计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
47(10): 1785-1796, 2010

一种优化多重过滤的序列查询算法

戴东波　汤春蕾　邱伯仁　熊　赟　朱扬勇
(复旦大学计算机科学技术学院　上海　200433)
(daidongbo@fudan.edu.cn)

An Algorithm for Sequence Similarity Query with Optimized Multiple Filtering

Dai Dongbo, Tang Chunlei, Qiu Boren, Xiong Yun, and Zhu Yangyong
(*School of Computer Science, Fudan University, Shanghai* 200433)

Abstract　Sequence data is an important data type, ubiquitous in many domains such as text, Web access log and biological database. Similarity query in this kind of data is a very important means for extracting useful information. One key factor for high performance of similarity query in huge sequence database is the filtering level of query algorithm, namely, designing those filters that can quickly filter out the unpromising strings for query string is very important. Combining metric property of sequences' distance with sequences' characteristics, an algorithm called SSQ_MF based on multiple filters is proposed, whose filtering level is further improved compared with those query algorithms with a single filter. The filters used in SSQ_MF are length filter, prefix filter, and reference-based filter. Then, the statistical information about the string database is pre-computed and some data structures are used to store those information. Furthermore, every filter's filtering size is effectively estimated and a model in which optimal filtering order is determined by every filter's filtering size is built. Comprehensive experimental results demonstrate that in terms of query performance, SSQ_MF is better than algorithms with a single filter and algorithms with multiple filters but executing in a random order.

Key words　sequence data; similarity query; filter; filtering order; metric space

摘　要　序列数据一类重要的数据类型,在文本、Web访问日志文件、生物数据库等应用中普遍存在,对其进行相似性查询是一种获取有用信息的重要手段.在大型序列数据库中进行高效相似性查询的关键因素之一就是查询算法的过滤能力,即设计能快速过滤与查询序列不相关序列集的过滤器十分重要.提出了结合序列距离的度量性质和序列自身特征的多重过滤算法SSQ_MF,SSQ_MF使用了长度过滤器、前缀过滤器和基于参考集的过滤器,使得算法过滤能力较基于单一过滤器算法进一步增强.此外,设计了有关数据结构对查询数据库的一些统计信息进行了预计算和保存,有效估计了各过滤器的过滤集大小,并构建了一个由过滤集大小确定的最优过滤顺序模型,使得算法的过滤代价最低.实验结果表明,算法SSQ_MF的查询性能优于单一过滤器算法和随机过滤顺序的多过滤器算法.

关键词　序列数据;相似性查询;过滤器;过滤顺序;度量空间

中图法分类号　TP311.13

收稿日期:2009-05-11;**修回日期**:2009-10-12

基金项目:上海市重点学科建设基金项目(B114);上海市科委基金项目(08511500203)

0 引 言

序列数据是一类重要而特殊的数据类型,在许多应用领域广泛存在[1-4],如文本数据中的单词序列、交易数据库中的用户购买序列和生物数据库中的DNA序列和蛋白质序列等.序列数据的特殊性体现在除了其组成元素项外,还包含元素项之间的序关系,这些序关系体现为空间序(如字符序列)或时间序(如Web访问序列).本文的研究对象是字符序列,如不作特别说明,本文以后讨论的序列就是字符序列.

序列相似性查询在基因数据库中的功能预测、数据清洗中的实体副本检测、文本编辑器中的拼写错误检查和数据通信中的错码纠正等应用中是一种重要的技术手段.表征2条序列相似性的的常见度量是编辑距离.2条序列 S_1 和 S_2(长度分别为 m 和 n)的编辑距离 $ED(S_1,S_2)$ 定义为将 S_1 映射到 S_2 所需要的最少操作步骤,其中的操作包括插入、删除和替代[5],$ED(S_1,S_2)$ 的计算复杂度为 $O(m\times n)$.因此,在大型的长序列数据库中用朴素的完全搜索方法来进行相似性查询,其效率是非常低下的.解决此问题的关键技术之一就是设计有效的过滤器,使得大部分与查询序列不相似的数据能快速过滤掉,只在小部分数据集中进行代价较大的后处理来获得最后结果.同时,用过滤器进行过滤的代价要小于直接计算序列编辑距离代价.序列的一些特征如序列长度、不同 q-gram(连续的 q 长子序列)数目以及 q-gram的位置信息可以用来进行过滤,这些特征是序列本身固有的,我们称之为序列的"内在属性".由于序列间的编辑距离是一个度量,度量中的三角不等式性质(称之为序列的"外在属性")也可以用来帮助快速过滤.现有大部分算法只注重基于序列本身特征的过滤方法,没有充分利用序列的"内在属性"和"外在属性"的有机结合来进一步增强算法过滤能力.此外,在基于序列本身特征的多过滤器中,整体的过滤代价是对各过滤器的过滤顺序敏感的,但现有算法没有分析过滤顺序对过滤代价的影响,使得整体的查询算法性能没有得到优化.

针对上述问题,本文提出一种基于多重过滤的高效序列相似性查询算法SSQ_MF,其主要贡献是:1.设计了结合序列距离的度量性质和序列自身特征的多重过滤器,从序列的"内在属性"和"外在属性"两个正交角度对序列进行过滤,使得SSQ_MF过滤能力进一步增强;2.有效地估计了各过滤器的过滤集大小;3.构建了一个由过滤集大小确定的最优过滤顺序模型,使得算法的过滤代价最低.实验结果表明,相对只利用序列自身特征来进行过滤的查询算法,SSQ_MF在查询性能上有较大提高,且优于基于随机过滤顺序的多过滤器算法.

1 相关工作

常见的序列相似性查询有两类:范围查询和 k 近邻查询.k 近邻查询是在给定序列数据库中返回与查询序列最相似的前 k 条序列.由于 k 近邻查询可利用范围查询完成[6],本文只着重考虑范围查询.范围查询的目的就是在序列数据库中返回与查询序列距离小于给定半径的所有序列数据,其定义如下:给定数据库 $D=\{S_1,S_2,\cdots,S_n\}$ 和查询 $Q(S_q,r)$(S_q 为查询序列,r 为查询半径),所求查询结果为 $R=\{S_i\mid S_i\in D,\text{且 } ED(S_i,S_q)\leqslant r\}$.

由于序列间编辑距离计算的高代价和序列数据库的大规模限制,朴素的完全搜索方法变得不可行.采用过滤技术来加速算法运行是提高性能的有效途径之一.现有过滤算法分别基于序列自身特征和度量空间性质两类过滤器,并且多数算法是以序列自身特征序列的过滤方法为主.

1.基于序列自身特征的过滤算法

序列自身的特征包括序列长度、q-gram数目、q-gram位置信息和序列中各组成元素的统计信息等.当两序列 S_1 和 S_2(长度为别为 m 和 n)的编辑距离满足 $ED(S_1,S_2)\leqslant k$ 时,则两序列至少共享 $\max(m,n)+1-(k+1)q$ 个 q-gram[7],从而可用序列间共享的 q-gram下界条件快速过滤不满足范围查询要求的序列.文献[8]中使用了序列中不同 q-gram的相对序关系,在每隔一定距离对 q-gram进行抽样来获得相似序列必须共享的保持序关系的 q-gram下界条件.由于不使用序列的全部 q-gram,该方法的查询性能明显提高.文献[9]中进一步给出了当 $ED(S_1,S_2)\leqslant k$ 时的 q-gram位置过滤条件:序列 S_1 中的 q-gram不能对应 S_2 中位置相差大于 k 的某个 q-gram,且序列长度必须满足 S_1 和 S_2 的长度差不能超过 k.这2个过滤条件结合文献[7]中的过滤条件一起支持关系数据库中的近似字串连接(approximate string joins)操作.但这种方法的过滤条件是没有先后之分的,不同的过滤顺序往往会产生不同的执行效率.由于文献[7]中为了验证两序列

共享的 q-gram,必须比较两序列 q-gram 集中所有元素,这对于较长序列会降低其查询性能.因此 Chaudhuri 等人[10]设计了一种基于 q-gram 数目过滤的改进方法:前缀过滤(prefix-filtering)方法.该方法是给所有可能的 q-gram 集定义一个序,两序列 S_1 和 S_2 的 q-gram 集 $G(S_1)$和 $G(S_2)$按定义的序存放在一个数组里.当 $G(S_1)$和 $G(S_2)$的交集大于某一阈值时,$G(S_2)$中第 1 个元素的序值必须小于 $G(S_1)$中某个元素的序值,该过滤方法不必考察 $G(S_1)$和 $G(S_2)$中所有元素,使得过滤性能有所提高.

