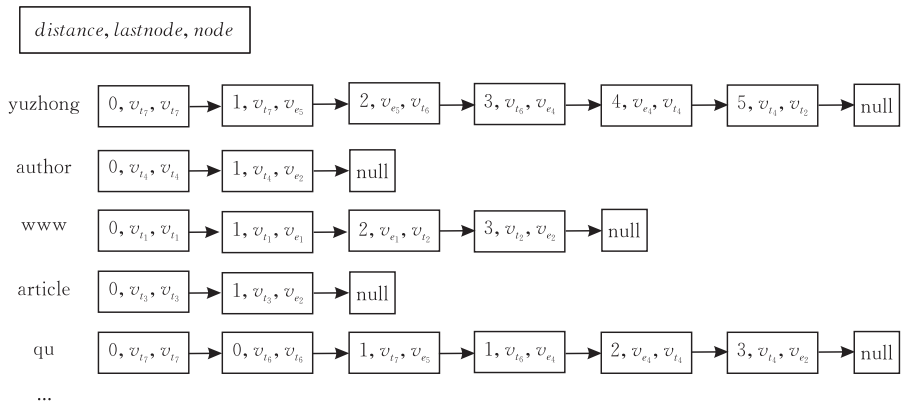

图 3 索引结构

因为实体三元组关联图中边的权值均为 1,本文采用广度优先搜索的方法求两点之间的最短通路. 设实体三元组关联图 $D(T)$ 的顶点集合为 V,边集合为 E,所有关键词的集合为 W. 从任一点出发的广度优先搜索的时间复杂度为 $O(|V|+|E|)$,计算所有顶点对之间的最短通路时间复杂度为 $O(|V|\cdot(|V|+|E|))$. 在最坏情况下,索引占用的空间是 $3\cdot|W|\cdot|V|$. 图的连通性是影响索引大小的关键因素.

当面对大规模实体三元组关联图时,索引全部顶点对之间最短通路的时间和空间开销比较大. 在具体应用中可以只索引小于等于某个阈值 d(索引步长参数)的最短通路,即 d 步内可达的顶点间最短通路. 设 $\overline{deg^-(D(T))}$ 表示实体三元组关联图 $D(T)$ 中所有顶点的平均入度,$\bar{\rho}$ 表示每个关键词平均击中的顶点数目,则索引 d 步内可达顶点对占用的空间是 $3\cdot|W|\cdot(\bar{\rho}\cdot\overline{\deg^-(D(T))}^d)$. 根据定义 2 可知实体三元组关联图 $D(T)$ 中所有三元组顶点的入度均为 1,实体顶点的入度等于 T 中所有以该实体为客体的三元组数目,即等于 *RDF* 图中该实体的入度. 据统计[18],目前语义网中 *RDF* 数据实体的平均度数为 3.44. 因此,$\overline{\deg^-(D(T))}$ 通常为一个很小的数值. 通过调整 d 值,可以控制索引时间和空间开销,5.2.1 节给出了不同 d 值下索引时间和空间开销的数值.

4.2 关键词查询算法

给定实体三元组关联图 $D(T)=\langle V_E\cup V_T,E,l_{V_T}\rangle$ 和关键词查询 $Q=\{q_1,q_2,\cdots,q_m\}$. 关键词查询算法首先取出关键词 $q_i(i=1,2,\cdots,m)$ 对应的索引. 然后依次扫描 m 条索引的每个记录,判断当前顶点是否为具名实体顶点且同时可达 m 个查询关键词. 如是,找到一个以该顶点为树根的查询结果,利用索引恢复树根到关键的最短通路,为其评分并插入到查询结果列表中. 算法继续,直到找到前 k 个结果. 关键词查询算法描述如下.

算法 1. 关键词查询算法.

