软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(8):2002−2017 [doi: 10.3724/SP.J.1001.2012.04122]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

XML 关键词检索的查询理解*

李求实[1,2+], 王秋月[1,2], 王 珊[1,2]

[1](数据工程与知识工程教育部重点实验室(中国人民大学),北京 100872)

[2](中国人民大学 信息学院,北京 100872)

Query Understanding for XML Keyword Search

LI Qiu-Shi[1,2+], WANG Qiu-Yue[1,2], WANG Shan[1,2]

[1](Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), Ministry of Education, Beijing 100872, China)

[2](Information School, Renmin University of China, Beijing 100872, China)

+ Corresponding author: E-mail: frankduns@126.com

Li QS, Wang YQ, Wang S. Query understanding for XML keyword search. *Journal of Software*, 2012,23(8): 2002−2017 (in Chinese). http://www.jos.org.cn/1000-9825/4122.htm

Abstract: Compared with flat textual documents, XML documents are annotated with many meaningful tags, which give information retrieval systems a clearer understanding on queried documents. In addition to structured query languages, such as SQL, XQuery and XPath, keyword queries are widely used for XML retrieval because of their simplicity and ease of use. Although a single keyword and its query intention may be ambiguous, two or more keywords can clarify the query intention if possible occurring contexts and interrelationships are considered. This paper proposes the XNodeRelation algorithm to understand users' keyword queries in XML retrieval. In contrast to existing approaches, the study infers users' query intention by taking into account both schematic and statistical information of the XML data and considering the possible occurring contexts and interrelationships of query keywords. Extensive experiments verify the effectiveness of this algorithm.

Key words: XML keyword search; query understanding; target node type; conditional node type

摘 要: 与纯文本文档集相比,使用语义标签标注的半结构化的 XML 文档集,有助于信息检索系统更好地理解待检索文档.同样,结构化查询,比如 SQL,XQuery 和 Xpath,相对于纯关键词查询更加清晰地表达了用户的查询意图.这二者都能够帮助信息检索系统获得更好的检索精度.但关键词查询因其简单和易用性,仍被广泛使用.提出了 XNodeRelation 算法,以自动推断关键词查询的结构化信息(条件/目标节点类型).与已有的推断算法相比,综合了 XML 文档集的模式和统计信息以及查询关键词出现的上下文及其关联关系等推断用户的查询意图.大量的实验验证了该算法的有效性.

关键词: XML 关键词检索;查询理解;目标节点类型;条件节点类型

中图法分类号: TP311 **文献标识码**: A

* 基金项目: "核心电子器件、高端通用芯片及基础软件产品"国家科技重大专项(08XNG040); 国家高技术研究发展计划(863)(2009AA01Z149)

收稿时间: 2010-12-20; 定稿时间: 2011-09-01

随着 XML 的广泛引用,在很多领域出现了大量的 XML 文档,比如科学研究、商业、数字图书馆等等.尽管诸如 XPath,XQuery 等 XML 检索语言具有很强的表达能力,但对于大部分的用户来说,使用诸如 XPath,XQuery 对他们来说太困难了.用户不但需要了解查询语言本身,而且还需要知道被检索文档集的结构信息.而关键词查询由于简单、易用,仍然是使用最广泛的检索方式.但是,关键词查询因其表达的不精确性,往往导致检索系统返回大量的不相关结果.

与传统信息检索中的纯文本文档集相比,XML 文档具有自身的结构层次和语义信息.比如图 1(a)的一段文本,在标注之前,搜索引擎根本无法知道文本的结构和语义信息;而对应的标注过的文本,如图 1(b)所示,则提供了更多的信息给搜索引擎,比如“Tolga Yurek”是一个“author”.这些信息可以帮助检索系统获得更高的检索精度.同样,关键词查询“Tolga Yurek View Maintenance”,搜索引擎只是当作一组关键词处理,无法知道其表达的确切语义.但如果查询表示为“〈author〉Tolga Yurek〈/author〉 〈title〉View Maintenance〈/title〉 〈masterthesis〉”,则搜索引擎可以理解用户是要查找“author”为“Tolga Yurek”的,“title”中包含“View Maintenance”的“masterthesis”,从而准确地返回图 1(b)中的 XML 片段.在面向数据的 XML 文档集中存在大量标签.这些标签不但描述了其包含的文本片段的语义信息,而且标签之间构成了层次化的逻辑结构.大部分的关键词查询请求,通过指定 1 个或者多个元素(字段)上的条件信息,希望返回某个元素类型的 1 个或者多个 XML 元素.然而,让一个普通用户提供这样的结构化查询请求是不现实的.在本文中,我们提出一种方法来自动地推断用户输入的关键词查询的结构信息.

Tolga Yurek, Efficient View Maintenance at Data Warehouses.
1997 University of California at Santa Barbara, Department of Computer Science.

(a) 一个示例文档片段

```
〈mastersthesis key=“ms/Yurek97”〉
    〈author〉Tolga Yurek〈/author〉
    〈title〉Efficient View Maintenance at Data Warehouses.〈/title〉
    〈year〉1997〈/year〉
    〈school〉University of California at Santa Barbara, Department of Computer Science〈/school〉
〈/mastersthesis〉
```

(b) 一个示例 XML 文档片段

Fig.1

图 1

1 相关工作

大部分现有的面向数据的关键词检索系统基于最低公共祖先(lowest common ancestor,简称 LCA)及其变体返回查询结果,比如修剪掉不相关节点的 LCA[1]、节点相互连接的 LCA[2]、有意义的 LCA(MLCA)[3]、最小 LCA(SLCA)[4]、有价值的 LCA(VLCA/CVLCA)[5]、有意义的最低公共实体祖先 MLCEA[6],等等.

文献[7,8]首先考虑了关键词检索的目标节点识别问题,但它们不仅需要 XML 文档集的模式信息,而且还需要用户或者管理员的干预.例如,文献[7]需要系统管理员分割模式信息,文献[8]需要管理员指定模式中的关联边权重以及用户指定返回节点的“度(degree)”和“势(cardinality)”的约束.XSeek[9]使用 XML 的模式信息把文档中的元素标注为实体节点、属性节点和连接节点,然后通过分析用户关键词的查询模式,划分为搜索谓词和返回节点.如果不能直接从查询分析出返回节点,XSeek 使用隐含推断算法猜测返回节点,即主实体节点(包含所有查询关键词的最低最小公共实体节点,如果不存在,即为 XML 文档的根节点).XSeek 的隐含推断算法可被认为是 SLCA 算法的变体,并没有考虑查询关键词的关联和出现上下文.而且,其基于查询模式和隐含推断算法的返回节点识别是基于单个查询结果实例的,一个显而易见的问题是,因为查询关键词歧义的影响,XSeek 会返回许多不同的节点类型作为用户的查询目标节点.比如“ICDE 2000”,XSeek 不但会返回〈proceedings〉作为返回节点,同时还会返回〈inproceedings〉,〈article〉等节点,而实际上,用户只是希望返回关于 ICDE 2000 会议的信息.

XReal[10]提出了 search for 和 search via 节点(对应于目标节点和条件节点)类型的概念,并且提出了目标节

点和条件节点类型的识别算法.文献[10]使用关键词出现的频率来定位返回节点类型.因其忽略了关键词出现的上下文,对于 DBLP[11]数据集上的查询“ICDE 2000”,XReal 错误地返回⟨inproceedings⟩而不是⟨proceedings⟩作为目标节点类型.原因在于,DBLP 中虽然存在包含⟨title⟩…ICDE…2000⟨/title⟩的⟨proceedings⟩元素,同时也存在大量的论文引用会议 ICDE 2000 上的文章,这意味着有更多的⟨inproceedings⟩元素包含⟨cite⟩…ICDE 2000…⟨/cite⟩.作为 XReal 的改进算法,DynamicInfer[12]认为 XReal 中存在目标节点识别的不一致性和异常性问题,认为问题产生的原因在于其公式中的衰减因子 r 不能根据文档集的变化而动态变化,据此提出了衰减因子动态调整算法. DynamicInfer 尽管在某些情况下提高了目标节点的识别精度,但并没有解决 XReal 中存在的问题.XBridge[13]的目标节点类型的评分函数考虑了查询关键词在结果子树中出现的节点频率、结果子树的结构以及查询关键词之间的距离等多种因素,其算法可被看作一个LCA变体,即包含所有查询关键词的以LCA为根节点的子树,其下包含查询关键词的子节点(关键词的出现频率不考虑)越多,子树越紧凑(即包含关键词的节点到 LCA 节点的非累加距离和最小),这棵子树的评分就越高.某一个候选目标节点类型的 top-k 个结果实例的评分和,即为此候选目标节点类型的评分.XBridge 考虑了查询关键词之间的关系,但仍然忽略了 XML 元素中的语义信息和关键词出现的上下文.

