基于索引过滤的汉语短文本模糊匹配计算方法*

曹鞏 [1,2]，邬晓钧 [2]，夏云庆 [2]，郑方 [2]
(1. 清华大学 计算机科学与技术系，北京 100084;
2. 清华信息科学技术国家实验室技术创新和开发部语音和语言技术中心，北京 100084)

文　摘： 在当前的中文信息处理中，对短文本进行模糊匹配有广泛的应用。而现有的模糊匹配算法时间复杂度常常无法满足实际的在线需求。本文从索引检索代替顺序计算的思想出发，提出了基于索引过滤的中文短文本模糊匹配计算方法，包括长度过滤和字命中过滤两种方法，能够大大地减少模糊匹配的计算量。实验表明，本文提出的算法在不影响召回率的前提下，能够极大地减少模糊匹配任务的计算时间。

关键词：TP391；编辑距离；模糊匹配；索引；过滤
中图分类号： TP391

近年来随着互联网和通信产业的飞速发展，汉语文本处理面临着更多的问题与挑战。商业搜索引擎已经成为用户访问互联网的重要接口。现有的信息检索系统大部分采用基于关键词匹配的检索技术[1]。在实际应用中，用户的输入往往无法精确表述自己的检索意图，检索系统在构建数据集时也可能会引入错误[2]。同时，聊天室、即时通信软件、手机短信也逐渐成为人们的日常生活和工作中十分重要的部分。短信、聊天软件的文本与搜索引擎的用户查询的共同特点是：文本长度不长，数量十分巨大，可能含有比较多的错误，通常的文本处理方法难以奏效。我们将这类文本称为短文本。

由于短文本的广泛存在，对短文本进行过滤、分类和聚类有巨大的实际需求。这三类需求在技术解决方案上有一个共同点，就是计算两个文本之间的相似度，换个角度来说，就是在一定的阈值之内对短文本进行模糊匹配。

研究短文本的模糊匹配对于解决这类问题也有重大意义。此外，模糊匹配方法还广泛用于其他领域，如入侵检测、信息过滤、基因检测等[4][5]。

对文本进行模糊匹配，需要寻求一种统一的方法表征两个文本之间的相似程度。编辑距离（Edit Distance）是计算文本相似度时普遍采用的技术[4]。目前常见的编辑距离在线计算方法的时间复杂度无法满足互联网海量数据处理的要求，离实际应用存在一定差距。

针对上述问题，本文提出了一种离线建立索引并对待匹配文本进行预先过滤的方法，在保证召回率的同时，可大大减小模糊匹配的时间复杂度。我们的实验也表明，所提出的方法能够有效地降低短文本模糊匹配的用时，基本满足实际应用的需要。

下面的正文部分首先介绍模糊匹配的概念、常见的模糊匹配算法以及目前实际应用中面临的问题；然后提出基于索引过滤的汉语模糊匹配计算方法；随后，介绍实验的设计与实验结果；最后，总结全文并展望未来的工作。

1　相关研究工作

1.1 编辑距离与模糊匹配

目前，普遍采用编辑距离来刻画两个字符串的差异[4]。设 A、B 为两个字符串，狭义的编辑距离定义为把 A 转换成 B 需要的最少删除（删除 A 中一个字符）、插入（在 A 中插入一个字符）和替换（把 A 中的某个字符替换成另一个字符）的次数，用 $ED(A,B)$ 来表示。直观来说，两个串互相转换需要经过的步骤越多，差异越大。

除了编辑距离以外，两个字符串的最长公共子串和最长公共子序列的长度也常常被用来衡量两个字符串的相似程度。最长公共子串可以看作最长公共子序列的一种特殊情况。当替换代价为无穷大的时候，求最长公共子序列就等同于求编辑距离。所以，在计算意义上，三种相似度度量方式可以统一。因而，在本文中将只针对编辑距

*基金项目：国家自然科学基金项目：搭配驱动意见挖掘研究（编号：60703051）
作者简介：曹鞏（1984-），男（汉），湖北人，硕士生。
通讯联系人：郑方，研究员，博士生导师，fzheng@tsinghua.edu.cn

