基于分词矩阵模型的模糊匹配查重算法研究

李成龙 杨冬菊 韩燕波
(大规模流数据集成与分析技术北京市重点实验室 北京 100144)
(北方工业大学云计算研究中心 北京 100144)

摘 要 针对中文文本查重的需求,利用分词的结果,将待查重的目标文本和查重样本文本转换为分词矩阵模型,然后扫描和分析矩阵,得到查重结果。由此提出了一种查重算法,并通过实例验证了该算法具有一定的实用效果。

关键词 相似度,分词矩阵模型,模糊匹配,查重算法
中图法分类号 TP301 **文献标识码** A

Research on Fuzzy Matching Duplicate Checking Algorithm Based on Matrix Model of Word Segmentation

LI Cheng-long YANG Dong-ju HAN Yan-bo
(Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, Beijing 100144, China)
(Research Center for Cloud Computing, North China University of Technology, Beijing 100144, China)

Abstract Aiming at the need of Chinese text duplicate checking, based on the result of word segmentation, we converted target text and sample text into matrix model of word segmentation, then scanned and analyzed matrix to get the result. Therefore an algorithm of duplicate checking was developed, and the usefulness of the method was demonstrated by practical examples.

Keywords Similarity, Matrix model of word segmentation, Fuzzy matching, Duplicate checking algorithm

1 引言

随着中文信息处理技术的发展,在信息检索、机器翻译、文本查重等领域中,文本相似度的计算作为一个非常基础而又关键的问题被广泛探讨和研究,长期以来也一直是人们研究的热点和难点。相比于计算机对英文的处理,计算机对中文的处理具有更大的难度,主要体现在对中文文本的分词处理上。随着研究的深入,人们越来越发现分词是中文文本相似度计算的基础和前提。

近些年来,网络数据、记录数据以及生物数据等文本数据增长迅速,在文本上进行精确的子串查询是工业界和学术界中的一个常见应用,同时也是子串近似匹配的一个基础操作[1]。这种方式虽然简便且直观,但是得出的结果有一定局限性。例如,通常两段看似重复的文字,用上述字符匹配的方式并不一定能查出,因为大多是通过加上一些无意义的"停止词"(Stopwords)或者将主、谓、宾顺序颠倒等方式来规避这些查重系统的检测,因此传统方法在查全率方面面临了极大的挑战。通过分词方法去掉一些停止词或虚词,是目前逐渐被采用的主流方法。在这种方法下,如何合理地分段和分词,成为提高查全率和查准率需要解决的关键问题。目前用得比较多的是按句分词比对的方式,由于句子长短不一,大部分比较出来的结果虽然在实际中的作用不大,但却会对整体相似比有较大影响。

本文在合理利用分词手段的基础上,提出了一种构造矩阵模型的比对算法和矩阵扫描的策略,使得传统的对文本的比对操作转换成对矩阵的扫描分析工作;同时探讨了与基于空间向量模型算法的优势之处。这一课题的研究以及系统的实现,对于中文信息处理领域的文本相似度计算的比较问题有着一定的参考价值和应用前景。

2 研究问题和相关工作

自然语言处理分析也是学术界和互联网企业的研究重点,中文分词作为中文信息处理的基本环节,近年来重新得到了广泛的关注[2]。随着计算机处理中文信息种类和规模的增加,汉语自动分词技术的研究在自然语言处理、人工智能以及信息检索等领域越来越深入。目前的分词研究仍多基于语言数据进行匹配,耗费大量的存储空间来存储这些语言资源,由分词切分算法对语言资源进行切分[3]。直至今日,自然语言处理问题仍然是学术界研究的热点之一。

本算法涉及文本查重的概念。文本查重是根据文本之间

本文受国家自然科学基金面上项目(61672042),支持流式大数据实时联动的数据服务模型及方法研究资助。
李成龙(1993-),男,硕士,主要研究方向为云存储、分布式计算,E-mail:lichenglong_email@163.com;**杨冬菊**(1975-),女,博士,副研究员,主要研究方向为服务计算、行业资源中心关键技术、创建模式及其在领域中的示范应用,E-mail:yangdongju@ncut.edu.cn;**韩燕波**(1962-),男,博士,教授,CCF会员,主要研究方向为云计算、信息系统集成,E-mail:yhan@ict.ac.cn。

