13. IF $(r,s)\notin H_0$

14. *Opt-hash*(r,s);

15. $H_0\leftarrow H_0\cup(r,s)$;

16. ENDIF

17. ENDFOR

18. $count_r\leftarrow count_r+1$;

19. *E.push*$(r,p_r+MultiLen_{ult},sp_r)$ and free H_0;

20. ENDWHILE

在 Opt-join 算法中,我们调用算法 2(*Opt-hash*(r,s))来对候选对进行过滤(第 14 行).同时,为了防止候选对重复计算,在候选对生成之后,增加了一个小哈希表 H_0(第 13 行).在 p_r 到 $p_r+MultiLen_{ult}$ 之间生成的所有候选对,都要通过 H_0 过滤.$MultiLen_{ult}$ 个 Token 处理完成后,释放小哈希表 H_0(第 19 行).

算法 4 负责处理所有 sp_r=1 的前缀事件.前缀事件堆 E 的初始化阶段是将所有 sp_r=1 的前缀事件插入到堆中(详见第 2 节),此时,堆的大小为$|R|$.在对所有$|R|$个前缀事件处理地同时,为每条记录初始化一个计数器(第 6 行),记录当前记录处理的次数.

算法 4. *Bach-one*(·).

1. FOR i=0 TO $|R|$

2. $\langle r,p_r,sp_r\rangle\leftarrow E.pop(\cdot)$;

3. Generate candidate pairs (r,s) via index search;

4. *Opt-hash*(r,s);

5. Update the inverted index and T;

6. $count_r\leftarrow 0$; //Intialize the record execution times

7. *E.push*(r,p_r+1,sp_r);

8. ENDFOR

在 Token 批处理过程中,由于 sim_k 与 θ_{rs} 之间的差值,不可避免地增加了某些记录前缀的长度,从而相应增加了生成的候选对数量.但是这一策略加快了 sim_k 的迭代速度,提高了过滤技术的过滤能力(主要是 Size 过滤和位置过滤),实际上是减小了需要验证的候选对的数量,这在实验的结果中会有所体现.

4 实验评估

本节对 Opt-join 和 Top*k*-join 进行了实验比较.实验环境为:Dell T320,1.9GHz Xeon(R) E5-2420 六核处理器,16G 内存和 1T 磁盘;操作系统为 Ubuntu 12.04;程序设计语言为 C++,编译器为 gcc-4.5.

为了能更好地看出哈希优化策略和 Token 批处理技术带来的性能提升,实验中比较了如下 3 种算法.

(1) Top*k*-join:文献[3]中提出的 Top-*k* 相似连接算法;

(2) 不使用 Token 批处理的 Opt-join(OPT1):该算法在 Top*k*-join 算法的基础上,只使用了哈希优化策略;

(3) Opt-join:该算法在事件驱动框架基础上,结合了 Token 批处理技术,且包含了 OPT1 的优化技术.

4.1 实验数据

与文献[2,3]类似,我们使用 DBLP 和 TREC 数据集来进行实验.DBLP 包含了 0.9M 的记录(作者+标题);TREC 选自于 MEDLINE 数据库,我们从中抽取 0.34M 作为实验数据.预处理之后,数据集的统计信息见表 6.与文献[3]类似,我们使用 Jaccard 和 Cosine 相似度函数衡量记录对的相似度.

Table 6 Dataset statistics

表 6 数据集统计信息

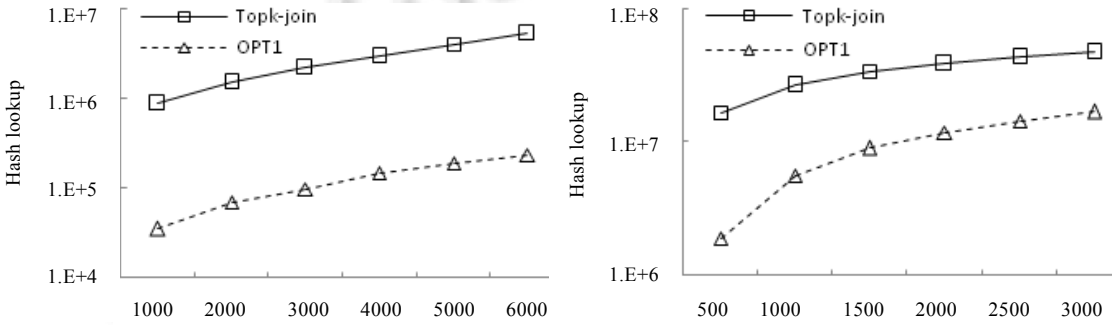
Dataset	*N*	Min.Len	Max.Len	Avg.Len
DBLP	861 567	3	284	14.3
TREC	345 969	24	609	114.4

