近似串匹配过滤算法研究

孙德才[1],王晓霞[2]
(1. 渤海大学 信息科学与技术学院,辽宁 锦州 121013;
2. 渤海大学 大学计算机教研部,辽宁 锦州 121013)

摘 要:近似串匹配在众多研究领域都有广泛的应用,如文本检索、生物信息学等。文中对基于过滤技术的 Off-line 模式近似串匹配算法进行了相关研究。首先介绍了串匹配的基础知识和近似串匹配技术的应用分类;然后阐述了 Off-line 模式近似串匹配算法常用的索引结构;接着详细介绍了近似串匹配过滤算法的研究现状,并阐述了几个经典过滤算法的过滤原理;最后在实验中对比了这些经典过滤算法的性能差异,实验数据显示提高过滤效率和减少过滤时间是加快过滤算法匹配速度所要解决的关键问题。研究表明,基于留空 q-gram 的过滤算法是近似串匹配未来研究的方向。

关键词:串匹配;近似串匹配;过滤算法;q-gram 过滤
中图分类号:TP391.3 **文献标识码**:A **文章编号**:1673-629X(2015)04-0171-06
doi:10.3969/j.issn.1673-629X.2015.04.039

Research on Filtering Algorithm of Approximate String Matching

SUN De-cai[1], WANG Xiao-xia[2]
(1. College of Information Science and Technology, Bohai University, Jinzhou 121013, China;
2. Teaching and Research Institute of College Computer, Bohai University, Jinzhou 121013, China)

Abstract: Approximate string matching is widely used in many areas, such as text retrieval, computational biology, etc. In this paper, a survey on filter-based approximate string matching algorithm of Off-line mode is done. First, the preliminaries of string matching and the classifications of approximate string matching techniques are introduced. Next, some index structures which are often used in Off-line approximate string matching algorithms are illustrated. Then, the research status quo of approximate string matching is described in detail, and some classical filter algorithms are illustrated. Last, the performance of these classical filtering algorithms is given in experiment, and experimental data shows that enhancing filtration efficiency and decreasing filtration time are two key issues of improving matching speed. The research shows that the filter algorithms based on gapped q-gram is a further research direction of approximate string matching.

Key words: string matching; approximate string matching; filter algorithm; q-gram filter

0 引 言

字符串是定义在有限字母表上的一个字符序列。字符串匹配是在一个长字符串 T 中搜索给定字符串 P 所有出现位置的过程,其中 T 称为文本串,P 称为模式串。字符串匹配包括精确串匹配和近似串匹配二种。近似串匹配(Approximate String Matching)[1]与精确匹配不同,它是允许有“错误”发生的字符串匹配。这个“错误”数也称“距离”,可以采用编辑距离、汉明距离、最长公共子串距离或 N-gram 距离等表示。近似串匹配技术在众多研究领域都有广泛的应用,如文本检索、生物信息学、信号处理、入侵检测、模式识别、数据挖掘和手写识别等。

编辑距离[2]是指把一个字符串 A 经过编辑操作转变成字符串 B 所要进行的最小操作次数,这里的编辑操作包括插入、修改或删除三种,是衡量字符串间相似程度最常用的描述方式,用 ED(A,B) 表示。编辑距离

收稿日期:2014-05-11 **修回日期**:2014-08-05 **网络出版时间**:2015-01-20
基金项目:国家自然科学基金资助项目(61173142);2014 年辽宁省博士科研启动基金计划(20141138);辽宁省社科联 2014 年度辽宁经济社会发展立项重点课题(2014lslktzdian-04)
作者简介:孙德才(1979-),男,讲师,博士,研究方向为近似串匹配、复制取证、基因序列比对等。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150120.2202.034.html

计算的最基本方式是动态程序法,其主要过程是填充字符串 A 和 B 的编辑距离矩阵 $\boldsymbol{C}_{0\cdots|A|,0\cdots|B|}$,其中 $|X|$为串 X 的长度。编辑距离矩阵的计算过程[1] 如下:

$$C_{i,0}=i$$

$$C_{0,j}=j$$

$$C_{i,j}=\begin{cases}C_{i-1,j-1} & A[i]=B[j]\\ 1+\min(C_{i-1,j},C_{i,j-1},C_{i-1,j-1}) & A[i]\neq B[j]\end{cases} \tag{1}$$