文献[11]中作者虽然把这几种基于序列自身特征的过滤条件集成在一棵过滤树(filter tree)中,结合有效的 q-gram 倒排链表合并方法来执行序列数据的近似查询,但并没有分析几种过滤条件的过滤顺序对过滤代价的影响.文献[12]中作者提出了最小编辑误差的概念,设计了基于位置错配和基于内容错配的 2 个过滤器,并证明是编辑距离的下界,使其应用在序列的相似性连接中.Li 等人[13]通过序列数据库构造变长的 q-gram 字典,然后把每条序列通过字典分离出各自的 q-gram 集,并给出了相似序列必须共享的变长 q-gram 下界来进行过滤.同样地,作者在随后的研究[14]中进一步给出了构造的变长 q-gram 字典与查询性能的关系.

此外,文献[6,15]中分别提出了基于序列各字母频率的频率距离 FD 和二分频率变换距离 TPFD,且证明了这两个距离都是其编辑距离的下界,从而可实现快速过滤查询.

2.基于序列度量空间性质的过滤算法

由于表征序列间相似性的编辑距离是一个度量,从而满足三角不等式.利用这个基本性质可以构造有效的索引结构来对序列的相似性查询进行过滤.VP-Tree[16](及其变型 MVP-Tree)与 M-Tree[17]是通过把数据空间划分为层次的球性区域而建立的平衡查询树,是一般度量空间的重要索引结构.Sahinalp 等人在 2003 年指出编辑距离的 2 种变型[18]:压缩距离和加权的编辑距离都是近似度量(almost metric)(即满足自反、对称和常数因子范围内的近似三角不等式),并修改 VP/MVP-Trees 中过滤规则来适应在 2 种近似度量空间的序列相似性查询.文献[19]中实现的查询算法是基于参考集索引的,其基本思想是为每个数据库中元素分配一个参考集.在查询 $Q(S_q,r)$提交时,利用编辑距离的三角不等式性质,由数据库中每个元素 S_i 的参考集可计算出 $ED(S_q,S_i)$的上界 UB 和下界 LB,由 UB 和 LB 来获得部分结果集和过滤集.同样的作者在文献[20]中通过对用水库抽样(reservoir sampling)的方法来维护查询的分布,使得生成的索引能随数据库的动态更新而更新.

上述基于序列自身特征和度量空间性质的 2 类过滤算法分别利用了序列的"内在属性"和"外在属性",但没有把两者进行有机的统一形成多过滤器来增强算法的过滤能力.本文的多重过滤器算法SSQ_MF 有效地结合了体现序列"内在属性"和"外在属性"的 2 类过滤器.

2 最优过滤顺序模型和过滤集大小估计

在采用多过滤器的序列相似性查询算法中,由于各过滤器的过滤集大小不同,使得采用不同的过滤顺序对过滤代价有较大影响,因此寻求低过滤代价的过滤顺序对提高算法的查询性能有很重要的意义.此外,过滤顺序的确定依赖于各过滤集大小的有效在线估计.2.1 节引入一个最优的过滤顺序模型,2.2 节给出了 SSQ_MF 算法中各过滤器的过滤集大小的在线估计.

2.1 最优过滤顺序模型

给定一个序列的相似性查询算法 A,设 A 中有 n 个过滤器 $F_1,F_2,\cdots,F_n$,在查询数据库 D 中的各过滤集依次为 $FS_1,FS_2,\cdots,FS_n$.在过滤器实行过滤时,要对每条序列验证是否能被过滤掉.每个过滤器的过滤代价由验证的序列数目与对每条序列验证的时间复杂度决定.过滤器的作用就是要避免代价较大的编辑距离计算,由于在 SSQ_MF 算法中引入的各过滤器对每条序列的验证代价都为 $O(L)$,L 为序列平均长度,我们近似假设每条序列的验证代价具体为 τ.首先,我们注意到各过滤器的过滤顺序对过滤集大小总和没有影响,即有下面定理 1.

定理 1. 算法 A 中 n 个过滤器 $F_1,F_2,\cdots,F_n$ 总的过滤集与 $F_1,F_2,\cdots,F_n$ 的过滤顺序无关.

证明. 因为对于任意过滤顺序的 $F_1,F_2,\cdots,F_n$,总的过滤集为 $FS_1\cup FS_2\cup\cdots\cup FS_n$. 证毕.

算法 A 的时间代价 C 可由下式确定:$C=VS\times\tau+|(D-FS_1\cup FS_2\cup\cdots\cup FS_n)|\times C'$,其中 VS 是过滤阶段验证的序列数目,C'是对每条序列的平均后处理时间代价.后处理方法一般是直接的编辑距离计算去除假阳性数据,我们可近似地认为 C'是 1 个常数.因此,决定 C 值的 2 个可变值是 VS 和 $FS_1\cup FS_2\cup\cdots\cup FS_n$,而 $FS_1\cup FS_2\cup\cdots\cup FS_n$ 由过滤

器的综合过滤水平决定,本文通过结合基于序列自身特征和度量空间性质的多重过滤器来提高 $FS_1 \cup FS_2 \cup \cdots \cup FS_n$ 值.在 n 个过滤器给定的情况下,由定理 1 知,其过滤顺序不会改变后处理集 $D-FS_1 \cup FS_2 \cup \cdots \cup FS_n$,这时算法 A 的时间代价 C 取决于过滤阶段验证的序列数目 VS,我们可以通过优化过滤顺序来降低 VS 值.

图 1 给出了一个不同过滤顺序影响过滤阶段验证的序列数目 VS 值的例子.设 3 个过滤器 F_1,F_2,F_3 的过滤集互不相交,且大小满足 $|FS_1|>|FS_2|>|FS_3|$.图 1(a)(b)中给出了两种不同的过滤顺序:F_1,F_2,F_3 和 F_2,F_3,F_1.从图 1 中可以看出,两种不同过滤顺序产生的过滤总集是一样的.但图 1(a)中的验证序列数目为 $VS=|D|+|D-FS_1|+|D-FS_1-FS_2|=3|D|-2|FS_1|-|FS_2|$,图 1(b)中的验证序列数目为 $VS'=|D|+|D-FS_2|+|D-FS_2-FS_3|=3|D|-2|FS_2|-|FS_3|$,从而有 $VS'-VS=2(|FS_1|-|FS_2|)+(|FS_2|-|FS_3|)>0$,即图 1(b)中的过滤代价大于图 1(a)中的过滤代价.直观上讲,在使用过滤器过程中,每次优先使用过滤集大的过滤器会使得过滤代价减少.下面定理 2 给出了在各过滤集互不相交的理想情况下的最优过滤顺序.

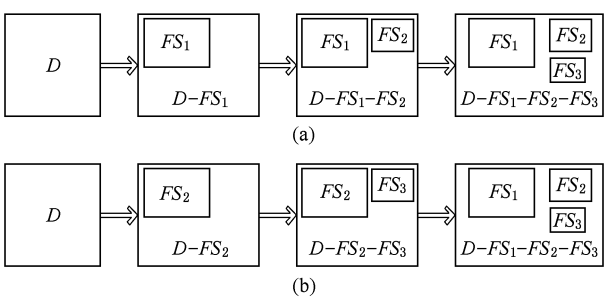


Fig. 1 Impact of different filtering order on the size of verification set. (a) Filtering order: F_1,F_2,F_3 and (b) Filtering order: F_2,F_3,F_1.

