doi:10.3969/j.issn.1005-3026.2017.01.005

基于压缩实体摘要图的RDF数据关键词查询

林晓庆[1,2], 马宗民[1]
(1. 东北大学 信息科学与工程学院, 辽宁 沈阳 110819; 2. 辽东学院 信息工程学院, 辽宁 丹东 118003)

摘 要: 提出一种将关键词查询转换为SPARQL查询的方法来进行RDF数据的搜索. 首先,根据RDF本身的关联特点,构建一个压缩实体摘要图;然后,借助关键词与所在实体的索引,将所查询的关键词在该摘要图上进行定位,通过图双向搜索算法找出包含关键词实体的前k子图,获得查询实体之间的关系,再联合最初的关键词及他们的属性,构建SPARQL查询;最后使用SPARQL搜索引擎执行查询. 实验结果表明,所提方法较其他方法有更快的响应时间及更高的准确率.

关 键 词: RDF;SPARQL;OPS索引;压缩实体摘要图;双向搜索
中图分类号: TP 311.13 **文献标志码**: A **文章编号**: 1005-3026(2017)01-0022-05

RDF Keyword Search by the Condensed Entity Summary Graph

LIN Xiao-qing[1,2], *MA Zong-min*[1]
(1. School of Information Science & Engineering, Northeastern University, Shenyang 110819, China; 2. School of Information Engineering, Eastern Liaoning University, Dandong 118003, China. Corresponding author: LIN Xiao-qing, E-mail: linda164@163.com.)

Abstract: A method of translating keyword queries to SPARQL queries was presented to implement RDF (resource description framework) keyword search. Firstly, a condensed entity summary was constructed according to connections of RDF data. Then, keywords were located on the designated nodes of the summary graph by the OPS (object predicate subject) index. Top-k subgraphs connecting all keyword entities would be found by a bidirectional search algorithm. Finally, SPARQL queries were obtained by incorporating inter-entity relationships of top-k subgraphs, keywords and their properties, and SPARQL queries were executed by a SPARQL search engine. The experimental results show that a faster response time and a higher accuracy than the existing ones are achieved.

Key words: RDF (resource description framework); SPARQL; OPS (object predicate subject) index; condensed entity summary graph; bidirectional search

资源描述框架(resource description framework,RDF),是对万维网上信息进行描述的一个框架[1],RDF已成为语义数据描述的标准,被广泛应用于元数据、本体及语义网的描述中,如Yago,DBLP,wikipedia等. 近几年来,由于可用RDF数据的大量增加,用户对RDF数据查询的需求日益强烈. 随着SPARQL的广泛使用,SPARQL已经被W3C推荐为RDF查询的标准语言[2],但是RDF数据的查询语言对于普通用户过于复杂,大多数用户掌握不了查询语言及待查询数据的模式,所以形式简单的关键词查询备受欢迎. 如何能够准确、高效地进行RDF数据查询仍是当前研究的主题,如果能将关键词查询转换为SPARQL查询,就会大幅度提高查询结果的准确率. 因此本文提出一种关键词查询转换为SPARQL查询的方法,根据RDF实体间关联特点,提出了压缩实体摘要图的模型,封装实体类型到实体顶点上,该摘要图在规模上远远小于传统的数据图,降低了RDF数据图索引大小,减少了索引建立的时间,通过该摘要图转换关键词查询. 本

收稿日期: 2015-09-13
基金项目: 国家自然科学基金资助项目(61370075); 教育部新世纪优秀人才支持计划项目(NCET-05-0288).
作者简介: 林晓庆(1979-),女,辽宁丹东人,东北大学博士研究生; 马宗民(1965-),男,山东金乡人,东北大学教授,博士生导师.

文方法的执行过程：首先对输入的关键词通过 OPS 索引定位于压缩实体摘要图的指定实体上；再在该摘要图上进行双向搜索，找出包含这些关键词实体的前 k 子图；最后根据子图与 SPARQL 查询的映射将关键词查询转换为 SPARQL 查询，由用户选择一个 SPARQL 查询执行，获得最终结果.