已有的查询理解的工作基本上都是基于 LCA/SLCA 及其变体的,它们使用 LCA/SLCA 或者实体 LCA/SLCA 作为候选目标节点或者候选目标节点类型.因而,无论返回何种节点类型,其最终的查询结果都只可能是一棵以 LCA/SLCA 作为根节点的子树.而很多情况下,用户的目标是这棵子树上的一个节点不一定是根节点.比如来自于 XSeek 的示例查询“team Rockets center”,用户的查询目的相当明确,即返回“Rockets”队中位置为“center”的“player”,但已有的所有算法都会把“team”或者其祖先节点作为目标节点返回.XBridge 考虑了查询关键词之间的节点关联,但也只是计算它们与 LCA 节点的距离而已,并没有考虑它们之间的语义关联.

在本文中,我们通过分析查询关键词的关联和它们出现的上下文以及上下文之间的整体语义联系,推断查询关键词的条件节点类型;然后,在每一个候选目标节点类型上计算查询关键词的信息贡献以及查询关键词节点类型的内容剩余信息量,成功地识别出用户查询的目标节点类型.本文的主要贡献可以概括如下:

(1) 通过集成查询关键词的频率、上下文以及内部关联,本文提出了一种推断条件节点类型的有效算法;
(2) 通过计算查询关键词在候选目标节点类型信息贡献及其候选节点类型的内容剩余信息量,本文提出了一种有效的目标节点类型识别算法;
(3) 通过大量的实验,验证了上述算法的有效性和容错性.

2 背景知识和问题定义

2.1 数据模型和概念

每一个 XML 文档都被建模为一个有向图 $G=\{V,E,root\}$.V 表示 XML 文档中所有的元素、属性和文本节点的集合;$root$ 为 XML 文档的根节点;$E=E_t\cup E_r,E_t$ 表示 XML 文档中的所有树边(即不考虑 XML 节点间的引用关系时,XML 节点树中的边)的集合,E_r 表示 XML 节点之间 ID/IDREF 表示的引用边的集合.图 2 显示了 DBLP 文档集中的一个 XML 片段的数据图.其中,节点“inproceedings(3)”分别通过节点“crossref(8)”和节点“cite(7)”引用节点“proceedings(4)”和节点“article(5)”.因而对于图 2 中的数据图,E_r={(7,5),(8,4)}.我们用 G_{tree} 表示只考虑了 G 中的树边得到的子图,即 $G_{tree}=\{V,E_t,root\}$.

为了便于进行算法描述,我们定义了如下概念:

(1) 节点类型.对于一个节点 t,其节点类型为 G_{tree} 中从 $root$ 到 t 的路径.如图 2 所示,节点“article(5)”的节点类型为“dblp.article”.在不产生混淆的情况下,我们使用路径的最后一个标签名表示节点类型.比如“dblp.article”简化为“article”.
(2) 节点实例.对于一个给定的节点类型 T,其节点实例为 V 中所有类型为 T 的节点.比如,图 2 中“inproceedings”的实例节点为节点 3 和节点 6.
(3) 实体节点.作为实体节点的 XML 节点,一般对应于现实世界中的实体,通常具有 1 个或者多个属性节

点,比如图 2 中的“article(5)”,“proceedings(2)”以及“inproceedings(3)”节点.实体节点描述了一个节点的语义类别,同样也可以描述一个节点类型,即实体节点类型.下面的属性节点、关联节点的定义与此类似,不再累述.

(4) 属性节点.属性节点描述了实体节点的某种属性,比如图 2 中的“title”和“year”节点.

(5) 关联节点.关联节点既不是实体节点,也不是属性节点,而是连接实体节点、属性节点以及其他关联节点的 XML 节点.比如图 2 中的关联节点“cite(7)”和“crossref(8)”.我们使用文献[9]中提出的算法来识别实体、属性以及关联节点.

(6) 父实体节点.节点 e 的父实体节点为其最低祖先实体节点,如果 e 为一个实体节点,其父实体节点即为 e 本身.

(7) 结构摘要树.我们采用文献[14,15]中的算法,可以为每个 XML 文档集创建一棵结构摘要树.摘要树中的每个节点表示一个节点类型,其索引信息包含节点类型标识符、语义节点类别、实例节点个数等基本的统计信息.节点类型标识符为结构摘要树的先序遍历次序号;语义节点类别表示当前节点类型是实体,属性还是关联节点类型.对于图 2 中的 XML 文档片段,其结构摘要树如图 3 所示.

(8) 父实体节点类型.节点类型 T_1 的父实体节点类型为其在结构摘要树上的最低祖先实体节点类型,如果 T_1 为一个实体节点类型,其父实体节点类型即为 T_1.

(9) 内容关键词.出现在 XML 节点值中的关键词,比如图 2 中的“information,xml,zhao”.

(10) 标签关键词.作为标签名称出现的关键词,比如图 2 中的“inproceedings,publisher”.一个节点包含的所有关键词为其内容关键词和标签关键词集合的并,比如图 2 中节点 9 的所有关键词为{title, information,retrieval}.一个节点的内容关键词会向上传播到其父节点,至父实体节点停止.比如图 2 中的节点 6 包含的所有关键词为{inproceedings,title,information,retrieval,author,sun}.另外,需要注意的是,一个标签关键词可能对应多个节点类型,比如关键词“title”,在图 2 中分别对应节点类型“inproceedings.title”和“proceedings.title”.

(11) 节点类型 T 包含关键词 k.至少存在 1 个类型为 T 的实例节点包含关键词 k.一个实例节点 t 包含关键词 k 表示 k 出现在 t 的关键词集合中.

(12) 实体节点关联图.为了处理实体节点之间的关联,我们根据图 $G=\{V,E,root\}$ 派生出实体节点关联图 $G_{entity}=\{V_e,E_e\}$.$V_e=\{v_i|v_i$ 是一个实体节点或者一个关联节点,如果 v_i 是一个关联节点,$v_i\neq root,v_i\in V\}$;$E_e=\{e_i|e_i=(v_i,v_j),v_i,v_j\in V_e$ 并且 $e_i\in E\}$.G_{entity} 中节点的标识符和语义节点类型仍然和它在图 G 中的一样.图 4 显示了一个派生于图 2 的实体节点关联图,因为根节点“dblp(1)”不是一个实体节点,所以它并没有出现在实体节点关联图中.

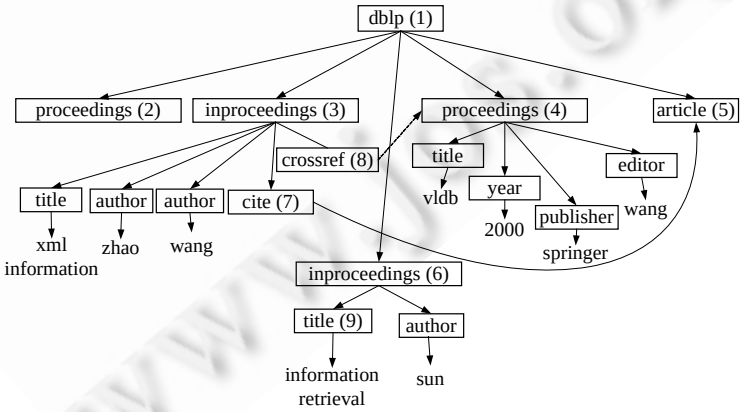

Fig.2 Data graph for a DBLP XML fragment

图 2 DBLP XML 文档片段数据图

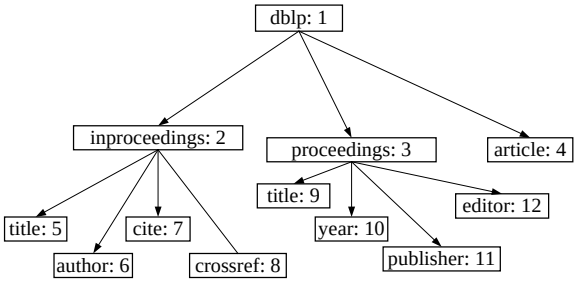

Fig.3 Structure summary tree for XML data graph in Fig.2

图 3 图 2 中的 XML 数据图的结构摘要树

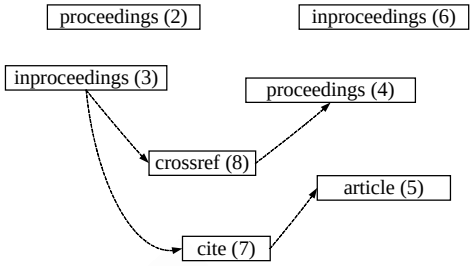

Fig.4 Entity node association graph for XML data graph in Fig.2

图 4 图 2 中的 XML 数据图的实体节点关联图

2.2 问题定义和假设

关键词查询 Q 由一组查询关键词组成,即 $Q=k_1,k_2,...,k_n$.用户可能在查询中显式地指定了查询的条件节点类型和目标节点类型,也可能没有.根据是否查询中包含了条件和目标节点类型,我们把关键词查询划分为以下 4 类:1) Q 既没有指定条件节点类型也没有指定目标节点类型;2) Q 指定了目标节点类型,但没有指定条件节点类型;3) Q 指定了条件节点类型但没有指定目标节点类型;4) Q 同时指定了条件和目标节点类型.

