文章编号：1003-0077(2007)06-0059-06

中文信息检索系统的模糊匹配算法研究和实现

王静帆，邹晓钧，夏云庆，郑 方

（清华大学计算机系 清华信息科学与技术国家实验室
技术创新和开发部 语音和语言技术中心，北京 100084）

摘 要：在现代中文信息检索系统中，用户输入的字符串和实际数据库中的条目往往存在局部偏差，而基于关键词匹配的检索技术不能很好地解决这一问题。本文参考并改进了 Tarhio 和 Ukkonen 提出的过滤算法[1]，针对汉字拼音输入法中常出现的同音字/近音字混用现象，将算法进一步扩展到广义的 Edit Distance 上。实验表明，本文提出的算法能有效提高中文信息检索系统的召回率，在实际应用中可达到“子线性”的效率。

关键词：计算机应用；中文信息处理；模糊匹配；过滤算法；动态规划

中图分类号：TP391 **文献标识码**：A

An Approximate String Matching Algorithm for Chinese Information Retrieval Systems

WANG Jing-fan, WU Xiao-jun, XIA Yun-qing, ZHENG Fang

(Dept. of Computer Sci. & Tech. Tsinghua University,
Center for Speech and Language Technologies, Division of Technical Innovation and Development,
Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China)

Abstract: In the modern Chinese information retrieval systems, classical keyword based string matching can not work when the input string is different from the entries in the database. This paper proposed a method based on Tarhio and Ukkonen's filtering algorithm to solve the problem. Because the Chinese Pinyin typewriting usually consists Chinese characters with the same or similar pronunciations, we defined a special Edit Distance and expended our method accordingly. The experimental results showed that our algorithm can improve the recall rate of the retrieval systems and obtain practical sub-linear complexity.

Key words: computer application; Chinese information processing; approximate matching; filter algorithm; dynamic programming

1 引言

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[2]。在实际应用中，用户往往凭借印象进行检索，有时只能模糊地描述查询目标，输入的关键词无法和数据集合中保存的数据完全一致；另一方面，在构建数据集时引入的错误（如 OCR 识别错误等）也可能造成这些数据无法被用户获取。在上述情况下，传统的检索系统将难以从数据集中查找到所需要的信息。本文采用模糊匹配方法查找数据集中和用户输入相似的项，并根据相似度排序输出结果，以部分解决上述问题。模糊匹配方法还可以用于其他领域，如入侵检测、信息过滤、基因检测等[3,4]

中文用户大部分使用拼音输入法。用户输入查询串时选词错误造成的同音字替换是很典型的一种现象；方言、发音习惯等造成的音近字替换（如南方方言中，zh 和 z 不分）是第二种典型的错误现象。本文针对这些错误，提出了一种考虑同音字/近音字替换的距离度量方法，在此基础上建立模糊匹配算法。

收稿日期：2007-01-09 **定稿日期**：2007-09-10

作者简介：王静帆（1982—），女，硕士生，研究方向为自然语言处理；邹晓钧（1976—），男，博士，助研，研究方向为口语对话系统和自然语言处理；郑方（1967—），男，博士，教授，研究方向为语音信号处理。

2 背景简介

字符串的模式匹配(精确匹配)问题是：给定目标字符串 *str* 和模式串 *pat*，在 *str* 中寻找 *pat* 的匹配位置。其经典算法有 Knuth-Morris-Pratt (KMP)[5]，Boyer-Moore (BM)[6]等。实际应用中，BM 算法及其改进型(如 BMH[7]等)能达到极高的效率(子线性)，被各种检索系统广泛使用[3,8]。

类似的，字符串的模糊匹配目标是在 *str* 中查找与 *pat* 相似的字串位置。普遍采用 Edit Distance (ED)来刻画两个字符串的距离[3]。设 A，B 为两个字符串，狭义的 ED(A，B)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)或替换(把 A 中的某个字符替换成另一个字符)次数。直观地，两个串互相转换需要经过的步骤越多，相差越多。模糊匹配问题转化为对给定正整数 k，找出 *str* 的所有子串 s'，使得 ED(s'，*pat*)$<k$。

模糊匹配技术的策略主要有以下四种[3]：1. 动态规划[9]，2. 自动机[10]，3. 位平行策略[11,12]，4. 过滤策略[1]。将它们结合使用，常常可以获得更高的时间效率[13~15]。