1　相关研究

RDF 数据关键词查询根据其存储结构的不同主要有下面几类方法：一是将 RDF 数据以三元组的形式存储在关系数据库中[3-4]；二是基于 XML 格式的关键词查询[5]；三是目前较为流行的基于图的搜索方法[6-10]，利用图的结构索引和有效的搜索、剪枝算法，在 RDF 数据图上查找匹配的子图. 文献[10]将大图划分为若干个小的子图或块，建立子图(块)内和子图(块)间的路径索引. 文献[11]则先利用摘要图缩小搜索的空间，再在小的数据范围内使用向后搜索查找包含关键词的子图，从而在搜索性能上有了很大的提高. 以上几种方法都属于直接查询方法，即直接从 RDF 数据图中找到满足条件的子图，数据图匹配方法存在的主要问题是图索引开销较大，建索引时间过长，匹配的结果不全或匹配结果不够准确等. 文献[11]将关键词查询转换为形式化查询语句，再借助相应的搜索引擎进行查询. 本文提出压缩实体摘要图模型，减少了搜索空间，构建 SPARQL 查询，减少了查询响应时间，并得到了较高的准确率. 本文的系统框架如图 1 所示.

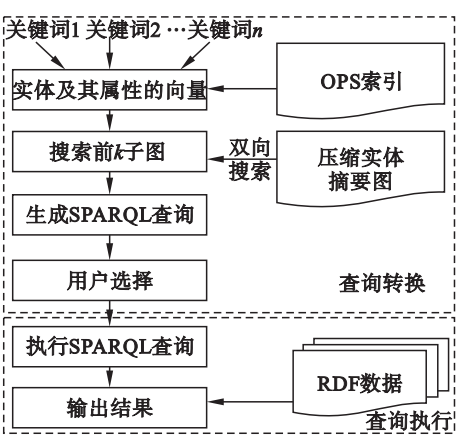

图 1　系统框架图

Fig. 1　System architecture

2　RDF 与 SPARQL

2.1　RDF 三元组和 RDF 图

RDF 三元组是一个(S,P,O)的元组，S 代表主语，P 代表谓词，O 代表宾语，其中，主语使用国际化的资源标识符 IRIs(internationalized resource identifiers)，谓词表示主语和宾语之间关系，宾语为另一实体或对应的属性值.

2.2　SPARQL 查询

SPARQL 是为 RDF 开发的一种查询语言和数据获取协议，可以对任何以 RDF 表示的信息资源进行查询，已经被 W3C 推荐为 RDF 查询的标准语言. 下面是 SPARQL 查询语句的格式.

SPARQL：SELECT varlist WHERE (graph pattern)

其中，varlist = (v_1,v_2,…,v_n)是查询变量的列表，查询变量由符号“?”or“$”标记；WHERE 子句是查询条件，复杂的 SPARQL 查询还包括用于过滤条件的正则表达式等.

3　关键词查询向 SPARQL 查询转换

3.1　压缩的实体摘要图(condensed entity summary graph，CESG)

RDF 数据图是一个描述实体与实体或者实体与值关系的连接图. SPARQL 查询的构建需要实体之间的关系，因此为了实现查询转换，本文提出一种具有实体关系的压缩摘要图模型.

封装节点所属类型到节点上有助于提高查询转换效率，不必再利用索引或搜索算法查找实体类型，压缩实体[9]将实体类型和关键词都封装到实体节点. 图 2 是根据表 1 数据构建的一个压缩实体摘要图实例. 文献[11]将同一种类型的所有实体绑定在一个类型上，缺少了一些实体关系. 本文通过 RDF 数据构建只有实体及其关系的摘要图 CESG. 图 2 中“pro1/Project”节点表示实体为“pro1”其类型为“Project”. 简单描述一下构建过程，表 1 为 RDF 数据的三元组形式，其中实体节点为“pro1”，“pub1”，“rest1”，“rest2”，“inst1”. 简单描述一下 CESG 的构建过程，将表 1 中包含实体关系的三元组，即 1,3,5,10 行保留；包含谓词“type”的三元组，即 2,4,6,9,11 行也要保留. 属于实体与值关系的三元组则不需要. 根据表 1 数据构建的 CESG 数据如图 2 所示，CESG 定义如下.

定义 1　CESG = (V,L,E)，其中节点集合 $V = V_{(E,type)}$，V_E 表示实体节点，它所属类型(type)封装在实体节点上，边标签集 $L = L_R$(表示两个实体间关系)，边集 E 中的元素为 $e(v_1, v_2)$，而且 $e \in L$. $e(v_1, v_2) \in E$，当且仅当 RDF 数据图中存在边 $e(v_1', v_2') \in E'$. 对于大规模的 RDF 数据图是

不切实际的,本文构建的 CESG 图在规模上远远小于数据图,所以单层索引的双向搜索算法[10]适合本文. 本文通过 OPS 索引,查找 CESG 前 k 子图,预先建立向后搜索链表,LEN(v),向前搜索矩阵,$M_{\mathrm{NE}}(v_i,v_j)$,LEN(v)链表存储所有能够到达 v 的节点,并按它们到达 v 的距离排序. M_{NE}存放任意一个实体节点到达其他实体节点的最短路径,$M_{\mathrm{NE}}(v_i,v_j)$为返回节点 v_i 到达节点 v_j 的最短路径.