表 1 显示了几个关键词查询的示例.Q_1 没有指定任何条件或者目标节点类型,Q_2 指定了目标节点类型为“article”,Q_3 指定条件节点类型为“author”,Q_4 指定了条件和目标节点类型.

Table 1 Examples of keyword queries

表 1 示例关键词查询

编号	查询	类别	查询意图(目标节点类型)
Q_1	TWIGSTACK	1	不确定
Q_2	ARTICLE JIM GRAY	2	⟨article⟩
Q_3	AUTHOR JIM GRAY	3	⟨article⟩, ⟨inproceedings⟩, ⟨author⟩
Q_4	AUTHOR JIM GRAY ARTICLE	4	⟨article⟩

即使用户的关键词查询中指定了目标和条件节点类型,但因为 1 个标签关键词往往对应于多个节点类型,以及条件和目标节点类型的位置的不固定性,我们仍然需要推测查询关键词在查询中具体对应的节点类型,以及这个节点类型是目标还是条件节点类型.在给出形式化的问题定义之前,我们做以下假设:

假设 1. 一个查询只有一个目标节点类型.大部分面向数据的关键词查询往往仅需要返回 1 种目标节点类型.对于那些需要返回多种节点类型,甚至多种不同节点类型的连接查询,我们留作未来的工作.

假设 2. 查询中的关键词是次序相关的.查询关键词是按照一定的次序排列的.如果两个关键词彼此临近,那么它们更可能相互关联.比如,查询“Jim Gray xml”不应该被表示为“Jim xml Gray”.

假设 3. 所有的查询关键词对于返回相关的信息都是必需的,即没有查询关键词是无用的.比如,如果一个用户希望查找“Jim Gray”的一般信息,那么他不应该提交一个像“Jim Gray 1900”这样的查询.

基于上述假设,关键词查询的理解问题被定义为:给定一个由一系列关键词构成的查询 $Q:k_1,k_2,...,k_n$,我们返回一个目标节点类型 T_{target} 和多个条件节点类型 $T_{ci}(i=1,\ldots,m)$,并且每个关键词 k_i 被指定到一个包含它的条件节点类型 T_{ci},即算法的输出为$\langle T_{target}\rangle:\langle T_{c1}\rangle k_1,\ldots,k_{i1}\langle/T_{c1}\rangle,\ldots,\langle T_{cm}\rangle k_{im},\ldots,k_n\langle/T_{cm}\rangle$.

3 推断条件和目标节点类型

为了推断查询关键词应属于的条件节点类型,我们综合考虑了关键词在节点类型中出现的频率和包含该关键词的节点类型的区分能力.正如前面的示例所示,“title”节点类型对于“proceedings”节点类型而言,其重要性应该大于“cite”对于“inproceedings”.因而,如果一个用户提交查询“ICDE 2000”,那么其查询目的更可能是查找

有关 ICDE 2000 的“proceedings”而不是引用了发表在 ICDE 2000 上的“inproceedings”,虽然“ICDE 2000”在“inproceedings.cite”中出现的频率比在“proceedings.title”中出现的频率要高.此外,关键词出现的条件节点类型可以通过考察查询中的所有关键词可能出现的上下文之间的关联进一步明确.比如,单个查询关键词“java”可能是一个国家名称,也可以是一个编程语言名称.但如果和其他查询关键词组合在一起,比如“Java population”,我们就能够推断出“Java”应该是指一个国家的名称.

基于上述观察,通过考虑关键词出现上下文(即包含关键词的节点类型)的区分能力、所有查询关键词出现上下文之间的相互关联以及 XML 节点类型之间的语义联系,我们提出一种新方法来理解 XML 关键词查询.具体来说,我们的查询理解算法是:首先,通过计算查询关键词出现的上下文的区分能力及其相互关系,我们推断出查询关键词的条件节点类型;其次,我们计算第 1 步推断出的所有条件节点类型在结构摘要树上的最低公共祖先(LCA),并提取出结构摘要树中以此 LCA 为根节点的子树中的所有实体节点类型作为候选目标节点类型集合;最后,通过计算特定条件信息熵(specific conditional entropy)来度量每一个候选目标节点类型和条件节点类型的关联强度,以及候选目标节点类型的剩余内容信息量.我们为每一个候选目标节点类型得到一个评分,评分最高的即为关键词查询的目标节点类型.

3.1 查询关键词出现的上下文的重要性权重

正如我们之前所讨论的那样,出现在不同的上下文中的关键词有不同的重要性.我们使用节点包含的所有关键词的逆节点频率的平均值,并归一化到(0-1)来度量其重要性权重,如公式(1)所示.其中,T_a 为某一属性节点类型,K_a 表示 T_a 包含的不重复的内容关键词的集合,N_a 表示 T_a 的实例节点总数,$n_{a,i}$ 表示 T_a 中包含关键词 $k_i(k_i \in K_a)$的实例节点总数,N_c 表示文档集中的所有非值节点总数.

$$w(T_a)=2\times\arctan\left(\frac{\sum_{k_i\in K_a}(N_a/n_{a,i})+N_c/N_a}{|K_a|+1}\right)\Big/PI \tag{1}$$

示例 1:如图 3 所示的节点类型“dblp.inproceedings.title”(T_a),K_a={xml,information,retrieval},$|K_a|$=3;T_a 仅有两个实例节点,因而 N_a=2;T_a 中包含关键词“xml”,“information”和“retrieval”的节点个数分别为 1,2,1,总的节点个数 N_c=17,则 $w(T_a)$=2×arctan[(2/1+2/2+2/1+17/2)/(3+1)]/PI=2×arctan[27/8]/PI.

3.2 度量实体节点之间的关联

我们使用两个实体节点在实体节点关联图 G_{entity} 上的距离来度量它们的关联性.

定义 1(实体节点距离). 如果实体节点 v_i 和 v_j 在 G_{entity} 上是连通的,它们之间的实体节点距离 $Dist(v_i,v_j)$等于二者在 G_{entity} 上的最短路径长度;否则等于 G_{entity} 上的最大路径长度 Max$Dist$.如果 v_i=v_j,则 $Dist(v_i,v_j)$=0.

示例 2:如图 4 所示,实体节点 3 和节点 5 之间的距离 $Dist$(3,5)=2;因为实体节点 2 和节点 6 之间是不连通的,所以 $Dist$(2,6)=Max$Dist$.

3.3 推断条件节点类型

用户通过查询关键词指定了 1 个或者多个实体对象的属性,通过这些属性最终定位具体的目标实体对象(或者其属性)信息,因而我们定义关键词要修饰的最终实体节点的类型为其条件节点类型.

假定包含关键词 k_1 的实体节点类型为 $TS_1=\{U_1,U_2,\ldots,U_n\}$,包含关键词 k_2 的为 $TS_2=\{V_1,V_2,\ldots,V_m\}$,查询关键词指定的标签名称对应的节点类型集合为 $QS=\{T_{q,1},T_{q,2},\ldots,T_{q,s}\}$.下面我们讨论如何推断每个关键词的条件节点类型.

3.3.1 度量单个关键词 k_1 在候选条件节点类型 U_i 上的置信度

原理 1(最大化单个查询关键词的区分能力). 为了定位期望的信息,用户会选择一个可以尽可能精确地表达其查询需求的查询关键词.比如,一个用户打算查找某人的信息,如果可能的话,他会直接输入这个人的名字而不是年龄来表示自己的查询需求.与年龄相比,这个人的名字有更强的区分能力.