的相似程度，计算两篇或多篇文本之间的重复部分所在的位置，以及该部分在所在文本中所占的比重等参数，将中间结果进行存储、分析、整合的过程。基于大规模文本集统计的方法，通常采用向量空间模型(Vector Space Model，VSM)和隐性语义索引模型(Latent Semantic Indexing，LSI)等方法[4]。目前常见的文本相似度算法有：基于关键词频和逆向文本频率的相似度计算方式(TF-IDF)[5]、基于空间余弦相似度的算法[6]、传统的基于精确匹配的相似度算法[7]、基于距离度量的差异比对算法[8]以及以上算法的改良优化版本。

对于两个文本的相似程度，可以从两个方面来考量，即相似性和差异性。因此，现在常见的度量计算文本相似程度或者差异的方式大体分为两种：距离度量(Distance)和相似度度量(Similarity)。

距离度量用于衡量个体在空间上存在的距离，与相似度度量相反，距离越远说明个体间的相似度越小，反之说明相似度越大。常见的距离度量的计算方式包括欧氏距离(Euclidean Distance)[9]、明可夫斯基距离(Minkowski Distance)[10]、曼哈顿距离(Manhattan Distance)[11]、切比雪夫距离(Chebyshev Distance)[12]等。其中欧氏距离是一个通用的距离度量方式，它用于衡量多维空间中各个点之间的绝对距离。由于存在差值运算，因此算法的前提是必须保证特征值间单位的统一。其公式如下：

$$dist(A,B)=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2} \tag{1}$$

相似度度量，即计算个体间的相似程度。相似度度量的值越大，说明个体间的相似度越大，差异越小。常见的计算相似度度量的方法包括皮尔森相关系数(Pearson Correlation Coefficient)[13]、Jaccard 相似系数(Jaccard Coefficient)[14]、Tanimoto 相似系数(Tanimoto Coefficient)[15]等。在众多相似度计算方法中，余弦相似度保持了许多优越的性质[16]。

定义 A,B 两个个体，它们都包含了 N 维特征，分别可以得出两个空间向量，即 $A:\vec{a}=(a_1,a_2,a_3,\cdots,a_n)$，$B:\vec{b}=(b_1,b_2,b_3,\cdots,b_n)$，则 A 与 B 的相似度可以由式(2)计算得出：

$$sim(A,B)=\cos\theta=\frac{\vec{a}\cdot\vec{b}}{|\vec{a}|\cdot|\vec{b}|}=\frac{(a_1,a_2,a_3,\cdots,a_n)\cdot(b_1,b_2,b_3,\cdots,b_n)}{\sqrt{\sum_{i=1}^{n}(a_i)^2}\times\sqrt{\sum_{i=1}^{n}(b_i)^2}} \tag{2}$$

其中，θ 为向量$\vec{a}$与$\vec{b}$的空间夹角。

3 算法原理

本文研究的查重算法，需要两个理论前提作为基础，才能更好地应用于实际。首先是分词技术，再者是相似度计算的方式。

通常的方法主要分为两类[17]：第一类主要基于字典、词库的匹配和词的频度统计，这类方法实用、具体，比较容易实现；第二类方法主要基于句法、语法分析，并结合语义分析，通过对上下文内容所提供信息的分析对词进行定界，这类方法试图让机器具有人类的理解能力，其原理较为晦涩，一般不易实现。对于中文的词法分析、语法分析以及语义分析，目前已经有了大量相关研究，也取得了丰硕的成果，可以满足对汉语的以词为单位的切分要求。

分词技术的流程图如图 1 所示。

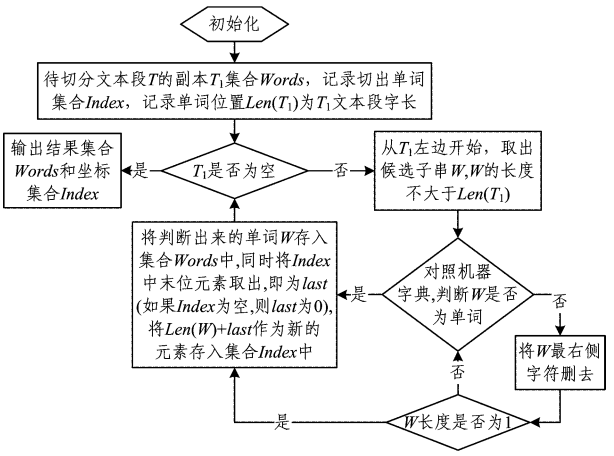

图 1 分词流程图

相似度计算在查重算法中至关重要，直接反映在查重结果的反馈上。将这部分说明如下：一个中文文本段 S 可以按如下的公式表示：

$$S=\{w_1,w_2,w_3,\cdots,w_n\}=\{w_k\mid k=1,2,\cdots,n\} \tag{3}$$

其中，w_k 即 S 中的元素(可以是 w 也可以是 d，与字母无关)，代表分词后的一个单词；n 代表分词后的词的个数；ϕ 代表空文本段，也可以代表一个空的单词；$Len(S)$代表 S 文本段的长度；$Len(w_k)$代表 w_k 单词的长度；$S(k)$代表第 k 个单词 w_k。将 S 的子串定义如下：