本文基于 Tarhio 和 Ukkonen 提出的过滤算法[1](TU 过滤算法)，配合带剪枝的动态规划算法[9]，在以下两方面进行了改进：1. 扩展了原有过滤算法，使之能处理更一般的情况；2. 针对中文中特有的同音字/近音字替换问题定义了广义的 Edit Distance，并扩展了过滤算法和动态规划算法以解决该距离度量下的模糊匹配问题。

本文的余下部分安排如下：第 3 节简单介绍动态规划及剪枝算法[9]和 TU 算法[1]；第 4 节介绍我们对 TU 算法的扩展；第 5 节针对同音字/近音字问题定义了广义的 Edit Distance，并提出该距离度量下解决模式匹配的基于 TU 过滤和剪枝动态规划算法；第 6 节给出实验结果和分析，最后是结论。

3 动态规划和 TU 过滤算法

为了更好的说明模糊匹配算法，首先定义可能用到的符号：字符集 Σ 上，目标字符串为 *str*，长度为 n，模式字符串 *pat*，长度为 m。两个匹配串之间的最大 Edit Distance 为 k，错误率上界为 α，$\alpha=k/m$。用 $s[0,\cdots,i]$表示字符串 s 的一个子串，下标由 0 开始。

str 的每个子串可以表示为其前缀子串的后缀子串，动态规划方法[9]计算了 *pat* 的前缀子串 *pat*$[0,\cdots,i-1]$和 *str* 的前缀子串 *str*$[0,\cdots,j-1]$的后缀子串的最小 Edit Distance，记入 $D(i,j)$。

$$D(0,j)=0 \quad (j=0,1,\cdots,n)$$

$$D(i,0)=D(i-1,0)+1 \quad (i=1,2,\cdots,m)$$

$$D(i,j)=\min\{D(i-1,j-1),D(i-1,j),D(i,j-1)\}+1 \quad \text{if } str[j-1]\neq par[i-1]$$

$$=D(i-1,j-1) \quad \text{if } str[j-1]=pat[i-1]$$

$$(i=1,2,\cdots,m,j=1,3,\cdots,n)$$

$D(i,j)$只和 $D(i-1,j)$，$D(i,j-1)$，$D(i-1,j-1)$有关，分别对应于对 *pat*$[i-1]$的删除、插入、替换或匹配操作，计算$(m+1)\times(n+1)$的动态规划矩阵 D，复杂度为 $O(mn)$。

Ukkonen[3,9]证明，在狭义 ED 定义下，D 沿对角线从左上到右下，元素值非严格单调递增。对每列的最后一个元素 $D(m,j)$，如果 $D(m,j)>k$ 说明在该位置上不能找到和 *pat* 匹配的子串。若 $D(i,j)>k$，其具体值不影响后来的计算。记下列 j 中不大于 k 的最后一个元素的位置 *last*，对 $j+1$ 列，只需计算 $D(0,j+1)$，$D(1,j+1)$，$\cdots$，$D(last+1,j+1)$。这种方法被称为 cut-off heuristic 剪枝，把时间复杂度减小到 $O(kn)$。配合 cut-off heuristic 剪枝的动态规划是目前为止最快，也是唯一具有实用价值的基于动态规划的模糊匹配方法。

和动态规划方法不同，过滤算法不能直接计算得到 ED 值，必须配合其他方法使用。分为两个阶段：①在较少的时间消耗下过滤掉大部分无法匹配的位置；②对余下可能匹配的位置调用其他方法定量计算 ED 值。过滤算法目标是在快速过滤掉无法匹配的位置。

TU 算法出发点是用近似方法找到并过滤掉 D 中必然大于 k 的 $D(i,j)$，实现上借鉴了 BMH 算法[7]中从右向左扫描 *str* 串，寻找失败位置，跳跃性移动 *pat* 串，以及通过预处理减少查找时间的思想，通过查找“坏字符”，滤掉 *str* 中 Edit Distance 必然大于 k 的大部分子串位置。