对于每一个实体节点类型 $U_i \in TS_1$,我们依照公式(2)和公式(3)来计算 U_i 作为关键词 k_1 的条件节点类型的置

信度 $C(U_i,k_1)$.这里,u_i 是 U_i 的一个实例节点.在公式(3)中,SA_i 表示在 u_i 中包含关键词 k_1 的属性节点集合(如果关键词 k_1 作为 u_i 的标签关键词出现,则此集合包含 u_i 本身),T_a 是属性节点 t_a 的节点类型.$w(t_a,k_1)$是关键词 k_1 在属性节点 t_a 中出现的频率,依照公式(4)计算.在公式(4)中,$tf(k_1,t_a)$是关键词 k_1 在 t_a 的关键词内容中出现的次数,$|t_a|$是 t_a 的关键词内容长度.

$$C(U_i,k_1)=\max(C(u_i,k_1)) \tag{2}$$

$$C(u_i,k_1)=\sum_{t_a\in SA_i}[w(T_a)\times w(t_a,k_1)\times w(T_a,QS)] \tag{3}$$

$$w(t_a,k_1)=tf(k_1,t_a)/|t_a| \tag{4}$$

函数项 $w(T_a,QS)$表示用户指定的查询节点类型对条件节点置信度的影响,其中,$ancesor\text{-}or\text{-}self(T_q,T_a)$表示 $T_q=T_a$ 或者 T_q 是 T_a 的在 QS 集合中的最低祖先节点类型.即,如果一个查询关键词 k_1 出现的节点类型 T_a 被用户查询关键词对应的节点类型 T_q 包含,则这个查询关键词出现的候选条件节点类型的置信度被放大$(1+w(T_q))$倍.比如在图 2 所示的示例数据上的查询“vldb 2000 PROCEEDINGS”,其中,k_1=“vldb”的候选条件节点类型 U_1=“PROCEEDINGS”,u_1=“proceedings(4)”,T_a=“PROCEEDIGNS.TITLE”.由于 QS={“PROCEEDINGS”},则 T_q=“PROCEEDINGS”.$C(u_1,k_1)=w(T_a)\times w(t_a,k_1)\times(1+w(T_q)$.

$$w(T_a,QS)=\begin{cases}1, & (\neg\exists T_q\in QS)\wedge ancestor\text{-}or\text{-}self(T_q,T_a)\\ 1+w(T_q), & (T_q\in QS)\wedge ancestor\text{-}or\text{-}self(T_q,T_a)\end{cases} \tag{5}$$

3.3.2 度量两个关键词 k_1,k_2 分别在候选条件节点类型 U_i,V_j 上的联合置信度

原理 2(最大化两个查询关键词的组合区分能力). 如果两个关键词组合起来的区分能力大于它们单个的区分能力之和,这两个关键词就组合到一起;否则,它们被单独处理.因为单个关键词的内在歧义性,其表达的信息可能关于不同节点类型的实体节点.如图 2 所示,关键词“wang”可能是一个“inproceedings”中一个“author”的姓,也可能是一个“proceedings”中一个“editor”的姓.但如果这个关键词和“VLDB”组合到一起表示一个查询时,即“wang VLDB”,用户更可能是查找有关 VLDB,并且有一个姓“wang”的“editor”的“proceedings”信息.因而,我们可以推断出查询关键词“VLDB wang”的条件节点类型为“dblp.proceedings”(“dblp.proceedings.editor”为一个属性节点类型,其父实体节点类型为“dblp.proceedings”).

我们使用两个查询关键词的联合置信度来度量它们组合后的区分能力,根据其联合置信度来决定它们是属于同一个条件节点类型,还是不同的条件节点类型.U_i 作为 k_1 的条件节点类型,V_j 作为 k_2 的条件节点类型的联合置信度 $C(U_i,V_j,k_1,k_2)$依照公式(6)计算,u_i 和 v_j 分别是 U_i 和 V_j 的实例节点.

$$C(U_i,V_j,k_1,k_2)=\max(C(u_i,v_j,k_1,k_2)) \tag{6}$$

定义 2(关键词距离). $Dist(u_i,v_j,k_1,k_2)$如公式(7)定义所示.假定实体节点 u_i 和 v_j 中包含关键词 k_1 和 k_2 的属性节点分别为 t_{a1} 和 t_{a2};$d(k_1,k_2,t_{a1})$是关键词 k_1 和 k_2 在属性节点 t_{a1} 中的最小距离;$length(x)$为节点 x 的总的关键词内容长度,如果 x 包含的关键词 k_i 为标签关键词,$length(x)=1$;$Dist(u_i,v_j)$是 u_i,v_j 之间的实体节点距离(见定义 1);如果 k_1 和 k_2 在一个实体节点中出现多次,那么它们之间的关键词距离定义为所有出现的关键词对之间的最小距离.

$$Dist(u_i,v_j,k_1,k_2)=\begin{cases}1, & t_{a,1}=t_{a,2},\text{并且}k_1,k_2\text{有且只有1个是标签关键词}\\ d(k_1,k_2,t_{a,1}), & t_{a,1}=t_{a,2},\text{并且}k_1,k_2\text{都是内容关键词}\\ (length(u_i)+length(v_i)-1)\times(Dist(u_i,v_i)+1), & t_{a,1}\neq t_{a,2}\end{cases} \tag{7}$$

示例 3:如图 2 所示,关键词“xml”和“information”在实体节点 3 中的关键词距离等于 1;“vldb”和“2000”在实体节点 4 中的关键词距离等于(4+4−1)×(0+1)=7;“zhao”和“vldb”之间的关键词距离等于(4+4−1)×(2+1)=21,实体节点 3 和节点 4 之间的实体节点距离等于 2,它们的文本长度都等于 4.

我们依照公式(8)计算公式(6)中的实例关联置信度 $C(u_i,v_j,k_1,k_2)$.关键词 k_1 和 k_2 的条件节点类型 T_{c1} 和 T_{c2} 分别等于它们的联合置信度取最大值时的节点类型,$T_{c1}\in TS_1$ 并且 $T_{c2}\in TS_2$.

$$C(u_i,v_j,k_1,k_2)=(C(u_i,k_1)+C(v_j,k_2))/Dist(u_i,v_j,k_1,k_2) \tag{8}$$

3.3.3 度量多个查询关键词的 $k_1,k_2,k_3,\ldots$的整体条件节点类型置信度

对于包含多个关键词的查询,我们首先计算 k_1 和 k_2 之间的关联置信度,判别它们是否属于同一个条件节点类型.如果它们属于同一个条件节点类型 T_c,那么这两个关键词被合并到一个关键词分组 KG.然后计算 k_2 和 k_3 之间的关联置信度,如果 k_2 和 k_3 都属于条件节点类型 T_c,我们添加 k_3 到关键词分组 KG 中;否则,我们保持 k_1 和 k_2 的条件节点类型不变,依次计算 k_3 和之后的关键词的条件节点类型,直到所有的查询关键词都被处理完毕.

但通过相邻查询关键词的关联分析仍然存在歧义性问题,比如 DBLP 上的查询“Digital Libraries 2000 PROCEEDINGS”,用户希望查找 2000 年有关“Digital Libraries”的会议.通过上述相邻查询关键词的联合置信度计算,可以得出“Digital Libraries”的条件节点类型为“dblp.article”.这显而易见与用户的查询目的不相吻合.出现这种情况的原因在于,仅仅分析相邻关键词的关联关系,在很多情况下仍无法确定其在查询中表达的确切语义.

定义 3(查询关键词分组(query keywords group)). 每一个查询关键词分组 KG_i 包含:1) 属于此分组的关键词列表 KL:$\{k_i,k_{i+1},\ldots,k_{i+h}\}$;2) 此分组的候选条件节点类型列表以及对应的评分 TL:$\{\{T_{c1},Score(T_{c1})\},\{T_{c2},Score(T_{c2})\},\{T_{c3},Score(T_{c3})\},\ldots\}$;3) 包含所有关键的实体节点列表 EL:$\{e_1,e_2,\ldots,e_d\}$;4) T_c,保存此分组计算过程中的当前最高评分的候选条件节点类型,以及此关键词分组计算后的最终条件节点类型.

定义 4(候选条件节点类型距离). 两个关键词分组 KG_1,KG_2 对应的候选条件节点类型 T_{c1},T_{c2} 之间的距离 $Dist(T_{c1},T_{c2})$定义是:$KG_1.EL$ 中类型为 T_{c1} 与 $KG_2.EL$ 中类型为 T_{c2} 的实体节点两两之间距离的最小值.

$$UnionScore(T_{c1},T_{c2},\ldots,T_{cm})=\frac{\sum_{i=1}^{m}Score(T_{ci})}{\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}Dist(T_{ci},T_{cj})} \tag{9}$$

为了解决上述问题,在计算相邻关键词的两两关联置信度时,不是仅仅保存两两关联置信度最高的节点类型,而是保存置信度评分最高的 k(参数,一般设为 3)个节点类型以及相应的置信度评分作为候选条件节点类型.即经过上述处理,查询关键词列表被转换为一个查询关键词分组列表 KGL:$\{KG_1,KG_2,\ldots\}$.然后依照公式(9)计算每一组候选条件节点类型与其他候选条件节点类型的组合的整体评分 $UnionScore(T_{c1},T_{c2},\ldots,T_{cm})$作为最后的判别依据,评分最高的候选节点类型组合作为每一个关键词分组的条件节点类型.