图 1 不同过滤顺序对验证序列数目的影响.(a) 过滤顺序:F_1,F_2,F_3;(b) 过滤顺序:F_2,F_3,F_1

定理 2. 设查询算法 A 在查询序列数据库 D 中 n 个过滤器 $F_1,F_2,\cdots,F_n$ 的过滤集 $FS_1,FS_2,\cdots,FS_n$ 满足:对于任意的 i 和 j,$FS_i \cap FS_j=\varnothing(1\leqslant i\neq j\leqslant n)$,且 $|FS_1|\geqslant|FS_2|\geqslant\cdots\geqslant|FS_n|$,则验证序列数目最少的过滤顺序是:$F_1,F_2,\cdots,F_n$.

证明. 略.

定理 2 得到的最优过滤顺序是在各过滤集互不相交的理想情况下得到的.在基于过滤方法的现实查询算法中,各过滤器的过滤集一般是有重叠的,这时有可能导致定理 2 中的过滤顺序不是最优的,但从直观上看,前 k 个过滤器总的过滤集 FS 越大,则第 $k+1$ 个过滤器的过滤集 FS' 与 FS 相交的可能性越大,即第 $k+1$ 个过滤器产生的新增过滤集 NFS 的可能性越小.因此,我们可假设 NFS 的大小为:$|NFS|=\dfrac{|D|-|FS|}{|D|}\times|FS'|$.在此假设条件下,下面的定理 3 证明了定理 2 中的最优过滤顺序仍不变.

定理 3. 设查询算法 A 在查询数据库 D 中 n 个过滤器 $F_1,F_2,\cdots,F_n$ 的过滤集为 $FS_1,FS_2,\cdots,FS_n$,且满足:$|FS_1|\geqslant|FS_2|\geqslant\cdots\geqslant|FS_n|$.设其中任意前 $k(1\leqslant k\leqslant n-1)$ 个过滤器 $F_{i_1},F_{i_2},\cdots,F_{i_k}$ 总的过滤集为 FS,若 $F_{i_{k+1}}$ 在 $F_{i_1},F_{i_2},\cdots,F_{i_k}$ 过滤后产生的新增过滤集 NFS 满足:$|NFS|=\dfrac{|D|-|NS|}{|D|}\times|FS_{i_{k+1}}|$,则 A 中验证序列数目最少的过滤顺序 τ 为 $F_1,F_2,\cdots,F_n$.

证明. 设 n 个过滤器 $F_1,F_2,\cdots,F_n$ 的任意一个过滤顺序 $F_{i_1},F_{i_2},\cdots,F_{i_n}$ 为 σ,其中 $i_1,i_2,\cdots,i_n$ 是集合 $\{1,2,\cdots,n\}$ 的任意一个全排列.若 σ 不是 $F_1,F_2,\cdots,F_n$,则总存在 $j(1\leqslant j\leqslant n-1)$,使得 $|FS_{i_j}|\leqslant|FS_{i_{j+1}}|$.我们把过滤顺序 $F_{i_j},F_{i_{j+1}}$ 调换为 $F_{i_{j+1}},F_{i_j}$ 称为一次有效移动.设 σ 中验证序列数目总数为 VS_σ,σ 中前 $j-1$ 个过滤器 $F_{i_1},F_{i_2},\cdots,F_{i_{j-1}}$ 总的过滤集为 FS.

在过滤顺序 σ 中,进行 $F_{i_j},F_{i_{j+1}}$ 两次过滤所验证的序列数目为 $VS=(|D|-|FS|)+\left(|D|-|FS|-\dfrac{|D|-|FS|}{|D|}\times|FS_{i_j}|\right)$.当 σ 中的 $F_{i_j},F_{i_{j+1}}$ 过滤器经过一次有效移动后,设过滤顺序 σ 变为 σ'.则在过滤顺序 σ' 中,进行 $F_{i_{j+1}},F_{i_j}$ 两次过滤所验证的序列数目为 $VS'=(|D|-|FS|)+\left(|D|-|FS|-\dfrac{|D|-|FS|}{|D|}\times|FS_{i_{j+1}}|\right)$.由 $|FS_{i_{j+1}}|\geqslant|FS_{i_j}|$,易知 $VS'\leqslant VS$.由定理 1 知,σ 中过滤器 $F_{i_1},\cdots,F_{i_j},F_{i_{j+1}}$ 和 σ' 中过滤器 $F_{i_1},\cdots,F_{i_{j+1}},F_{i_j}$ 所产生的过滤集是相同的,则 σ 中和 σ' 中过滤器 $F_{i_{j+2}},\cdots,F_{i_n}$ 验证的序列数目也是相同的.所以,$VS_\sigma\geqslant VS_{\sigma'}$.若 σ' 还不是 $F_1,F_2,\cdots,F_n$,则 σ' 中总存在两过滤器可进行有效移动变为 σ'',使得 $VS_{\sigma'}\geqslant VS_{\sigma''}$.一直重复上述过程,直到最后过滤顺序变为 τ:$F_1,F_2,\cdots,F_n$,此时的 VS_τ 是所有过滤顺序中验证

序列数目最少的,即过滤顺序 $F_1,F_2,\cdots,F_n$ 是最优的. 证毕.

我们在算法 SSQ_MF 中通过有效地估计各过滤器的过滤集大小,应用定理 3 中给出的过滤顺序对序列数据库进行实际过滤,实验结果表明此过滤顺序的过滤代价在大多数情况下都低于其他的随机过滤顺序,这说明定理 3 中对一个过滤器新增过滤集大小所作的假设符合一定的实际意义.

2.2 各过滤集大小的在线估计

2.2.1 各过滤器的基本过滤原理

在讨论各过滤器基本原理之前,我们给出几个基本概念:

1. 字符序列及其位置 q-gram. 一条字符序列 S 是字母表 $\Sigma=(a_1,a_2,\cdots,a_T)$ 上的有序字符串,记作 $S=s_1s_2\cdots s_n$,$s_k\in\Sigma(1\leqslant k\leqslant n)$ 称为 S 的元素,其中 $T=|\Sigma|$ 称为字母表的基数,$n=|S|$ 称作 S 的长度. 对于任意的 $1\leqslant i\leqslant j\leqslant n$,$S[i,j]=(s_i,s_{i+1},\cdots,s_j)$ 称为序列 S 的子串,也称为 q-gram,其中 $q=j-i+1$. 序列 S 所有的 $n-q+1$ 个 q-gram 集 $G(S)$ 可以通过在序列 S 上每次移动一个大小为 q 的窗口来获得. 用 (p,G) 表示 q-gram G 起始于序列 S 中的第 p 个位置. 序列 S 的所有位置 q-gram 元素的集合记作 $PG(S)$.

2. 位置 q-gram 的匹配关系. 若 $ED(S_1,S_2)\leqslant r$,S_1 中的位置 q-gram (p,G) 与 S_2 中的某个位置 q-gram (p',G') 满足条件 $|p-p'|\leqslant r$ 和 $G=G'$,我们称 (p,G) 和 (p',G) 是匹配的位置 q-gram.

3. 位置 q-gram 集合的序映射. 给定一个定义在所有 q-gram 集上的序 O,序列 S 的位置 q-gram 集合 $PG(S)$ 中元素的 q-gram 值按 O 排列后称之为 $PG(S)$ 的序 O 映射,记作 $O(PG(S))$. 如:$q=2$,字母表 $\Sigma=\{a,b\}$,设定序 O 为:$aa\rightarrow ab\rightarrow bb\rightarrow ba$,其中 $x\rightarrow y$ 表示在序 O 中,a 位于 b 之前. 序列 $S=bab$ 的 $PG(S)$ 为 $\{(ba,1),(ab,2)\}$,则 $O(PG(S))$ 为 $\{(ab,2),(ba,1)\}$.

我们在算法 SSQ_MF 引入 3 个过滤器:基于序列自身特征的长度过滤器 LenFilter、前缀-位置 q-gram 过滤器 Pre_PosFilter 以及基于序列度量空间性质的参考集索引过滤器 RefFilter. 给定数据库 $D=\{S_1,S_2,\cdots S_n\}$,查询半径 r 以及查询序列 S_q,各过滤器的过滤原理如下.