4.2 实验结果分析

下面我们将从哈希代价、候选对过滤能力、前缀事件堆大小、候选对数量以及运行时间 5 个方面对上述 3 种算法进行比较.受篇幅限制,我们只列出了部分代表性实验结果.

(1) 全局哈希表代价对比

哈希表的代价主要来自于哈希查找.图 3 展示了在 Top*k*-join 算法和 OPT1 算法中,哈希查找次数的对比.*x* 轴表示 *k* 值,*y* 轴表示哈希表查找的次数.可以发现:相对于 Top*k*-join,OPT1 的哈希查找次数明显减少.这是因为在 Top*k*-join 算法中,所有的候选对都要进行哈希查找,只有查找失败的候选对才进行过滤(位置过滤和后缀过滤);而在 OPT1 算法中,先做位置过滤和后缀过滤,然后才进行哈希查找,大量不符合条件的候选对已经被过滤掉了.所以,OPT1 算法的哈希代价会远小于 Top*k*-join.



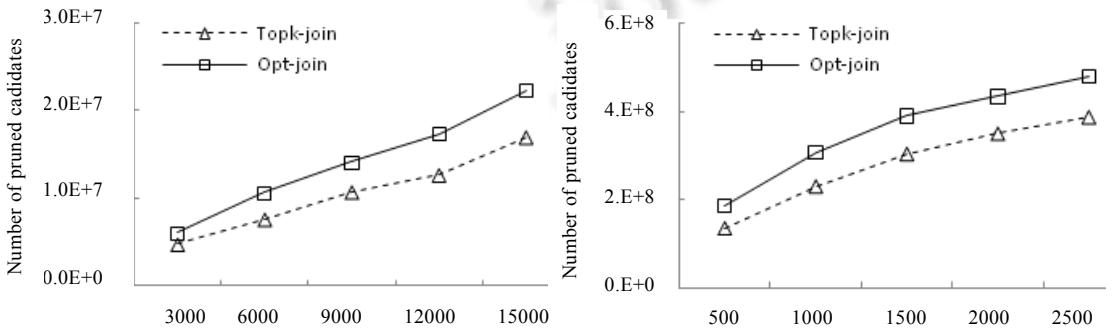
(a) DBLP, Cosine (b) TREC, Jaccard

Fig.3 Cost of hash lookup

图 3 哈希表查找代价

(2) 候选对过滤能力

图 4 和图 5 分别比较了 Top*k*-join 和 Opt-join 算法 Size 过滤和位置过滤的过滤能力,其中,*x* 轴表示 *k* 值,*y* 轴表示被过滤掉的候选对个数.