输入:查询关键词 $Q=\{q_1,q_2,\cdots,q_m\}$,实体三元组关联图 $D(T)=\langle V_E\cup V_T,E,l_{V_T}\rangle$,$k$

输出:前 k 个查询结果

```
1. Begin
2.   //cursorqueue 为查询关键词索引指针队列
3.   //resultlist 为查询结果列表,按评分升序排列
4.   //nodeset_i 为可达 q_i 的顶点集合,初始化为空
5.   ansnumber=0;
6.   flag=TRUE;
7.   threshold=Integer.MaxValue
8.   cursorqueue←Q 的索引指针队列;
9.   while(flag)
10.  {
11.    cursor←cursorqueue.dequeue;
12.    node←cursor.node;
13.    distance←cursor.distance;
```

```
14.  if (ansnumber>k & distance>threshold)
15.  {
16.      return resultlist.sublist(0,k-1);
17.      break;
18.  }
19.  nodeset_cursor.add(node);
20.  if ((node∈nodeset_i, i=1,2,…,m)&(node∈U))
21.  {
22.      result←以 node 为根的查询结果;
23.      if(!resultlist.contains(result))
24.      {
25.          resultlist.insert(result);
26.          ansnumber++;
27.          threshold=第 k 个结果评分;
28.      }
29.  }
30.  cursor 移向下一个索引记录;
31.  cursorqueue.enqueue(cursor);
32. }
33. End
```

由于只需返回前 k 个查询结果，算法中保持一个结果列表，里面存储已找到的结果中最小的前 k 个，将第 k 个结果的评分作为阈值. 如果当前索引指针返回的距离大于或等于该阈值，算法结束. 因为如果索引指针返回的距离已经大于或等于该阈值，对应的结果评分必然也大于或等于阈值. 而且，因为索引记录按照距离的升序排序，后继找到的结果评分必然大于当前列表中结果评分，算法可以结束.

算法找到的查询结果树（设树根为 r）中，r 到关键顶点的通路为两点间最短通路，满足文献[17] 1-层树的定义. 因此，该查询结果与指定树根 r 的组斯坦纳树最优解之间的近似比为 m.

设实体三元组关联图顶点集合为 V，查询关键词的数目为 m. 算法取出 m 条关键词索引，其中每条索引最多包含 $|V|$ 个记录，依次扫描每个记录找到 top-k 个查询结果. 算法的时间复杂度为 $O(m\cdot|V|)$.

另外，在关键词查询问题中，用户以 RDF 图为查询对象，返回给用户的查询结果也应该是 RDF 图. 因此，本文将算法 1 中得到的查询结果树（实体三元组关联图模型）转化为 RDF 图呈现给用户. 根据性质 2 可知，转化后的 RDF 图也是一个连通图.

5　实验分析

5.1　实验设置

为验证本方法效率和有效性，对一个真实 RDF 数据集（http://lsdis.cs.uga.edu/projects/semdis/swetodblp）进行实验，数据主题是关于计算机科学领域的文章发表信息. 该数据的 RDF 图共包括 13041580 条边，5342190 个顶点，将其转化为实体三元组关联图用时 507s，存储占用 62.9MB. 实验环境为一台 Intel 双核 2.4GHz 主频，3GB 内存的 PC 机.

本文在数据集上测试了两组共 20 个查询（表 1），分别包含 2～5 个查询关键词. 第 1 组 Q_1～Q_{10} 是 BLINKS[13] 方法中使用的测试查询，不包含属性和关系名. 第 2 组 Q_{11}～Q_{20} 是本文构造的包含属性和关系名的测试查询. 分别对查询响应时间及查询结果的相关性和正确性进行评价，评价原则和结果的评测指标参见第 5.2 节.

表 1　查询示例

查询	关键词
Q_1	algorithm 1999
Q_2	Michael database
Q_3	Kevin statistical
Q_4	Jagadish optimization
Q_5	Carrie carrier
Q_6	Cachera cache
Q_7	Jeff dynamic optimal
Q_8	Abiteboul adaptive algorithm
Q_9	Hector Jagadish performance improving
Q_{10}	Kazutsugu Johanna software performance
Q_{11}	ISWC publisher
Q_{12}	Swoogle author
Q_{13}	Tim Finin homepage
Q_{14}	2003 AAAI editor
Q_{15}	Peter Haase workplacehomepage
Q_{16}	article author Ian Horrocks
Q_{17}	Data mining article author
Q_{18}	article book title iswc
Q_{19}	Falcon Yuzhong Qu year
Q_{20}	Article author Motta Yuangui Lei

实验对比了本文提出的方法（KREAG）、IQTQA 方法和单层索引的 BLINKS 方法（称为 SLINKS，仅考虑单层索引的关键词查询方法，BLINKS 和 SLINKS 均不支持对属性和关系名的查询）.