表 1 RDF 三元组

Table 1 RDF triples

S	*P*	*O*
pub1	hasProject	pro1
pro1	type	Project
pub1	author	rest2
rest2	type	Researcher
rest2	worksAt	inst1
inst1	type	Institute
pro1	name	"Big data"
pub1	year	"2015"
rest1	type	Researcher
pub1	author	rest1
pub1	type	Publication

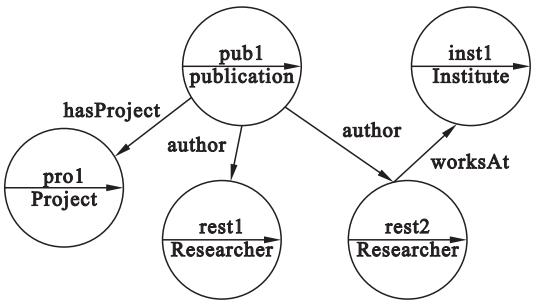

图 2 压缩的实体摘要图

Fig. 2 Condensed entity summary graph

在计算 top – k 子图过程中,本文通过向前搜索,同时利用 $M_{\mathrm{NE}}(v_i,v_j)$,返回任何两实体节点间的距离 dist_i,距离和 $\mathrm{sumDist}=\sum_i^m \mathrm{dist}_i$,$M_{\mathrm{NE}}(v_i,v_j)$是通过运行 Floyd 算法获得的. 为了减少搜索范围,本文设置一个剪枝条件,把第 k 个子图的路径长度和记为 T_{prune},在搜索过程中剪掉 sumDist 大于 T_{prune}的子图,每次访问一个节点时检查更新一下 T_{prune},算法的伪代码显示在表 2.

3.2 关键词与所在实体索引

既然 CESG 中不包括关键词,怎么找到关键词所在的实体呢? 本文使用 OPS[4] 索引将关键词定位于它所在的实体(S),结构如图 3 所示,$\boldsymbol{P}$ 表示属性,O 表示宾语(值的节点,主要指关键词),S 代表主语(表示实体),每个宾语 O_i(关键词)对应一组属性($\boldsymbol{P}$)的列表,每一个(O_i,P_j)的组合指向一个实体(S)的列表,例如,在表 1 数据上建立 OPS 索引后,"2015"会指向一个属性 $\boldsymbol{P}$ 向量,这个 $\boldsymbol{P}$ 向量包含一个属性,即"year",在这里,该属性只对应一个相关的主语 S 列表,"pub1". 对输入的每一个关键词,通过 OPS 索引可以找到一组对应关键词的属性列表,然后再联合各自的关键词属性,获得关键词所在的主语.

表 2 双向搜索算法

Table 2 Bidirectional search algorithm

输入:包含关键词$\{k_1,k_2,\cdots,k_m\}$的实体元素$\{v_1,v_2,\cdots,v_m\}$,CESG = $\{V,L,E\}$,$L_{\mathrm{EN}}(v_i)(i\in[1,m])$,$\boldsymbol{M}_{\mathrm{NE}}$矩阵,$T_{\mathrm{prune}}=\infty$;向量 $\boldsymbol{R}$ 记录根节点到所有关键词节点的路径;$\boldsymbol{R}$ 与向量 $\boldsymbol{A}$ 的每个元素为 $<\mathrm{root},\mathrm{dist}_1,\mathrm{dist}_2,\cdots,\mathrm{dist}_m>$;

输出:$\boldsymbol{A}$, top – k 子图;

```
for each i ∈ [1,m] do
    PQ_i ← new PriorityQueue(LEN(v_i)); //每个关键词所在节点建立优先队列;
for each PQ_j and j ∈ [1,m]
    v_i ← PQ_j.poll();
    if v_i is not visited then
        for each l ∈ [1,m]
            R[v_i].dist_l ← M_NE(v_i, v_l);
        R.add(<v_i, dist_1, ..., dist_m>); //向前搜索
        if sumDist(v_i) < T_prune then
            A.add(R[v_i]) and update T_prune with
            sumDist(v_i);
```