长度过滤器 LenFilter 的过滤条件由性质 1 给出.

性质 1[9]. 给定两序列 S_1 和 S_2,若其编辑距离 $ED(S_1,S_2)\leqslant r$,则 $||S_1|-|S_2||\leqslant r$.

长度过滤器 LenFilter 刻画了不同序列在组成元素数量上的差异. 当查询序列与 D 中各序列长度相差较大时,LenFilter 的过滤水平较为突出.

序列中 q-gram 的数目和位置信息都可看成是序列的特征"信号". 文献[12]中作者提出了前缀-位置过滤方法,基本过滤原理如下:

性质 2[12]. 给定序列 S_1 和 S_2 的位置 q-gram 集合 $PG(S_1)$ 和 $PG(S_2)$,若 $ED(S_1,S_2)\leqslant r$,则对于任意的序 O,$O(PG(S_1))$ 和 $O(PG(S_2))$ 各自的前 $r\times q+1$ 个元素中至少有一个匹配的位置 q-gram.

过滤器 LenFilter 和 Pre_PosFilter 是基于序列自身特征的,而参考集索引过滤器 RefFilter 主要利用了序列间的编辑距离是一个度量,即编辑距离满足三角不等式. RefFilter 基本过滤原理如下:在序列数据库 D 找到一个大小为 k 的参考集 R,然后预计算数据库中各元素 S_i 与 R 中所有元素的编辑距离. 在查询 $Q(S_q,r)$ 提交时,由 D 中每个元素 S_i 的参考集可计算出 $ED(S_q,S_i)$ 的下界 LB,即:$LB=\max(\bigcup_{v_j\in R}|ED(S_q,V_j)-ED(V_j,S_i)|)$,若 $r<LB$,S_i 被过滤掉,否则把 S_i 加入候选集作后处理.

在估计各过滤集大小时,我们使用文献[21]中提到的下面假设:

假设 1. 任何序列中每个字符都是由字母表中元素均匀独立地产生.

根据这个假设,我们易知,一条随机序列的产生与另一条的产生是相互独立的,且一个 q-gram 在序列的各位置出现也可近似看成是相互独立的.

2.2.2 过滤集大小的在线估计

2.1 节定理 3 中给出的最优过滤顺序模型要求在进行实际过滤操作之前,必须获得各过滤器的过滤集大小,SSQ_MF 算法以离线的方式,预先对序列数据库 D 维护必要的统计信息. 然后,对于不同的在线查询(其查询序列和查询半径各不相同),SSQ_MF 算法利用统计信息在线估计 3 个过滤器的过滤集大小,并按定理 3 中的最优过滤顺序进行实际过滤. 给定查询 $Q(S_q,r)$,下面给出各过滤器在 D 中过滤集大小的在线估计方法:

1. 长度过滤器 LenFilter 的过滤集大小估计.

当查询 $Q(S_q,r)$ 到达时,由性质 1 知,当 D 中任意序列 S 的长度满足 $|S|>|S_q|+r$ 或 $|S|<|S_q|-r$ 时,则 S 是在过滤器 LenFilter 的过滤集中. 为此,我们对 D 中不同长度值的序列数进行统计,并按长

度值升序存放各序列长度值的序列数,然后求出不超过某一长度值的所有序列数. 我们用链表 *Len_list* 的每一个结点升序存放长度值以及不超过此长度值的所有序列数. 利用 *Len_list* 链表可精确地求出 D 中所有长度不在 $|S_q|-r$ 和 $|S_q|+r$ 之间的序列数目.

2. 前缀-位置 q-gram 过滤器 Pre_PosFilter 的过滤集大小估计.

为了应用过滤器 Pre_PosFilter 对 D 进行过滤,我们对 D 进行离线预处理. 由性质 2 知,我们可分 3 步来求得对各查询 $Q(S_q,r)$ 过滤有用的统计信息:

1) 对 D 中各序列 S_i 的位置 q-gram 集合 $PG(S_i)$ 进行序 O 映射,求出 $O(PG(S_i))$ 中 $r\times q+1$ 个前缀集合,记作 $Pre_S_i(r\times q+1)$;

2) 对 D 中各序列 S_i 的 $Pre_S_i(r\times q+1)$ 中不同位置 q-gram (p,G),统计 G 在对应位置 p 下的序列数目;

3) 当 r 值不同时,D 中各序列 S_i 的 $Pre_S_i(r\times q+1)$ 也不同,重复步骤 2),增量求出不同 $Pre_S_i(r\times q+1)$ 中每个位置 q-gram 的统计信息.

对于步骤 1),给定一个基数为 T 的字母表 $\Sigma=(a_1,a_2,\cdots,a_T)$,我们先定义一个在 Σ 上的简单函数 $f: f(a_i)=i-1, 1\leqslant i\leqslant T, a_i\in\Sigma$. 对 Σ 上的任意 q-gram $G=a_{i_1}\cdots a_{i_q}, a_{i_k}\in\Sigma(1\leqslant k\leqslant q)$,给 G 赋整数值 $\sum_{k=1}^{q} f(a_{i_k})\times T^{q-k}$,且取序 O 就为整数序. 然后通过一遍扫描数据库 D,获得 D 中所有序列的位置 q-gram 集. 对任意序列 S_i 的 $PG(S_i)$ 进行序 O 映射,这样可以在 $O(PG(S_i))$ 中取出任意长度的前缀.

步骤 2)是对 D 进行预处理的关键一步. 因为要对 $Pre_S_i(r\times q+1)$ 中不同元素产生统计信息,我们根据 $Pre_S_i(r\times q+1)$ 中每个元素 (p,G) 的 G 值,形成 G 关于位置升序的倒排链表 *InvetList*(G),*InvertList*(G) 中每个结点的结构是 (SID,pos),其中 *SID* 记录了所有包含 G 的序列 ID 号,*pos* 是 G 在此序列中的起始位置信息且升序排列. 图 2 给出了某 G 的 *InvetList*(G) 例子. 为了获得对各查询 $Q(S_q,r)$ 过滤有用的统计信息,我们从 *InvetList*(G) 生成另外一个含有统计信息的链表 *PrePos_list*(G),其中每个结点的结构是:$[pos,(r_val,sum,rep)]$,其中 (r_val,sum,rep) 是数组形式,因为 *r_val* 值表明此结点中的统计信息是由各序列在序 O 映射的 $r\times q+1$ 前缀得到的,不同的 $r\times q+1$ 前缀得到的统计信息是不一样的. *sum* 是对 *InvetList*(G) 统计 G 在位置 *pos* 的序列总数(各结点中的 *pos* 值仍是递增顺序的),利用 *InvertList*(G) 中的序列 *SID* 信息,*rep* 指针指向离此结点最近的前向结点 *node*,且此结构的 *sum* 条序列中,存在一条序列与 *node* 中的某条序列重复(若与多个前向结点有重复序列,可设置多个 *rep* 指针),*rep* 指针主要用来避免重复序列的计算, *InvetList*(G) 经统计后,变成了如图 2 所示的 *PrePos_list*(G).

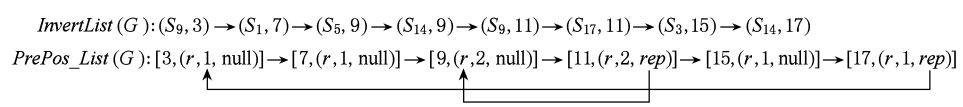


Fig. 2 *InvertList*(G) and *PrePos_list*(G) of q-gram G.