在动态规划矩阵 D 中，记录下每个元素 $D(i,j)$ 的来源(即 $D(i-1,j)$，$D(i,j-1)$，$D(i-1,j-1)$之一)，可跟踪每个 $D(i,j)$的生成路径。$D(m,j)$的生成路径称为 j 上的最小化路径，若 $D(m,j)<k$，该路径称为成功的最小化路径。每条最小化路径对应于一个成功的匹配。Tarhio 和 Ukkonen 证明了：

在 D 矩阵中，成功的最小化路径不能跨越多于 k 条对角线。在此基础上，定义坏字符如下：对 pat 中的位置 i，若字符 $a \notin \{pat[i-k], pat[i-k+1], \cdots, pat[i+k]\}$，把 a 定义为位置 i 上的一个“坏字符”。在成功的匹配中，一个坏字符将造成 Edit Distance 至少增加 1。若一条最小化路径经过对角线 h（设 $D(i, j)$ 对应对角线为 $h=j-i$），则 $str[h, \cdots, h+m-1]$ 中的“坏字符”数不大于 k。若找到 $k+1$ 个坏字符，该位置的匹配失效，跳跃到下一位置；否则调用动态规划方法计算该位置附近子串 $str[h-k, \cdots, h+m-1+k]$ 和 pat 的最小 Edit Distance。

每种匹配方案对应 str 和 pat 的一种对齐方式。一个成功匹配最多允许 k 次操作，所以在 str 中的连续 $k+1$ 个字符中至少有一个等于 pat 中的某个字符，也就是说，考察单个字符 a 在 pat 中出现的位置将可以近似排除 pat 和 str 中不可能的匹配位置，指导 pat“跳跃”到下一个可能匹配的位置。

TU 算法过程如下：

预处理阶段，可通过扫描一次 pat 构建最小跳跃距离表格 d 和坏字符表格 bad。$bad[i,a]$ 指示在 $pat[i]$ 和 $str[j]=a$ 对齐时，a 是否为坏字符；$d[i, a]$ 指示对应情况下，下次可能匹配成功的最大跳跃的长度。

$$bad[i,a]=\begin{cases}\text{false} & \text{if}(\exists l),\\ & st.\ pat[l]=a\\ & \text{and } l-k\leqslant i\leqslant l+k\\ \text{true} & \text{otherwize}\end{cases} \tag{1}$$

$$d[i,a]=\min\{l \mid (l=m) \text{ or } (0<l<m \text{ and } pat[i-l]=a)\}\\ a\in\sum, m-k-1\leqslant i<m \tag{2}$$

字符串比较过滤阶段，在当前位置 h 上把 $pat=pat[0,\cdots,m-1]$ 串和 str 的子串 $s'=str[h,\cdots,h+m-1]$ 对齐（h 初始化为 0），从 $str[h+m-1]$ 开始向左扫描 str 子串。对 $str[h+i]$ $(i=m-1, m-2, \cdots, 0)$，查询 bad 表确定是否坏字符。同时对 $i\in[m-1-k, m-1]$，查表获得 $\text{nextd}=\min\{d[i, str[h+i]]\}$ 作为下个可能匹配的最小跳跃距离。若找到大于 k 个坏字符，说明经过该对角线的所有的“最小化路径”的权值之和均大于 k，不能满足模糊匹配的要求，可以跳到下一个位置。Ukkonen 证明了，当 $2k+1\leqslant m$ 时，若在 $str[h+k,\cdots,h+m-1]$ 这 $m-k$ 个字符中，有多于 k 个坏字符，可以把 pat 向右移动至少 $k+1$ 个位置，即 $\text{nextd}=\max\{\text{nextd}, k+1\}$。

在 α 较小，$|\Sigma|$ 较大的情况下，由于第一阶段快速滤去了大部分不匹配的位置，该方法时间效率能达到“子线性”。

4　改进的 TU 过滤算法

文献[1]证明了，在 $str[h+, \cdots, h+m-1]$ 这 $m-k$ 个字符中，若有多于 k 个“坏字符”，可以把 pat 向右移动至少 $k+1$ 个位置。但是，当 $2k+1>m$ 时，$str[h+k, \cdots, h+m-1]$ 中不可能找到多于 k 个坏字符，原过滤算法不可用。

在这种情况下，我们证明了：若从后向前扫描到的第 $k+1$ 个“坏字符”为 $str[h+u]$ $(0\leqslant u<k)$，则至少可以移动 $u+1$ 个位置，即下一步只需比较 pat 和 $str[h+u+1, \cdots, h+u+m]$。