示例 4:对于 DBLP 上的查询“Digital Libraries 2000 PROCEEDINGS”,经过相邻关键词关联分析,KG_1 的关键词列表为{Digital,Libraries},候选条件节点类型列表为{article,inproceedings,proceedings}(列表已经依据置信度评分排序,下同);KG_2 的关键词列表为{2000,proceedings},候选条件节点类型列表为{proceedings,inproceedings,article}.根据公式(9)计算出的 KG_1 和 KG_2 条件节点类型都为“proceedings”.

3.4 推断目标节点类型

正如第 2.2 节中所分析的,即使用户在查询中指定了标签名称作为条件或目标节点类型的“提示”(因为一个标签关键词往往对应多个节点类型),也需要判断具体的目标节点类型.但是从另一方面来说,如果用户的查询模式毫无歧义地定义了自己的目标节点类型,那么我们也应该能够正确地识别出来.基于上述认识,我们采用两种策略识别用户的目标节点类型:1) 基于查询模式分析;2) 基于特定条件信息熵的查询关键词关联分析.

设查询关键词指定的标签名称对应的节点类型集合为 $QS=\{T_{q1},T_{q2},\ldots,T_{qs}\}$,经过上述第 3.3 节条件节点类型分析之后的结果为 KGL:$\{KG_1,KG_2,\ldots\}$,每个关键词分组 KG_i 对应的条件节点类型为 T_{ci},其条件节点类型集合表示 $CS=\{T_{c1},T_{c2},\ldots,T_{cm}\}$.

3.4.1 基于查询模式的目标节点类型识别

如果一个查询关键词 k 最终被识别为一个节点类型 T_q,那么 T_q 要么作为一个条件节点类型,限定其他查询关键词出现的上下文;要么作为目标节点类型.即,对于任何一个由查询关键词映射的节点类型 $T_{q,i}$,如果其不包含任何其他查询关键词,并且存在 $T_{ci}\in CS$,使得 $T_{q,i}=T_{ci}$ 或者 T_{ci} 是 $T_{q,i}$ 的祖先节点类型,则 $T_{q,i}$ 为目标节点类型.

比如 Mondial 上的查询“CITY Andorra POPULATION”,经过分析之后,$QS=\{T_{q1}$=mondial.country.city,T_{q2}=mondial.country.population,T_{q3}=mondial.country.city.population$\}$.

“CITY Andorra”的条件节点类型为“mondial.country.city”,而“POPULATION”的条件节点类型为“mondial.country.city.population”,即 $CS=\{T_{c1}$=mondial.country.city,T_{c2}=mondial.country.city.population$\}$,因为 T_{q3} 没有包含任何其他查询关键词,并且 $T_{q3}=T_{c2}$,则可以判定其为目标节点类型.

如果使用上述模式分析方法无法确定查询的目标节点类型,则使用第 3.4.2 节中基于特定条件信息熵的目标节点推断算法推断条件节点类型.

3.4.2 基于特定条件信息熵的目标节点类型推断

在推断每一个关键词的条件节点类型之后,我们在结构化摘要树上计算所有条件节点类型的 LCA,然后提取以 LCA 为根节点的类型节点子树中的所有实体节点类型作为候选目标节点类型集合.基于下面的两个指导原则,我们从上述候选目标节点类型中推断最终的目标节点类型:

指导原则 1. 由条件节点类型以及对应的查询关键词定义的搜索条件在目标节点类型上的信息贡献越大越好,即搜索条件对于目标节点类型上的实例节点的区分能力应该尽可能地大.如果信息贡献为 0,则表明对于返回结果没有任何约束能力,这显然与假设 3 相矛盾.

指导原则 2. 一个目标节点类型的剩余内容信息应该大于 0.比如,查找有关姓名为“wang wei”的”author”的有关信息的示例查询 Q:“wang wei”,如果返回节点的文本内容仅仅包含“wang wei”,那么这样的结果对于用户来说是毫无意义的,因为它没有给用户提供任何新信息.

给定一个候选目标节点类型 T 以及查询关键词的条件节点类型列表$\{T_{c1},\ldots,T_{cm}\}$,我们使用 $IG(T|C_1,\ldots,C_m)$ 度量搜索条件 $C_1,\ldots,C_m$ 对候选目标节点类型 T 的信息贡献,使用 $IC(T|C_1,\ldots,C_m)$度量候选目标节点类型 T 的剩余内容信息量.此处,C_i 对应于一个由查询关键词分组 KG_i(参见定义 3)定义的搜索条件.基于上述的两个指导原则,我们定义了公式(10)来计算每一个候选目标节点类型的评分,取得最高评分的节点类型即为最后的目标节点类型.

$$Score(T)=2\times\arctan(IG(T|C_1,\ldots,C_m))/PI\times(2\times\arctan(IC(T|C_1,\ldots,C_m))/PI)\wedge a \tag{10}$$

在公式(10)中,a 是一个用户参数,调整 $IC(T|C_1,\ldots,C_m)$和 $IG(T|C_1,\ldots,C_m)$之间的相对重要程度.通过使用反三角正切函数 arctan,$IC(T|C_1,\ldots,C_m)$和 $IG(T|C_1,\ldots,C_m)$都被归一化到$[0,\pi/2]$之间.

设 T 是一个随机变量,其所有的实例节点 t_i 为事件空间,则 $p(T=t_i)=count(t_i)/|T|$,$|T|$表示 T 的所有实例节点总数,$count(t_i)$表示实例节点 t_i 出现的次数,则如公式(11)定义 T 的熵;类似地,如公式(12)定义其特定条件信息熵:

$$H_e(T)=\sum p(T=t_i)\log(1/p(T=t_i)) \tag{11}$$

$$H_e(T|C_1,...,C_m)=\sum p(T=t_i|C_1,...,C_m)\log(1/p(T=t_i|C_1,...,C_m)) \tag{12}$$

定义 5. 搜索条件 $C_1,\ldots,C_m$ 在节点类型 T 上的信息贡献定义为 $IG(T|C_1,\ldots,C_m)$.其计算如公式(13)所示.

$$IG(T|C_1,\ldots,C_m)=H_e(T)-H_e(T|C_1,\ldots,C_m) \tag{13}$$

示例 5:如图 2 所示,设 T 为“dblp.inproceedings”,则 $p(T$=节点 3)=1/2,$p(T$=节点 6)=1/2,因而

$$H_e(dblp.inproceedings)=1/2\times\log2+1/2\times\log2=\log2.$$

设条件 C_1 为“节点类型 inproceedings 包含关键词 wang”,则 $p(T$=节点 $3|C_1)=1$,$p(T$=节点 $6|C_1)=0$,因而 $H_e(T|C_1)=0$,从而得出 $IG(T|C_1)=\log2-0=\log2$.

对于一个条件节点类型 T 和关键词 k,我们使用最大似然估计计算 k 出现在 T 中的概率,即 $p(k|T)=f_{k,T}/length(T)$.$f_{k,T}$表示 k 在节点类型 T 中出现的次数,$length(T)$表示 T 的内容关键词总长度.依照公式(14)计算节点类型 T 的内容信息量 $H_c(K_T)$,K_T 表示 T 包含的不重复的内容关键词集合,同时也表示代表关键词的随机变量.

$$H_c(K_T)=\sum_{k\in K_T} p(k|T)\log(1/p(k|T)) \tag{14}$$

定义 6. 节点类型 T 在条件 $C_1,\ldots,C_m$ 约束下的剩余内容信息量定义为 $IC(T|C_1,\ldots,C_m)=H_c(K_T|C_1,\ldots,C_m)$,其计算如公式(15)所示.

$$H_c(K_T|C_1,...,C_m)=\sum_{k\in K_T}[p(k|C_1,...,C_m)\times\log(1/p(k|C_1,...,C_m)] \tag{15}$$

示例 6:如图 2 所示,让 T=“dblp.inproceedings”,则 $length(T)=7$,$K_T=\{$xml,information,retrieval,zhao,wang,sun$\}$,

$H_c(K_T)$=5×1/7×log(7)+2/7×log(7/2).假设 C_1 为“dblp.inproceedings 包含关键词 wang”.仅实例节点“inproceedings (3)”满足条件,则 $H_c(K_T|C_1)$=4×1/4×log(4)=log4.

4 主要算法和实现

4.1 索引创建

在索引创建中,所有的属性节点都被转换为元素节点处理.我们除了创建常规的节点倒排索引、关键词倒排索引之外,还参照文献[14,15]创建了数据集的结构摘要树和实体节点关联图.在查询处理中,结构摘要树和实体节点关联图被完全载入内存;节点倒排表和关键词倒排表都保存在 Berkeley DB 的数据表中,并且分别创建了 hash 和 B^+ tree 索引.

4.2 关键词查询理解算法

如算法 1 所示,对于每一个关键词查询,首先提取其中的标签关键词,并把对应的节点类型保存到查询节点类型集合 *QS* 中(第 1 行),然后计算关键词查询的条件节点类型(第 2 行).如果查询节点类型集合 *QS* 不为空,我们通过查询模式分析识别目标节点类型(第 4 行~第 6 行),如果识别失败,获取候选目标节点列表,使用信息贡献和剩余内容信息量推断目标节点类型(第 7 行~第 10 行).最后,输出目标节点类型 id.