图 2 q-gram G 的倒排链表 *InvertList*(G) 和统计信息链表 *PrePos_list*(G)

通过步骤 2)得到的统计信息链表是只根据某特定查询 $Q(S_q,r)$ 中 r 值建立的. 因为不同查询有不同的查询半径 r,从而 D 中每条序列 S_i 的 $Pre_S_i(r\times q+1)$ 也有所变化,如何建立关于所有变化后的 $Pre_S_i(r\times q+1)$ 的统计信息链表呢? 朴素的方法就是由不同的 r 值,按步骤 2)中方法再为所有序列的 $Pre_S_i(r\times q+1)$ 中的所有位置 q-gram 建立新的统计信息链表. 但这种方法需为每个位置 q-gram 维护关于不同 r 值的统计信息链表,会占用大量存储空间. 因为当 $r'>r$ 时,序列 S_i 的 $Pre_S_i(r\times q+1)$ 中所有元素也是 $Pre_S_i(r'\times q+1)$ 中的元素,所以我们在步骤 3)中增量建立所有位置 q-gram 关于不同 r 值的的统计信息链表.

假设某 q-gram G 关于 r 的 *InvertList*(G) 和 *PrePos_list*(G) 已经建立好,我们增量建立 G 关于 $r+1$ 的 *PrePos_list*(G). 首先,对于 D 中所有序列 S_i 的 $Pre_S_i(r\times q+1)$ 到 $Pre_S_i((r+1)\times q+1)$ 的新增位置 q-gram 集中,找出所有位置 q-gram(p,G),按 p 值升序的方式将其插入到 *InvertList*(G) 中. 若 p 值在 *PrePos_list*(G) 中的某个结点 *node* 的 *pos* 域出现过,且在 *node* 中的数组 (r_val,sum,rep) 中存在 *r_val* 值为 $r+1$ 的元素,将此元素中 *sum* 值增 1,若在 *node* 中的数组 (r_val,sum,rep) 中不存在 *r_val* 值为 $r+1$ 的元素,则新增一数组元素

$(r+1, sum_L+1, rep)$,其中 sum_L 是数组最后一个元素中的 sum 值;若 p 值不在 $PrePos_list(G)$ 中任何结点的 pos 域出现,则新增一结点[p,$(r+1,1,rep)$],按 p 值升序方式将其插入 $PrePos_list(G)$ 中.利用 $InvertList(G)$,新增数组元素中的 rep 域设置方法与步骤 2)相同.如在图 2 所示例子中,若 S_1 中的 $Pre_S_1(r\times q+1)$ 到 $Pre_S_1((r+1)\times q+1)$ 的新增位置 q-gram 集中有$(12,G)$,S_9 中的 $Pre_S_9(r\times q+1)$ 到 $Pre_S_9((r+1)\times q+1)$ 的新增位置 q-gram 集中有$(15,G)$,则加入这两个位置 q-gram 后,$InvertList(G)$ 和 $PrePos_list(G)$ 的更新如图 3 所示:

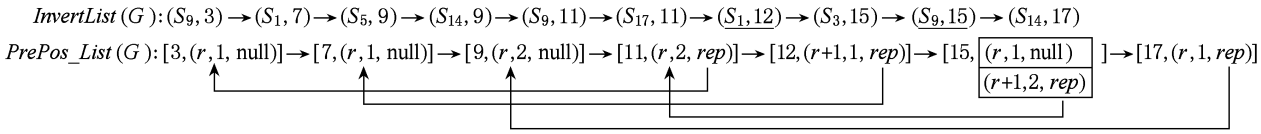


Fig. 3 Updated $InvetList(G)$ and $PrePos_list(G)$ after position q-gram $(12,G)$ and $(15,G)$ being inserted.

图 3 位置 q-gram$(12,G)$和$(15,G)$插入后所更新的 $InvetList(G)$和 $PrePos_list(G)$

当对查询 $Q(S_q,r)$的 $PG(S_q)$进行序 O 映射后,若 $Pre_S_q(r\times q+1)$中存在某位置 q-gram (p,G),则可通过 $PrePos_list(G)$求出各序列 S_i 的 $Pre_S_i(r\times q+1)$中与(p,G)至少有一个匹配的位置 q-gram 的序列总数 $MS^r_{(p,G)}$.假设图 3 中的 $PrePos_list(G)$就是经过处理所有序列后得到的 G 关于 r 和$r+1$ 的完整统计信息链表,且设 $r=4$ 和 $p=13$.则与(p,G)匹配的位置 q-gram 的位置值应在 9~17 之间.由图 3 中 $PrePos_list(G)$的位置递增顺序和 r_val 域知,有 4 个结点满足条件:[9,$(r,2,$null$)$],[11,$(r,2,rep)$],[15,$(r,1,$null$)$],[17,$(r,1,rep)$].由于[11,$(r,2,rep)$]结点中的 rep 指向[3,$(r,1,$null$)$],而[3,$(r,1,$null$)$]不在满足条件的 4 个结点中,所以[11,$(r,2,rep)$]中的 2 条序列与 4 个结点的前向结点中的序列没有重复.但[17,$(r,1,rep)$]指向的结点[9,$(r,2,$null$)$],[9,$(r,2,$null$)$]落在满足条件的 4 个结点中,所以[17,$(r,1,rep)$]中的序列与[9,$(r,2,$null$)$]中的某 1 条序列重复.所以 $MS^r_{(p,G)}=2+2+1+1-1=5$.

当另一查询 $Q(S'_q,r+1)$到来时,若 $Pre_S'_q((r+1)\times q+1)$中存在某位置 q-gram (p,G),则仍可通过 $PrePos_list(G)$求出所有序列的 $Pre_S_i((r+1)\times q+1)$中与(p,G)至少有一个匹配的位置 q-gram 的序列总数 $MS^{r+1}_{(p,G)}$.求解方法与上述方法基本一样,只是在找符合位置要求的结点时,结点中 r_val 域值应满足:不超过 $r+1$ 且是最大的.仍以图 3 中的 $PrePos_list(G)$以及 $r=4$ 和 $p=13$ 为例,则满足要求的结点为:[9,$(r,2,$null$)$],[11,$(r,2,rep)$],[12,$(r+1,1,rep)$],[15,$(r+1,2,rep)$],[17,$(r,1,rep)$].由上述方法可求得 $MS^{r+1}_{(p,G)}=6$.

由于查询序列 S_q 的各位置 q-gram 的 $MS^r_{(p,G)}$ 条序列中有重复序列,因此无法精确得到 $Q(S_q,r)$的过滤集大小 $FSPre_Pos$,但我们可根据假设 1 来近似估计 $FSPre_Pos$.我们用$\frac{MS^r_{(p,G)}}{|D|}$来近似任意序列 S_i 的 $Pre_S_i(r\times q+1)$中与 S_q 的(p,G)至少有一个匹配的位置 q-gram 的概率,由假设 1 知,任何一个位置 q-gram 和另一个位置 q-gram 在序列中的出现都是相互独立的,因此查询 $Q(S_q,r)$的 $Pre_S_q(r\times q+1)$中与查询数据库 D 中各序列的 $Pre_S_i(r\times q+1)$中没有任何匹配的位置 q-gram 的概率为

$$\prod_{G\in Pre_S_q(r\times q+1)}\left(1-\frac{MS^r_{(p,G)}}{|D|}\right),$$

此概率也是任意序列 S_i 被过滤掉的概率,从而过滤集大小可近似表示为

$$FSPre_Pos\approx|D|\times\prod_{G\in Pre_S_q(r\times q+1)}\left(1-\frac{MS^r(p,G)}{|D|}\right).$$

3. 参考集索引过滤器 RefFilter 的过滤集大小估计.