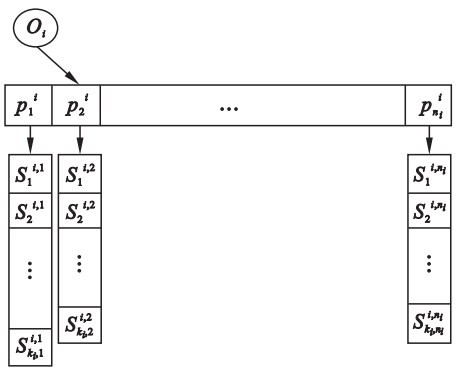

图 3 OPS 索引

Fig. 3 OPS index

3.3 关键词查询向 SPARQL 查询转换

关键词查询转换为 SPARQL 查询的关键在于查询变量间的关系获得,CESG 包含着所有实体关系,通过双向搜索可以快速找到包含关键词实体的 top – k 子图. 表 3 算法将 top – k 子图翻译成 SPARQL 查询. 简单描述一下该算法,由表 2 输出的 top – k 子图向量 $\boldsymbol{A}$ 中的每个元素结构为 $<\mathrm{root},\mathrm{dist}_1,\mathrm{dist}_2,\cdots,\mathrm{dist}_m>$,其中 root 是一个关联节点,连接所有的关键词实体,假设图 4a 是获得的一个子图,节点 rest2 是 root 节点,root 节点

连接 pub1 和 rest1 节点，边的权值表示实体关系，表 3 的第 2 行将子图向量 $\boldsymbol{A}$ 中的每一个顶点映射到 SPARQL 查询变量，如图 4b 所示，再根据关联节点与各个关键词的路径生成 SPARQL 的查询模式，即表 3 的 3，4 行，最后将生成的 SPARQL 查询存放于 $\boldsymbol{R}$ 向量当中，返回转换后的 top－k SPARQL 查询 $\boldsymbol{R}$. 对于 k 的取值，如图 5 所示，取查询长度（lenth）为 2～4 的 20 个查询，分别计算 top－10，top－15，top－20 的平均查询时间，查询时间随着 k 增加，逐渐变长，从图 5 可见，查询性能在 $k=10$ 时，查询长度对查询结果的影响最小，因此本文测试时取 $k=10$.

表 3　Top－k SPARQL 查询翻译

Table 3　Top－k SPARQL query translation

输入：top－k 子图，$\boldsymbol{A}$；关键词 $k_1,k_2,\cdots,k_n$，属性值 $p_1,p_2,\cdots,p_n$；

输出：$\boldsymbol{R}$；　//top－k SPARQL 查询；

1　for $a_i\in\boldsymbol{A},i\in[1,k]$ do

2　　？v_1，？v_2，？r，…，？$v_n\leftarrow v_1,v_2,r,\cdots,v_n$；// 子图顶点映射到 SPARQL 查询变量；

3　　中间过程：select ？v_1，？v_2，…，？v_n where（？v_1，R_1，？r），（？v_2，R_2，？r），…，（？v_n，R_n，？r）；

4　　a_i 的边标签 $l_1,l_2,\cdots,l_n$ 替换步骤 3 中的 R_1，$R_2,\cdots,R_n$；

5　　联合关键词条件（？r，type，t_1）and（？v_1，p_1，k_1），（？v_2，p_2，k_2），…，（？v_n，p_n，k_n）；

6　　$\boldsymbol{R}\leftarrow$生成 SPARQL 查询：Select？v_1？v_2，…，？v_n where ｛？v_1l_1？r.？v_2l_2？r.，…，？v_nl_n？r？r　type　t_1.？$v_1p_1k_1$.？$v_2p_2k_2$.，…，？$v_np_nk_n$｝；

7 返回 $\boldsymbol{R}$

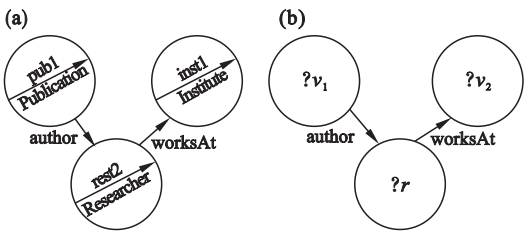

图 4　子图到 SPARQL 查询映射

Fig. 4　Mapping of subgraph to SPARQL query

（a）—子图；（b）—SPARQL 查询图.