$$S(i,L)=\{w_i,w_{i+1},w_{i+2},\cdots,w_{i+L}\}\subset S$$
$$i\geqslant 1,i\leqslant n,L\leqslant n-i \tag{4}$$

其中，L 代表子字符串的长度。假定有两个中文文本段 S_1 和 S_2，并且 $S_1(i,L)\subset S_1$，$S_2(i,L)\subset S_2$，从中文文本段 S_1 中找到一个子串 $S_1(i,L)$，从中文文本段 S_2 中找到一个子串 $S_2(j,L)$，两个子串分别是所有能找到的子串集合中相似度函数值最大的(接近于 1)，且 L 尽可能大。如果子串不等于 ϕ，那么定义文本段 S_1 和 S_2 存在重复，重复位置分别是 i 和 j。计算两个文本段相似度的函数有很多种。由于相似性的比较最终会落脚到字符串中单词的比较上，因此，首先定义一种两个单词之间的比较操作：

$$w_k\otimes w_j=\begin{cases}1, & w_k=w_j\\ 0, & w_k\neq w_j\end{cases},\ w_k\otimes\phi=0,\ \phi\otimes\phi=0 \tag{5}$$

对于两个中文文本段 $S_1=\{w_k\mid k=1,2,\cdots,n\}$和 $S_2=\{d_k\mid k=1,2,\cdots,m\}$(其中 w_k 和 d_k 都代表各自字符串中分词后的单词元素)，定义它们之间的相似度函数为：

$$R(\tau)=\sum_{k=1}^{n}(S_1(k)\otimes S_2(k+\tau))*Len(w_k)=\sum_{k=1}^{n}(w_k\otimes d_{k+\tau})*Len(w_k) \tag{6}$$

其中，$-(m-1)\leqslant\tau\leqslant(n-1)$，$R(\tau)$代表计算出来的一个相似子串的长度，$S_1(k)$代表 S_1 字符串中的第 k 个元素，τ 代表这个相似子串在 S_2 中的偏移量(因为 S_2 的元素个数最多为 m，所以 $0\leqslant k+\tau\leqslant m$ 并且 $0\leqslant k\leqslant n$，经过计算得出 τ 的范围)。

式(6)代表的含义是，设置一个起始偏移量，将 S_1 中的每一个元素(从 1 到 n)都与 S_2 中的偏移量以后的元素进行顺序比对，如果相同，就将这个元素的长度累加到 $R(\tau)$的结果中，并且得出两文本段间的相似度函数定义为 $R(\tau)$的最大值：

$$R(S_1,S_2)=\frac{\max\{R(\tau)\}*2}{Len(S_1)+Len(S_2)} \tag{7}$$

根据偏移量 τ 的变化，可以得出一组不同的结果$R(\tau)$，$\max\{R(\tau)\}$代表这组 $R(\tau)$中最大的一个，因此通过上面相似度函数的计算，可以得出两段文本的相似度。当 S_1 和 S_2 完全相同时，这个函数的值最大：$R(S_1,S_2)=1$。当$R(S_1,S_2)=0$ 时，表明字符串 S_1 和 S_2 完全不同。

4　算法实现

4.1　算法流程

随着分词技术的不断成熟，文件查重系统由以往的以字为单位进行比对查重分析进化为以词为基本单位来进行查重比较和分析，这样，在不破坏原有句子的本意的前提下使得查重工作的结果更加精准。因此，本文对一个基于分词的文件重复率比对的算法(见图 2)进行讨论和分析。