给定查询 $Q(S_q,r)$,由过滤器 RefFilter 的过滤原理知:对于序列数据库 D 中的每条序列 S_i 和参考集中 R 的每个参考序列 V_j(参考集大小为 k),S_i 能够通过 V_j 被 $Q(S_q,r)$过滤掉的条件是:$|ED(S_q,V_j)-ED(V_j,S_i)|>r$,即 $ED(S_q,V_j)>ED(V_j,S_i)+r$ 或 $ED(S_q,V_j)<ED(V_j,S_i)-r$.因为各$ED(V_j,S_i)(1\leqslant j\leqslant k,1\leqslant i\leqslant|D|)$已经预计算好,所以我们可以按照 LenFilter 统计不同长度序列数的类似方法,把每个参考序列 V_j 的所有 $ED(V_j,S_i)$值$(1\leqslant j\leqslant k)$按升序统计不超过某一值的所有序列数.借助这些统计信息,$Q(S_q,r)$可获得通过 V_j 能够过滤掉的序列数 $FS^q_{V_j}$.我们用 $FS^q_{V_j}/|D|$来近似 S_i 通过 V_j

被过滤掉的概率,由假设 1 易知,S_i 通过各 V_j 进行过滤的事件是独立的,则 S_i 通过 R 中所有参考序列都不能被过滤的概率是:$\prod_{j=1}^{k}(1-FS^{Q}_{V_j}/|D|)$,从而 RefFilter 过滤集大小 $FSRef$ 可被近似估计为:$FSRef \approx |D| \times \left(1-\prod_{j=1}^{k}(1-FS^{Q}_{V_j}/|D|)\right)$. 类似文献[20]中提到的方法,我们对 D 进行随机采样,在样本中提取前 k 条最大距离方差的序列作为 R,并将 R 中序列与 D 中所有序列的编辑距离的统计信息存储在链表 Ref_list 中.

3 SSQ_MF 算法实现策略

当在线查询 $Q(S_q,r)$ 到达时,基于多重过滤的序列相似性查询算法 SSQ_MF 先采用 2.2 节提出的方法,估计 3 个过滤器 LenFilter,Pre_PosFilter 和 RefFilter 各自的过滤集大小 $FSLen$,$FSPre_Pos$ 和 $FSRef$,然后根据过滤集大小,采用 2.1 节定理 3 中的过滤顺序进行实际的过滤操作. 为了加速实际的过滤操作,可离线对序列数据库 D 进行预处理. 我们为 D 中每条序列 S 设计一种便于 3 个过滤器进行过滤的数据结构 $TriF_DS$:(SID, Len, $q\text{-}gramsByOrder$, $RefDis$),其中 SID 是序列 S 的 ID 号,Len 是 S 的长度,$q\text{-}gramsByOrder$ 是 S 在序 O 下的 $O(PG(S))$,$RefDis$ 是序列 S 与参考集 R 中 k 条序列间的编辑距离. 对于预处理中的最大可能查询半径 r_{max},我们在实验中取数据集 D 中最长序列长度的 10%.

图 4 给出了为估计 3 个过滤器过滤集大小进行的预处理和对 D 中每条序列的预处理算法 Preprocess_MF,此算法主要是产生为估计各过滤集大小和为实际过滤准备的各种统计信息. 图 5 给出了为在线查询 $Q(S_q,r)$ 返回查询结果的算法 SSQ_MF. 其中函数 $FOder_Gen$ 返回按过滤集大小降序排列的过滤顺序,$PartialED$ 是后处理函数,从中找出满足要求的结果序列集. 文献[6,20]中的后处理都是直接计算查询序列和后处理集 $D-FSet$ 中序列的编辑距离,这样会影响后处理的效率. 我们采用文献[22]中最优比对表的对角区域计算方法 $PartialED$ 来提高计算效率,$PartialED$ 方法判断 $D-FSet$ 中序列 S 是否是查询结果的时间复杂度为 $O(r\times\min(|S_q|,|S|))$,这在 r 较小时效率提高较为明显.

```
Preprocess_MF:
Input: Sequence database D={S1,S2,…,Sn}, length of q-gram q,
       maximum possible query radius rmax, the size of reference set
       k.
Output: Link-list Len_list for LenFilter, statistical list PrePos_list
        for Pre_PosFilter, reference set R and Ref_list for
        RefFilter, TriF_DS for every sequence in D.
SN←0; /* SN is a variable for SID */
Len_link←Preprocess_Len(D); /* pre-processing for LenFilter */
PrePos_list←Preprocess_PrePos(D,q,rmax); /* pre-processing
for Pre_PosFilter */
Ref_list←Preprocess_Ref(D,k); /* pre-processing for RefFilter */
Initialize TriF_DS;
For(every sequence S in D)
    SN++; TriF_DS.SID←SN; TriF_DS.Len←length of S;
    TriF_DS.Q-gramsByOrder←O(PG(S));
    TriF_DS.RefDis←edit distances between S and every sequence
    in R
```

Fig. 4 Algorithm Preprocess_MF.

图 4 算法 Preprocess_MF

```
SSQ_MF:
Input: Sequence database D={S1,S2,…,Sn}, query Q(Sq,r);
Output: query result of Q(Sq,r) QRes.
Initialize FSet; /* FSet stores the filtered sequences */
/* GetFsize_Len, GetFsize_PrePos, GetFsize_Ref are functions
which return the estimation of filtering size for LenFilter,
PrePosFilter and RefFilter respectively */
FSLen←GetFsize_Len(Sq,r);
FSPre_Pos←GetFsize_PrePos(Sq,r);
FSRef←GetFsize_Ref(Sq,r);
  FO←FOder_Gen(FSLen,FSPre_Pos,FSRef);
For(every string S in D)
  if(S can be first filtered by LenFilter, Pre_PosFilter or RefFilter
    in the order of FO using TriF_DS)
      FSet←S;
QRes←PartialED(Sq,r,D-FSet);
```

Fig. 5 Algorithm SSQ_MF.

图 5 算法 SSQ_MF

4 实验结果及其分析

所有实验在 Windows XP 系统和内存为 2 GB 的机器上进行,算法用 Java 语言实现. 实验数据集采用两个不同应用领域的真实数据集:一个是 DBLP 网站上(http://www.informatik.uni-trier.de/~ley/db)的快照数据,共有大约 90 万条记录,每条记录主要包含了作者名字和发表论文题目的字串数据,我们从中取出 10 万条记录作为实验数据集,这个数据集也在文献[12-13]中使用过;另一个数据集 UNIREF 是从 UniProt 项目中下载(http://beta.uniprot.org/)的蛋白质数据集 UniRef90,每条蛋白质序列是从大小为 20 的字母表中取值的字符序列.

我们也从中抽取 10 万条序列作为我们的实验数据集.

算法 SSQ_MF 需要进行预处理建立一些含有统计信息的数据结构,但对于一个数据集,预处理只进行一次,其时间开销可被后面不断到达的在线查询所平摊后而忽略不计.因此,下面各实验的时间性能都是基于预处理完后而进行计算的.

DBLP 和 UNIREF 的查询数据集分别在作为和不作为实验数据集中随机选取.除非特别说明,各查询的查询半径采取与文献[6]中类似的方法,在查询序列长度的 1%~10%之间取值.实验结果都是由查询集上所获得的性能指标值取平均得到.

4.1 整体过滤水平

文献[9,11]分别在序列数据库中求近似交和进行相似性查找,但都是只基于序列自身特征如长度、q-gram 数量和位置而进行过滤的.我们把只利用过滤器 LenFilter 和 Pre_PosFilter 的 SSQ_MF 算法称为 SSQ_IntFt.图 6 和图 7 分别给出了在数据集 DBLP 上,SSQ_MF 和 SSQ_IntFt 过滤的序列数和整体时间开销对比情况.其中参数 q 设为 4.从图中可以看出,随着查询半径 r 不断增大时,两算法的过滤序列数都在不断下降,但 SSQ_MF 的过滤序列数和运行时间要优于 SSQ_IntFt,这说明加入基于序列度量空间性质的过滤器 RefFilter,能改善算法的性能.

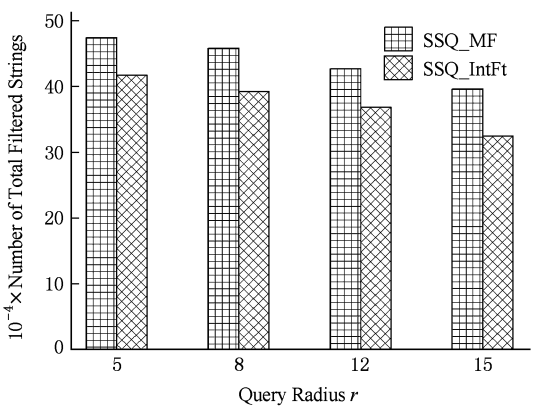


Fig. 6 Comparison of number of total filtered sequences.