(a) DBLP, Cosine (b) TREC, Jaccard

Fig.4 Comparison of size filtering capability

图 4 Size 过滤能力的对比

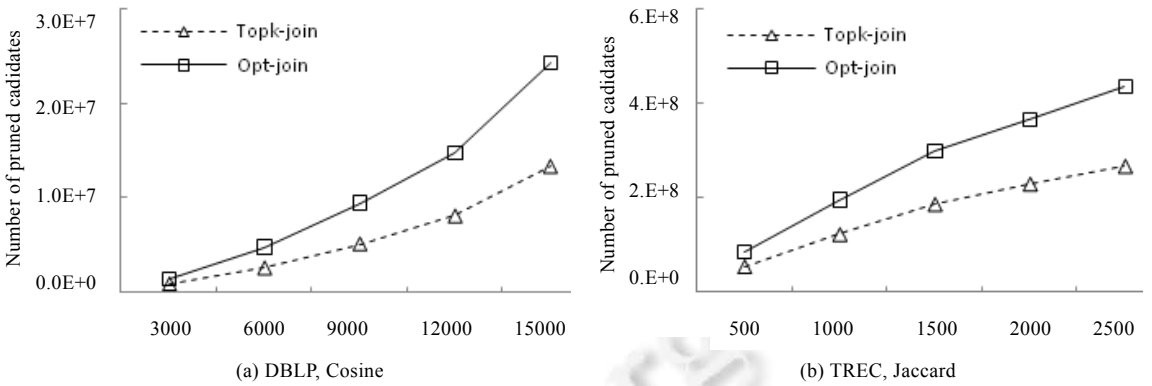


Fig.5 Comparison of positionalfiltering capability

图 5 位置过滤能力的对比

可以发现:在 Opt-join 算法中,Size 过滤和位置过滤的能力均得到了加强.这是由于在 Opt-join 算法中,Token 批处理使得临时结果堆顶的候选对相似度 sim_k 迭代加快.由公式(1)可知,Size 过滤能力主要依赖于 sim_k.sim_k 增长越快,Size 过滤的过滤区间就越小,其过滤能力就越强.所以,sim_k 的快速增长,使得 Size 过滤能力得到加强.

位置过滤能力增强的原因有:

1) 公式(2)和公式(5)中,∂值越大,位置过滤的过滤能力越强.见表 2 中,∂值的大小依赖于 sim_k.因此, sim_k 的迭代加快导致阈值∂迅速增大,从而使得位置过滤能力得到加强;

2) 定理 3 中,使用公式(5)代替公式(2)执行位置过滤后,位置过滤能力得到加强.

由定理 4 可知:使用公式(6)代替公式(3)后,后缀过滤能力将得到加强.但是,由于大量不符合条件的候选对已被 Size 过滤和位置过滤剔除掉,致使后缀过滤能力增加的不明显.以 TREC 数据集 Top-500 查询为例,在 Top*k*-join 算法中,后缀过滤剔除的不符合条件的候选对为 25 662 651;而在 Opt-join 中,剔除的不符合条件的候选对为 26 868 795,后缀过滤的能力只提高了 4.7%左右.

(3) 前缀事件个数

图 6 比较了 Top*k*-join 和 Opt-join 中前缀事件的个数.在 Top*k*-join 中,一个前缀事件只对应一个 Token;而使用了 Token 批处理技术后,一个前缀事件对应于多个 Token.在记录前缀不变的情况下,使用 Opt-join 算法时前缀事件明显减少,这就意味着前缀事件堆护所需的代价也相应降低.

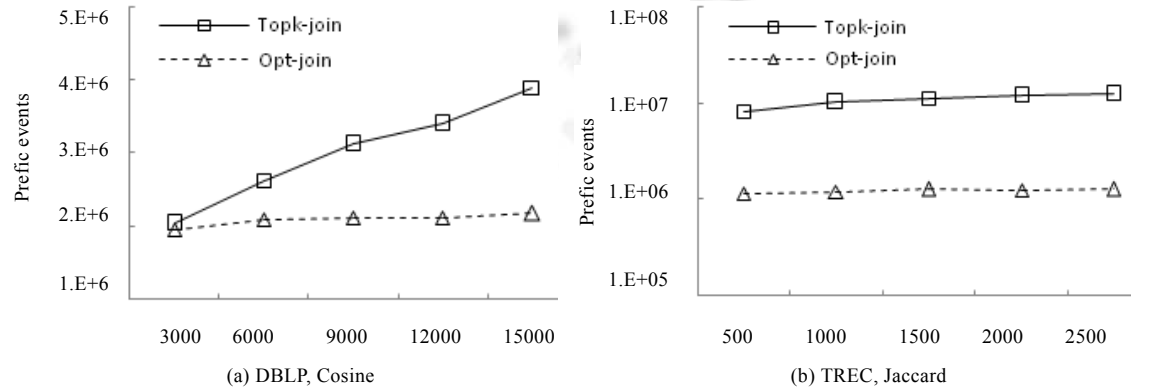


Fig.6 Comparison of the number of prefix events

图 6 前缀事件个数的比较

从图 7 中可以看出:TREC 数据集中,前缀事件减少了 10 倍左右;而在 DBLP 数据集中,前缀事件数目的减少

不如 TREC.主要原因是 DBLP 中记录长度太短,记录的相似度太高,导致计算出的 *MultiLen* 值太小.