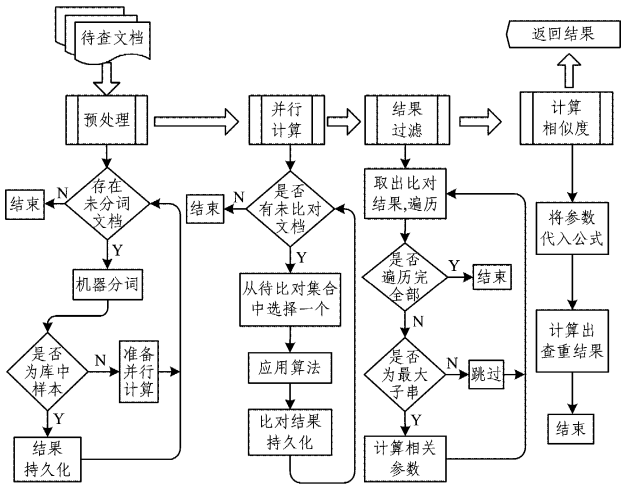

图 2　查重流程

当要查找两篇文章的重复率时，首先根据文本预处理流程将每一篇文章的内容提取出来，并分别进行解析与处理。利用分词方式将自然语言变成一个字符型的数组，并且记录下每个词在文本字符串中的开始位置。这样，便得到了两个开始位置由小到大排列的字符数组。

本系统使用 IKAnalyzer 分词器。IKAnalyzer 分词器采用了特有的“正向迭代最细粒度切分算法”，具有 60 万字/s 的高速处理能力[18]。例如句子：“这是一个中文分词的例子，我爱北京天安门!”经过分词处理以后的结果就是“这是/一个/中文/分词/例子/我/爱/北京/天安门”。本系统将分词结果以及每个单词在文本中出现的位置分别进行持久化，以方便查重算法使用。

当进行查重比对时，首先取出两段文本所对应的分词后的数组。假设这两个数组的长度分别为 m 和 n，这时可以将两个数组分别作为横轴和纵轴，从而形成了一个 $m*n$ 的矩阵 M；并且对于矩阵中的每一个元素，如果“行对应的词”与“列对应的词”相同则置为 1，否则即为 0。例如，对两句话切词以后的结果分别为：1) BDCABA 和 2) ABCBDAB(其中每个英文字母代表一个中文的词)，则构造的矩阵为：

$$\begin{array}{c} \\ \mathrm{A} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{B} \\ \mathrm{D} \\ \mathrm{A} \\ \mathrm{B} \end{array}\begin{array}{c}\begin{array}{cccccc}\mathrm{B} & \mathrm{D} & \mathrm{C} & \mathrm{A} & \mathrm{B} & \mathrm{A}\end{array}\\ \begin{bmatrix}0&0&0&1&0&1\\1&0&0&0&1&0\\0&0&1&0&0&0\\1&0&0&0&1&0\\0&1&0&0&0&0\\0&0&0&1&0&1\\1&0&0&0&1&0\end{bmatrix}\end{array}$$

因此得到的矩阵为：

$$\mathcal{M}=\begin{bmatrix}0&0&0&1&0&1\\1&0&0&0&1&0\\0&0&1&0&0&0\\1&0&0&0&1&0\\0&1&0&0&0&0\\0&0&0&1&0&1\\1&0&0&0&1&0\end{bmatrix}$$

此时，矩阵构造完成。我们在矩阵 M 中寻找一个子方阵 A，这个方阵元素为 a_{ij}，其中 i 和 j 代表方阵元素 a_{ij} 的横坐标和纵坐标，方阵的阶为 k。找到的方阵 A 可以在原始文本中映射为两段文本字段，说明两段文字重复，而映射出的两段文本即为相似查重的一个结果。查找方阵 A 有如下两个原则。

(1)存在一个乘积 $a_{1j_1}*a_{2j_2}*\cdots*a_{kj_k}$ 不为 0(其中 j_1，$j_2,\cdots,j_k$ 是 $1,2,\cdots,k$ 的一个排列)。例如，如果方阵 A 为一个4×4的矩阵，形式如下：

$$A=\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

则存在一个乘积 $a_{13}*a_{22}*a_{34}*a_{41}=1$，则 $j_1,j_2,\cdots,j_k$ 代表的排列就是$\{3,2,4,1\}$，因此方阵 A 满足条件。

如果方阵 A 是一个 4×4 的矩阵，形式如下：

$$A=\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&1&0\end{bmatrix}$$

则存在一个乘积 $a_{11}*a_{22}*a_{34}*a_{43}=1$，则 $j_1,j_2,\cdots,j_k$ 代表的排列就是$\{1,2,4,3\}$，因此方阵 A 满足条件。