4　实　　验

本文使用 Lehigh 大学的开放基准数据集 LUBM（Lehigh university benchmark，http://swat.cse. leghigh. edu/projects/lubm/.）和 DBLP（http://dblp. uni－trier. de/xml/）进行测试，LUBM（50，0）是由 Lehigh 大学提供的 UBA 生成器产生的 50 个有关大学、部门、教师、等之间关系的本体数据. DBLP 是对计算机领域内的国际会议和期刊出版物进行描述的数据集. 本文使用 Jena 来存储和查询 RDF 数据，表 4 是本文构建的针对 LUBM 数据集的 10 个关键词查询，图 6 和图 7 是使用表 4 查询作的对比实验结果.

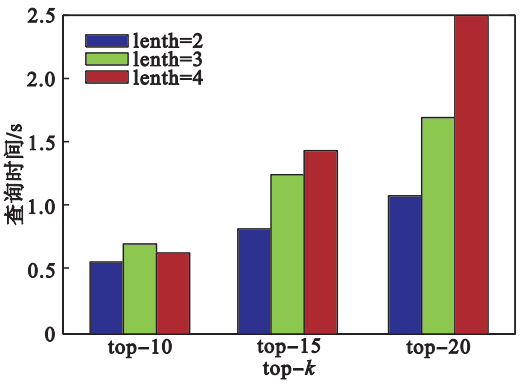

图 5　查询性能受不同 k 影响程度

Fig. 5　Query performance with different k

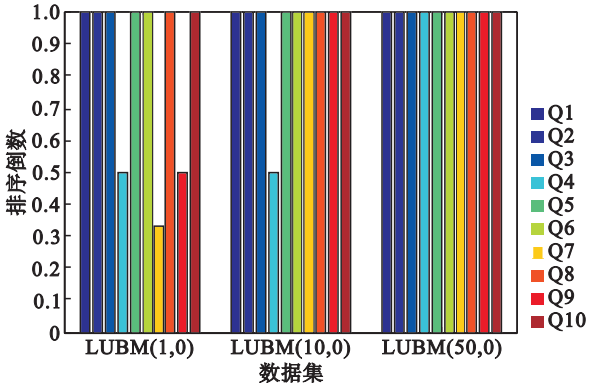

图 6　Q1～Q10 在不同数据集上的排序倒数

Fig. 6　Q1～Q10 reciprocal rank on different databases

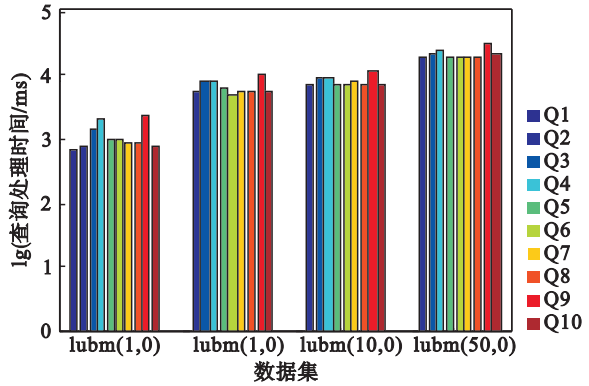

图 7　Q1～Q10 在不同数据集上的查询处理时间

Fig. 7　Q1～Q10 processing time on different datasets

本文使用国际上通用的对搜索算法进行评价的机制，MRR（mean reciprocal rank）$=1/n$，即如果返回的第一个结果匹配，分数为 1，第二个匹配分数为 0. 5，第 n 个匹配分数为 $1/n$，如果没有匹配的结果，分数为 0. 在保证结果正确的前提下，

本文没有与文献[11]方法进行比较,因为该方法的摘要图缺少了部分实体间的关系. 本文查询响应时间包括查询转换的时间加上查询执行的时间. 随着 RDF 数据的增加,构建的 CESG 具有更完备的实体关系,因此得到更高的 MRR 值,如图 6 所示;本文取 10 次查询时间的平均值作为查询时间,从图 7 可以看出,随着数据的增加,查询时间也随之变长,原因是 OPS 索引及 CESG 的大小都随之变大. 程序的循环次数因关键词数量增加而变多,所以 Q9 处理时间较其他查询更长些. 本文采用文献[10]中的 10 个查询,与文献[7]及几个利用图索引的方法,1000BFS,1000METIS,300BFS,300METIS 进行查询性能比较,实验结果如图 8 所示,本文的方法都优于文献[7]方法.

