基于城市安全知识图谱的多关键词流式并行检索算法

管 健 汪璟玢 卞倩虹

(福州大学数学与计算机科学学院 福州 350116)

摘 要 我国智慧城市安全概念的普及和建设的逐渐落地,以及大数据在智慧城市安全建设方面的深度应用,对关键词检索的处理响应速度提出了更高的要求。针对这一问题,提出了基于城市安全知识图谱的流式知识图谱多关键词并行检索算法(MKPRASKG),该算法能够根据用户输入的查询关键字,通过关联类图的构建、剪枝和融合操作实时构建基于知识图谱实体的查询子图集,再结合评分函数,以高评分的查询子图为指引,在知识图谱实例数据中进行并行搜索,最终返回 Top-k 查询结果。实验结果证明,该算法在实时搜索、响应时间、搜索效果以及可扩展性等方面均具有较大的优势。

关键词 知识图谱,流式,多关键词检索,实时

中图法分类号 TP319 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.02.006

Multi-keyword Streaming Parallel Retrieval Algorithm Based on Urban Security Knowledge Graph

GUAN Jian WANG Jing-bin BIAN Qian-hong

(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China)

Abstract With the popularization and construction of the concept of smart city security in China, and the deep application of big data in the construction of smart city security, higher requirements on the processing response speed of keyword retrieval are needed. Aiming at this problem, this paper proposed a streaming multi-keyword parallel retrieval algorithm based on the urban security knowledge graph (MKPRASKG). This algorithm can construct a query subgraph set based on the entities of knowledge graph through the construction, pruning and fusion operation of the associated class graphs based on the query keywords input by the user in real time. And then combined with the scoring function, the high-scoring query subgraph is used as a guide, and the parallel search is performed in the knowledge graph instance data, and finally the Top-k query results are returned. Experimental results show that this algorithm has great advantages in terms of real-time search, response time, search effect and scalability.

Keywords Knowledge graph, Streaming, Multi-keyword search, Real time

1 引言

近年来,事故灾难、自然灾害、公共卫生事件、社会安全事件等公共突发事件呈现出频发性和多元化发展趋势,因此城市安全在应急响应的快速性和准确性等方面面临着严峻的挑战[1-2]。同时,在大数据时代下,数据变化的速度越来越快,与城市安全相关的数据同样增长迅猛,要求处理和响应的时间越来越短[3-4],因此对大数据 3V 特性[5]中的高速性(velocity)处理就显得尤为重要。综上,数据流的实时分析和流式处理[6-7]已然成为当前的热点研究领域之一。

目前,流式数据的实时查询[8]方法主要分为 3 类。1)基于传统数据库的存储查询方法。使用传统数据库来存储管理流式数据是一种比较常见的方法,如 Facebook insight[9]应用,其架构为"数据流处理系统+Hadoop+MySQL",即将日志数据流收集后上传到 Hadoop,定期更新数据摘要到前端的 Mysql 服务器,以便通过 OLTP[10]工具产生报表等。2)基于数据流管理系统的存储查询方法[11]。数据流管理系统是面向流数据设计的数据管理系统,可有效处理输入流数据并提供持续的检索功能[12]。目前比较成型的数据流管理系统有斯坦福大学的 STREAM 系统,布朗大学、布兰戴斯大学和麻省理工大学联合开发的 Aurora[13],以及美国加州大学伯克利分校的 TelegraphCQ[14]等。3)基于 HBase[15]的 NoSQL 数据库存储查询方法。例如,侯荣军等[16]提出了一种流式数据实时写入保障下的数据查询方法,该方法将传感设备实时采集到的数据存储到 Nosql 数据库中,并通过三级缓存策略来提升查询效率。

现存的流式数据实时查询方法存在查询规则复杂、可扩展性差、查询效率低以及无法有效处理多源异构的数据集等问题。因此,本文基于城市安全知识图谱提出了流式知识图谱多关键词并行检索算法(MKPRASKG)。该算法能够很好地处理多源异构的数据源,动态地生成相应的实体和实例数据,并且能够利用实体数据集生成查询子图,然后以查询子图

到稿日期:2018-07-13 返修日期:2018-11-23 本文受国家自然科学青年基金资助项目(61300104),福建省自然科学基金项目(2017J01755)资助。

管 健(1994-),男,硕士,主要研究方向为海量数据管理和智能技术等,E-mail:963016674@qq.com;**汪璟玢**(1973-),女,硕士,副教授,主要研究方向为海量数据管理、网络数据库和智能技术等,E-mail:wjbcc@263.net(通信作者);**卞倩虹**(1993-),女,硕士,主要研究方向为海量数据管理和网络数据库等。

为指引分布式地在实例数据上进行实时搜索，并返回 Top-k 查询结果。

2 城市安全知识图谱及相关定义

2.1 城市安全知识图谱

知识图谱是 Google 用于增强其搜索引擎功能的知识库。本质上，知识图谱旨在描述真实世界中存在的各种实体或概念及其关系，它们构成一张巨大的语义网络图，节点表示实体或概念，边则由属性或关系构成。现在的知识图谱已被用来泛指各种大规模的知识库。本文通过实体建模方法构建了一个概念模型，以捕捉城市安全顶层术语的概念，随后基于这个概念模型，利用语义标记及实体链接等技术构建了一个统一的大规模城市安全知识图谱。知识图谱的实体和实例都是用 RDF 的格式来存储。本文所构建的城市安全知识图谱的部分实体图如图 1 所示。