如果方阵 A 是一个 4×4 的矩阵，形式如下：

$$A=\begin{bmatrix}1&0&1&0\\0&1&0&0\\0&0&0&0\\1&0&1&0\end{bmatrix}$$

则在所有 j_1,j_2,j_3,j_4 的排列中一定会有 4，因此在 a_{1j1}，a_{2j2}，a_{3j3}，a_{4j4} 中必有一个元素为 0，所以不为 0 的乘积不存在。因此，方阵 A 不满足条件。

(2)这个方阵 A 的维度 k 尽可能大。

如果 M 矩阵的子方阵可以在同一个范围中取得图 3 所示的两种满足条件的形式：

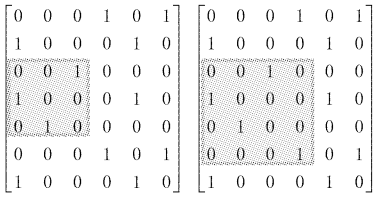

图 3 维度规则说明

由于 A_2 的维度 $k=4$，A_1 的维度 $k=3$，因此选择 A_2 作为更优的结果。

根据常识，当将每一个词都取出在海量文本中进行匹配时，几乎每一个词都被其他文本引用，所以当逐词进行查询时每个词都可以被算作重复的文本，因此需要定义一个词的数量长度，当小于这个长度数量的词出现重复时，应该跳过，不予以判断。例如，设置这个词的最小长度为 $len=10$，即当一篇文章的重复文本长度达到 10 个时，才需要将其算作重复内容，进而计算其重复率。此时，如何高效、迅速地扫描这个矩阵，成为摆在我们面前的另一个难题。

子方阵 A 的寻找方式是这个算法的另一个重点。行列遍历都可以，下面不妨选择列遍历进行说明。

(1)遍历列标 0 到 m，若遍历结束则完成计算。

(2)设置 *begin* 为列标的上界，初始为 0，设置 *end* 为列标的下界，并且取 *end* 为 *begin*＋*len*，如果 *end* 大于 m 则遍历结束。

(3)遍历行标，从 0～n 截取长度为 *end*－*begin* 的两个行标 i 和 j，这就得到了一个子方阵，即第 *begin* 行到第 *end* 行与第 i 列与第 j 列之间围成的方阵即为子方阵，判断这个方阵是否满足原则(1)。若不满足，则将 i，j 增长并继续判断，直到方阵满足原则(1)时将 i 位置记录并进入步骤(4)，如果没有方阵满足条件，则进入步骤(6)。行遍历与列遍历示意图如图 4、图 5 所示。

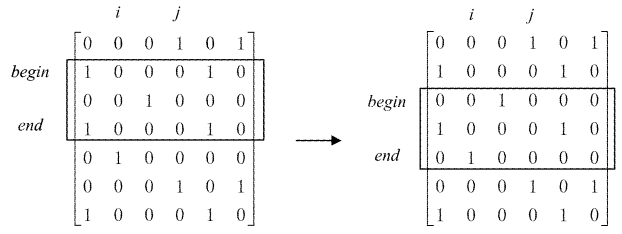

图 4 行遍历

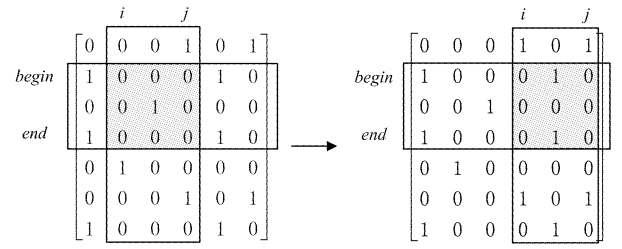

图 5 列遍历

(4)将 *begin* 减 1(最多减 5)，*end* 不变，重新进行步骤(3)的操作，并且将 i 位置记录传入，只在 i 附近寻找方阵 A，以提高遍历的时间效率，由此可以找到开始下标的最小值，进入步骤(5)。向上维度扩展示意图如图 6 所示。

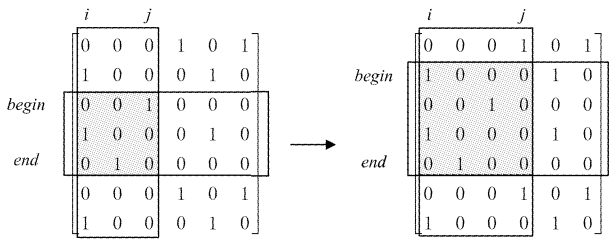

图 6 向上维度扩展

(5)*begin* 不变，将 *end* 加 1(最多加到 m)，重新进行步骤(3)的操作，并且将 i 位置记录传入，只在 i 附近寻找方阵 A，以提高遍历的时间效率，由此可以找到结束下标的最大值。每次找到一个方阵 A 即可以对应一个横轴的词的子集合与一个纵轴的词的子集合。向下维度扩展示意图如图 7 所示。