离来进行讨论和研究。

如果给定目标字符串 *str* 和模式串 *pat*，模糊匹配问题可抽象地描述为：给定正整数 *k* 作为阈值，找出目标字符串 *str* 中的所有子串 *s*，使得 $ED(s, pat) \le k$。一般地，针对不同长度的 *pat*，我们用 $\alpha = k / m$ 来作为阈值，其中，*m* 是 *pat* 的长度。

显然，在 k 为 0 时，模糊匹配退化为经典精确匹配问题，即给定目标字符串 *str* 和模式串 *pat*，在 *str* 中寻找 *pat* 的匹配位置。在实际应用中，BM 算法及其改进算法（如 BMH[6]等）能达到极高的效率（子线性），被各种检索系统广泛使用[6][7]。而 BM 算法的“跳跃”策略，在模糊匹配的很多计算方法如 TU 过滤算法中，都得到了借鉴和利用。

模糊匹配技术中计算编辑距离的主要策略有以下几种[6]：动态规划[8]、自动机[9]、位平行策略[10]。在实际的应用中，常常将它们结合使用以获得更高的时间效率[11]。但是，在文献[12]中已经证明，计算两个长度分别为 *m* 和 *n* 的字符串的最长公共子序列的时间复杂度不可能低于 $O(m \log n)$。从前面的分析可以知道，计算编辑距离的时间复杂度不会低于这个复杂度，在很多情况下这是无法满足实际应用需要的。

文献[3]使用带有剪枝的动态规划算法和扩展的 TU 过滤算法进行汉语模糊匹配，取得了非常好的效果。在该文献的实验中，*str* 为 600 左右的歌手、乐队名和 5000 左右的歌曲名，*pat* 为用户日志中的一次用户输入，它们的长度大多在 10 个汉字以内，在 *k* 限制最小且算法运行最快的情况下，一次模糊匹配所需要的时间是 10 毫秒左右。如果待匹配的 *str* 进一步增加，模糊匹配运行的时间也将相应线性增加。

1.2 存在的问题

主流商业搜索引擎，如百度[1]、谷歌[2]等，常常需要利用模糊匹配技术处理基于用户日志对某些含错的用户查询输入进行查询修正，或者利用模糊匹配技术对检索结果进行再聚类。这些搜索引擎数据量巨大，每天的用户访问次数以亿计，每次检索都要求及时响应，而此类搜索引擎对于一次用户查询的响应时间通常不超过 10 毫秒。如果在发现查询结果数较少的后对查询进行改写，那么在查询日志中进行模糊匹配所消耗的时间必定无法过长，否则用户查询的响应时间也将会随之增长，搜索引擎对大批量用户查询的处理能力也必将受到限制。同样的，对搜索结果进行再聚类也面临着同样的时间消耗上的限制。

在已有的研究中，文献[3]中的算法在时间复杂度的优化上已经达到了同类算法中比较成熟的地步。但还是和实际的线上应用需求存在不小的差距。虽然可以通过采用更好的硬件、部署更多的机器进行分布式计算来满足应用需求，但如果能在算法层面上有效降低时间复杂度，就能大大节省计算资源。

2 索引过滤

以基于索引的检索代替顺序匹配是本文提出的过滤算法的主要思想。

在实际应用中，目标字符串 *str* 通常由很多长短不一的文本组成。在很多应用情况下，如查询修正和搜索结果聚类，对于不同的模式串 *pat*，待匹配的目标字符串 *str* 集合基本保持不变。因此，如果在匹配之前先对 *str* 进行预处理建立索引，在实际匹配的过程中直接忽略那些绝对不可能满足匹配要求的字符串，就可以大大减少编辑距离的计算次数，提高整个匹配过程的效率。

2.1 长度过滤

假设模式串 *pat* 的长度为 *m*，要求的编辑距离阈值为 $k(k < m)$，令 $\alpha = k / m$。若目标字符串 *str* 中某一个子串 S_1 的长度为 *x*，则由编辑距离的阈值定义可知：$ED(S_1, pat) \ge |m - x|$。如果文本 S_1 满足阈值 *k*，那么 $ED(S_1, pat) \le k$，即 $|m - x| \le k = \alpha m$，从这个式子中可以得到：

$$(1-\alpha)m \le x \le (1+\alpha)m$$

上式表明，只有长度在这个范围之内的文本，与模式串 *pat* 的编辑距离才有可能小于等于阈值 *k*。因而，对于目标字符串 *str* 中的一系列长短不一的子串，可以在预处理的时候按照长度建立索引，当输入某个模式串 *pat* 的时候，只在那些长度符合阈值的文本中进行匹配和编辑距离计算。对于长度分布比较均匀的数据集合，这种基于长度限定进行过滤的方法能够大大地降低时间复杂度。