通过公式(1)可计算出矩阵的所有元素,其中元素 $C_{i,j}$ 表示字符串 $A[1,\cdots,i]$ 和字符串 $B[1,\cdots,j]$ 间的最小编辑次数,即它们间的编辑距离。矩阵的最后一个元素 $C_{|A|,|B|}$ 就是字符串 A 和 B 间的最小编辑距离,即 $\mathrm{ED}(A,B)=C_{|A|,|B|}$。

因不同应用对近似串匹配的需求也不同,近似串匹配算法可分为全局匹配、局部匹配等[3]。全局匹配是指在文本串中查找所有模式串的匹配串的过程。局部匹配是指在文本串中找出所有模式串子串的匹配串的过程,一般用参数 w 来限定模式串子串的最短长度。局部匹配并不是多次全局匹配的简单组合,它需要在匹配过程中充分利用模式串中多个子串的重叠信息来加快匹配速度。

近似串匹配算法又根据应用中是否允许对文本串进行查询前的预处理,可分为 On-line 模式和 Off-line 模式[3]。On-line 模式的匹配算法只允许对模式串进行预处理,而对文本串不能。该模式的匹配算法已被广泛研究[1],一些算法的时间复杂度已经达到了 $O(k|T|)$,甚至 $O(|T|)$。但当文本串非常长,且要频繁使用不同的模式串进行匹配时,即使最好的 On-line 算法的匹配速度也无法达到实际应用的要求。相对而言,Off-line 模式的匹配算法允许匹配前对文本串进行预处理,在处理过程中收集和存储文本串的组成信息,最后构建一个索引[4]。实际使用中索引的创建时间并不计算在查询时间之内,因查询中可以充分利用索引中已存储的信息,所以其匹配速度要远快于 On-line 模式的算法。

最早的近似串匹配算法都是基于动态程序法的 On-line 算法。动态程序法是 1970 年由 Needleman 和 Wunsch[5] 提出来的,后来众多学者在此基础上进行了改进,其中最著名的是 1981 年由 Smith 和 Waterman 提出的 Smith-Waterman 算法[6]。Smith-Waterman 算法能找出所有的局部匹配,但该算法的时间复杂度是平方阶的,并且会花费大量时间去计算那些并不存在匹配串的文本区域。为提高近似串的匹配速度,尤其是在较大的文本库中进行匹配,此后一些学者提出了具有启发式的算法,例如,BLAST 家族[7]、FASTA[8]、PatternHunter[9]、SST[10] 等。但启发式算法可能丢失真实匹配,且其匹配速度仍无法满足当前应用的需要。过滤算法[3] 是采用了过滤技术的新一代匹配算法,其思想兴起于 90 年代中期。经过众多学者的研究,过滤算法中涌现出了众多优秀的匹配算法。

文中主要探讨 Off-line 模式下近似串匹配过滤算法的基本知识、研究现状、关键问题以及研究趋势,希望能为近似串匹配技术的研究以抛砖引玉的作用。

1 Off-line 模式常用索引结构

在 Off-line 模式的近似串匹配中,文本串是由大量文本文件构成的文本库。为提高算法的匹配速度,首先需要对文本库进行预处理,即建立一个适当的索引结构。在匹配过程中可以充分利用索引中已存储的信息加快匹配速度。因此选择适当的索引结构是设计一个 Off-line 匹配算法的前期基础。

目前,索引主要分为词索引和序列索引两种。词索引是以文本库中的词为基本单位建立的索引,主要应用于自然语言处理方面。建立词索引首先需要一个所处理领域的词典,建立索引时需要根据词典从文本库中抽取每个文本文件中包含的词。序列索引是以文本库中的字符为基本单位建立的索引,在近似串匹配中应用较广泛。序列索引有多种组织形式,常用的索引结构[4] 有后缀树、后缀数组和倒排索引等。