5.2　实验评测标准和实验结果分析

5.2.1　索引耗费时间和空间的分析比较

当面对大规模实体三元组关联图时，索引全部顶点对之间最短通路的开销较大. 本文只索引距离小于等于索引步长参数 d 的顶点对间最短通路. 据统计，实验数据集转化为实体三元组关联图后，平均每个关键词击中的顶点数目为 44，每个顶点的入度为 1.19. 通过调整 d 值，可以控制索引时间和空间开销，表 2 给出 d 选取不同数值的结果. IQTQA 方法的索引分为实体索引、关系索引和结构图三部分，大小分别为 129MB、92MB、1.36MB，建立时间

为 161544s. SLINKS 方法利用 Dijkstra 算法建立索引,大小为 809MB,建立时间为 100800s.

表 2 d 不同取值的索引时间和空间

d	索引空间/M	索引时间/s
4	670	28360
6	941	41400
8	1130	51724
10	1239	63424
12	1298	70989

可以发现,IQTQA 方法的索引空间较小,除了建立实体索引和关系索引,只存储了根据结构相似性将实体划分为若干外延(extension),将数据抽象为以外延为顶点,外延间关系为边的结构图. 但其索引建立时间较长,原因是它要判断所有实体对间结构是否相似. 相对本文方法,SLINKS 直接在 RDF 图上进行最短通路索引,索引空间较小. 但由于其利用 Dijkstra 算法索引了全部顶点对间的最短通路信息,时间消耗较大. 本文方法通过调整 d 值,可以灵活地控制索引时间和空间开销.

5.2.2 查询响应时间的分析比较

图 4 分别显示了 IQTQA、SLINKS 和 KREAG 对表 1 中查询的响应时间,为 10 次查询的平均值. 其中,IQTQA 方法的响应时间由查询转换(QT)和查询结果生成(QA)两部分组成. 因为 QT 时间远长于 QA 时间,而且图中采用对数坐标,为了便于观察,我们将 QA 时间放在柱形图的下部,QT 时间放在上部. SLINKS 和 KREAG 均设置为返回前 10 个查询结果. IQTQA 方法通过查询转换得到查询图再生成查询结果,且一个查询图可能生成多个查询结果,无法精确设置返回前 10 个结果. 图 4 显示的是查询结果个数最接近 10 的响应时间. 另外,KREAG 方法的 d 值设置为 8,IQTQA 方法在结构图中的最大搜索步长也设置为 8.

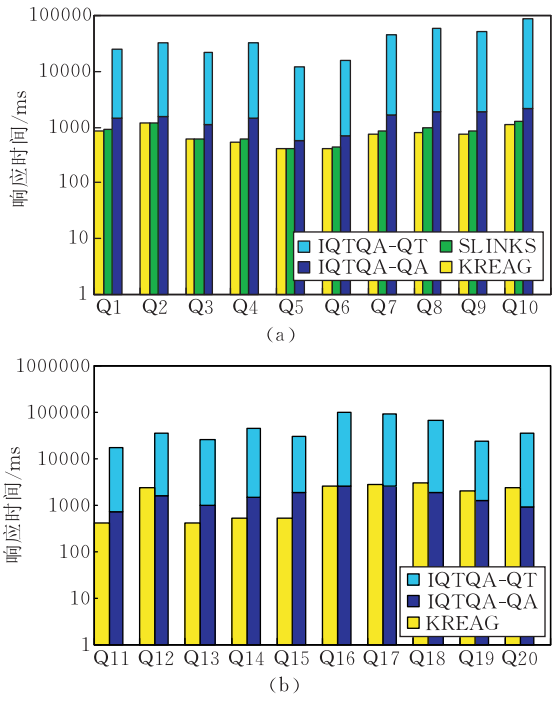

图 4 查询响应时间

可以观察对于第 1 组查询 SLINKS 和 KREAG 的响应时间基本相同,可以在一二秒内返回前 10 个查询结果. IQTQA 方法的响应时间较长,原因是 QT 过程需要在结构图中进行子图搜索,虽然结构图规模远小于 RDF 图,但直接从中搜索子图的响应时间并不理想. 另外,IQTQA 的 QA 时间也相对较长,原因是在查询转换过程中产生较多的无效查询图,无法从这些查询图生成查询结果. 对于第 2 组查询,由于其中包含了击中属性或关系的关键词,SLINKS 方法无法处理. IQTQA 方法的响应时间依然较长. KREAG 则基本可以在几秒内返回前 10 个查询结果,Q_{12},Q_{16},Q_{17},Q_{18} 和 Q_{20} 的响应时间略长于其它查询,原因是其中包含了 author,title 关键词,会击中大量的三元组顶点,导致响应时间略长.