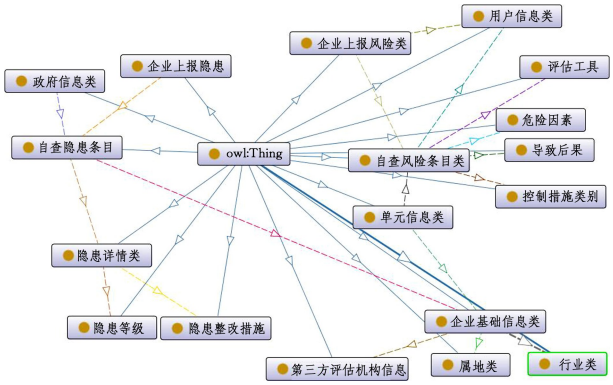

图 1 实体图

Fig. 1 Entity graph

2.2 相关定义

问题定义：给定关键词查询 $Q=\{q_1,q_2,\cdots,q_i,\cdots,q_m\}$ 以及 RDF 数据图 G，返回 Top-k 查询结果。下面给出 RDF 关键词搜索的相关定义。

定义 1（RDF 三元组） 设 RDF 三元组可表示为 $t\langle s,p,o\rangle$，其中 s 表示 t 的主语，p 表示 t 的谓语，o 表示 t 的宾语。$s\in(\mathrm{IUB})$，$p\in(\mathrm{IUB})$，$o\in(\mathrm{IUBUL})$，I 是 URI 顶点的集合，B 是空白顶点集合，L 是文本顶点集合。

定义 2（RDF 图） 设 $G=\{t_1,t_2,\cdots,t_i,\cdots,t_m\}$ 表示 RDF 图。一个 RDF 图可由一组 RDF 三元组定义。一个 RDF 图表示为一个有向标记图，每个三元组 $t_i=\langle s_i,p_i,o_i\rangle$ 的主语 s_i 和宾语 o_i 作为 RDF 图顶点，谓语 p_i 是由主语指向宾语的一条有向标记边。

定义 3（关联类图，记为 GS_i） 设 $GS_i=\{T_1,T_2,\cdots,T_n\}$ 表示关联类图，给定一关键词所属的类 C_i，将与其相关联的类连接起来，即将 $T_i\langle S_i,P_i,O_i\rangle$ 加入到 GS_i 集合中，其中 $S_i=C_i$ 或者 $O_i=C_i$。

定义 4（类图剪枝） 多个关键词构建成多个 GS，在所有的 GS 中，仅出现一次的模式三元组 $\langle S_i,P_i,O_i\rangle$ 就是松散挂起的节点，将其删除不会对查询结果图产生影响。

定义 5（类图融合去重） 多个关键词构建成多个 GS，在所有的 GS 中，出现多次的模式三元组 $\langle S_i,P_i,O_i\rangle$ 则为关系紧密的节点，去掉重复的三元组，保留一份，从而形成新的图关联类图。

定义 6（实体查询子图） 多个 GS 进行融合后，进行三元组连接操作，形成实体查询子图集 Gsk。

定义 7（三元组连接） 在构建实体查询子图或者结果子图时，需要对匹配的模式三元组或者实例三元组进行连接操作，其中任意两个三元组通过主语、宾语或者其他三元组相连接。三元组连接的形式化表示为：对于模式三元组或者实例三元组集合 $Set=\{T_1,T_2,\cdots,T_i,\cdots,T_m\}$，给定 $T_i\langle S_i,P_i,O_i\rangle$ 和 $T_j\langle S_j,P_j,O_j\rangle$，其中 $\exists i,j\in\{1,2,\cdots,m\}$，如果 $(S_i=S_j\&\&O_i\neq O_j)$ 或者 $(S_i=O_j\&\&O_i\neq S_j)$ 或者 $(O_i=S_j\&\&S_i\neq O_j)$ 或者 $(O_i=O_j\&\&S_i\neq S_j)$，则称 T_i 与 T_j 相邻，可以进行三元组连接。

定义 8（查询结果，记为 R） 已知 RDF 数据图 G 和关键词查询 Q，查询结果是一组包含所有查询关键词的三元组构成的连通子图，其中任意两个三元组通过主语、宾语或者其他三元组连接起来。设 $R=\{t_1,t_2,\cdots,t_k,\cdots,t_r\}$，其中 $\exists i,j\in\{1,2,\cdots,r\}$，$t_i\langle s_i,p_i,o_i\rangle$ 和 $t_j\langle s_j,p_j,o_j\rangle$，则有 $(S_i=S_j$ 且 $O_i\neq O_j)$ 或 $(S_i=O_j$ 且 $O_i\neq S_j)$ 或 $(O_i=S_j$ 且 $S_i\neq O_j)$ 或 $(O_i=O_j$ 且 $s_i\neq s_j)$ 或 $(t_i-t_k-t_j)$。若两个三元组集合中的元素不完全相同，则认为是不同的查询结果。

定义 9（相关性评分函数，Score Estimation，记为 SE） 输入查询 $Q=\{q_1,q_2,\cdots,q_i,\cdots,q_m\}$，对应 RDF 实体实例类 $C=\{c_1,c_2,\cdots,c_i,\cdots,c_m\}$，假定 Q 对应的一个实体查询子图 $Gsk=\{g_1,g_2,\cdots,g_n\}$，其中 $g_k\in C$。

$$SE(Gsk)=\alpha*len(Gsk)+(1-\alpha)*pageRanks(Gsk) \quad (1)$$

其中，$len(Gsk)=\dfrac{1}{Length(Gsk)}$，$Length(Gsk)=\sum\limits_{i\in(1,2,\cdots,m)}dis(c_i,c_j)$，$pageRanks(Gsk)=\sum\limits_{i\in(1,2,\cdots,m)}pageRanks(c_i)$。