(1)后缀树。

在文本串中,从某字符开始到串结束构成的子串被定义为文本串的一个后缀串。后缀 trie 是由文本串的所有后缀串构成的一个树形数据结构,其中每一个叶子节点对应一个后缀串的开始位置,每个中间节点表示一个在文本串中出现一次以上的不同子串。每个文本串中的子串都能从树的根节点出发找到一条路径,当到达叶子节点时可继续从根节点查询。后缀树是后缀 trie 的压缩存储形式,它要求除根节点外不允许其他内部节点只含有一个子节点,它的空间耗费要低于后缀 trie。后缀树建立时间消耗为 $O(n)$,空间消耗为 $O(n)$。在后缀树中查找文本串子串的时间复杂度为 $O(m)$。后缀树的缺点是占用内存空间巨大,一般未压缩的索引占存储空间是文本串的 12 倍,另外,后缀树的存储空间也难于管理。

(2)后缀数组。

后缀数组是后缀树的一种弱化的以数组存储的压缩形式,它比后缀树需要更少的存储空间,但查询效率有所下降。一个后缀数组构建过程是先提取出文本串的所有后缀串及其地址,然后按字母表顺序排列这些后缀串,此时地址顺序所构成的一维数组就是后缀数

组。后缀数组建立时最坏情况下所消耗的时间为 $O(n\log_2 n)$,平均时间为 $O(n\log_2\log_2 n)$。在后缀数组中能够应用后缀树的匹配算法,但以增加 $O(\log_2 n)$ 匹配时间为代价。

(3)倒排索引。

倒排索引也称为反向索引、置入档案或反向档案,是文本检索中一种最常用的索引方法。倒排索引存储的是某个单词或字符串在一个文本或一组文本中存储位置的映射,倒排索引由词汇表和倒排列表两部分组成。词汇表为文本库中所有不同词汇的集合。词汇根据提取形式不同,可以是词,也可以是字符串。词汇表常采用 B* 树或哈希表形式存储。倒排列表也称地址列表,是当前索引项在文本库中出现的所有地址的集合。倒排索引分为词倒排和序列倒排两种,其中序列倒排在近似串匹配中应用较多。

2　近似串匹配过滤算法

2.1　过滤算法基础

过滤算法是一种先使用过滤条件剔除大量文本片段,再采用 On-line 算法验证未过滤掉文本片段的近似串匹配方法。过滤算法因采用了过滤技术能在前期快速去除大量文本区域,特别适合 Off-line 模式下的大文本库近似匹配,具有匹配速度快、实现简单等优势,具有广阔的应用前景。过滤算法主要分为二个阶段[3]:过滤阶段和验证阶段,如图 1 所示。

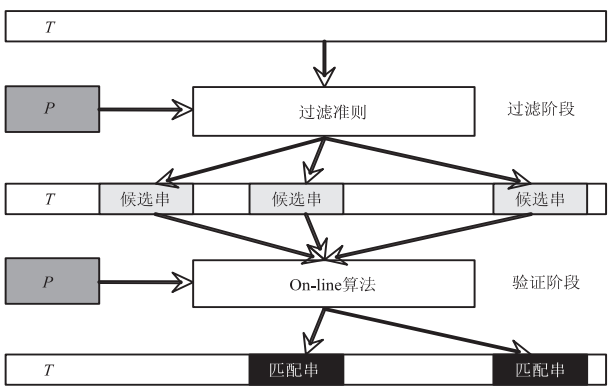


图 1　过滤算法的二个阶段

如图 1 所示,过滤算法的过滤阶段使用过滤准则快速地过滤掉文本串中大量的文本区域,而通过过滤的文本区域则为匹配串和非匹配串的集合,称这些文本区域为“候选串”。验证阶段则采用 On-line 算法对每个候选串进行精确验证,最终找出所有匹配串,以及这些匹配串与模式串或模式串子串间的编辑距离。

如过滤算法的过滤准则限制得非常严格,则不会过滤掉任何一个匹配串,即无漏报,这样的过滤算法称为无损过滤算法。如过滤准则限制不严格,允许漏掉一小部分可能存在匹配串的文本区域,即存在漏报,则称为有损过滤算法。一个过滤算法根据不同应用中过滤准则的设置可能是无损的,也可能变成有损的。