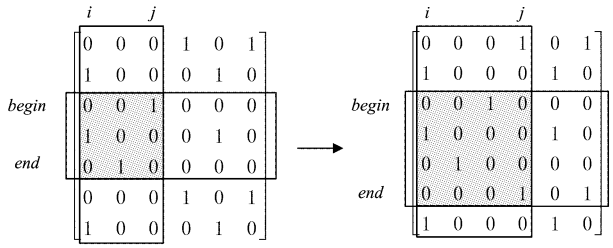

图 7 向下维度扩展

两个子集合以及对应文章的位置就可以对应原文的一段文字，该段文字即为找到的相似的字符串。将中间结果进行持久化。

如图 8 所示，查找出来的一个相似子串即为 $\overline{BDCAB}$ 和 $\overline{CBDAB}$。

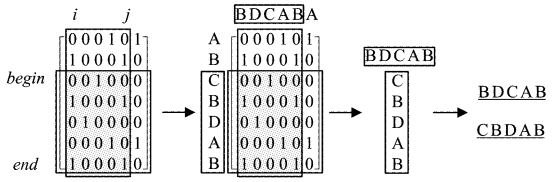

图 8 得到的扫描结果

(6)若 *begin* 与 *end* 间无满足条件的方阵，则令 *begin*＝*end*＋1，进行步骤(2)的操作。

4.2 算法的伪代码实现

由于可以使用并行计算模型将多个查重文档之间的批量查重过程简化到一对一的比对计算，因此本算法就可以针对一对一的文件对比需求来实现一种文件比对方式。需要定义的函数有 compare(S_{one}，$Words_{one}$，$Index_{one}$，S_{two}，$Words_{two}$，$Index_{two}$，$minLen$) 函数，在该函数中调用了 generateMatrix($Words_{one}$，$Words_{two}$)用来构造和生成矩阵、search(M，*begin*，*end*)用来搜索矩阵行区间以及 save(*begin*，*end*，*index*，S_{one}，S_{two})函数用来保存分析出来的结果。下面给出 compare 函数的伪代码实现过程及基于分词矩阵模型的文本查重算法的整体框架。

DEFINITION：

1)S_{one}：需要比对的文本段。

2)$Words_{one}$：S_{one} 分词后的结果。

3)S_{two}：需要比对的一个样本数据。

4）$Words_{two}$：S_{two} 的分词结果。

5）$minLen$：最小重复文本词数。

6）$Index_{one}$：记录了目标 S_{one} 分词后的每个单词的起始位置。

7）$Index_{two}$：记录了样本 S_{two} 分词后的每个单词的起始位置。

```
1. FUNCTION compare(Sone, Wordsone, Indexone, Stwo, Wordstwo, Indextwo, minLen)
2. BEGIN
3.    M=CALL FUNCTION generateMatrix(Wordsone, Wordstwo);
      /* 根据分词结果构造分词矩阵模型 */
4.    begin=0;
      /* 行起始值 */
5.    FOR begin<M. length
6.      end =begin+minLen;
        /* 行结束值，使用 minLen 为单位遍历，使得小于 minLen 的文本段被自动忽略 */
7.      IF end >M. length THEN
        /* 如果遍历行越界，则终止遍历 */
8.      BREAK;
9.      END IF
10.     IF index=CALL FUNCTION search(M, begin, end) !=-1 THEN
        /* 调用 search 函数来扫描行区间内是否存在满足条件的方阵 */
11.       temp_begin=begin-1;
12.       FOR temp_begin>0
          /* 找到满足条件的方阵后，向前寻找最大方阵 */
13.       IF index=CALL FUNCTION search(M, temp_begin, end) ==-1 THEN
          /* 调用 search 函数，遍历行区间 */
14.         begin=temp_begin+1;
15.         BREAK;
16.       END IF
17.       IF temp_begin == 0 THEN
          /* 遍历到上边界时，遍历终止 */
18.         begin=temp_begin;
19.       END IF
20.       temp_begin--;
21.     END FOR
22.     temp_end =end+1;
23.     FOR temp_end< M. length
        /* 找到满足条件的方阵后，向后寻找最大方阵 */
24.       IF index=CALL FUNCTION search(M, begin, temp_end) == -1 THEN
          /* 调用 search 函数来遍历行区间 */
25.         end=temp_end-1;
26.         BREAK;
27.       END IF
28.       IF temp_end == M. length-1 THEN
          /* 遍历到下边界时，遍历终止 */
29.         end=temp_end;
30.       END IF
31.       temp_end++;
32.     END FOR
33.     CALL FUNCTION save(begin, end, index, Sone, Stwo, Indexone, Indextwo);
        /* 调用 save 函数，得出比对结果，并持久化 */
34.   END IF
35.   begin=end + 1;
36.  END FOR
37. END
```