相关性评价函数由结构紧密度评分 $len(Gsk)$ 和内容关联度评分 $sim(C)$ 两部分构成，α 为调节参数，本文中 $\alpha=0.5$，表示两者的影响程度一样。$dist(c_i,c_j)$ 表示实例类节点 c_i 和 c_j 在实体查询子图上的距离，若实例类 c_i 和 c_j 不可达，则距离为 $dist(c_i,c_j)=+\infty$。

$Length(Gsk)$ 等于对实体查询子图上两两实例类的顶点间的距离求和，求和距离越短，则 $1/Length(Gsk)$ 值越大，说明内容联系越紧密。pageRank 算法是 Google 提出的一种用于评价网页等级（权重）的一种计算模型，本文引用 pageRank 算法来计算实体查询子图中各个类节点的权重，以评估实体查询子图的内容相关性。pageRanks(Gsk) 表示实体查询子图上各个类的 pageRank 值之和，当 pageRanks(Gsk) 值越大时，说明 Gsk 的内容相关性越高。RDF 数据图上的一个 RDF 实例顶点可映射到 RDF 实体图上的一个实例类，在 RDF 数据图上关系越紧密的实例顶点对应的实体实例类间的关系越紧密，结果评分值也越高。

3 MKPRASKG 算法

为了避免直接在大规模知识图谱实例数据上进行搜索，该算法先结合知识图谱的实体图，通过关联类和类剪枝及类融合方法，为输入关键词实时构建若干个查询实体子图集，然后利用 SE 评分函数进行评分排序，最后以高评分的查询实体子图为指引，在知识图谱的实例数据中进行实时并行搜索，最终返回 Top-k 查询结果。算法的总体框架设计如图 2 所示。

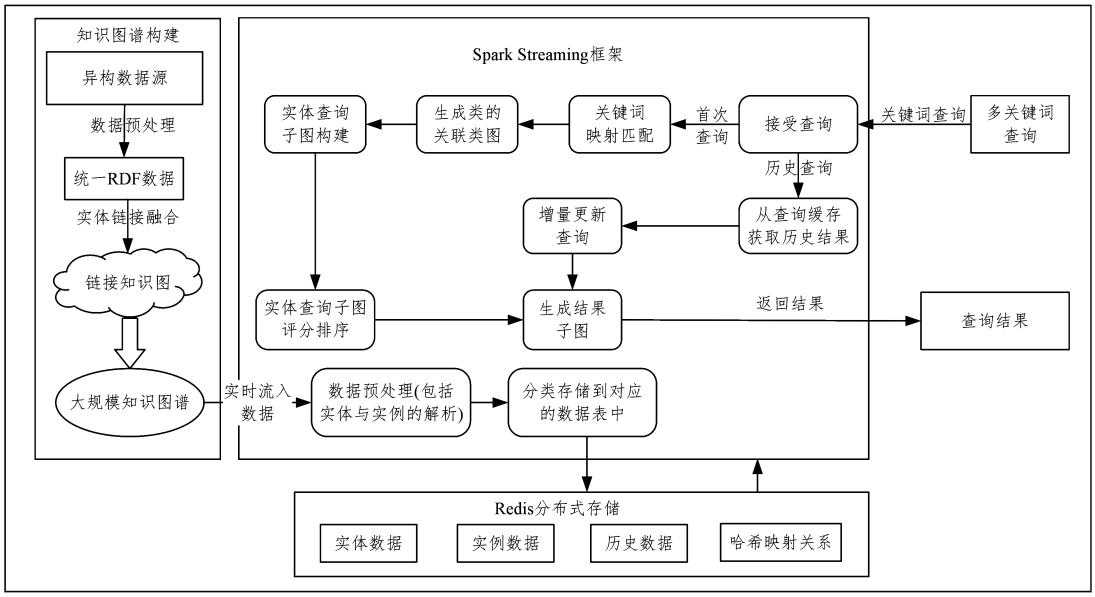

图 2　MKPRASKG 总体框架图

Fig. 2　Overall framework of MKPRASKG

3.1　MKPRASKG 算法的分布式存储方案的设计

本节介绍 MKPRASKG 算法的分布式存储方案。该算法存储的数据包括实体数据和实例数据,由于知识图谱数据具有显著的结构特征,同一类型的实例三元组数据间的语义关系较密切,因此本文依据实体数据类型将大规模的实例三元组数据分类进行分布式存储,从而有效提高查询效率。该算法使用 Redis 内存数据库集群作为数据存储的媒介。集群中 Redis 内存数据库的数量可以根据需求动态增加或减少。考虑到流式数据的动态性,本文在实体数据存储和实例数据存储的基础上,还增加了对历史数据的存储。具体的表及存储内容的说明如下。

Rdf_Entity 表:$Rdf\_Entity=>\{Cindex_1,Cindex_2,\cdots,Cindex_n\}$,其中 $Cindex_n$ 为实体中的类,$Cindex_n=\{Class_n=>n\}$表示键为类名、值为编号的结构。

SubClassOf 表:$Class\_Sub=>\{CS_1,CS_2,\cdots,CS_n\}$,其中 $CS_n=\{Class_n=>\{SubClass:[]\}\}$表示键为类名、值为该类的父类集合的存储结构。

C_C_Property 表:$C\_C\_Property=>\{P_1,P_2,\cdots,P_n\}$,其中 $P_n=(Property_n=>\{[C_1,C_2],\cdots,[C_j,C_k]\})$表示键为属性名、值为该属性对应的类对构成的集合的存储结构。