评价算法性能的最基本标准是时间效率和空间效率。空间效率由过滤算法的设计结构决定。过滤算法的匹配时间主要分为过滤时间和验证时间二部分。另外,过滤效率也是评价过滤算法的一个重要参数,定义如下:

$$f_e = \frac{n - n_f}{n - n_t} \tag{2}$$

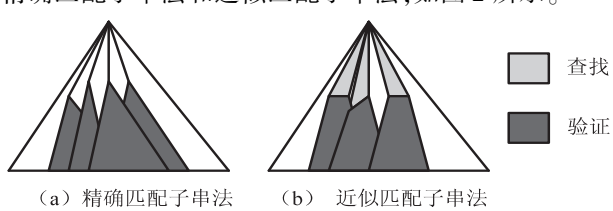
式中,n 表示整个文本库的长度,即 $n = |T|$;n_f 表示过滤算法在本次匹配中未过滤掉的文本区域的总长度;n_t 表示本次匹配文本库中真实存在的所有匹配串长度总和。

过滤效率的大小体现了过滤算法抛弃与匹配无关文本的能力,即过滤能力。

过滤效率和验证时间是一对密切相关的参数,过滤效率高,则验证时间就短,反之验证时间则长。过滤效率和过滤时间也是一对矛盾的参数,过滤效率的高低与过滤过程中使用的过滤准则密切相关,如过滤准则简单,则过滤时间短,但过滤效率一般较低;如过滤准则复杂,则过滤时间长,但过滤效率一般较高。优秀的过滤算法能在较短的过滤时间内获得较高的过滤效率,因此如何平衡过滤时间和过滤效率是提高过滤算法整体匹配速度的关键,也是众多过滤算法研究者所要解决的关键问题。

2.2　过滤算法的研究现状

基于过滤思想的匹配算法主要源于 90 年代初,已有算法根据采用的匹配方法不同,大致可归为二类[4]:精确匹配子串法和近似匹配子串法,如图 2 所示。



(a) 精确匹配子串法　　(b) 近似匹配子串法

图 2　近似串匹配过滤算法的分类

(1)精确匹配子串法。

精确匹配子串法利用索引查找模式串的部分子串在文本串中出现的所有精确匹配位置,然后再对出现精确匹配子串附近的文本区域进行验证,从而找到满足要求的匹配串,如图 2(a)所示。如文献[11-12]把模式串分成 $k+s$ 部分,在文本串中至少出现 s 部分且位置正确的文本区域才会被验证,例如文献[11]中采用了 $s=1$,而文献[12]中采用了 $s>1$。如文献[13-15]对文本串和模式串都进行连续的 q-gram 拆分,匹配时对至少存在 $|P|+1-(k+1)q$ 个 q-gram 的文本区域进行验证。如文献[16]间隔地从文本串中抽取 q

-sample,在匹配中对出现一定数目 q -sample 且位置正确的文本区域进行验证。

(2)近似匹配子串法。

近似匹配子串法利用索引找出与模式串子串间有一定错误偏差的文本串的近似子串,然后再验证这些近似子串附近的文本区域,从而得到满足要求的匹配串,如图 2(b)所示。如文献[17]把模式串分成 j 部分,对存在至少一部分错误不大于 $\lfloor k/j \rfloor$ 的文本区域进行验证。如文献[18]把模式串分割成 q -sample,匹配时在索引中查找与模式串 q -sample 的近似匹配,并对出现 q -sample 的文本区域进行验证。

以上众多近似串匹配算法中,在过滤过程中常采用长度为 q 的子串(称为 q -gram 或 q -sample)进行过滤,这里称为 q -gram 过滤算法,如文献[13-16,18]等。q -gram 过滤算法具有过滤效率高、过滤速度快的特点,但它们都有个通病,即当匹配要求的错误率超过某个阈值时,q -gram 过滤算法的过滤效率会突变为 0,即发生崩塌[19]。