证明：对坏字符 $str[h+i]$ $(u\leqslant i<m-1)$，由坏字符的定义，$str[h+i]\notin\{pat[i-u], pat[i-u+1], \cdots, pat[i-1]\}$ 所以对角线 $h+1, h+2, \cdots, h+u$ 范围内所有的最小化路径的 Edit Distance 都大于 k。

因此，若在 $str[h, \cdots, h+m-1]$ 中找到 $k+1$ 个坏字符，则下一步的移动距离为 $nextd=\max\{nextd, u+1\}$，至少移到对角线 $h+u+1$ 上，从 $str[j+m+u]$ 开始向左扫描，不会漏解。若在 $str[h, \cdots, h+m-1]$ 中找不到 k 个坏字符，则 $str[h-k, \cdots, h+k+m-1]$ 的某个子串 s' 和 pat 可能满足 $ED(pat, s')\leqslant k$，调用动态规划算法精确计算。这里，最坏的情况下可能出现只移动 1 个位置的情况。

综合以上两种情况，改进后的算法可描述为：令第 $k+1$ 个坏字符的下标为 $h+u$，最小跳跃距离 $nextd\geqslant\min\{k+1, u+1\}$。

5　基于广义 Edit Distance 的扩展

5.1　新的 Edit Distance 定义

给插入、删除、替换以不同的代价，可以得到广义的 Edit Distance[3]。

令对 pat 插入和删除一个字符的代价分别为 c_{ins} 和 c_{del}，替换的代价根据不同的替换字符定义，只需满足距离的一般定义，即：

1. $\text{cost}(a\to b)>0$ （当 $a\neq b$）
2. $\text{cost}(a\to a)=0$

3. $\text{cost}(a\to b)\leqslant\text{cost}(a\to c)+\text{cost}(c\to b)$

且所有代价均大于0小于正无穷。

我们将过滤算法和扩展的TU算法进行了扩展，以处理这种广义的Edit Distance。

5.2 扩展的动态规划剪枝算法

扩展后的递推公式如下：

$$D(0,j)=0 \quad (j=0,1,\cdots,n)$$

$$D(i,0)=D(i-1,0)+c_{del} \quad (i=1,2,\cdots,m)$$

$$\begin{aligned}D(i,j)=\min\{&D(i-1,j-1)\\&+\text{cost}(pat[i-1]\to str[j-1]),\\&D(i-1,j)+c_{del},\\&D(i,j-1)+c_{ins}\\&\}\end{aligned}$$

$$(i=1,2,\cdots,m,j=1,3,\cdots,n) \qquad (3)$$

我们证明了(详细证明见附录A)，在5.1定义下，动态规划矩阵中的主对角线依旧是非严格单调递增，可以采用cut-off heuristic剪枝算法。

5.3 扩展的改进TU过滤算法

接下来，再将改进的过滤算法扩展到广义的Edit Distance意义下。

设 $q=k/min\{c_{del},c_{ins}\}$，与前类似，一条成功的最小化路径不能跨越多于 q 条对角线，因为从当前对角线到达相邻对角线，必然经过水平或者竖直方向的转移，每一步转移的代价不小于 $\min\{c_{del},c_{ins}\}$。成功的最小化路径 h 被限制在以 h 为中心的 $2q+1$ 条对角线范围内。

设 $c_{\min}=\min\{c_{del},c_{ins},\text{cost}(a\to b)\}$，新的“坏字符”定义如下：

设 $str[j]$ 和 $pat[i]$ 对齐，若 $str[j]\notin\{pat[i-q],pat[i-q+1],\cdots,pat[i+q]\}$，把 $str[j]$ 称为该比较对角线上的一个“坏字符”，“坏字符”必然造成该匹配位置上ED值增加至少 $c_{\min}$。若一条成功的最小化路径经过对角线 h，则它在 h 上的“坏字符”数不大于 $k/c_{\min}$。新的算法为：对齐 pat 和 str 的子串 $s'=str[h,\cdots,h+m-1]$，从右向左扫描 s'，若找到 $k/c_{\min}+1$ 个坏字符，该位置的匹配失效，跳跃到下一位置，设第 $k/c_{\min}+1$ 个坏字符的下标为 $h+u$，最小跳跃距离 $\text{nextd}=\min\{q+1,u+1\}$；否则调用动态规划方法计算该位置附近子串 $str[h-q,\cdots,h+m-1+q]$ 和 pat 的最小ED。