Subject_Class 表:$Subject\_Class=>\{C_1,C_2,\cdots,C_n\}$,其中 $C_n=(Class_n=>\{Property_1=>ObjectClass_1,\cdots,Property_j=>ObjectClass_j\})$表示键为主语所属类的名称、值为属性和宾语所属类的组合的存储结构。

Object_Class 表:$Object\_Class=>\{C_1,C_2,\cdots,C_n\}$,其中 $C_n=(Class_n=>\{[P_1,SC_1],\cdots,[P_j,SC_j]\})$表示键为宾语所属类的名称、值为属性和主语所属类的组合的存储结构。

Literal_Triple 表:$Literal\_Triple=>\{L_1,L_2,\cdots,L_n\}$,其中 $L_n=(Literal_n=>\{P_1S_1,\cdots,P_mS_m\})$表示键为文本标签、值为数据属性和实例的存储结构。

Instance_Class 表:$Instance\_Class=>\{IC_1,IC_2,\cdots,IC_n\}$,其中 $IC_n=(Instance_n=>\{Class:[]\})$表示键为实例、值为实例对应的类的集合的存储结构。

SC_OP_OC 表:$SC\_OP\_OC=>\{SO_1,SO_2,\cdots,SO_n\}$,其中 $SO_n=(Subject_n=>Object_n)$表示键为实例三元组的主语、值为实例三元组的宾语的存储结构。

OC_OP_SC 表:$OC\_OP\_SC=\{OS_1,OS_2,\cdots,OS_n\}$,其中 $OS_n=(Object_n=>Subject_n)$表示键为实例三元组的宾语、值为实例三元组的主语的存储结构。

His_Class_Graph 表:$His\_Class\_Graph=\{HCG_1,HCG_2,\cdots,HCG_n\}$,存储历史记录中搜索过的查询子图,其中 $HCG_n=(Keywords_n=>EntityGraphs_n)$表示 key 为搜索过的关键词,$value$ 为实体查询子图集。

3.2　知识图谱实体查询子图集的构建

为了避免直接在大规模的知识图谱实例数据图上进行关键词匹配,本文提出在知识图谱实体数据图上构建查询子图的算法。该算法通过关联类图的构建、剪枝和融合操作,最终生成关键词对应的查询子图集,不仅能有效挖掘关键词间的语义信息,还能提高关键词的搜索效率。

3.2.1　关联类图的构建

关联类图是构建实体查询子图的基础。用户输入的多个关键词可能映射为类、属性、实例或文本。若映射为类,则生成对应类的关联类图。因此,算法对于每一个关键词,首先根据 C_C_Property 表判断该关键词是否映射为属性,若为属性,则先存储该属性以备构建实体查询子图阶段使用;若不是属性,则通过 Rdf_Entity,C_C_Property,Instance_Class,Literal_Triple 这 4 张表,可以确定该关键词可能映射的实体类。

由于 Redis 采用的是 key-value 的存储形式,在查询阶段的时间复杂度是 O(1),因此对每个关键词均可以快速定位到其匹配的类;同时,考虑到一个关键词可能会映射多个类,多个关键词匹配的类会有重复的现象,需要去除重复的实体类。给定任意一个类,通过定义 3 进行关联,都可以生成该类对应的关联类图 GS。

3.2.2　关联类图的剪枝

一个类生成的关联类图中包含的模式三元组会很多,其

中会掺杂着多余的模式三元组，为了去掉不会对查询结果产生影响的边，需要对关联类图进行剪枝操作。根据定义 4，在关联类图中，如果一条路径$\langle s,p,o\rangle$中 s,p,o 只有一个可以映射为查询关键词对应的类或属性，则可以认为该路径不会对查询结果起作用，因此可以对其剪枝，最终得到剪枝后的关联类图 *GSSet*。剪枝算法的具体步骤如算法 1 所示。

算法 1 cutGS 剪枝算法

功能：去掉不会对查询结果起作用的边

输入：关联类图的集合 GSSet，属性集合 Pset，类的集合 Cset

输出：剪枝后的 GS

```
1. For t_i〈s_i,p_i,o_i〉∈GS&i =1,2,…,n // * 遍历关联类图 GS *
//如果模式三元组的谓语在查询关键词对应的属性集合 Cset 中
2.   IF Pset.contain(p_i)
3.     continue;
     //如果该模式三元组的主语和宾语都在查询关键词对应的类集合 Cset 中
4.   ELSE IF Cset.contain(s_i)&& Cset.contain(o_i)
5.     continue;
     //如果上边两种情况都不满足，则认为该模式三元组不会对查询结果起作用
6.   ELSE
7.       GS.remove(t_i);
8.   END IF
9. END FOR
10. Return GS.
```

代码第 1 行开始遍历关联类图集合 *GS*，第 2－5 行进行判断，如果 t_i 的属性 p_i 是存在于查询关键字中或者 t_i 的主语和宾语都在查询关键字对应的类的集合 *Cset* 中，那么 t_i 可以保留在 *GS* 中，算法继续；第 6－8 行，如果不是以上两种情况，则认为 t_i 不会对查询过程产生影响，将 t_i 移除 *GS* 即可。

3.2.3 关联类图的融合

融合的目的是将剪枝后的关联类图连接起来。属于父子关系的关联类图应该分开，与其他关联类图做融合。在一个需要融合的关联类图集中，迭代判断两个关联类图是否存在公共边，并根据公共边将两个关联类图融合成一个关联类图，以此得到一个或多个最终关联类图。融合过程的输入是经过剪枝后的 *GSSet*，融合算法如算法 2 所示。