图 5 是 top-k 的 k 值对查询响应时间的影响. 其中图(a)是第 1 组查询在不同 k 值下的平均响应时间,分别测试了 3 种方法. 图(b)是第 2 组查询在不同 k 值下的平均响应时间,只测试了 IQTQA 和 KREAG. 可以观察,k 值的增加使得 SLINKS 和

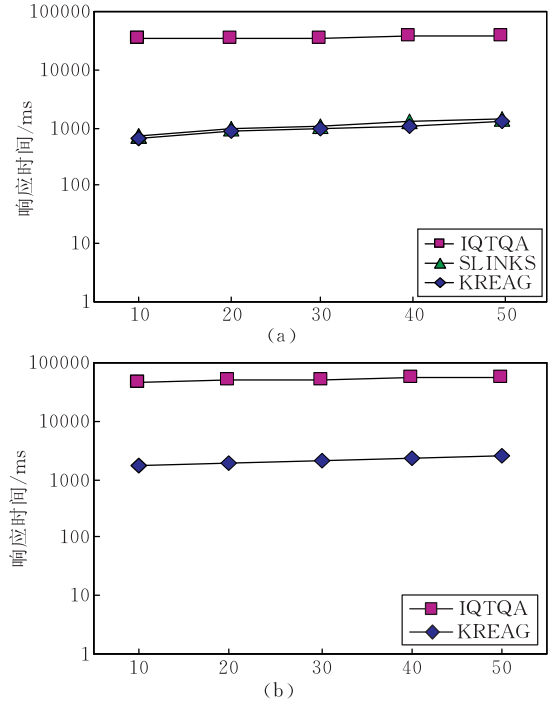

图 5 top-k 的 k 值影响

KREAG 的响应时间随之增大，主要原因是用户要求的查询结果越多，算法中需要执行的循环次数也越多. 由于 IQTQA 方法由一个查询图可以生成多个查询结果，所以其响应时间的增长相对平缓.

5.2.3　查询效果的分析比较

主要从查询准确率和平均排序倒数两个方面进行评价. 对于查询准确率，本文采用 *Precision@k* 和 *AP@k* 来衡量. *Precision@k* 是前 k 个查询结果的查询准确率，常用来衡量大多数用户关注的前 k 个查询结果的准确率. *AP@k* 目的是描述前 k 个查询结果的准确率平均值，以衡量前 k 个查询结果中正确结果的排序情况. 平均排序倒数采用 *MRR*(Mean Reciprocal Rank)来评价. 用户对返回的正确结果判断最相关的一个，*MRR* 关心的是第一个最相关结果的排序位置. 这样，3 个指标在一起能够更全面地对 top-k 查询结果进行评价.

Precision@k 的计算方式是：*Precision@k*＝前 k 个查询结果中正确的结果数/*num*，*num* 是返回的结果个数. *AP@k* 的计算方式是：$AP@k=\frac{1}{num}\sum_{rank_j\le k}\frac{j}{rank_j}$，其中，*num* 表示前 k 个结果中正确结果的个数；j 表示前 k 个结果中第 j 个正确结果；$rank_j$ 表示第 j 个正确结果的排序. *RR*(Reciprocal Rank)的计算方式是第一个最相关结果排序位置的倒数，如果没有返回正确结果，则记为 0. *MRR* 是对测试查询的 *RR* 值求平均. 测试中取 $k=10$.

图 6 给出了 3 种方法的 *Precision@k* 值，可以观察，KREAG 对 Q_8，Q_{10} 查询的准确率较低，原因是其查询结果较少，KREAG 除了返回正确结果外，还给出了其它将查询关键词关联起来的结果. 对于第 1 组的其它查询，KREAG 和 SLINKS 方法的查询准确率相近，明显高于 IQTQA 方法，原因是 IQTQA 方法中一个错误的查询图会生成多个错误查询结果，导致准确率降低. 同时，对于 SLINKS 方法不能处理的第 2 组查询，KREAG 的查询准确率基本达到 100%.