算法 1. *proceessQuery*.

输入:一个次序相关的关键词列表 *KL*:$k_1,k_2,\ldots,k_n$.

输出:目标节点类型 id:*targetEleTypeId*.

1. *fillNodeTypeSet*(*KL*,*QS*); //从查询关键词列表中查找所有的标签关键词,并且把每个标签对应的节点类型放到查询节点类型集合 *QS* 中.
2. *identifyConditionNodeType*(*KL*,*KGL*); //计算条件节点类型
3. int *targetEleTypeId*=−1; //初始化目标节点类型 id
4. if (*QS* 集合不空)
5. *targetEleTypeId*=*computeTargetTypeByQueryPattern*(*KGL*,*QS*); //根据查询模式推断目标节点类型
6. end if
7. if (*targetEleTypeId*<0) //无法推断出目标节点类型
8. *getCandidateTargetTypeList*(*KGL*,*CTL*); //在结构摘要树上计算查询关键词分组 *KGL* 中的条件节点类型对应的 *LCA*,然后保存以 *LCA* 为根节点的子树中的实体节点类型到候选目标节点类型列表 *CTL* 中.
9. *targetTypeId*=*inferTargetTypeByNodeAssociation*(*KGL*,*CTL*); //推断目标节点类型
10. end if
11. 输出 *targetTypeId*;

4.2.1 识别查询关键词的条件节点类型

如算法 2 所示,我们首先从查询关键词列表中读取一个关键词到 k_1,然后获取其实体节点列表并计算 k_1 在对应实体节点上的条件节点可信度(第 1 行、第 2 行).接着,我们读取下一个关键词到 k_2,如果 k_2 不空,做类似的计算(第 4 行~第 7 行).然后根据公式(5)计算 top-*k* 个关联可信度评分以及对应于每个查询关键词的候选节点类型列表 TL_1,TL_2(第 8 行,*TL* 中的每个元素内容参见定义 3).如果输出结果列表 *KGL* 为空,或者输出结果的最后一个关键词分组 KG_{last} 并不包含 k_1,则表明 k_1,k_2 属于新的关键词分组,我们添加判断结果到输出列表中(第 9 行、第 10 行).如果 k_1 出现在 KG_{last} 的关键词列表中,则表明 k_1 和其前面的关键词同属一个已经判别出的条件节点类型,此时,我们判别 k_2 是否也属于 KG_{last} 中的条件节点类型,如果是,则添加 k_2 到 KG_{last} 中,并合并相应的实体节点列表(第 11 行~第 15 行).保存 k_2 的信息到 k_1 中,清空 k_2,继续下一个循环(第 17 行).在循环退出之前,如果 k_1 属于一个新的条件节点类型,则添加 k_1 到结果列表中(第 19 行~第 22 行).最后,枚举所有的候选条件节点类型组

合,根据公式(9)计算最终的条件节点类型.

算法 2. *identifyConditionNodeType*. //识别查询关键词的条件节点类型

输入:一个次序相关的关键词列表 KL:$k_1,k_2,\ldots,k_n$.

输出:查询关键词分组列表 KGL:$KG_1,KG_2,...,KG_m$(KG_i 的具体内容参见定义 3).

1. k_1←从关键词列表 KL 中取出第 1 个关键词;
2. *getEntityList*(k_1); //获取包含 k_1 的所有实体节点,并且根据公式(3)计算 k_1 在每一个实体节点上的置信度,同时记录 QS 中的每个节点类型 T_{qi} 是否包含其他查询关键词.
3. while (*true*)
4. k_2←取下一个关键词;
5. KG_{last}←取结果列表 KGL 中的最后一个元素;
6. if (k_2 不空)
7. *getEntityList*(k_2);
8. 根据公式(5)计算 k_1,k_2 的 top-k 个关联可信度以及对应的节点类型 $T_{c,1},T_{c,2}$;
9. if (KG_{last} 为空 OR k_1 没有出现在 KG_{last} 的关键词列表中)
10. *add2ResultList*($k1,k2,T_{c,1},T_{c,2}$); //以 k_1 创建一个新的关键词分组,如果 k_2 和 k_1 的条件节点类型相同,则添加到新分组中.
11. else
12. if ($T_{c,1}$=$T_{c,2}$=KG_{last} 中的条件节点类型)
13. 添加 k_2 到 KG_{last} 的关键词列表中;
14. 赋值 KG_{last} 的实体节点列表为 k_2 和 KG_{last} 的实体节点列表的交集;
15. end if
16. end if
17. k_1=k_2,$T_{c,1}$=$T_{c,2}$,清空 $k_2,T_{c,2}$;
18. else
19. if (KG_{last} 为空 OR k_1 没有出现在 KG_{last} 的关键词列表中)
20. *add2ResultList*(k_1,*NULL*,$T_{c,1}$,*NULL*);
21. end if
22. break; //退出循环
23. end if
24. end while
25. 枚举所有的候选条件节点类型组合,根据公式(8)计算最终的条件节点类型;

4.2.2 推断目标节点类型

推断目标节点算法分为基于模式分析的目标节点识别算法 *computeTargetTypeByQueryPattern*(具体描述见第 3.4.1 节,算法略)以及基于信息贡献和剩余内容信息的推断算法 *inferTargetTypeByNodeAssociation*(见算法 3 和算法 4).如算法 4 所示,为了计算某一个节点类型 T 的评分,我们首先要求出此节点类型对应的所有实例节点列表 *eleList*(T):$e_1,e_2,...,e_n$(第 1 行).然后,逐个处理 *eleList*(T)中的实体节点 *ele*,判断节点 *ele* 是否满足每一个关键词分组 KG 中的实体节点列表定义的搜索条件.即在实体节点图中,*ele* 至少与 KG 中的 1 个实体节点是联通的.如果满足所有的搜索条件,则保存 *ele* 到 *satisfiedEleList*(第 3 行~第 13 行).根据这两个列表,计算搜索条件在候选目标条件节点类型上的信息贡献和候选目标节点类型的内容剩余信息量,并最终得到类型的评分(第 14 行~第 16 行).

算法 3. *inferTargetTypeByNodeAssociation*. //推断目标节点类型

输入:条件节点类型分析输出的查询关键词分组列表 KGL:$KG_1,KG_2,...,KG_m$,候选目标节点类型类表 CTL:

$T_1,T_2,T_3,\ldots$

输出:一个目标节点类型 id.

1. 初始化当前最大类型评分 max*TypeScore*=0.0;
2. for each 节点类型 *T* in *CTL*
3. *typeScore*=*computeTypeScore*(*T*,*KGL*);
4. if (*typeScore*>max*TypeScore*)
5. max*TypeScore*=*typeScore*;
6. *eleTypeId4*max*Score*=当前 *typeScore* 对应的节点类型 id;
7. end if
8. end for
9. 输出 *eleTypeId4*max*Score*;

算法 4. *computeTypeScore*. //计算一个节点类型的评分

输入:节点类型 *T*,查询关键词分组列表 *KGL*.

输出:对应节点类型 *T* 的评分.

1. 定位节点类型 *T* 对应的所有实例节点列表 *eleList*(*T*);
2. 初始化满足搜索条件的实体节点列表 *satisfiedEleList* 为空;
3. for each *ele* in *eleList*(*T*)
4. *reachableEleList*←*ele* 在实体节点图 G_{entity} 上的可达节点列表;
5. 初始化布尔变量 *satisfied*=*true*;
6. for each KG_i in *KGL*
7. if (*reachableEleList* 和 KG_i 中的实体节点列表交集为空)
8. *satisfied*=false;
9. break;
10. end if
11. end for
12. if (*satisfied* 为 true),添加 *ele* 到 *satisfiedEleList* 中;
13. end for
14. 根据公式(13)从 *eleList*(*T*)和 *satisfiedEleList* 计算查询条件在候选目标类型上的信息贡献 *IG*;
15. 根据公式(15)从 *eleList*(*T*)和 *satisfiedEleList* 计算候选目标节点类型的剩余内容信息量 *IC*;
16. 根据公式(10)计算最终的类型评分 *typeScore*;

5 实验和结果分析

我们使用 C++和 Berkeley DB 5.0.26 实现了基于上述算法的一个 XML 数据上的关键词检索的原形系统,命名为 XNodeRelation.我们实现了 3 个最近最相关工作 XReal[10],DynamicInfer[12]以及 XBridge[13]中的算法,与它们进行判断用户查询意图(即目标节点类型)准确性方面的比较.因为本文的工作集中在能够正确理解用户提出的关键词查询,如何高效地实现我们提出的算法将作为进一步的工作来完成.XBridge 提出了两种算法:基于穷举计算的 INL 算法和基于 XSketch 模糊估算的 SDI 算法.SDI 算法的执行效率更高,但 INL 算法目标节点识别精度更高.因为我们的实验是比较目标节点类型的识别精度,所以只实现了其 INL 算法.我们的实验平台为 Windows Server 2008,5G 内存,双核 intel 志强 3.6GHz CPU.