算法 2 融合算法

功能：多个关联类图做连接生成实体查询子图集

输入：关联类图组成的集合 GSSet

输出：实体查询子图集合

```
1. Set result={};//存储最终生成的实体查询子图
2. Set allSet={};//暂存已经查过的模式三元组
3. Set temp={};//为了取交集的临时变量
4. For Gs_i∈GSSet & i=1,2,…,m
5. Gs_i.removeAll(result);
6.   temp.clear();
7.   temp.addAll(allSet);
8.   temp.retainAll(Gs_i);
9.   temp.retainAll(Gs_i);
10.  result.addAll(temp);
11. Gs_i.removeAll(temp);
12.  allSet.addAll(Gs_i);
13. End For
14. return result.
```

代码的第 1－3 行定义了 3 个存储变量，它们分别是 *result*，存储最终生成的实体查询子图集；*allSet*，存储已经查过的模式三元组的集合；temp，为了取关联类图交集的临时变量。第 4 行开始遍历关联类图的集合 *GSSet*。第 5 行对 Gs_i 先删除掉已经查询到的模式三元祖，以避免重复。第 6－10 行获取 *allSet* 集合与 Gs_i 集合的交集，并将结果添加到 *result* 集合中；第 11－12 行将 Gs_i 中的非交集添加到 *allSet* 中。

对于 DBPedia 数据集，假设用户输入关键词“Kobe，Lakers”，其中，“Kobe”可能映射为“Athlete”类，“Lakers” 可能映射为“SportsTeam”类。各个关键词从关联类的构建（见图 3(a)）到剪枝（见图 3(b)），再到融合（见图 3(c)），最后到查询子图集生成（见图 3(d)）的全过程如图 3 所示。

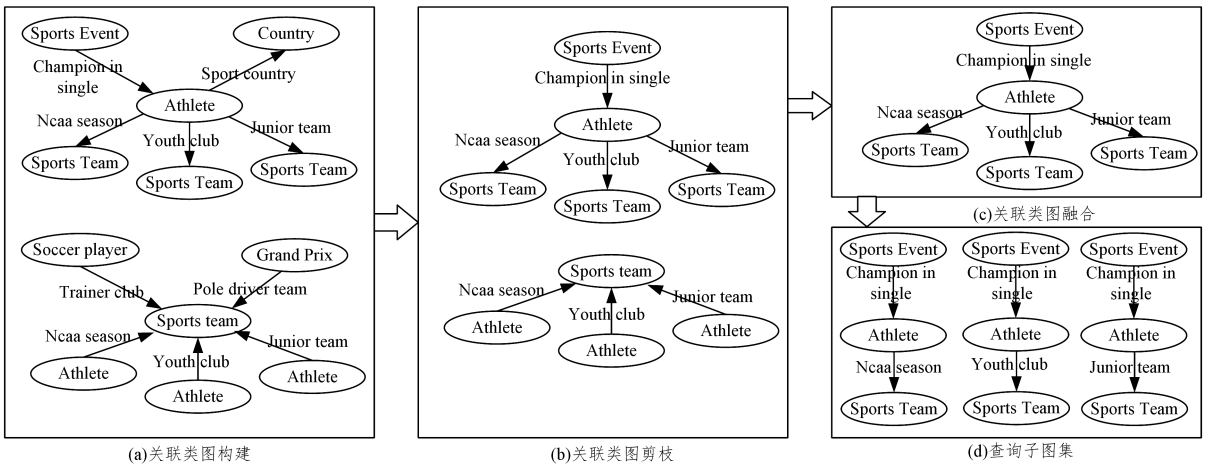

图 3 实体查询子图集生成过程（DBPedia）

Fig. 3 Process of generating entity query subsets (DBPedia)

对于本文所构建的城市安全知识图谱数据集，假设用户输入关键词“张三，1 号机，自然灾害”，其中，“张三”可能映射为“用户信息”类，“1 号机”可能映射为“单元信息”类，“自然灾害” 可能映射为“自查风险条目”类。各个关键词从关联类的构建（见图 4(a)）到剪枝（见图 4(b)），再到融合（见图 4(c)），最后到查询子图集生成（见图 4(d)）的全过程如图 4 所示。