特定应用中，可能给某些字符替换定义了很高的相似度，导致 $c_{\min}$ 很小，过滤时必须扫描很多字符，更糟糕的是，无法保证 $k/c_{\min}\leqslant m$，即扫描了整个串都找不到足够的“坏字符”。总之，考虑每个细节将使过滤算法失掉高效的优点。这时可以略微放宽限制，只保证不滤掉可能解，及能滤掉大部分不可能解，把细节的考察放到动态规划评分中。改进思路是：在过滤阶段只选用其中足够大的插入/删除/替换代价计算，把很“相似”的当成相等处理：

$$\text{cost2}(a\to b)=\begin{cases}\text{cost}(a\to b) & \text{if cost}(a\to b)>\text{tcost}\\ 0 & \text{else}\end{cases} \qquad (4)$$

其中，a，b 不全为空。tcost是一个阈值，用来分开“相似”和“不相似”，必须满足 $k/\text{tcost}<m$。tcost过小可能导致过滤阶段时间效率降低，过大可能导致第二阶段的计算量增加。我们认为空字符和任何其他的字符都不相似，在这种假设下，可以令 $\text{tcost}=\min\{c_{ins},c_{del}\}$。这样的替换不会漏掉可能的匹配位置。

5.4 用于同音字/近音字处理

在中文信息检索的实际应用中，大量用户由于使用拼音输入法经常产生同音字/近音字的错误。同音字是现代汉语里语音相同但字形、意义不同的字。所谓语音相同，一般是指声母、韵母和声调完全相同.如“江”和“姜”。近音字是在现代汉语里语音相近，但字形，意义不同的字。在这里我们处理了几种常出现的近音字混用：1.声韵母相同，但声调不同，如“江”和“讲”2.声母、韵母存在差异但发音相近的字，包括前后鼻音混用(ing/in，ang/an，eng/en)，平舌卷舌混用(z/zh，c/ch，s/sh)。在进行模糊匹配时，我们给同音字/近音字替换赋予较小的权重(cost为0.5)，给其他替换和插入删除操作赋予较高的权重(cost为1)，采用以上的过滤算法配合带剪枝的动态算法计算Edit Distance，取得了很好的效果。

6 实验结果

本次实验的数据包括歌曲名称数据库和用户查询日志，实验环境为Windows XP，记录的时间并非程序的绝对运行时间，只是开始和结束时间之差，并未除去系统调度的时间。所以我们只考虑时间的相对意义而不考虑其绝对意义。被检索的是一个歌曲数据集合，包含了600左右歌手/乐队名和5 000左右的歌曲名称。查询词是一段时间内的用户查询日

志，包含大约 200 000 个查询串。

在 $m=6$ 时对三种算法进行对比：精确匹配 BMH 算法，狭义的 Edit Distance 汉字匹配的过滤算法（$\alpha=0.4$，$m=6$），以及带同音字/近音字广义 Edit Distance 扩展的过滤算法（$\alpha=0.4$，$m=6$，取同音字/近音字替换代价为 0.5），后两个算法的第二阶段均采用带剪枝的动态规划算法。实验结果如图 1 所示，对于相同的数据集合，模糊匹配比精确匹配可以检索到更多的条目，而考虑同音/近音字能进一步提高召回率。

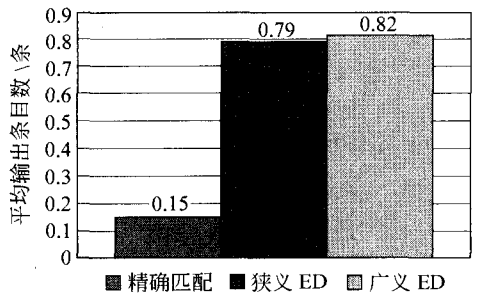

图 1　不同距离度量下的输出比较

接下来，我们对四种算法（在汉字集合上的动态规划和过滤算法，带同音字/近音字的扩展 Edit Distance 下的动态规划和过滤算法）的时间效率进行比较，实验结果如图 2 所示。