2.2 字命中过滤

对于目标字符串 *str* 中一共有 *N* 个文本。其中一个长度为 *x* 的文本 S_1，假设在长度为 *m* 的模

[1] www.baidu.com
[2] www.google.cn

式串 *pat* 中，有 *h* 个汉字在 S_1 中也出现，那么根据编辑距离的定义有：

$$ED(S_1, pat) \geq \max(m-h, x-h)$$

为了满足 $ED(S_1, pat) \leq k$，有：

$$\begin{cases} h \geq x - \alpha m, & \text{当} x \geq m \text{时} \\ h \geq (1-\alpha)m, & \text{当} x < m \text{时} \end{cases}$$

在实际应用的时候，对 *str* 中的各个文本串按照汉字建立索引，每个汉字对应的索引记录出现过这个汉字的文本编号。当输入特定的 *pat* 时，就可以用很低的时间复杂度得到各个文本的 *h* 值：这个计算实质上是一个索引归并的过程，在我们实际的编程实现中，用一个初始化全为 0 的长度为 *N* 的一维数组来记录每个文本的 *h* 值，然后遍历 *pat* 的对应 *m* 个汉字索引链，每出现一个文本就把数组对应的位置加 1，遍历完成之后，就得到了全部文本的 *h* 值。这个过程时间复杂度是和 *pat* 中 *m* 个汉字的索引链的总长度成正比的。

得到了每个文本的 *h* 值之后，就可以根据 *x* 和 *m* 的大小，只对 *h* 值满足相应条件的文本进行后续的计算。

3 实验

为了验证本文提出的过滤算法在应用时能够保证准确率和召回率，并且满足实际应用的时间复杂度要求，本文设计了如下的实验。

3.1 实验设计

本文实验所用的数据由 Sogou 实验室无偿提供。我们选取了某一周内用户查询次数最多的 10 万条用户查询输入构成待匹配的目标字符串 *str*，同时在另一个时间段的用户日志中随机挑选了 1 万条用户查询，作为 1 万条模式串 *pat*。在实际的实验中，针对不同的阈值 α，我们从 *str* 中查找与当前 *pat* 的相似度满足阈值的所有文本。

实验比较了四种模糊匹配方法，包括：

（1） 文献[15]中介绍的经典的在线计算方法（带有剪枝的动态规划算法和 TU 过滤算法的结合），并将其作为基线；

（2） 长度过滤后再进行在线计算；

（3） 汉字命中过滤后再进行在线计算；

（4） 同时进行长度过滤和汉字命中过滤后再进行在线计算。

本文一共设计了三组实验：实验一主要验证采用两种过滤算法并不会造成召回率的下降；实验二主要考察对于不同的阈值 α，采用不同的过滤算法能够带来的时间性能上的提升；实验三主要考察对于长度不同的 *pat*（*m* 值不同），采用不同的过滤算法能够取得的时间性能上的提升。

3.2 实验结果和分析

在实验一中，我们分别采用四种模糊匹配方法对 1 万条模式串 *pat* 在目标字符串 *str* 中进行模糊匹配。最终四种方法得到的匹配结果完全一样，表明本文提出的过滤算法并不会影响最终的匹配结果。

实验二的结果如图 1 所示。图中横坐为相似度阈值。由于在一般的应用中阈值不会很大，因此在实验中阈值最大取到 $\alpha = 0.7$，两个文本的相似性已经很小了）。图中纵坐标表示计算所用时间占基线算法所用时间的百分比。