经过以上代码的运算，将所有的待查重文件中的重复文本段持久化到数据库或文件中，再经过结果过滤、相似度计算等步骤就可以得到查重的结果。

5　评价方法

为了更好地测试本文所提算法的效果和时间消耗，使用了传统的字符精确匹配算法[19]与其进行比较。比较的目的是希望从查询所消耗的时间、查询出的直观结果、查询结果的准确率等几个方面，来全面地进行分析并显示出更为直观的结果。本次测试使用的样本源来自实验室合作单位提供的各类申报书等材料，使用了十万字符的文本片段作为测试样本，并将其打散为一一对应的 20 组查重样本，每个组有多份成对的文本片段进行一对一的文本查重测试，将得出的结果进行整理和计算，从而得出最终结果。

下面选取一段具有说明性的文本片段来对比对结果进行直观的说明。

其中样本 A 出自于百度百科对“操作系统”的定义[20]；样本 B 在 A 的基础上进行了一定调整，以保证两段文本有足够的相似度。本文所提算法传统的字符串比对算法的计算结果如图 9、图 10 所示。

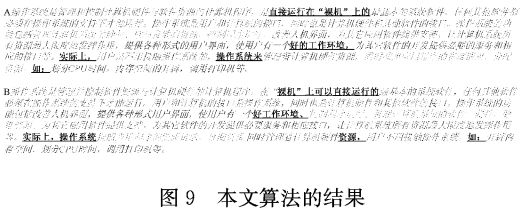

图 9　本文算法的结果

传统基于精确的字符串匹配的算法：

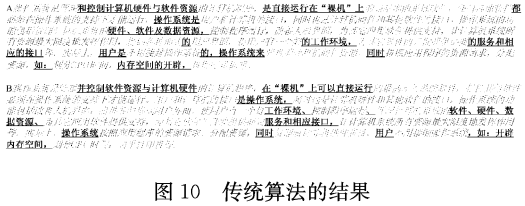

图 10　传统算法的结果

图 9 和图 10 中，带有下划线黑色的部分是未找到的对应相似片段，浅色部分对应的是两段文本中的相似部分。与大多数查重结果一样，本文算法查找到的相似部分多于传统算法查找到的部分，而且本文算法命中的文本片段明显比较长，一共只有 12 个片段，找到了 272 个字，平均每个片段 22.67 个字。而基于字符串匹配的传统算法则一般命中到更小的文本片段，数量达到了 20 个片段，共找到了 222 个字，平均每个片段长 11.1 个字。从图 11 所示统计结果中可以看出，本算法在命中长文本上有更大的优势。

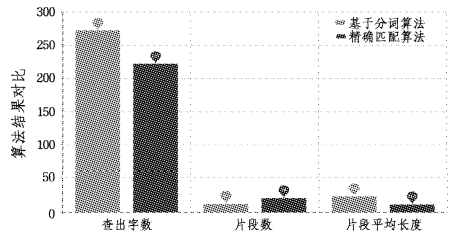

图 11 样本结果

在所用时间上，本文算法耗时 9ms，而传统算法耗时 5ms，相差了 0.004s。而在找到字数与总字数的比值方面，本文算法达到了 90.37%，传统算法仅有 73.75%，说明本文算法可以找到传统算法无法识别的部分。从查询结果（见图 12）来看，本算法与传统算法所查找出的对应文本全部合格，没有出现错查的情况。