近年来,基于留空 q -gram 的匹配算法已被广泛研究[3,9,20],该类算法介于精确匹配子串法和近似匹配子串法之间。留空 q -gram 是一些长度为 $s(s > q)$ 的模式,模式中除 q 个固定位置的字符外,其他位置的字符被认为是空位。在匹配过程中只需匹配这 q 个固定的字符,而无需关心空位上的字符。目前,该类算法的研究主要集中在种子优化上,该类匹配算法具有错误容忍度高,匹配速度快,可使用多留空 q -gram 进行匹配等优势,是近似串匹配未来的研究趋势。

2.3 经典过滤算法及原理

1992 年,Wu 和 Manber[11]提出一个全局近似匹配的基础过滤准则,即把模式串分割成 $k+1$ 个不重叠的子串,如果文本串中存在与模式串编辑距离不大于 k 的匹配串,那么文本串中一定存在至少一个完整匹配的模式串子串,如图 3 所示。该过滤准则后来被推广为分割成 $k+s$ 部分,至少存在 s 个完全匹配且位置也满足要求。当 $s=1$ 时就是 Wu 和 Manber 提出的过滤算法,这里称为 KS1 算法。

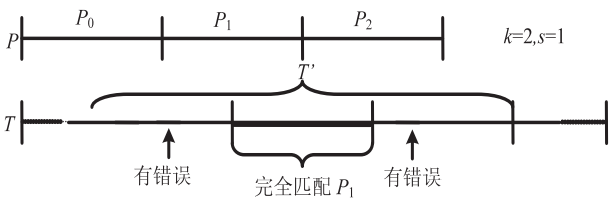


图 3 KS1 算法过滤原理图

1999 年,Burkhardt[14]对 Jokinen 和 Ukkonen[13]的工作进行了改进,提出一种近似串局部匹配算法 QUASAR,解决了在基因库中找出所有与查询串中长度不小于 w 的子串满足最大 k 不同的匹配问题。该算法中采用查找表和后缀数组相结合的索引结构。算法中首先把文本串分割成连续重叠的逻辑块,分块大小为 b,$b \geqslant 2w$,为避免遗漏任何匹配串,相邻分块间重叠 $b/2$,如图 4 所示。匹配过程中通过移动窗口技术处理查询串,提取索引中的 q -gram 地址并计算各个逻辑块内的 q -gram 命中数目,并使用计数器进行存储,最后通过一次扫描计数器抛弃那些命中数目不满足 Jokinen 和 Ukkonen 基础过滤定理[13]要求的逻辑块。该算法在过滤区选择和局部匹配过程上进行了精心设计,进行局部匹配时速度较快。

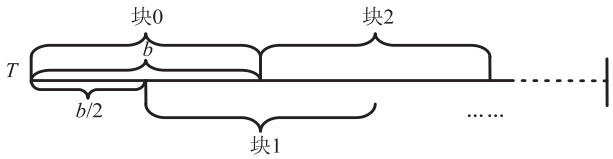


图 4 QUASAR 算法过滤原理图

2006 年,Rasmussen[15]从基因检索算法 FASTA[8]中受到了启发,提出一种近似串局部匹配算法 SWIFT。该算法采用对角线过滤技术,解决了局部匹配中的 ε 匹配问题,即找出与查询串中子串长度不小于 w 的且错误率不大于 ε 的所有匹配串。该算法中使用了平行四边形的过滤区,如图 5 所示。为避免遗漏匹配串,相邻过滤区间重叠 k 个对角线。SWIFT 算法中的过滤标准仍是 Jokinen 和 Ukkonen 基础过滤定理[13]。该算法虽然在过滤阶段的时间开销有所增加,但它的过滤效率非常高,因而局部匹配速度也非常快。