直接插入 Excel 图，现在的图太模糊了。图例可以放在图的下面。可以调整图中的点在横坐标刻度上，而不是在中间。

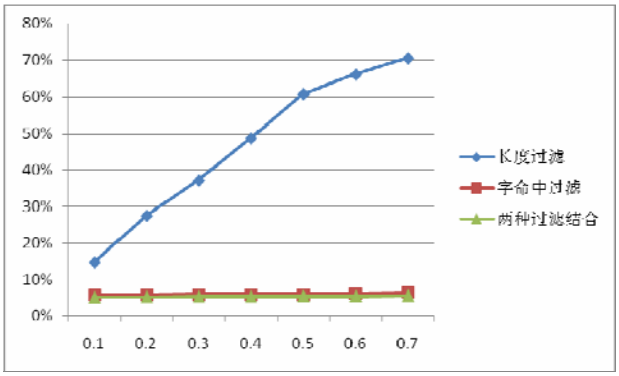

图 1：在不同的阈值下过滤算法的表现

横坐标为相似度阈值，纵坐标为消耗时间相对于无过滤时的比例

从实验二的结果可以看出：

（1） 字命中过滤是一个很有效的过滤算法，所用时间不到基线算法的十分之一。字命中过滤结合长度过滤，可以进一步减少计算的时间。

（2） 随着阈值的增大，长度过滤方法的性能呈下降趋势。这是因为阈值越大，满足公式(1)的长度范围也就越大，因而能够直接过滤掉的文本的数量会减少。

（3） 随着阈值的增大，后两种过滤算法的性能虽然也下降，但是不明显。这是因为，仅命中一个汉字的文本数量非常少，阈值的改变引起符合命中过滤条件的文本数量的变化不大。同时，过滤算法大部分的时间消耗在根据索引归并计算每个文本的汉字命中次数，而索引归并的计算量在不同阈值条件下基本不变。

在实验三中，我们固定相似度阈值为 $\alpha = 0.3$，并对 1 万个 *pat* 按照所含汉字个数进行

统计。实验三结果如图 2 所示。其中，横坐标表明 *pat* 所含的汉字个数，纵坐标与图 1 纵坐标的意义相同。由于长度小于 2 或者大于 12 的 *pat* 非常少，实验不对它们进行统计。

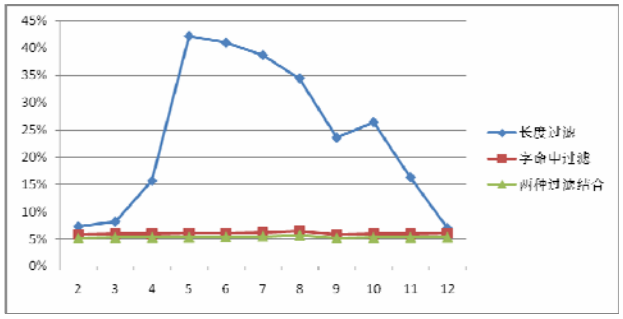

图 2：在不同长度的 *pat* 集合中过滤算法的表现
横坐标为 *pat* 的长度，纵坐标同图 1

从图 2 可以看出，在不同长度的 *pat* 集合中，字命中过滤算法的性能没有大的影响。但是长度过滤的性能随着 *pat* 长度的改变有不同的表现。在 *pat* 不短不长时，由于符合公式(1)的阈值范围的待匹配文本数量较多，所以长度过滤算法的性能较低；而当长度取极限值时，由于满足阈值的待匹配文本数量较少，所以长度过滤算法表现较好。

需要说明的是，在实验二和实验三的时间统计中，我们并没有加上离线建立索引的时间。在实验使用的一台 PC 机（P4 3.0，2G 内存）上，对 10 万条文本建立索引的时间不到 1 秒。而且，这个过程是在开始进行模糊匹配之前离线进行的，对于在线模糊匹配的性能没有影响。

对于查询修正这样的实际应用，待匹配的 *str* 通常较长时间维持不变，我们提出的过滤算法能够取得很好的效果。对于搜索结果聚类来说，先对结果文本建立索引并过滤，再进行聚类，比直接进行成千上万次模糊匹配来过滤，同样能够大大地减少计算时间。但是，对于那些随着 *pat* 改变待匹配文本也变化毕竟频繁的应用来说，每次 *str* 的变化就意味着要进行一次新的索引，从而带来更多的索引时间消耗，因而，实际的时间性能可能会有所下降。