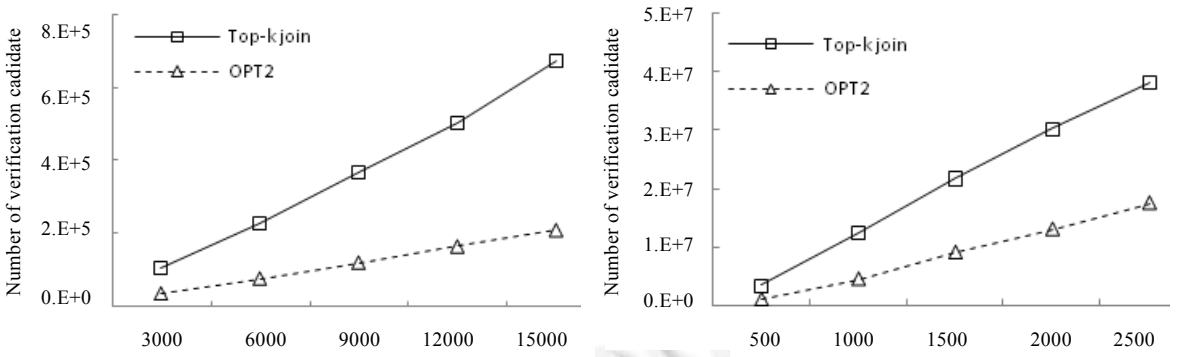


Fig.7 Comparison of the number of verificationpairs

图 7 验证对的比较

(4) 验证候选对数量对比

图 7 给出了 Top*k*-join 和 Opt-join 算法中,验证候选对的数量.验证候选对指通过过滤技术和哈希查找后,需要精确计算相似度的候选对.精确计算是算法代价的主要来源之一,验证候选对越少,精确计算的代价就越低.通过图 7 可以看出:Opt-join 算法使用 Token 批处理技术后,需要验证的候选对数量明显小于 Top*k*-join 算法(DBLP 减少了约 3 倍,TREC 约 2 倍).这是由于 Token 批处理技术使得 sim_k 的迭代加快(更快地达到实际的阈值),从而增强了候选对过滤能力(如图 4 和图 5 所示),使得需要精确计算相似度的候选对数量极大地减少.

(5) 程序运行时间

图 8 给出了 Top-*k* join,OPT1 和 Opt-join 算法程序运行时间的对比.为了方便比较,我们定义加速比为 Opt-join(OPT1)和 Top*k*-join 时间的比值.可以发现:OPT1 算法明显优于 Top*k*-join,主要原因是 Opt-join 算法通过变更哈希查找和过滤操作的位置,使得哈希表查找代价显著减小.Opt-join 算法优于 OPT1,主要是 Opt-join 在 OPT1 的基础上,又使用了 Token 批处理技术,使其在候选对过滤能力、前缀事件个数、候选对数量上都优于 OPT1(如实验 1~实验 4 所示).如图 8(a)、图 8(c)和图 8(d)所示,OPT1 比 Top*k*-join 算法的运行时间减少了 20%左右.在图 8(b)中,OPT1 的性能只提高了 8%,其原因是在相同 *k* 值下,Cosine 相似函数对应的相似阈值高于 Jaccard(例如在 *k*=500 时,Cosine 相似函数下的阈值为 0.96,Jaccard 相似函数下的阈值为 0.92),所以使用 Cosine 相似函数时生成的候选对个数远小于 Jaccard 下的个数(*k*=500 时,Cosine 相似函数下的候选对个数为 140 万,Jaccard 相似函数下的候选对个数为 253 万).候选对越少,Top*k*-join 中哈希代价越低,OPT1 性能提高的就越少.可以发现:随着 *k* 值增加,生成的候选对变得越多,OPT1 的优化也就越明显.