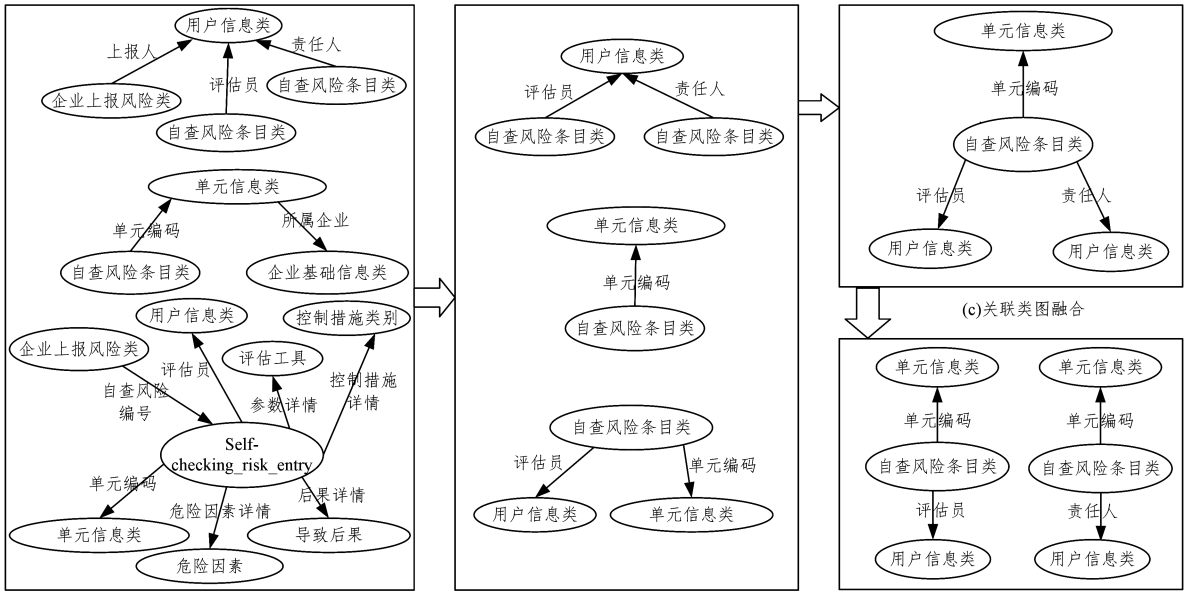

图 4　实体查询子图集生成过程(城市安全知识图谱)

Fig. 4　Process of generating entity query subsets (Knowledge graph of urban safety)

3.3　结果子图的构建

当完成实体查询子图的构建排序后,取出优先级最高的实体查询子图进入 Map 阶段,获取结果子图。首先,结合实体查询子图中的模式三元组,在知识图谱实例数据中搜索对应的键值对记录;然后,将模式三元组中的谓语和键值对记录中的主语和宾语进行组合,得到匹配的实例三元组;最后,通过定义 7 的连接方法对匹配的实例三元组进行连接,得到对应的结果子图。

关键词集合中可能包含文本类型的关键词,可以在 O(1)的时间内搜索 Keywords_Inf 得到文本类型关键词对应的实例,根据模式三元组和文本类型关键词对应的实例可以得到匹配的实例三元组,具体步骤详见算法 3 中的 Step2。此外,关键词集合中可能包含实例类型的关键词,可以在 O(1)的时间内搜索 SC_OP_OC 和 OC_OP_SC,从而得到以该实例为 key 的键值对记录,根据模式三元组和键值对记录得到匹配的实例三元组,具体步骤详见算法 3 中的 Step3。生成结果子图的具体过程如算法 3 所示。

算法 3　生成结果子图

输入:优先级最高的实体查询子图,实例数据

输出:结果子图集

Step1　获取 Keywords_Inf 中文本类型关键词对应的实例,组合得到包含数据属性的模式三元组匹配的实例三元组;

Step2　获取 SC_OP_OC 和 OC_OP_SC 中以实例类型关键词为 key 的记录,组合得到包含对象属性的模式三元组匹配的实例三元组;

Step3　获取 SC_OP_OC 和 OC_OP_SC 中其他模式三元组匹配的实例三元组;

Step4　通过定义 7 对匹配的实例三元组进行连接,得到结果子图;

Step5　将构建的多个结果子图添加到结果子图集合;

Step6　输出并传递结果子图集给编码反转阶段进行处理,跳到 Step7;

Step7　算法结束。

3.4　时间复杂度分析

本文算法的时间复杂度由两部分组成,即构造实体查询子图的时间复杂度和生成结果子图的时间复杂度。其中,实体查询子图构建阶段包括构建关联类图、类图修剪和类图融合;结果子图生成阶段包括 Map 阶段和 Reduce 阶段。假设用户输入关键词的数量为 M,并且每个关键词生成 m 个关联类图。每个关联类图都有 n 条边,生成的实体查询子图的数量为 K,关键词匹配的实例三元组的数量为 p,则:构造实体查询子图阶段的时间复杂度为 $O(M)+O(m*n)+O(m)$;生成结果子图阶段的时间复杂度为:Map 阶段 $O(K)$,Reduce 阶段 $O(p*K)$;

因此,MKPRASKG 算法的时间复杂度为 $(O(M)+O(m*n)+O(m))+O(K)+O(p*K)$。由于知识图谱实体图的比例相对较小,因此 m,n 和 K 相对较小,因此算法的总时间复杂度相对较低。

4　实验分析

4.1　实验环境

实验搭建的硬件环境为 Intel(R) Core(TM) i5-3570,CPU 3.40 GHz,内存 8 GB;软件环境为 Ubuntu 操作系统,采用 Java 作为编程语言,开发环境为 Eclipse。实验室 Spark 集群环境中部署了 1 个 Master 节点和 8 个 Worker 节点。

4.2　实验数据集及设置

本次实验所采用的数据集包括 DBPedia 以及来自于“城市风险管理系统”和“隐患排查治理系统”这两个系统所生成的与 500 家企业相关的大量真实数据。

实验所用的 5 个有代表性的查询搜索示例如表 1 所列,每个查询分别包含 2～4 个关键词。其中,第 1 组 Q_1-Q_3 为测试城市安全知识图谱的查询,第 2 组 Q_4-Q_5 为测试 DBPedia 的查询。

本次实验将 MKPRASKG 算法与文献[17]提出的 DK-

SRM 算法在相同的环境下进行对比。为了便于进行实验，将定义 9 中的相关性评价函数式(1)中的参数 α 设置为 0.5，Top-k 的 k 值设置为 10。

表 1 搜索示例

Table 1 Search example

搜索	关键词集合
Q_1	爆炸，冲击波
Q_2	张三，1 号机，自然灾害
Q_3	危险品，泄露，人体，受伤
Q_4	Aristotle，Western_philosophy
Q_5	Bush，Hussein，Iraq

由于 MKPRASKG 算法实现了多关键词的实时搜索，因此在测试 MKPRASKG 算法时采用如下方法：每间隔 2s 向 Spark 数据实时流入模块流入 8MB 的数据，并且在数据集实时流入的过程中不间断地进行实时多关键词搜索，同时在完成数据集流入时同样对其进行多关键词搜索，最后取多次关键词搜索结果的平均值作为最终的结果；对比实验中其他两种算法的数据都取 10 次搜索的平均值。