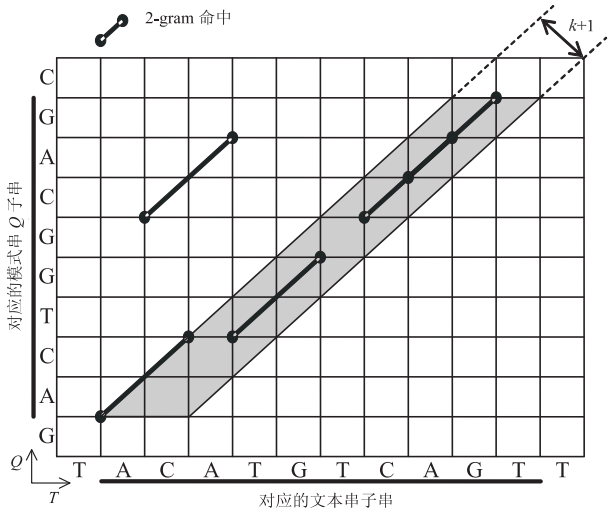


图 5 SWIFT 算法过滤原理图

3 算法性能对比分析

为在同一环境下对比这些经典近似串匹配过滤算法的性能差异,文中用 C++实现了 KS1[11]、QUASAR[14]和 SWIFT[15]。

3.1 实验环境和参数设置

文中的实验数据来源于美国国家生物技术信息中

心 NCBI 的人类不同完整基因序列(ftp. ncbi. nih. gov//repository/UniGene/Homo_sapiens/Hs. seq. uniq. gz),共约 164 MB 基因序列文本,共 123 252 个完整人类不同基因序列。为进行批量匹配实验,文中从基因库中随机选取并构建了一个由 500 个长度都为 2 000 的查询串集合。

因文中实验只对比算法进行全局匹配的性能差异,所以设置 QUASAR 和 SWIFT 的窗口长度等于查询串的长度以进行全局匹配,即 $w = |P|$。实验中 QUASAR 的块大小设置为 $2w$, SWIFT 的 Bin 大小设置为 $k + 2^x + 1, x \in \mathbb{N}, 2^x > k$,SWIFT 和 QUASAR 的 q 值都为 11。

3.2　算法性能对比分析

不同算法在不同匹配错误率下具有不同的匹配速度。为分析匹配错误率 e 对各个算法性能的影响,实验中对查询串集分别采用 KS1、QUASAR 和 SWIFT 进行批量匹配实验。实验中设置的匹配错误率分别为 0,0.005,0.01,0.02,0.03,0.04。实验中分别统计了不同匹配错误率下各个算法的平均过滤时间、平均过滤效率、平均验证时间和平均匹配时间,如图 6 所示。