图 6 SSQ_MF 和 SSQ_IntFt 过滤的序列数比较(DBLP 数据集)

文献[19]是在序列数据库中进行相似性查询,但只基于序列度量空间性质进行过滤.我们把只利用序列度量空间性质过滤器的 SSQ_MF 算法称之为 SSQ_ExtFt.图 8,9 分别给出了在数据集 UNIREF 上,SSQ_MF 和 SSQ_ExtFt 过滤的序列数和整体

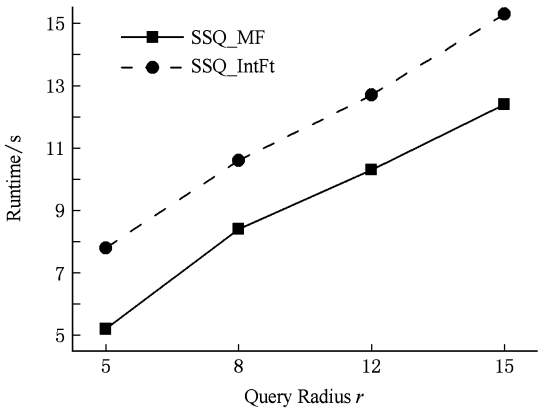


Fig. 7 Comparison of runtime between between SSQ_MF and SSQ_IntFt (DBLP dataset) SSQ_MF and SSQ_IntFt (DBLP dataset).

图 7 SSQ_MF 和 SSQ_IntFt 运行时间比较(DBLP 数据集)

时间开销对比情况,其中参数 q 设为 6.从图中可以看出,在查询半径 r 取不同值时,SSQ_MF 的过滤序列数和运行时间明显要优于 SSQ_ExtFt,这说明加入基于序列自身特征的过滤器 LenFilter 和 Pre_PosFilter,也能明显改善算法的性能.从 SSQ_MF 与 SSQ_IntFt,SSQ_ExtFt 的算法性能比较可以看出,基于序列自身特征的过滤器 LenFilter 和 Pre_PosFilter 与基于序列度量空间性质的过滤器 RefFilter 尽管过滤集有重叠,但一类过滤器还是能在另一类过滤器过滤的前提下产生较大的新增过滤集,这主要是由于两类过滤器是从序列的"内在属性"和"外在属性"两个正交角度对序列进行过滤的,从而两类过滤器结合使用,使得算法 SSQ_MF 的整体过滤水平较高.

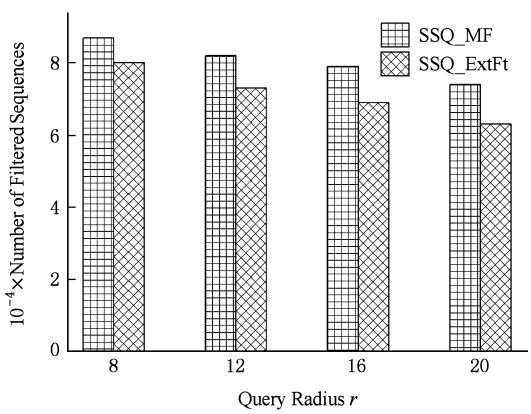


Fig. 8 Comparison of number of filtered sequences between SSQ_MF and SSQ_ExtFt (UNIREF dataset).

图 8 SSQ_MF 和 SSQ_ExtFt 过滤的序列数比较(UNIREF 数据集)

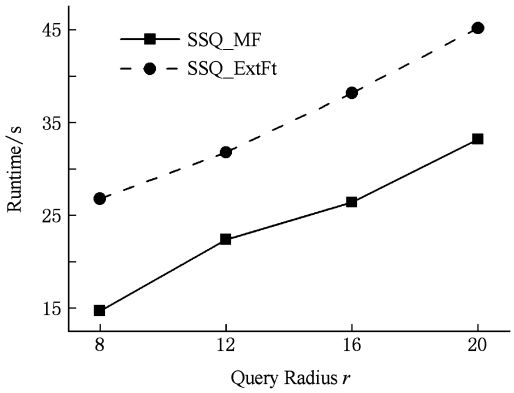


Fig. 9 Comparison of runtime between SSQ_MF and SSQ_ExtFt (UNIREF dataset).

图 9 SSQ_MF 和 SSQ_ExtFt 运行时间比较(UNIREF 数据集)

4.2 过滤代价

我们把 SSQ_MF 算法中随机确定 3 个过滤器过滤顺序的版本称为 SSQ_RanMF. 除了按过滤集大小降序排列的过滤顺序外,我们在其他 5 种(3!−1)过滤顺序中随机选取. 我们在实验数据集 DBLP 上对比了 SSQ_MF 和 SSQ_MF3 在序列过滤中的验证次数和时间效率,图 10 和图 11 给出了实验结果. 从图中可以看出,随着查询半径 r 的增大,由于各过滤器的过滤水平在下降,从而为了过滤 1 条序列进行的过滤验证次数在不断增加. 由于 SSQ_MF 中优化的过滤顺序,使得在序列验证次数上,SSQ_MF 少于 SSQ_RanMF,从而运行时间也得以减少. 在基于数据集 UNIREF 的实验中,我们也得到了类似的结论.

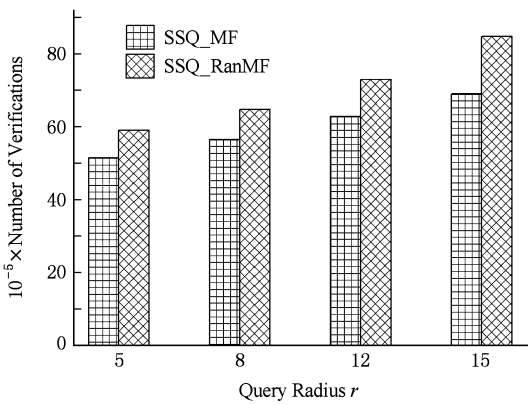


Fig. 10 Comparison of number of verifications between SSQ_MF and SSQ_RanMF (DBLP dataset).

图 10 SSQ_MF 和 SSQ_RanMF 序列验证次数的比较(DBLP 数据集)

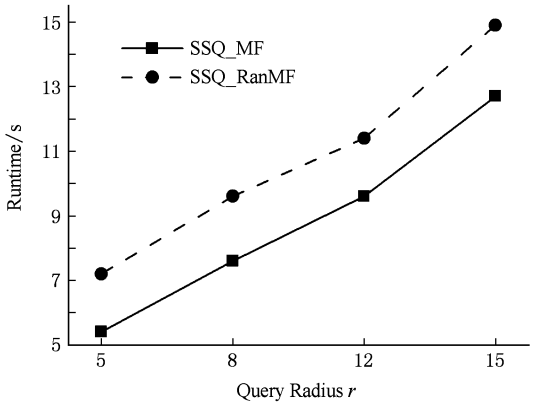


Fig. 11 Comparison of runtime between SSQ_MF and SSQ_RanMF (DBLP dataset).

图 11 SSQ_MF 和 SSQ_RanMF 运行时间比较(DBLP 数据集)

为了考察 3 个过滤集大小的在线估计方法对过滤顺序的影响,我们在 UNIREF 数据集中,对每个查询半径 r,选取 20 条不同的查询序列构成 1 个查询集. 我们共构造 3 组查询半径分别为 5,8 和 12 的查询集 Q_5,Q_8 和 Q_{12}. 我们把 SSQ_MF 中由过滤集大小在线估计方法确定的过滤顺序称为 EstFO,由各过滤器实际过滤集大小确定的过滤顺序(仍按过滤集大小降序排列的过滤顺序)称为 RealFO,表 1给出了算法 SSQ_MF 中 EstFO 接近 RealFO 的程度.