4.3 实验结果与分析

从搜索响应时间、查准率和查全率 3 个方面来分析实验结果，并在不同节点个数的集群和不同大小规模的数据上进行实验分析，以验证算法具有良好的可扩展性。

4.3.1 搜索响应时间及正确性的分析比较

对比实验结果如图 5－图 7 所示。其中，图 5 为查询响应时间对比图，图 6 为查准率对比图，图 7 为查全率对比图。

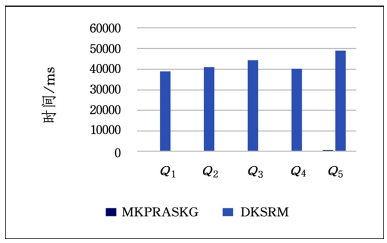

图 5 平均搜索响应时间对比

Fig. 5 Comparison of average search response time

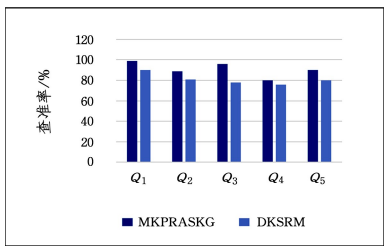

图 6 查准率对比

Fig. 6 Comparison of precision

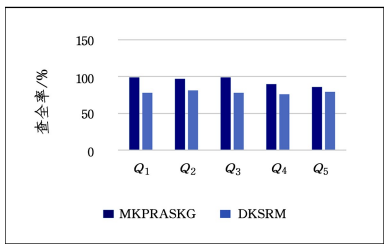

图 7 查全率对比

Fig. 7 Comparison of recall

由图 5 可知，DKSRM 算法在查询时需要构建大量的路径索引，大量的路径分布在集群的不同节点上，进行查询和连接操作时，MapReduce 并行度大，网络传输开销大，因此查询耗时较长。而 MKPRASKG 算法基于实体语义模型图，能够快速构建查询关键词对应的实体子图集，因此查询耗时较少。由图 6 和图 7 可知，MKPRASKG 算法具有比 DKSRM 算法更高的查准率和查全率。因为 DKSRM 算法通过在 RDF 数据图上直接匹配关键词连接生成结果子图，没有考虑关键词间的语义关联关系，存在查询结果信息不全或不准确的情况。

4.3.2 算法可扩展性分析

为了验证 MKPRASKG 算法的可扩展性，在节点个数分别为 2，4，6 和 8 的集群环境下，利用加速比衡量算法并行化的性能和效果。加速比指同一个任务在串行系统中运行消耗的时间和在并行处理系统中运行消耗的时间的比率。算法的加速比对比结果如图 8 所示。

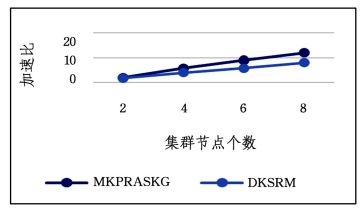

图 8 算法加速比的对比

Fig. 8 Comparison of speedup ratio

从图 8 可以看出，随着集群节点个数的增加，MKPRASKG 算法具有更高的加速比和更好的查询性能。这是因为集群节点个数增加时，DKSRM 算法开启多个 MapReduce 任务的时间也会延长，且 DKSRM 算法构建的大量路径索引数据会分布在不同的节点，从而导致了节点间数据传输时间的延长。

4.3.3 MKPRASKG 算法的实时查询分析

MKPRASKG 算法不仅能完成实时查询处理，而且能很好地支持增量更新查询，提高实时查询效率。因此，本节设计两种实验方案来进行验证。

(1)实际查询中，用户提交的查询与历史查询操作相同，可以直接复用历史查询结果，通过执行增量更新查询来加速查询。在数据集上分别采用 1%，10%，20%和 50%的增量数据占比，重复查询 Q_1－Q_3，分别执行增量更新查询和不使用增量直接查询，实验结果如图 9 所示。

从图 9 可以看出，相比不使用增量的直接查询，MKPRASKG 算法的增量更新查询在效率上提高了近 50%。对于重复的查询操作，增量更新查询直接复用相同的历史查询的实体子图集和历史结果三元组集合，只需计算当前数据新增的一小部分三元组数据，从而提高了查询效率。

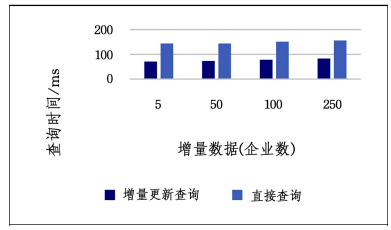

图 9 增量数据下 MKPRASKG 算法的查询时间

Fig. 9 Query time for MKPRASKG algorithm for incremental data

(2)实际查询中,可能存在历史查询为下次用户提交查询的一部分的情况。以表 1 中查询 Q_2,Q_3 为例,假定查询缓存中已存在 Q_2 和 Q_3 查询的历史查询结果,下一次查询用户可能想查询 Q_2'和 Q_3',Q_2 和 Q_3 的查询关键词集合分别为 Q_2'和 Q_3'查询关键词集合的真子集。具体的查询关键词如表 2 所列。对 Q_2,Q_3,Q_2'和 Q_3'进行查询,分别执行增量更新查询和不使用增量直接查询,实验结果如图 10 所示。

表 2　增量更新的搜索示例

Table 2　Search example for incremental updates

搜索	关键词集合
Q_2	张三,1 号机,自然灾害
Q_2'	张三,1 号机,自然灾害,伤亡人数
Q_3	危险品,泄露,人体,受伤
Q_3'	危险品,泄露,人体,受伤,企业