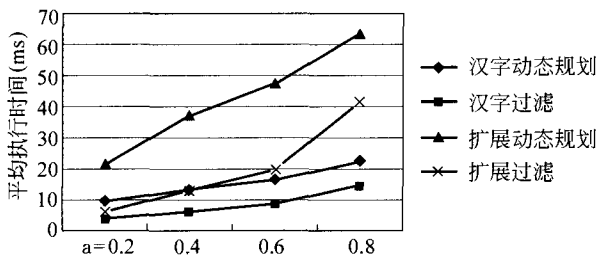

图 2　各算法时间效率比较

实验证明：

1. 两种过滤算法的时间效率都优于动态规划，过滤算法能有效提高时间效率。汉字集合上的过滤算法时间效率最高，因为汉字字符集合比拼音集合大（汉字 20 000 多个，常用的有 4 000 多个，中文音节数量约 400 个），可以滤掉更多字串。因此，在时间要求较高时，可以在过滤时适当放宽 k 值，把对拼音的处理放在第二阶段的动态规划中。

2. 在 α 变大时，动态规划的剪枝效果降低，过滤算法过滤掉的位置也减少，因此两者的耗时都上升。

通过理论分析和实验验证，我们得到以下结论：

动态规划可给出精确的输出，但是耗时太多，不适合直接使用，需要配合过滤算法。我们还把原有过滤算法的适用范围扩展到 $\alpha>0.5$ 的情况，过滤算法可以有效提高时间效率，但是在 α 较大的时候时间性能下降。

7　总结和展望

本文参考并改进了基于 BMH 算法的 TU 模糊匹配过滤算法，用于解决查询系统中用户输入字串和数据库中词条的局部偏差造成的召回率下降；并针对汉字拼音输入法带来的同音字/近音字混用现象定义广义的 Edit Distance，将模糊匹配算法推广到该广义定义上。从理论和实验上证明我们的方法能在有限的时间代价下有效提高检索的召回率。

由于汉字字符集的规模很大，基于汉字实现过滤在时间效率上有相当明显的优势，但空间利用效率不高。使用 Hash Table 可以在一定程度上解决这一问题。

除了同音字/音近字外，本算法还可以对替换的代价作不同的定义，用于处理词义类似，字形类似或错别字混用等检索系统中可能出现的问题。

本文是在线匹配的算法，不能预处理文本建立索引，对于大规模的语料扫描需要耗费大量时间；在 α 很大的时候，过滤效果不明显，时间效率较低。我们将进一步寻找有效的方法解决这些问题。

参考文献：

[1] Tarhio. J, Ukkonen E. Approximate Boyer-Moore string matching. [J]. SIAM Journal on Computing. 1993 22(2):243-260. Preliminary version in SWAT'90 (LNCS, vol. 447, 1990).

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

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

[4] 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.

[5] Knuth D E, Morris J H, Pratt V R. Fast pattern matching in strings. [J]. SIAM Journal on Computing, 1977. 6(2): 323-350.

[6] R. S. Boyer , J. S. Moore. A fast string searching algorithm. [J]. Comm. ACM 1977 20(10) :762-772.

[7] Horspool N. Practical Fast Searching in Strings. [J].

Software Practice and Experience,1980, 10.

[8] GONNET. G, BAEZA-YATES. R. Handbook of Algorithms and Data Structures, 2nd ed[M]. Addison-Wesley. 1991. 251-284.

[9] Ukkonen, E. Algorithms for approximate string matching. [J]. Information and Control. 1985a. 64, 100-118. Preliminary version in Proceedings of the International Conference Foundations of Computation Theory (LNCS, vol. 158,1983).

[10] Ukkonen, E. Finding approximate patterns in strings. [J]. Algor. 1985b,6 132-137.

[11] Wu. S, Manber. U. Fast text searching allowing errors. [J]. Commun. ACM 35, 1992 ,10, 83-91.

[12] Myers, G. A fast bit-vector algorithm for approximate string matching based on dynamic programming. [J]. Journal of the ACM (JACM), 1999,395-415.

[13] G Navarro, M Raffinot. Fast and flexible string matching by combining bit-parallelism and suffix automata, [J]. Journal of Experimental Algorithmics (JEA), 2000,5(4).