5.1 数据集和测试查询

我们选取了 3 个测试数据集:Baseball[16],mondial[11]和 DBLP[11],表 2 是它们的基本统计信息.Baseball 数据集最小,结构清晰.因为 XML 节点之间不存在引用关系,这个数据集的逻辑结构为一棵树.Mondial 数据集比

Baseball 稍大一些,与 Baseball 数据集的最大差别是其 XML 节点之间存在通过 ID/IDREF 建立的引用关系,因而其逻辑结构为一个图.DBLP 不仅数据集最大(127M),也是 3 个数据集中最复杂的一个.与 Mondial 相比,DBLP 中存在大量的“噪声”数据以及不完整的 ID/IDREF 关联.为了测试我们的算法的容错性,我们没有对这个数据集做任何的手工修改和转换.

Table 2 Basic statistics of test datasets

表 2 测试数据集的基本统计信息

数据集	大小(k)	节点总数	关键词总数	最大深度	平均深度
Baseball	414	26 432	25 555	5	4.958 893
Mondial	1 743	69 846	110 951	5	3.592 74
DBLP	130 726	3 736 406	11 925 921	6	2.902 28

我们对实验室中 20 多个同学提交的测试查询,过滤掉了一些不符合我们查询理解假设的查询,比如查询意图不明确的查询,然后为每个数据集选择了 12 个查询,共 36 个测试查询.如表 3 所示,编号为 Q_{1x} 的查询是在第 1 个数据集 Baseball 上的查询;编号为 Q_{2x} 的查询是在第 2 个数据集 Mondial 上的查询;编号为 Q_{3x} 的查询是在第 3 个数据集 DBLP 上的查询.

Table 3 Test queries

表 3 测试查询

数据集	编号	查询	查询意图
Baseball	Q_{11}	Abbott Outfield	PLAYER
	Q_{12}	Tigers Steve	PLAYER
	Q_{13}	Tigers Starting Pitcher	PLAYER
	Q_{14}	PLAYER Mike Relief Pitcher	PLAYER
	Q_{15}	PLAYER GIVEN_NAME Mike POSITION Relief Pitcher TEAM White Sox	PLAYER
	Q_{16}	PLAYER TEAM White Sox SURNAME Ward	PLAYER
	Q_{17}	Cubs TEAM	TEAM
	Q_{18}	Central Chicago	TEAM
	Q_{19}	American 1998 Kansas City	TEAM
	Q_{110}	TEAM PLAYER Scott Eyre	TEAM
	Q_{111}	CITY New York PLAYER Rigo Beltran TEAM	TEAM
	Q_{112}	DIVISION PLAYER Scott Eyre	DIVISION
Mondial	Q_{21}	Rufiji	RIVER
	Q_{22}	Rufiji COUNTRY	MONDIAL.COUNTRY
	Q_{23}	Andorra la Vella	CITY
	Q_{24}	CITY Andorra	CITY
	Q_{25}	CITY Andorra POPULATION	CITY.POPULATION
	Q_{26}	Europe democracy COUNTRY	COUNTRY
	Q_{27}	Roman Catholic Caribbean Sea	COUNTRY
	Q_{28}	Bosnia and Herzegovina Serb	ETHNICGROUPS
	Q_{29}	Bosnia and Herzegovina Serb PERCENTAGE	ETHNICGROUPS.PERCENTAGE
	Q_{210}	Group 8 member parliamentary democracy	COUNTRY
	Q_{211}	republic multiparty	COUNTRY
	Q_{212}	Papua New Guinea ORGANIZATION	ORGANIZATION
DBLP	Q_{31}	ICDE 2000	PROCEEDINGS
	Q_{32}	APPSEM 2000 Georges Gonthier	INPROCEEDINGS
	Q_{33}	Wisconsin Madison Database Workload	MASTERPTHESIS
	Q_{34}	PHDTHESIS Singh Mumick	PHDTHESIS
	Q_{35}	Database Snapshots ARTICLE	ARTICLE
	Q_{36}	Addison Wesley 1992 Relational Databases BOOK	BOOK
	Q_{37}	Digital Libraries 2000 PROCEEDINGS	PROCEEDINGS
	Q_{38}	INPROCEEDINGS DVSS CITE	INPROCEEDINGS.CITE
	Q_{39}	Database Snapshots JOURNAL IBM Research Report	ARTICLE
	Q_{310}	BOOK Automated Reasoning Normal Forms INCOLLECTION	INCOLLECTION
	Q_{311}	INCOLLECTION ADDS BOOK	BOOK
	Q_{312}	PROCEEDINGS VLDB Entity Relationship INPROCEEDINGS	INPROCEEDINGS

5.2 实验结果及分析

表 4~表 6 分别给出了测试查询在数据集 Baseball,Mondial 和 DBLP 上的实验结果.因为 Baseball 数据集结构清晰、逻辑简单,对大部分查询,4 个系统都能正确地推断出查询意图.对于 Q_{13},查询意图是返回“TEAM”名称中包含“Tigers”,位置为“Starting Pitcher”的球员的信息,XReal 和 DynamicInfer 依据关键词在节点类型中出现频率返回 LCA 节点“TEAM”作为目标节点类型,而 XBridge 基于查询结果实例同样返回 TEAM.而我们的算法通过分析条件节点类型,分别得出“Tigers”的条件节点类型为“TEAM”;“Starting Pitcher”的为“PLAYER”(实际包含节点类型为”POSITION”,但其不是实体节点),则候选节点类型为{TEAM,PLAYER}.然后像第 3.5.2 节讨论的那样,计算它们各自的类型评分并最终返回 PLAYER 作为目标节点.XReal,DynamicInfer 和 XBridge 都以 *LCA* 为基础对节点类型评分,倾向于返回包含所有关键词的 SLCA 节点或其祖先节点,比如查询 Q_{15},Q_{16},Q_{18},Q_{19}.而 XNodeRelation 通过分析查询关键词出现的上下文以及它们的关联关系,成功地推断出目标节点.

Table 4 Results on the Baseball dataset

表 4 Baseball 数据集上的测试结果

查询	XReal	DynamicInfer	XBridge/INL	XNodeRelation	查询意图
Q_{11}	player	Player	player	Player	player
Q_{12}	Team	Team	division	player	player
Q_{13}	Team	Team	team	player	player
Q_{14}	player	Player	player	player	player
Q_{15}	division	Division	team	player	player
Q_{16}	Team	Team	team	player	player
Q_{17}	Team	Team	team	team	team
Q_{18}	league	Division	division	team	team
Q_{19}	season	Season	season	team	team
Q_{110}	Team	Team	team	team	team
Q_{111}	Team	Team	team	team	team
Q_{112}	division	Division	division	division	division

Table 5 Results on the Mondial dataset

表 5 Mondial 数据集上的测试结果

查询	XReal	DynamicInfer	XBridge/INL	XNodeRelation	查询意图
Q_{21}	mondial	mondial	river.name	River	river
Q_{22}	river	river	River	Country	country
Q_{23}	country	country	city.name	City	city
Q_{24}	country	country	City	city	city
Q_{25}	country	country	City	city.population	city.population
Q_{26}	mondial	mondial	Mondial	river.located.country	mondial.country
Q_{27}	country	country	Mondial	country	country
Q_{28}	country	country	Country	ethnicgroups	ethnicgroups
Q_{29}	country	country	Country	ethnicgroups.percentage	ethnicgroups.percentage
Q_{210}	mondial	mondial	Mondial	country	country
Q_{211}	country	country	Government	country	country
Q_{212}	mondial	mondial	Mondial	organization	organization

Table 6 Results on the DBLP dataset

表 6 DBLP 数据集上的测试结果

查询	XReal	DynamicInfer	XBridge/INL	XNodeRelation
Q_{31}	INPROCEEDINGS	INPROCEEDINGS	PROCEEDINGS.KEY	PROCEEDINGS
Q_{32}	INPROCEEDINGS	INPROCEEDINGS	INPROCEEDINGS	INPROCEEDINGS
Q_{33}	INPROCEEDINGS	INPROCEEDINGS	MASTERSTHESIS	MASTERSTHESIS
Q_{34}	PHDTHESIS	PHDTHESIS	PHDTHESIS	PHDTHESIS
Q_{35}	ARTICLE	ARTICLE	ARTICLE	ARTICLE
Q_{36}	INPROCEEDINGS	INPROCEEDINGS	BOOK	BOOK
Q_{37}	INPROCEEDINGS	INPROCEEDINGS	PROCEEDINGS.TITLE	INPROCEEDINGS
Q_{38}	INPROCEEDINGS	INPROCEEDINGS	INPROCEEDINGS	INPROCEEDINGS.CITE
Q_{39}	ARTICLE	ARTICLE	ARTICLE	ARTICLE
Q_{310}	DBLP	DBLP	DBLP	INCOLLECTION
Q_{311}	DBLP	DBLP	DBLP	BOOK
Q_{312}	DBLP	DBLP	DBLP	INPROCEEDINGS