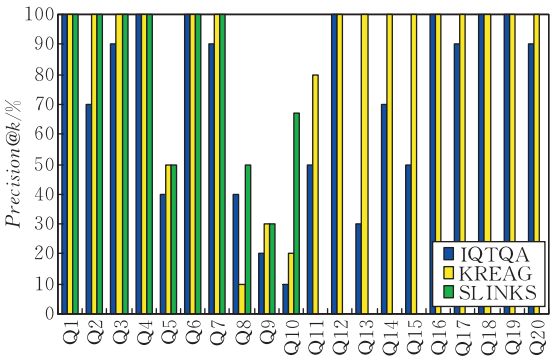

图 6　*Precision@k* 的比较

图 7 给出了 3 种方法的 *AP@k* 值，基本与 *Precision@k* 的结果一致.

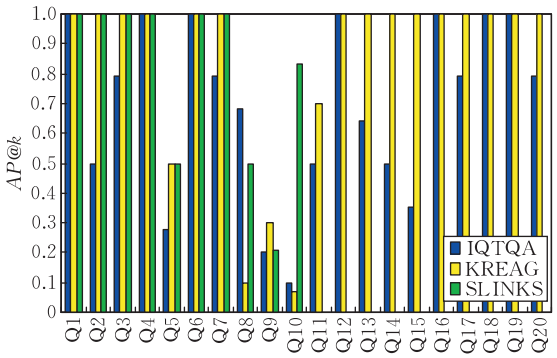

图 7　*AP@k* 的比较

通过表 3 给出的排序倒数指标值可以计算 KREAG 的 *MRR* 值为 0.98，SLINKS 方法的 *MRR* 值为 1，IQTQA 方法的 *MRR* 值为 0.69. 显然，KREAG 和 SLINKS 方法基本都能将最相关的查询结果排序第一返回.

表 3　排序倒数的比较

	排序倒数指标值									
	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}
KREAG	1	1	1	1	1	1	1	1	1	1/2
SLINKS	1	1	1	1	1	1	1	1	1	1
IQTQA	1	1/3	1/2	1	1/7	1	1/2	1/2	1	1

	排序倒数指标值									
	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}	Q_{16}	Q_{17}	Q_{18}	Q_{19}	Q_{20}
KREAG	1	1	1	1	1	1	1	1	1	1
IQTQA	1	1	1/2	1/4	1/5	1	1/2	1	1	1/2

6　总　结

随着 RDF 数据的增加，普通用户直接对其查询的需求也在不断增加. 关键词查询方法无需事先了解数据的模式信息也不需要掌握查询语言的语法结构，更适用于普通用户使用. 本文提出一个 RDF 数据的关键词查询方法(KREAG)，以实体三元组关联图为模型，封装文本信息到关联图顶点标签上，利用斯坦纳树问题的近似算法解决了 RDF 数据的关键词查询问题. 能够快速、有效地支持用户对数据中出现的所有关键词(包括属性和关系名)进行查询. 实验显示，KREAG 方法的索引时间和规模可以通过设置索引步长参数进行灵活地控制. 对于不包含属性或关系名的测试查询，KREAG 无论在响应时间和查询效果方面都与 SLINKS 方法相近，优于 IQTQA 方法. 对于包含属性和关系名的测试查询，

SLINKS 方法无法处理,而 KREAG 的响应时间和查询效果依然较好,明显优于 IQTQA 方法.

当用户对 RDF 数据进行增加或删除后,如何对索引进行更新维护使得代价最小是我们未来的研究工作之一.

参 考 文 献

[1] Berners-Lee T, Hendler J, Lassila O. The semantic Web. Scientific American, 2001, 284(5): 34-43

[2] Cheng G, Qu Y. Searching linked objects with falcons: Approach, implementation and evaluation. International Journal on Semantic Web and Information Systems, 2010, 5(3): 49-70

[3] Perez J, Arenas M, Gutierrez C. Semantics and complexity of SPARQL//Proceedings of the International Semantic Web Conference. Athens, GA, USA, 2006: 30-43

[4] Broekstra J. SeRQL: Sesame RDF query language//Ehrig M. SWAP Deliverable 3.2 Method Design. 2003: 55-68

[5] Lei Y, Uren V, Motta E. Semsearch: A search engine for the semantic Web//Proceedings of the EKAW. Podebrady, Czech Republic, 2006: 238-245

[6] Zhou Q, Wang C, Xiong M, Wang H, Yu Y. SPARK: Adapting keyword query to semantic search//Proceedings of the ISWC. Busan, Korea, 2007: 649-707

[7] Tran T, Wang H, Rudolph S, Cimiano P. Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data//Proceedings of the IEEE International Conference on Data Engineering. Shanghai, China, 2009: 405-416