Opt-join 在 OPT1 基础上增加了 Token 批处理,与 Top*k*-join 相比,Opt-join 显著地提高了算法的性能.如图 8(a)所示,Opt-join 取得了 1.28~1.46 加速比.随着 *k* 的不断增加,加速比有不断增大的趋势.在图 8(b)中,Opt-join 的加速比是 1.65~1.69,这是由于相同 *k* 值下,Cosine 相似函数对应阈值高于 Jaccard 下的阈值,所以记录前缀长度小于 Jaccard 下相应的长度,这就导致 Token 批处理的次数减小,从而减少了程序运行时间.在 TREC 数据集下,Opt-join 在 Jaccard 和 Cosine 相似度函数下的加速比分别为 2.61~3.09 和 1.59~2.27,如图 8(c)、图 8(d)所示.这是由于 TREC 数据集中的记录长度比较长,当 *k* 相同时,Cosine 相似函数下的 sim_k 值大于 Jaccard 相似函数下的值.由公式(7)可知:Cosine 相似函数下,*MultiLen* 的长度会小于 Jaccard 相似函数下相应的值.根据公式(11),后续 *MultiLen* 的增长在 Jaccard 相似函数下的值会更大,因此,其批处理次数会减少,相应的加速比较 Cosine 要更好.需要注意的是:在 DBLP 数据集中,记录的长度较短,sim_k 的值比较大,导致计算出的 *MultiLen* 变化较小,所以起决定作用的是前缀的长度.