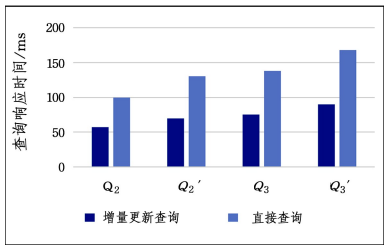

图 10　查询响应时间

Fig. 10　Query response time

从图 10 可以看出,采用增量更新查询能有效提高查询响应速度,因为它只需要在原有的实体查询子图的基础上再增加新部分的连接即可,不需要重新做整体的关键词连接,因此查询效率得到提高。

结束语　本文基于城市安全知识图谱提出了流式知识图谱多关键词并行检索算法(MKPRASKG),该算法能有效地将多源异构的数据集进行融合,解决了现有算法仅针对单一数据集的不足,并且结合 Spark 和 Redis 框架,根据查询关键字在知识图谱实体上实时构建查询子图集,再结合评分函数,以高评分的查询子图为指引,在知识图谱实例数据中进行并行搜索,从而在很大程度上提高了搜索的效率以及准确性。后续的研究将考虑有效的缓存策略,以便进一步加快查询的速率。

参 考 文 献

[1] CHEN H S,HAN Z,DENG S N. Analysis and Research of Big Data Security in Smart Cities[J]. Information Network Security,2015(7):1-6. (in Chinese)
陈红松,韩至,邓淑宁. 智慧城市中大数据安全分析与研究[J]. 信息网络安全,2015(7):1-6.

[2] WAN S,LU J,FAN P,et al. To Smart City:Public Safety Network Design for Emergency[J]. IEEE Access,2018,6(99):1451-1460.

[3] WANG Y Z,JIN X L,CHENG X Q. Network Big Data:Current Status and Prospects[J]. Chinese Journal of Computers,2013,36(6):1125-1138. (in Chinese)
王元卓,靳小龙,程学旗. 网络大数据:现状与展望[J]. 计算机学报,2013,36(6):1125-1138.

[4] MENG X F,CI X. Big Data Management:Concepts,Technologies and Challenges [J]. Journal of Computer Research and Development,2013,50(1):146-169. (in Chinese)
孟小峰,慈祥. 大数据管理:概念、技术与挑战[J]. 计算机研究与发展,2013,50(1):146-169.

[5] BUXTON B,GOLDSTON D,DOCTOROW C,et al. Big data:science in the petabyte era[J]. Nature,2008,455(7209):8-9.

[6] CHEN C. Streaming big data real-time processing technology,platform and application [J]. Big Data,2017,3 (4):1-8. (in Chinese)
陈纯. 流式大数据实时处理技术、平台及应用[J]. 大数据,2017,3(4):1-8.

[7] DUAN Z Y. Research on load balancing and fault tolerance mechanism of big data streaming system [D]. Beijing:North China Electric Power University,2017. (in Chinese)
段泽源. 大数据流式处理系统负载均衡与容错机制的研究[D]. 北京:华北电力大学,2017.

[8] HIRIYANNAIAH S,SIDDESH G M,SRINIVASA K G,et al. Real-Time Streaming Data Analysis Using a Three-Way Classification Method for Sentimental Analysis[J]. International Journal of Information Technology & Web Engineering,2018,13(3):99-111.

[9] BORTHAKUR D,GRAY J,SARMA J S,et al. Apache hadoop goes realtime at Facebook[C]// ACM SIGMOD International Conference on Management of Data(SIGMOD 2011). Athens,Greece,DBLP,2011:1071-1080.

[10] KEETON K,PATTERSON D A,HE Y Q,et al. Performance characterization of a Quad Pentium Pro SMP using OLTP workloads:Technical Report UCB//CSD-98-1001[R]. University of California at Berkeley,Computer Science Division,1998:15-26.

[11] CHEN H,MIGLIAVACCA M. StreamDB:A Unified Data Management System for Service-Based Cloud Application[C]// IEEE International Conference on Services Computing. IEEE Computer Society,2018:169-176.

[12] GUI H,FENG Y C,LI Y K. Research on data management system oriented to stream data[J]. Journal of Computer Applications,2005,22(1):88-90. (in Chinese)
桂浩,冯玉才,李又奎. 面向流数据的数据管理系统的研究[J]. 计算机应用研究,2005,22(1):88-90.

[13] CARNEY D,ETINTEMEL U,CHERNIACK M,et al. Monitoring streams:a new class of data management applications[C]// Proc. International Conference on Very Large Data Bases. Hong Kong,China,2002:215-226.

[14] CHANDRASEKARAN S,COOPER O,DESHPANDE A,et al. TelegraphCQ:continuous dataflow processing[C]// ACM SIGMOD International Conference on Management of Data. ACM,2003:668-668.

[15] HASSAN M,BANSAL S K. Semantic Data Querying over NoSQL Databases with Apache Spark[C]// IEEE International Conference on Information Reuse and Integration. IEEE Computer Society,2018:364-371.

[16] HOU R J,FANG J,ZHANG J J. A data query method for real-time write protection of streaming data[J]. Journal of Computer Applications,2014,31(9):2736-2740. (in Chinese)
侯荣军,房俊,张建静. 一种流数据实时写入保障下的数据查询方法[J]. 计算机应用研究,2014,31(9):2736-2740.

[17] VIRGILIO R D,MACCIONI A. Distributed Keyword Search over RDF via MapReduce[C]// European Semantic Web Conference. Springer,Cham,2014:208-223.