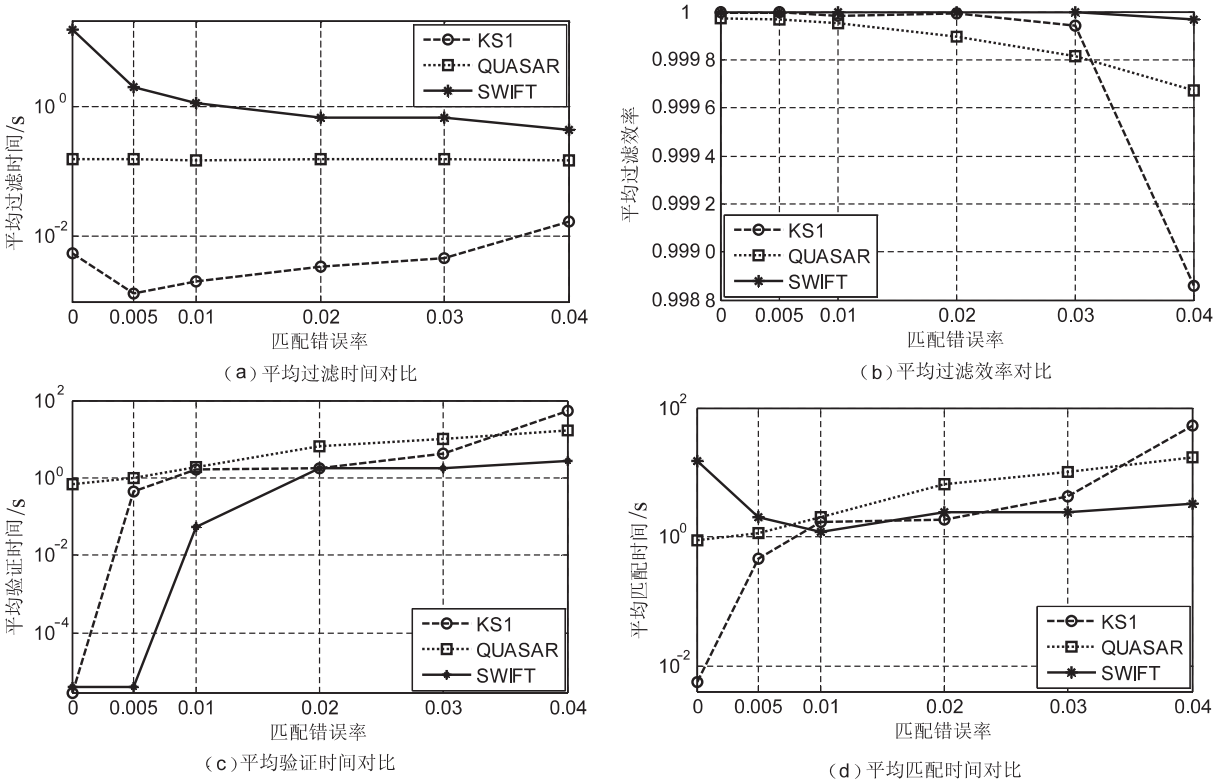


图 6　过滤时间、过滤效率、验证时间和匹配时间对比

从图 6(a)的平均过滤时间对比可知,KS1 的过滤时间最短,因其过滤过程最为简单;而 SWIFT 的过滤时间最长,因其采用了平行四边形的过滤区,过滤过程最为复杂;QUASAR 的过滤时间较稳定,因其过滤也较为简单。

从图 6(b)的平均过滤效率对比可知,SWIFT 的过滤效率最高,因其采用了复杂的过滤过程和严格的过滤准则,在过滤阶段抛弃了更多的无关文本;KS1 当匹配错误率较小时,分块数目较少,所以过滤效率也较高,但随着匹配错误率的增加,算法过滤效率快速下降;而 QUASAR 算法的过滤效率一直都比较低。

从图 6(c)的平均验证时间对比,并结合各算法的过滤效率可知,算法的过滤效率决定了验证时间的大小。QUASAR 的验证时间一直都较长;SWIFT 因其过滤效率最高而验证时间最短;当匹配错误率较低时 KS1 的验证时间处于中档,而匹配错误率较高时验证时间最长。

从图 6(d)的平均匹配时间对比可知,当匹配错误率较低时,KS1 的匹配速度最快;但随着匹配错误率的增大,SWIFT 的匹配速度逐渐变为最快;而 QUASAR 的匹配时间一直都较长。

4　结束语

文中首先介绍了串匹配的基础知识,并给出了 Off-line 模式下近似串匹配常用的索引结构。然后介绍了近似串匹配过滤算法的基本原理,并重点阐述了过滤算法的分类以及各类算法的研究现状,还详细介绍了几种经典过滤算法的匹配原理。最后通过实验分析并对比了这几种经典过滤算法的性能差异。

近似串匹配在文本检索、生物信息学、信号处理等领域都有广泛的应用,当前研究的关键问题是如何在

大数据环境下减少存储空间消耗和提高匹配速度。另外,基于留空 q -gram 的过滤算法具有错误容忍度高、匹配速度快等优势,是近似串匹配当前研究中的热点,也是未来的研究趋势。

参考文献:

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

[2] Levenshtein V. Binary codes capable of correcting deletions, insertions,and reversals[J]. Soviet Physics Doklady,1966,10(8):707-710.

[3] Burkhardt S. Filter algorithms for approximate string matching [D]. Saarland:Saarland University,2002.

[4] Navarro G,Baeza-Yates R,Sutineny E,et al. Indexing methods for approximate string matching[J]. IEEE Data Engineering Bulletin,2001,24(4):19-27.

[5] Needleman S,Wunsch C. A general method applicable to the search for similarities in the amino acid sequence of two proteins[J]. Journal of Molecular Biology,1970,48(3):443-453.