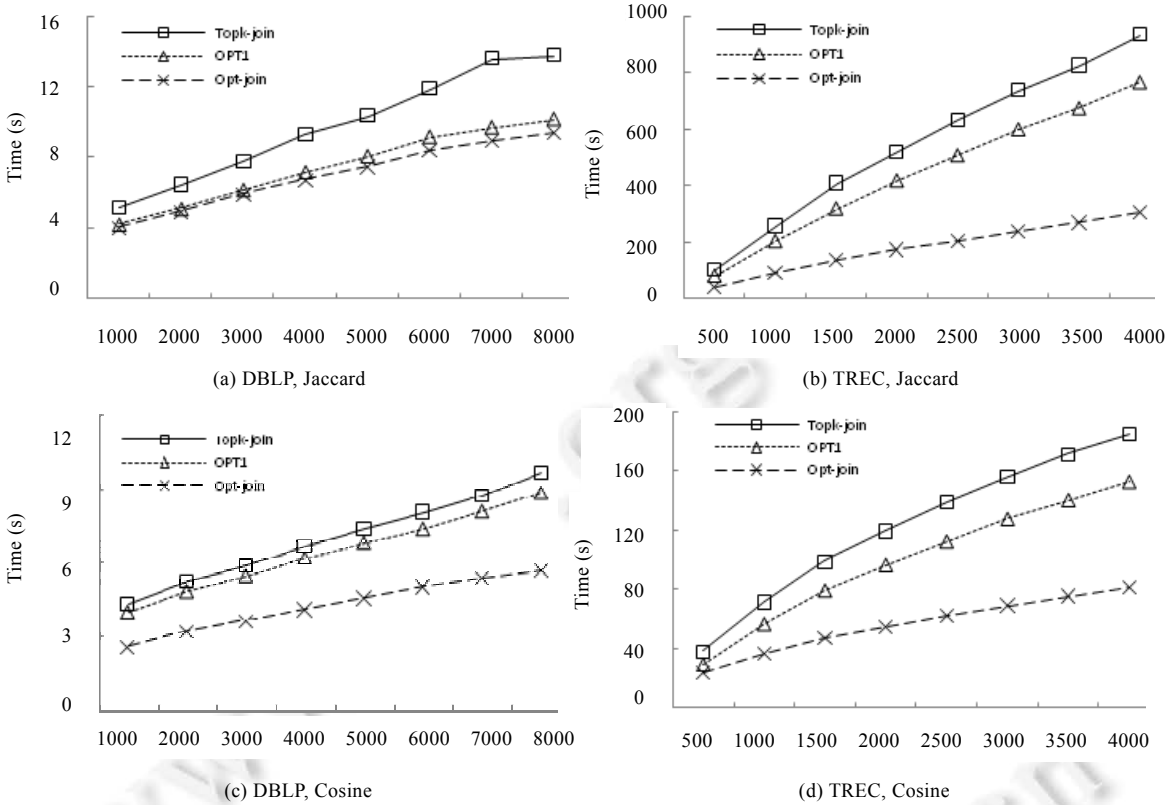


Fig.8 Running time

图 8 程序运行时间

总之,在使用了两个优化技术之后,Opt-join 可以取得 1.28~3.09 的加速比,且随着 k 值的增加或数据集中记录长度的增长,加速比有不断增大的趋势.

5 相关工作

相似连接算法在近几年已经获得广泛的关注.目前的相似连接算法可以归纳为两大类:基于阈值的相似连接和 Top-k 相似连接.

- 基于阈值的相似连接算法已经获得广泛的研究[1,2,4-9,10–23].文献[7]首先提出了前缀过滤算法;文献[1]利用倒排索引和各种过滤技术,对 Cosine 相似度下的相似连接算法进行优化;文献[2]在前缀过滤的基础上提出了位置过滤和后缀过滤,大大降低了需要验证的候选对的数量;文献[11,24]将相似连接算法应用与 Map-Reduce 中,可以处理 P 级以上的大数据集;
- Top-k 相似连接返回记录集中最相似的前 k 个记录对,主要应用于相似度阈值未知的情况[3,10,25].文献[3]中首次使用基于事件驱动的框架来处理 Top-k 相似连接算法;文献[10]使用编辑距(edit-distance)来计算 Top-k 的相似连接;文献[25]使用 Map-Reduce 框架来支持 Top-k 相似连接算法.

6 结 语

本文对 Top-k 相似连接算法进行了研究.在 Top$k$-join 算法的基础上,提出了哈希优化策略和 Token 批处理技术,并将这两个技术与前缀事件处理框架结合,设计了新的 Top-k 相似连接算法 Opt-join.

实验结果表明:Opt-join 在哈希代价、候选对过滤能力、前缀事件堆大小、候选对数量以及运行时间等 5

个方面都优于 Top$k$-join 算法,取得了 1.28~3.09 的加速比.随着 k 值的增加或数据集中记录长度的增长,加速比有不断增大的趋势.

References:

[1] Bayardo RJ, Ma Y, Srikant R. Scaling up all pairssimilarity search. In: Proc. of the 16th Int'l Conf. on World Wide Web. 2007. 131−140. [doi: 10.1145/1242572.1242591]

[2] Xiao C, Wang W, Lin X, Yu JX. Efficient similarityjoins for near duplicate detection. In: Proc. of the 17th Int'l Conf. on World Wide Web. 2008. 131−140. [doi: 10.1145/2000824.2000825]

[3] Xiao C, Wang W, Lin X, Shang H. Top-k setsimilarityjoins. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE). 2009. 916−927. [doi: 10.1109/ICDE.2009.111]

[4] Wang J, Li G, Feng J. Can we beat the prefix filtering: An adaptive framework for similarity join and search. In: Proc. of the SIGMOD Conf. 2012. 85−96. [doi: 10.1145/2213836.2213847]

[5] Arawagi S, Kirpal A. Efficient set joins onsimilarity predicates. In: Proc. of the SIGMOD Conf. 2004. 743−754. [doi: 10.1145/1007568.1007652]

[6] Deng D, Li G, Feng J. A pivotal prefix based filteringalgorithm for string similarity search. In: Proc. of the SIGMOD Conf. 2014. 673−684. [doi: 10.1145/2588555.2593675]

[7] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning. In: Proc. of the Int'l Conf. on Data Engineering (ICDE). 2006. [doi: 10.1109/ICDE.2006.9]

[8] Li C, Wang B, Yang X. Vgram: Improving performanceof approximate queries on string collections usingvariable-length grams. In: Proc. of the VLDB. 2007. 303−314.