4 结论与未来工作

本文中提出的两种过滤算法，在实验中取得了不错的效果，可以大大加快模糊匹配的速度，从而满足实际在线应用的需求。

本文提出的过滤算法对于待匹配文本数量比较多并且改变不明显的应用有较好的适用性，对于待匹配文本数量较少或者经常剧烈变化的文本，可能不太适用。

另外，本文的过滤算法基于经典的编辑距离计算，对于那些引入了汉字的语音或者语义知识对编辑距离的计算予以扩展的方法，无法直接适用。这一点，也是我们未来工作和改进的重点。

5 致谢

本文在研究工作中，使用了 Sogou 实验室无偿开放的文本数据和用户日志，在此表示感谢。

参 考 文 献

[1] R Baeza-Yates , B Ribeiro-Neto. . Modern Information Retrieve [M] . ACM press ,1999.

[2] 王静帆，邬晓钧，夏云庆等 中文信息检索系统的模糊匹配算法研究和实现 [J].中文信息学报 2007，21(6): 59-64
J. Wang, X. WU, Y. XIA, et al. An Approximate String Matching Algorithm for Chinese Information Retrieval Systems. [J]. Journal of Chinese Information Processing 2007, 21(6):59-64. (in Chinese).

[3] 黄永光，刘挺，车万翔 等. 面向变异短文本的快速聚类算法[A]. 全国网络与信息安全技术研讨会[C]. 北京，2005.
Y. HUANG, T. LIU, W. CHE, et al. Abnormal Short Texts Fast Clustering Algorithm[A]. NETSEC[C]. Beijing, 2005.(in Chinese).

[4] G. NAVARRO A guided tour to approximate string matching [J] . ACM Computing Surveys , 2001 , 33(1) : 31288.

[5] 陈儒. 面向短信过滤的中文信息模糊匹配技术[D] . 哈尔滨: 哈尔滨工业大学信息检索实验室 2003.
R. Chen. Approximate string matching algorithm for cell-phone notes filtering [D]. Information Retrieval Lab, Harbin Institute of Technology. 2003. (in Chinese).

[6] R. Boyer , J. Moore. A fast st ring searching algorithm. [J] . Comm. ACM 1977 20 (10) :7622772.

[7] Horspool N. Practical Fast Searching in St rings. [J] . Software Practice and Experience ,1980 , 10.

[8] E. Ukkonen Algorithms for approximate st ring matching. [J] . Information and Cont rol. 1985a. 64 , 1002118. Preliminary version in Proceedings of the International Conference Foundations of Computation Theory(LNCS , vol. 158 ,1983) .

[9] E. Ukkonen , Finding approximate patterns in strings. [J] . Algor. 1985b ,6 1322137.

[10] Wu. S , Manber. U. Fast text searching allowing errors. [J] . Commun. ACM 35 , 1992 ,10 , 83291.

[11] H Hyyro , G Navarro. Faster bit-arallel approximatest ring matching , [A] . Proc. 13th Combinatorial Pattern Matching (CPM’ 2002) [C] , LNCS , 2002 2 Springer ,23273.

[12] A. Aho, D. Hirschberg, J. Ullman. Bounds on the Complexity of the Longest Common Subsequence Problem. J.ACM 23, 1, 1～12(1976)

An Indexing and Filtering Algorithm for

Approximate Matching of Short Texts in Chinese

CAO Jiang[1,2], WU Xiaojun[2], XIA Yunqing[2], Thomas Fang Zheng[2]
(1. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;
2. Center for Speech and Language Technologies, Division of Technical Innovation and Development,
Tsinghua National Laboratory for Information Science and Technology, Beijing, 100084, China)

Abstract: Approximate matching of short texts is widely used in current Chinese information processing. However the popular approximate matching algorithms cannot satisfy the requirement of real online applications because of the high time complexity. We propose an indexing and filtering algorithm, which includes filtering by length and filtering by characters, to avoid sequential calculation of approximate matching of short texts in Chinese. According to the experiment result, the algorithm can greatly reduces the time cost of the approximate matching tasks while the recall rate does not decrease.

Key words: edit distance; approximate matching; indexing; filtering