Table 1 Impact of Online Estimation of Filtering Size on Filtering Order

表 1 在线过滤集大小估计对过滤顺序的影响

Query Data Set	Correct EstFOs	Accuracy (Computed by ((all correct EstFOs)/(all RealFOs)))/%
Q_5	17	85
Q_8	19	95
Q_{12}	18	90

从表 1 中可以看出,尽管 3 个过滤集大小是在线估计形成的(其中返回的 $FSLen$ 是精确的),但由它们产生的过滤顺序的准确度都在 85%以上,从而过滤集大小的在线估计方法是有效的.

5 结论和展望

本文提出一种基于多重过滤器的序列相似性查询算法 SSQ_MF,提出一种使得过滤代价最低的过滤顺序模型,并有效地在线估计各过滤器的过滤集大小. 实验结果表明,相对只基于序列自身特征的过

滤器或只基于序列度量空间性质的过滤器,算法SSQ_MF的过滤水平进一步提高.且大多数情况下,SSQ_MF中确定的过滤顺序在过滤代价方面优于随机的过滤顺序.今后我们的工作将对现有各类序列查询过滤器的最优组合方面进行研究.

参 考 文 献

[1] Dong G Z, Pei J. Sequence Data Mining [M]. Berlin: Springer, 2007

[2] Sarawagi S. Advanced Methods for Knowledge Discovery from Complex Data [M]. Berlin: Springer, 2005

[3] Zhu Yangyong, Xiong Yun. DNA sequence data mining technique [J]. Journal of Software, 2007, 18(11): 2766-2781 (in Chinese)
(朱扬勇, 熊赟. DNA序列数据挖掘技术[J]. 软件学报, 2007, 18(11): 2766-2781)

[4] Benson D A, Karsh-Mizrachi I, Lipman D J, et al. Genbank [J]. Nucleic Acids Research, 2000, 28(1): 15-18

[5] Gusfield D. Algorithms on Strings, Trees, and Sequences [M]. New York: Cambridge Press, 1997

[6] Kahveci T, Singh A K. An efficient index structure for string databases [J]. //Apers P, Ceri S, Paraboschi S, et al. eds. Proc of the 27th Int Conf on Very Large Data Bases (VLDB 2001). San Francisco: Morgan Kaufmann, 2001: 351-360

[7] Jokinen P, Ukkonen E. Two algorithms for approximate string matching in static texts [C] //Tarlecki A. Proc of the 16th Symp on Mathematical Foundations of Computer Science. Berlin: Springer, 1991: 240-248

[8] Sutinen E, Tarhio J. On using g-gram locations in approximate string matching [C] //Spirakis P G. Proc of the 3rd Annual European Symp on Algorithms. Berlin: Springer, 1995: 327-340

[9] Gravano L, Ipeirotis P, Jagadish H V, et al. Approximate string joins in a database (Almost) for free [C] //Appers P, Atzeni P, Ceri S, et al. Proc of the VLDB. San Francisco: Morgan Kaufmann, 2001: 491-500

[10] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning [C] //Liu L, Reuter A, Whang K Y, et al. Proc of the 22nd Int Conf on Data Engineering (ICDE). Los Alamitos, CA: IEEE Computer Society, 2006

[11] Li C, Lu J H, Lu Y M. Efficient merging and filtering algorithms for approximate string searches [C] //Proc of the 24th Int Conf on Data Engineering (ICDE). Los Alamitos, CA: IEEE Computer Society, 2008: 257-266

[12] Xiao C, Wang W, Lin X M. Ed-Jion: An efficient algorithm for similarity joins with edit distance constraints [C] //Proc of the 34th Int Conf on Very Large Data Bases (VLDB). Trondheim, Norway: VLDB Endowment, 2008: 933-944

[13] Li C, Wang B, Yang, et al. VGRAM: Improving performance of approximate queries on string collections using variable-length grams [C] //Koch C, Gehrke J, Garofalakis M N, et al. Proc of the 33rd Int Conf on Very Large Data Bases (VLDB). New York: ACM, 2007: 303-314

[14] Yang X C, Wang B, Li C. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently [C] //Wang J T L. Proc of the ACM SIGMOD Int Conf on Management of Data (ACM SIGMOD 2008). New York: ACM, 2008: 353-364

[15] Wang Guoren, Ge Jian, Xu Hengyu, et al. A sequence similarity query processing technique based on two-partitioning frequency transformation [J]. Journal of Software, 2006, 17(2): 232-241 (in Chinese)
(王国仁, 葛健, 徐恒宇, 等. 基于二分频率变换的序列相似性查询处理技术[J]. 软件学报, 2006, 17(2): 232-241)

[16] Yianilos P N. Data structures and algorithms for nearest neighbor search in general metric spaces [C] //Proc of the 4th Annual ACM-SIAM Symp on Discrete Algorithms (SODA). New York: ACM, 1993: 311-321

[17] Ciaccia P, Patella M, Zezula P. M-Tree: An efficient access method for similarity search in metric spaces [C] //Jarke M, Carey M J, Dittrich K R, et al. Proc of the 23rd Int Conf on Very Large Data Bases (VLDB'97). San Francisco: Morgan Kaufmann, 1997: 426-435

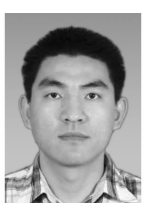
[18] Sahinalp S C, Tasan M, Macker J, et al. Distance based indexing for string proximity search [C] //Dayal U, Ramamritham K, et al. Proc of the 19th Int Conf on Data Engineering (ICDE). Los Alamitos, CA: IEEE Computer Society, 2003: 125-

[19] Venkateswaran J, Lachwani D, Kahveci T, et al. Reference-based indexing of sequence databases [C] //Dayal U, Whang K Y, Lomet D B, et al. Proc of the VLDB. New York: ACM, 2006: 906-917

[20] Venkateswaran J, Kahveci T, Jermaine C, et al. Reference-based indexing for metric spaces with costly distance measures [J]. The VLDB Journal, 2008, 17(5): 1231-1251

[21] Navarro G. A guided tour to approximate string matching [J]. ACM Computing Surveys, 2001, 33(1): 31-88

[22] Ukkonen E. Algorithms for approximate string matching [J]. Information and Control, 1985, 64: 100-118



Dai Dongbo, born in 1977. PhD. His main research interests include database, data mining and bioinformatics.

戴东波, 1977年生,博士,主要研究方向为数据库、数据挖掘、生物信息.

Tang Chunlei, born in 1976. PhD candidate. Her main research interests include database and data mining.

汤春蕾,1976 年生,博士研究生,主要研究方向为数据库、数据挖掘.

Qiu Boren, born in 1984. MS. His main research interests include bioinformatics and data mining.

邱伯仁,1984 年生,硕士,主要研究方向为生物信息、数据挖掘.

Xiong Yun, born in 1980. PhD and lecturer. Her main research interests include database, data mining and bioinformatics.

熊 赟,1980 年生,博士,讲师,主要研究方向为数据库、数据挖掘、生物信息.

Zhu Yangyong, born in 1963. Professor and PhD supervisor. His main research interests include database, data mining and bioinformatics.

朱扬勇,1963 年生,教授,博士生导师,主要研究方向为数据库、数据挖掘、生物信息.

Research Background

Our work is supported by Shanghai Leading Academic Discipline Project under grant No. B114 and the Development Fund of Shanghai Science and Technology Commission under grant No. 08511500203.

Similarity query in sequence data is a very important task in Web search engine, bioinformatics and pattern recognition. The rapid increase of sequence data in Web, biological database and other scientific computing field poses a great challenge for the efficient similarity query in these large databases. Up to now, many methods are proposed to tackle efficient similarity query problem, which are almost based on multiple filters running in a fixed filtering order. However, those filters are only exploiting strings' own features or only use the space properties formed by edit distance. Also, these algorithms don't investigate the impact of filtering orders on filtering cost. So, in this paper, we propose an algorithm called SSQ_MF, which combines metric property of sequences' distance with sequences' own characteristics. In addition, every filter's filtering size is effectively estimated and a model in which optimal filtering order is determined by every filter's filtering size is built. Comprehensive experimental results demonstrate that in terms of query performance, SSQ_MF is better than algorithms with a single filter and algorithms with multiple filters but executing in a random order.