表 4 查询示例

Table 4 Query examples

查询	关键词
Q1	Pub19, Lec6
Q2	Research5, FullProf9, Pub17
Q3	FullProf9, Grad0, Pub18, Dep0
Q4	Dep0, Grad10, Pub4, AssisProf2
Q5	FullProf1, Course1, UnderStud3,
Q6	GraduCour22, GraduStud1, Dep0
Q7	AssisProf6, GraduStud38, GraduCour37
Q8	FullProf2, GraduStud3, Pub5, Dep0, Course0
Q9	UnderStud12, Course10, Course26, Research13, AssocProf7, Dep0
Q10	FullProf4, GraduStud46, Pub18

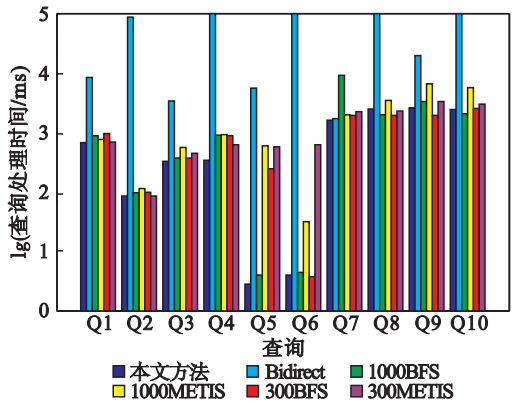

图 8 在 DBLP 上的查询性能对比

Fig. 8 Query performance comparison on DBLP

5 结 论

本文提出了一种利用压缩实体摘要图模型来实现关键词查询转换为 SPARQL 查询的方法. 该方法通过建立压缩实体摘要图缩减 top - k 子图的搜索范围,并引入了用户交互,在生成的 SPARQL 查询上,用户选择最适合的一个查询执行,得到了较高的准确率. 未来希望采取更有效的算法来更新该实体摘要图,考虑除了路径以外,融合更多因素来选择前 k 子图,比如,节点间的相关性、节点的受欢迎度、利用本体信息及尝试概率模型等.

参考文献:

[1] Beckett D. RDF/XML syntax specification [EB/OL]. (2004 - 02 - 10) [2015 - 09 - 02]. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar - 20040210/

[2] Prud'hommeaux E, Seaborne A, Harris S, et al. SPARQL query language for RDF [EB/OL]. (2013 - 03 - 21) [2015 - 9 - 13]. http://www.w3.org/TR/2013/REC - sparql11 - query - 20130321/

[3] Chen Y, Wang W, Liu Z. Keyword-based search and exploration on databases [C]// Proceedings of the 2011 IEEE 27th International Conference on Data Engineering. Washington D C: IEEE Computer society, 2011: 1380 - 1383.

[4] Weiss C, Karras P, Bernstein A. Hexastore: sextuple indexing for semantic web data management [C]// Proceedings of the VLDB Endowment. Berlin: Springer-Verlag, 2008: 1008 - 1019.

[5] Chen Y, Wang W, Liu Z, et al. Keyword search on structured and semi-structured data [C]// Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2009: 1005 - 1010.

[6] Kargar M, An A. Keyword search in graphs: finding r-cliques [C]// Proceedings of the VLDB Endowment. Berlin: Springer-Verlag, 2011: 681 - 692.

[7] Kacholia V, Pandit S, Chakrabarti S, el al. Bidirectional expansion for keyword search on graph databases [C]// Proceedings of the 31st International Conference on Very Large Data Bases. Trondheim, Norway, 2005: 505 - 516.

[8] Zou L, Mo J, Chen L, et al. gStore: answering SPARQL queries via subgraph matching [C]// Proceedings of the VLDB Endowment. Berlin: Springer-Verlag, 2011: 482 - 493.

[9] Le W, Li F, Kementsietsidis A, et al. Scalable keyword search on large RDF data [J]. *IEEE Transactions on Knowledge and Data Engineering*, 2014, 26(11): 2774 - 2788.

[10] He H, Wang H, Yang J, et al. Blinks: ranked keyword searches on graphs [C] // Proceedings of the 2007 ACM SIGMOD International Conference on Management of data. New York: ACM, 2007: 305 - 316.

[11] Tran T, Wang H, Rudolph S, et al. Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data [C]// IEEE International Conference on Data Engineering. Washington D C: IEEE Computer Society, 2009: 405 - 416.