[14] H Hyyro, G Navarro. Faster bit-parallel approximate string matching, [A]. Proc. 13th Combinatorial Pattern Matching (CPM'2002) [C], LNCS, 2002 - Springer,23-73.

[15] 陈开渠,赵洁,彭志威. 快速中文字符串模糊匹配算法[J]. 中文信息学报,2004,18(12):58-65.

附录 A

证明：定义新的 Edit Distance 如下：

对 pat 插入和删除一个字符的代价分别为 c_{ins} 和 c_{del}，替换的代价满足：

1. $\text{cost}(a\to b)>0$ （当 $a\neq b$）
2. $\text{cost}(a\to a)=0$
3. $\text{cost}(a\to b)\leqslant \text{cost}(a\to c)+\text{cost}(c\to b)$

且所有代价均大于 0 小于正无穷，则动态规划矩阵中的主对角线依旧是非严格单调递增的。

引理 1：令 S_0, S_1, S_2, …, S_m 为矩阵的一列，则 $S_i-S_{i-1}\in[-c_{ins}, c_{del}]$, $i=1,2,\cdots,m$。

证明：

(1) 对于第 0 列，由定义：

$$D(0,0)=0$$

$$D(i-1,0)-c_{ins}<D(i-1,0)\leqslant D(i,0)=D(i-1,0)+c_{del}\quad(i=1,2,\cdots,m)$$

$$\therefore\quad S_i-S_{i-1}\in[-c_{ins},c_{del}]$$

(2) 对 j 作归纳，假设对第 0, 1, …, $j-1$ 列引理 1 成立($j\geqslant 1$)，则对第 j 列，有：

$$\begin{aligned}D(i,j)=\min\{&D(i-1,j-1)\\&+\text{cost}(pat[i-1]\to str[j-1]),\\&D(i-1,j)+c_{del},\\&D(i,j-1)+c_{ins}\}\\\leqslant\ &D(i-1,j)+c_{del}\end{aligned}\tag{5}$$

$$\begin{aligned}D(i-1,j)=\min\{&D(i-2,j-1)\\&+\text{cost}(pat[i-2]\to str[j-1]),\\&D(i-2,j)+c_{del},\\&D(i-1,j-1)+c_{ins}\}\\\leqslant\ &D(i-1,j-1)+c_{ins}\end{aligned}\tag{6}$$

$$\begin{aligned}D(i,j)=\min\{&D(i-1,j-1)\\&+\text{cost}(pat[i-1]\to str[j-1]),\\&D(i-1,j)+c_{del},\\&D(i,j-1)+c_{ins}\}\\\geqslant\ &\min\{D(i-1,j-1),D(i-1,j)+c_{del},\\&D(i-1,j-1)-c_{ins}+c_{ins}\}\\&\text{(由归纳假设)}\\\geqslant\ &\min\{D(i-1,j)-c_{ins},D(i-1,j)+c_{del},\\&D(i-1,j)-c_{ins}\}\quad\text{(由式(6))}\\=\ &D(i-1,j)-c_{ins}\end{aligned}$$

引理 2：令 T_0, T_1, T_2, …, T_n 为扩展后矩阵中的一行，则 $T_j-T_{j-1}\in[-c_{del},c_{ins}]$, $j=1,2,\cdots,n$。

证法同引理 1。

证明：

由引理 1，$D(i,j-1)\geqslant D(i-1,j-1)-c_{ins}$；

由引理 2，$D(i-1,j)\geqslant D(i-1,j-1)-c_{del}$；

$$\begin{aligned}D(i,j)=\min\{&D(i-1,j-1)\\&+\text{cost}(pat[i-1]\to str[j-1]),\\&D(i-1,j)+c_{del},\\&D(i,j-1)+c_{ins}\}\\\geqslant\ &\min\{D(i-1,j-1),D(i-1,j)+c_{del},\\&D(i,j-1)+c_{ins})\\\geqslant\ &\min\{D(i-1,j-1),D(i-1,j-1)-c_{del}\\&+c_{del},D(i,j-1)-c_{ins}+c_{ins}\}\\=\ &D(i-1,j-1)\end{aligned}$$

因此，在我们定义的 Edit Distance 意义下，沿着矩阵对角线从左上到右下，每个元素的值依然是非严格单调递增的。