## 用于xml关键字搜索的查询细化框架

**简介**

XML关键字搜索的现有工作集中在如何为查询查找相关且有意义的数据片段，假设每个关键字都是其中的一部分。但是，在XML关键字搜索中，用户查询通常包含不相关或不匹配的术语，拼写错误等，这可能很容易导致空洞或无意义的结果。在本文中，我们介绍了内容感知XML关键字查询细化的问题，其中搜索引擎应该明智地决定在Q的处理期间是否需要细化用户查询Q，并找到有保证的精炼查询候选者列表，其中保证无需任何用户交互或第二次尝试即可在XML数据上获得有意义的匹配结果。为了实现这一目标，我们构建了一个新颖的内容感知XML关键字查询优化框架，包括两个核心部分：（1）我们构建一个查询排名模型来评估精炼查询RQ的质量，该查询RQ捕获形态/语义Q和RQ之间的相似性以及RQ关键字对XML数据的依赖性; （2）我们将RQ候选者的探索和匹配结果的生成集成为一个单独的问题，这可以在相关关键字倒排列表的一次性扫描中实现最佳。最后，一项广泛的实证研究验证了我们框架的效率和有效性。

##### 8、实验

在实验研究中，我们研究了第6节中提出的两种改进算法（即基于分区的方法和短列表急切方法）的效率和可扩展性，以及第4节中提出的查询排序模型的有效性。 请注意，我们不包括针对关系数据库中关键字查询清理而设计的最相关工作[38]的评估，因为将其扩展为适合XML文档几乎是不可行的，并且它不能保证存在 匹配清理的关键字查询的结果。

符号：（1）SLCA是指[53]中针对SLCA计算提出的扫描急切方法。（2）第6节中提出的短列表急切和基于分区的算法分别称为SLE和分区; Partition和SLE在计算查询的SLCA结果时采用扫描急切方法[53]。

设备：所有实验均在运行Windows XP且内存为3GB的1.9 GHz AMD DualCore PC上进行。所有代码都用Java实现，Berkeley DB Java Edition [1]用于存储关键字倒排列表。

数据集和查询集：由于我们的工作是与用户真实搜索体验密切相关的实证研究，因此我们使用真实数据集和实际用户查询而不是合成数据集和查询。据我们所知，XML关键字搜索领域的所有现有工作在研究其方法的实用性时遇到的一个常见问题是缺乏真实世界的数据集和用户查询。

由于缺乏真实的数据集，我们的实验中只使用了两个真实数据集DBLP [23]（420MB，深度= 2，直至2007/12/10）和Baseball4（1MB，深度= 5）。 DBLP包含计算机科学的出版物;棒球包含有关北美棒球联赛球队和球员的信息。这两个真实数据集在数据组织和数据应用方面各不相同：DBLP浅而宽，而棒球又深又窄。我们选择这些不同数据源的目标是了解我们的改进策略在不同的现实环境中的有用性。

关于现实世界的用户查询，从DBLP在线演示5的查询日志中选择最近的1000个查询，其中选择具有空结果的219个查询（平均长度为3.92个关键字）以形成池需要改进的查询，这与本文的主要动机相吻合。此外，我们随机选择100个具有有意义匹配结果的查询，并将它们添加到查询池中，为了增加查询的多样性。

与IR查询细化[16]相同，我们通过要求两个人类注释器手动细化上述219个查询，为本文采用的四个细化操作构建术语级别的细化规则集。但是，细化并不是立竿见影的，因为应用不同规则可能会产生许多可能的细化查询。关于细化规则r的相异度得分dsr，我们采用与第3.3节中描述的相同的度量，并且为单个术语删除指定dsr = 2。

8.1示例查询集

使用典型操作进行细化的样本查询分别显示在表3,4,5和6中，其中每个表中第3列显示我们的方法返回的细化，第4列显示基于相应的结果的基数RQ。 此外，涉及多个混合细化的查询，即QX1-QX6，如下所示。

8.2效率研究

（预解析、查询、检测器、建议器）

在本节中，我们通过测量发出查询与其返回的匹配SLCA结果的Top-K RQ之间的延迟来评估SLE和Partition的效率。我们首先在表3到表6以及QX1-QX6中的所有样本查询中评估SLE和Partition for Top-1查询优化（在8.1节中）。我们还将它们与一种天真的方法进行比较，我们首先处理初始查询，然后在必要时枚举所有RQ候选项，并逐个尝试（按查询排名的降序排列），直到用户满意为止 查询结果; 而不计算用户判断所花费的时间。 此外，我们记录了SLCA [53]处理初始查询所花费的时间，以了解RQ探索带来的额外成本。

图3a-b显示了需要细化的所有样本查询（在热缓存上）所经过的时间，其中我们有四个观察结果。

（1）对于所有样本查询，分区和SLE都优于天真的方法;分区大约比SLE快两倍。

（2）SLCA花费的评估时间最短，因为它只负责处理即使没有有意义的匹配结果的初始查询。相比之下，Partition带来了非常小的额外成本（平均约为30％），但同时用于生成top-1 RQ并在XML数据树中查找匹配结果。

（3）SLE优于QD2和QX3的分区，因为具有最短反转列表的关键字也在最终的Top-1 RQ中，因此避免了相应反转列表的完整扫描。

（4）有趣的是，我们发现对于QM10，QS3，QS11和QD7，分区甚至比不执行任何细化操作的SLCA更有效。这可以解释为，分区在计算RQ候选者的排名分数上花费的额外成本甚至小于SLCA在计算那些查询的无意义LCA（即文档根节点）时的成本。

最后，我们随机选择10个不需要任何细化的查询，并通过SLE，Partition和SLCA调查经过的时间。 如图3f所示，与SLCA相比，平均SLE和分区花费大约20％的额外时间，这是可接受的。

8.3可扩展性研究

（检测器、建议器）改变阈值

为了在Top-K查询细化中测试SLE和Partition的可伸缩性，我们设计了两个实验。

首先，我们测量不同K选择对Top-K查询细化评估时间的影响，其中K [1,6]。尝试了一组40个随机查询，平均长度为3.71（DBLP）和20个随机查询（平均长度为3.18），这些查询在五次执行中的平均时间如图4所示。如图4a所示。 ，分区一直很好，而当K> 3时，SLE的时间增加得更快。由于SLE必须在评估之前找到所有Top-K RQ，因此K越大，相异度计算的额外时间就越多，并且在使用现有方法查找SLCA结果时需要更多次关键字列表扫描。相反，对于分区方法，K越大，排名较低的RQ及其SLCA结果（在排名较高的查询之前检测到的结果）被保留（而不是被删除）的可能性就越高，因此额外成本更低介绍。对于棒球数据，两种算法都可以很好地扩展，如图4b所示。

其次，我们测量SLE和分区对不同大小的数据集的Top-3查询细化的响应时间，这些数据集是从DBLP（420MB）获得的，并且再次使用一批40个随机查询。如图5所示，两种方法都具有良好的数据大小可扩展性。请注意，SLE从60％显着增加到80％， 因为SLE的效率很大程度上依赖于检测到Top-K RQ的早期时间，从而影响关键字倒置列表的随机访问次数。

8.4建议查询的有效性

在验证了算法的效率后，我们在续集中评估了查询排名模型的有效性。