Mondial 数据集的逻辑模型为一个图结构,大部分节点之间从简单的 XML 树来看是平行的.比如 Country, Sea,River,Organization.而实际上,这些节点通过 XML 的 ID/IDREF 建立了紧密的语义联系,比如 Country 和 Sea 之间的相邻关系,Organization 和 Country 之间的包含关系.忽略了这些关系,简单地以频率来推测查询意图,倾向于返回包含所有关键词的 LCA 节点,通常是文档集的根节点,见表 5,XReal,DynamicInfer 和 XBridge 对查询 Q_{26},Q_{210},Q_{212} 的处理,由于忽略了查询关键词出现的上下文的语义信息内容,XBridge 对于查询 Q_{21},Q_{23},Q_{211} 仅仅把用户输入返回给了用户(XBridge 返回的上述目标节点中的信息基本上都包含在查询中).相比之下,XNodeReleation 基本上能够正确地推测出用户的查询意图.对于查询 Q_{26},XNodeRelation 返回“river.located.country”而不是“mondial.country”的原因是,因为尽管“river.located.country”作为一个关联节点(通过 ID 引用其他 Country 节点),但此关联节点同样有属性信息,因而被识别为了一个实体节点.

在 DBLP 中,相对于其他节点,“inproceedings”节点占了很大的比例,这基本上解释了为什么 XReal 和 DynamicInfer 会在很多查询中错误地返回“inproceedings”作为目标节点类型,比如查询 Q_{31},Q_{33},Q_{36},Q_{38}.与 Mondial 数据上的问题类似,XBridge 在某些查询返回的节点类型的信息量太少,比如 Q_{31} 和 Q_{37}.而且因为忽略节点引用关系,XReal,DynamicInfer 和 XBridge 对于查询 Q_{310},Q_{311},Q_{312} 都返回了文档根节点作为目标节点. 相比之下,XNodeRelation 能够推断出正确的查询意图.XNodeRelation 之所以对 Q_{37} 返回错误的结果,是因为在条件节点类型推断时,候选条件节点类型 top-k 设置的 k 值过小(k=5)而使得“proceedings”节点类型被提前过滤掉,从而不可能推断出正确的结果.当设置 k=10 时,则可以返回正确结果.

6 结论和未来工作

通过考虑查询关键词出现的上下文(条件节点类型)和它们之间的语义关联(目标-条件节点关联、内容剩余信息量)等,我们提出了一种 XML 关键词查询理解算法,并且在我们的 XML 关键词检索系统 XNodeRelation 中实现.经过从简单到复杂的各种数据集上的广泛实验,验证了该算法的有效性.尤其是在充满“噪声”的未经处理的 DBLP 数据集上的实验结果,更证明了我们算法的容错和有效性.

未来的工作包括改进算法的性能,比如预计算实体节点距离、多线程等等,以及研究基于查询理解的更有效的结果评分算法,如文献[17,18].

References:

[1] Guo L, Shao F, Botev C, Shanmugasundaram J. XRANK: Ranked keyword search over XML documents. In: Halevy AY, Ives ZG, Doan A, eds. Proc. of the 2003 ACM SIGMOD. San Diego: ACM Press, 2003. 16−27. [doi: 10.1145/872757.872762]

[2] Cohen S, Mamou J, Kanza Y, Sagiv Y. XSEarch: A semantic search engine for XML. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 45−56.

[3] Li Y, Yu C, Jagadish HV. Schema-Free XQuery. In: Nascimento MA, Özsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB, eds. Proc. of the 30th Int'l Conf. on Very Large Data Bases. Toronto: Morgan Kaufmann Publishers, 2004. 72−83. [doi: 10.1145/1066157.1066217]

[4] Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases. In: Özcan F, ed. Proc. of the 2005 ACM SIGMOD. Baltimore: ACM Press, 2005. 527−538. [doi: 10.1145/1321440.1321447]

[5] Li G, Feng J, Wang J, Zhou L. Effective keyword search for valuable LCAs over XML documents. In: Silva MJ, Laender AHF, Baeza-Yates RA, McGuinness DL, Olstad B, Olsen ØH, Falcão AO, eds. Proc. of the 16th ACM Conf. on Information and Knowledge Management. Lisbon: ACM Press, 2007. 31−40.

[6] Huang J, Xu J, Zhou J, Meng X. MLCEA: An entity based semantics for XML keyword search. Computer Research and Development, 2008,45(Suppl.):372−377 (in Chinese with English abstract).

[7] Hristidis V, Papakonstantinou Y, Balmin A. Keyword proximity search on XML graphs. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore: IEEE Computer Society, 2003. 367−378.

[8] Koutrika G, Simitsis A, Ioannidis YE. Précis: The essence of a query answer. In: Liu L, Reuter A, Whang KY, Zhang J, eds. Proc. of the 22nd Int'l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006. 69−78. [doi: 10.1109/ICDE.2006.114]

[9] Liu Z, Chen Y. Identifying meaningful return information for XML keyword search. In: Chan CY, Ooi BC, Zhou A, eds. Proc. of the 2007 ACM SIGMOD. Beijing: ACM Press, 2007. 329−340. [doi: 10.1145/1247480.1247518]

[10] Bao Z, Ling TW, Chen B, Lu J. Effective XML keyword search with relevance oriented ranking. In: Proc. of the 25th Int'l Conf. on Data Engineering. Shanghai: IEEE Computer Society, 2009. 517−528. [doi: 10.1109/ICDE.2009.16]

[11] http://www.cs.washington.edu/research/xmldatasets/

[12] Li J, Wang J. Effectively inferring the search-for node type in XML keyword search. In: Kitagawa H, Ishikawa Y, Li Q, Watanabe C, eds. Proc. of the 15th Int'l Conf. on Database Systems for Advanced Applications. Tsukuba: Springer-Verlag, 2010. 110−124. [doi: 10.1007/978-3-642-12026-8_11]

[13] Li J, Liu C, Zhou R, Wang W. Suggestion of promising result types for XML keyword search. In: Manolescu I, Spaccapietra S, Teubner J, Kitsuregawa M, Léger A, Naumann F, Ailamaki A, Özcan F, eds. Proc. of the 13th Int'l Conf. on Extending Database Technology. Lausanne: ACM Press, 2010. 561−572. [doi: 10.1145/1739041.1739108]

[14] Yu C, Jagadish HV. Schema summarization. In: Dayal U, Whang KY, Lomet DB, Alonso G, Lohman GM, Kersten ML, Cha SK, Kim YK, eds. Proc. of the 32nd Int'l Conf. on Very Large Data Bases. Seoul: ACM Press, 2006. 319−330.

[15] Goldman R, Widom J. DataGuides: Enabling query formulation and optimization in semistructured databases. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases. Athens: Morgan Kaufmann Publishers, 1997. 436−445.

[16] http://www.databasebasketball.com/

[17] Termehchy A, Winslett M. Effective, design-independent XML keyword search. In: Cheung DW, Song IY, Chu WW, Hu X, Lin J, eds. Proc. of the 18th ACM Conf. on Information and Knowledge Management. Hong Kong: ACM Press, 2009. 107−116. [doi: 10.1145/1645953.1645970]

[18] Kim J, Xue X, Croft WB. A probabilistic retrieval model for semistructured data. In: Boughanem M, Berrut C, Mothe J, Soulé-Dupuy C, eds. Proc. of the 31st European Conf. on IR Research. Toulouse: Springer-Verlag, 2009. 228−239. [doi: 10.1007/978-3-642-00958-7_22]

附中文参考文献:

[6] 黄静,徐俊劲,周军锋,孟小峰.MLCEA:一种基于实体的 XML 关键字查询语义.计算机研究与发展,2008,45(增刊):372−377.

李求实(1976－),男,河南周口人,博士,CCF 学生会员,主要研究领域为高性能数据库,XML 数据管理和基于 XML 的信息检索.

王秋月(1974－),女,博士,讲师,CCF 会员,主要研究领域为数据库管理系统,XML 数据管理,结构化信息检索.

王珊(1944－),女,教授,博士生导师,CCF 高级会员,主要研究领域为高性能数据库,数据库与信息检索,内存数据库,视频数据库.