[6] Smith T F,Waterman M S. Identification of common molecular subsequences[J]. Journal of Molecular Biology,1981,147(1):195-197.

[7] Altschul S F, Madden T L, Alejandro A S,et al. Gapped BLAST and PSI-BLAST:a new generation of protein database search programs[J]. Nucleic Acids Research,1997,25(17):3389-3402.

[8] Pearson W R,Lipman D J. Improved tools for biological sequence comparison[J]. Proceedings of the National Academy of Sciences of the United States of America,1988,85(8):2444-2448.

[9] Ma B,Tromp J,Li M. PatternHunter:faster and more sensitive homology search[J]. Bioinformatics,2002,18(3):440-445.

[10] Giladi E,Walker M G,Wang J Z,et al. SST:an algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size[J]. Bioinformatics,2002,18(6):873-879.

[11] Wu S,Manber U. Fast text searching allowing errors[J]. Communications of the ACM,1992,35(10):83-91.

[12] Chang Y I,Chen J R,Hsu M T. A hash trie filter method for approximate string matching in genomic databases[J]. Applied Intelligence,2010,33(1):21-38.

[13] Jokinen P,Ukkonen E. Two algorithms for approximate string matching in static texts[C]//Proceedings of the 16th international symposium on mathematical foundations of computer science. Berlin,Germany:Springer-Verlag,1991:240-248.

[14] Burkhardt S,Crauser A,Ferragina P,et al. Q-gram based database searching using a suffix array[C]//Proceedings of the annual international conference on computational molecular biology. New York,USA:ACM,1999:77-83.

[15] Rasmussen K R,Stoye J,Myers E W. Efficient q-gram filters for finding all epsilon-matches over a given length[J]. Journal of Computational Biology,2006,13(2):296-308.

[16] Sutinen E,Tarhio J. Filtration with q-samples in approximate string matching[C]//Proceedings of 7th annual symposium on combinatorial pattern matching. Berlin, Germany: Springer-Verlag,1996:50-63.

[17] Navarro G,Baeza-Yates R. A hybrid indexing method for approximate string matching[J]. Journal of Discrete Algorithms,2000,1(1):205-239.

[18] Navarro G,Sutinen E,Tarhio J. Indexing text with approximate q-grams[C]//Proceedings of CPM'2000. Berlin,Germany:Springer,2000:350-363.

[19] Sutinen E,Szpankowski W. On the collapse of the q-gram filtration[C]//Proceedings of international conferences on FUN with algorithms 1998. Waterloo, Canada: Carleton Scientific,1998:178-193.

[20] Egidi L,Manzini G. Better spaced seeds using quadratic residues[J]. Journal of Computer and System Sciences,2013,79(7):1144-1155.

(上接第 170 页)

2011:596-597.

[31] Keogh E,Lonardi S,Chiu W. Finding surprising patterns in a time series database in linear time and space[C]//Proc of the 8th ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM Press,2002:550-556.

[32] Chandola V. Anomaly detection for symbolic sequences and time series data[D]. Minnesota:The University of Minnesota,2009.

[33] 孙梅玉. 基于分形的非平稳时间序列挖掘关键技术研究[D]. 上海:东华大学,2009.

[34] 翁小清,沈钧毅. 多变量时间序列例外模式的识别[J]. 模式识别与人工智能,2007,20(3):336-342.

[35] 王 欣. 两阶段的多元时间序列异常检测算法[J]. 计算机应用研究,2011,28(7):2466-2469.

[36] 李 权,周兴社. 基于 KPCA 的多变量时间序列数据异常检测方法研究[J]. 计算机测量与控制,2011,19(4):822-825.

[37] Baragona R,Battaglia F. Outlier detection in multivariate time series by independent analysis[J]. Neural Computation,2007,19(7):1962-1984.

近似串匹配过滤算法研究

作者： 孙德才， 王晓霞， SUN De-cai， WANG Xiao-xia
作者单位： 孙德才,SUN De-cai(渤海大学 信息科学与技术学院,辽宁 锦州,121013)， 王晓霞,WANG Xiao-xia(渤海大学 大学计算机教研部,辽宁 锦州,121013)

刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2015(4)