[8] Lamberti F, Sanna A, Demartini C. A relation-based page rank algorithm for semantic Web search engines. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(1): 123-136

[9] Ding L, Pan R, Finin T, Joshi A, Peng Y, Kolari P. Finding and ranking knowledge on the semantic Web//Proceedings of the 4th International Semantic Web Conference. Galway, Ireland, 2005: 156-170

[10] Ladwig G, Tran T. Combining query translation with query answering for efficient keyword search//Proceedings of the 7th Extended Semantic Web Conference. Heraklion, Greece, 2010: 288-303

[11] Hristidis V, Gravano L, Papakonstantinou Y. Efficient IR-style keyword search over relational databases//Proceedings of the VLDB. Berlin, Germany, 2003: 850-861

[12] Bhalotia G, Nakhe C, Hulgeri A, Chakrabarti S, Sudarshan S. Keyword searching and browsing in databases using BANKS//Proceedings of the ICDE. San Jose, CA, 2002: 431-440

[13] He H, Wang H, Yang J. BLINKS: Ranked keyword searches on graphs//Proceedings of the SIGMOD. Beijing, China, 2007: 305-316

[14] Li G, Ooi B, Feng J, Wang J, Zhou L. EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data//Proceedings of the International Conference on Management of Data/Principles of Database Systems. Vancouver, BC, Canada, 2008: 903-914

[15] Wu Gang, Tang Jie, Li Juan-Zi, Wang Keng-Hong. Fine-grained semantic Web retrieval. Journal of Tsinghua University (Science and Technology), 2005, 45(1): 1865-1872 (in Chinese)
(吴刚,唐杰,李涓子,王克宏. 细粒度语义网检索. 清华大学学报(自然科学版), 2005, 45(1): 1865-1872)

[16] Charikar M, Chekuri C, Cheung T et al. Approximation algorithms for directed steiner problems. Journal of Algorithm, 1999, 33(1): 73-91

[17] Zelikovsky A. A series of approximation algorithms for the acyclic directed steiner tree problem. Algorithmica, 1997, 18(1): 99-110

[18] Ge W, Chen J, Hu W, Qu Y. Object link structure in the semantic Web//Proceedings of the 7th Extended Semantic Web Conference. Heraklion, Greece, 2010: 257-271

LI Hui-Ying, born in 1977, Ph. D. candidate, lecturer. Her main research interests include semantic Web and software technologies.

QU Yu-Zhong, born in 1965, Ph. D., professor, Ph. D. supervisor. His main research interests include semantic Web, software methods and technologies.

Background

The amount of available RDF data has grown rapidly, and the need for ordinary users to query RDF data has also dramatically increased. However, most RDF data management systems only support formal queries such as SPARQL.

It is too complicated for users to write even a relatively simple query without detailed knowledge of RDF data schema and query language. The keyword query has become a popular information query method on the Web because the user does not need to know any query language or underlying structure of data.

Approaches for computing structured answers can be divided into two different types. One is keyword query translation, which supports keyword search by computing translations in the form of queries and then processes these queries using an existing semantic search engine. This type of approach depends on the RDF schema or the previously computed summary graph to carry out keyword translation and matching sub graphs in the schema graph or summary graph. Processing the queries derived from them is carried out using the underlying RDF stores. Due to the coarse-grained schema and summary graph, the queries conducted by them often produce empty results. Alternatively, keyword search can be supported by searching for matching sub graphs directly in the data. The authors refer to this as direct keyword query answering. The basic idea of these systems is to map keywords to data elements, search for sub graphs that connect the data elements matching the keywords, and produce the top-k sub graphs based on a scoring function. Direct keyword search has several advantages. Answers are conducted directly on fine-grained data graph, thereby leading to non-empty results. Moreover, it can deal with RDF data without the help of RDF schema or background knowledge.

In this paper, a direct keyword query answering approach which can return ranked answers to keyword query without the help of data schema is proposed. With the proposed approach, RDF data is modeled as Entity-Triple Association Graph, and an answer to a query is modeled as a subgraph containing all query keywords. The authors present an approximation algorithm for searching top-k answers based on an indexing scheme. The experimental results show our approach is feasible and effective.

The work is supported by the National Natural Science Foundation of China under grant No. 60973024, the Natural Science Foundation of Jiangsu Province under grant No. BK2008290.