[9] Qin J, Wang W, Lu Y, Xiao C, Lin X. Efficient exactedit similarity query processing with the asymmetric signaturescheme. In: Proc. of the SIGMOD Conf. 2011. 1033−1044. [doi: 10.1145/1989323.1989431]

[10] Deng D, Li G, Feng J, Li WS. Top-k string similaritysearch with edit-distance constraints. In: Proc. of the Int'l Conf. on Data Engineering (ICDE). 2013. 925−936. [doi: 10.1109/ICDE.2013.6544886]

[11] Deng D, Li G, Hao S, Wang J, Feng J. Massjoin: Amapreduce-Based method for scalable string similarity joins. In: Proc. of the Int'l Conf. on Data Engineering (ICDE). 2014. 340−351. [doi: 10.1109/ICDE.2014.6816663]

[12] Kusumoto M, Maehara T, Kawarabayashi K. Scalable similarity search for SimRank. In: Proc. of the SIGMOD Conf. 2014. 325−336. [doi: 10.1145/2588555.2610526]

[13] Li G, He J, Deng D, Li J. Efficient similarity join and search on multi-attribute data. In: Proc. of the SIGMOD Conf. 2015. 1137−1151[doi: 10.1145/2723372.2723733]

[14] Li G, Deng D, Wang J, Feng J. Pass-Join: Apartition-Based method for similarity joins. PVLDB, 2011,5(3):253−264. [doi: 10.1145/2588555.2610526]

[15] Xiao C, Wang W, Lin X. Ed-Join: An efficient algorithmfor similarity joins with edit distance constraints. PVLDB, 2008,1(1): 933−944. [doi: 10.14778/1453856.1453957]

[16] Wang W, Qin J, Xiao C, Lin X, Shen HT. Vchunkjoin: An efficient algorithm for edit similarity joins. IEEE Trans. on Knowlogy Data Engineer, 2013,25(8):1916−1929. [doi: 10.1109/TKDE.2012.79]

[17] Sarawagi S, Bhamidipaty A. Interactive deduplication using activelearning. In: Proc. of the KDD. 2002. 269−278. [doi: 10.1145/775047.775087]

[18] Wang J, Li G, Feng J. Trie-Join: Efficient trie-basedstring similarity joins with edit-distance constraints. PVLDB, 2010,3(1): 1219−1230. [doi: 10.14778/1920841.1920992]

[19] Jiang Y, Li G, Feng J. String similarity joins: An experimental evaluation. In: Proc. of the VLDB. 2014. 625−636. [doi: 10.14778/2732296.2732299]

[20] Bocek BST, Hunt E. Fast similarity search in large dictionaries. Technical Report, ifi-2007.02, Department of Informatics, University of Zurich, 2007.

[21] Lam HT, Dung DV, Perego R, Silvestri F. An incremental prefix filtering approach for theall pairs similarity search problem. In: Proc. of the 12th Asia Pacific Web (APWEB). 2010. 188−194. [doi: 10.1109/APWeb.2010.30]

[22] Lin XM, Wang W. Set and string similarity queries: A survey. Chinese Journal of Computers, 2011,34(10):1853−1861 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01853]

[23] Arasu A, Ganti V, Kaushik R. Efficient exactset-similarity joins. In: Proc. of the VLDB. 2006. 918−929.

[24] Sarma AD, He Y, Chaudhuri S. Clusterjoin: Asimilarity joins framework using map-reduce. PVLDB, 2014,7(12):1059−1070. [doi: 10.14778/2732977.2732981]

[25] Kim Y, Shim K. Parallel top-*k* similarity join algorithmsusing mapreduce. In: Proc. of the Int'l Conf. on Data Engineering (ICDE). 2012. 510−521. [doi: 10.1109/ICDE.2012.87]

附中文参考文献:

[22] 林学民,王炜.集合和字符串的相似度查询.计算机学报,2011,34(10):1853−1862. [doi: 10.3724/SP.J.1016.2011.01853]

王洪亚(1976－),男,河南开封人,博士,教授,博士生导师,CCF 会员,主要研究领域为数据库系统与理论,实时计算,移动计算.

杨利宏(1988—),男,硕士,主要研究领域为数据挖掘,数据清理.

刘晓强(1968—),女,博士,教授,主要研究领域为数据管理,信息系统,语义 Web,分布式计算,软件设计与评测.