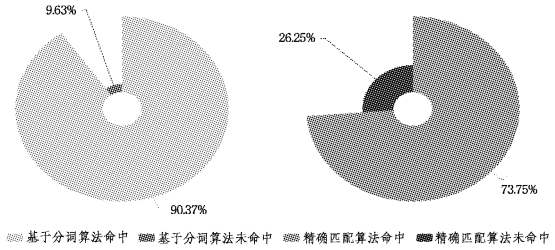

图 12 相似比对比图

在测试算法所消耗的时间方面，我们采用了 20 对文件样本进行多次测试并取平均值的方法获得了一组测试结果，将结果绘制成折线图（见图 13），以展示时间消耗随着样本规模增大的变化情况。

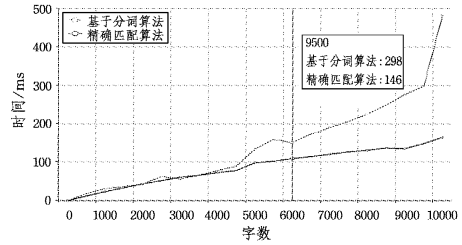

图 13 算法的时间趋势

通过观察结果可以得出，在样本小于 5000 字时，传统算法的时间消耗与本算法的走势几乎相同；而当样本大于 5000 字以后，本算法的耗时就要长于传统算法。经过分析，本算法的时间大部分消耗在词过程中，因此在应用过程中，可以考虑使用“生产者-消费者模式”[21]或者预处理的方式将分词过程与查重比对过程分离，这样会极大地提高算法的效率。

将耗时的工作在“线下”完成，同时将计算和分析后的结果用于“线上”的快速即时响应。比如，人们通过搜索引擎系统可以花费不到一秒钟的时间就可以在数以亿计的种类繁多的网页中找到所需要的信息，如此快速的响应就得益于庞大的线下操作。本文提出的基于分词矩阵模型的模糊匹配查重算法的系统实现，也是借鉴这样的思想，将查重过程进行预处理，即将待查样本库中的文本数据进行格式化并存储起来；并且一次处理，多次使用，极大地降低了查重过程中的时间消耗。

本文提出的基于分词矩阵模型的模糊匹配查重算法的实现仍然有改进的空间。如果将子方阵分析过程中，计算子方阵 A（阶为 k）的元素 a_{ij} 存在一个乘积 $a_{1j_1} * a_{2j_2} * \cdots * a_{kj_k}$（其中 $j_1, j_2, \cdots, j_k$ 是 $1, 2, \cdots, k$ 的一个排列）不为 0，这个过程简化为判断行或列不存在全为 0 的情况，虽然查准率有小幅度下降，但是判断过程的算法时间复杂度就会大幅度地降低，极大地提高查重效率；如在构建矩阵的过程中加入同义词字典的比对分析，那么就可以实现涵盖“语义”的无序文本查重算法，检测结果就会更加全面。

结束语 如今自然语言处理和文本相似度计算得到越来越多的关注，随着技术的不断成熟和发展，随时都会有新思想、新方法产生，同时也会带来更多新的难题需要不断地去攻克。本文在合理利用分词手段的基础上，提出了一种构造矩阵模型的比对算法和矩阵扫描的策略，使得传统的对文本的比对操作转换成对矩阵的扫描分析工作。未来可以在矩阵构造的方式中，结合语义分析，加入同义词字典的比对分析，对实现涵盖“语义”的无序文本查重算法进行一些合理的实验与评估。综上，这一课题的研究以及系统的实现，给中文信息处理领域的文本相似度计算问题提供了一种新的思路和方法。

参 考 文 献

[1] WANG J Y, WANG B, et al. Multi-core Parallel Substring Matching Algorithm Using BWT [J]. Journal of Northeastern University (Natural Science), 2016, 37(5): 624-628.

[2] SONG Y, CAI D F, et al. Approach to Chinese Word Segmentation Based on Character-Word Joint Decoding [J]. Journal of Software, 2009, 20(9): 2366-2375.

[3] ZHANG B Y, WEI B, et al. Chinese word segmentation algorithm based on pair coding [J]. Journal of Nanjing University of Science and Technology, 2014(4): 526-530.

[4] ZHANG P Y, CHEN C M, et al. Texts Similarity Algorithm Based on Subtrees Matching [J]. Pattern Recognition and Artificial Intelligence, 2014(3): 226-234.

[5] HUANG C H, YIN J, et al. A Text Similarity Measurement Combining Word Semantic Information with TF-IDF Method [J]. Chinese Journal of Computers, 2011, 34(5): 856-864.

[6] MAO Y F, ZHANG D L, WANG L. Directional evidence conflict measurement based on improved cosine similarity [J]. Systems Engineering and Electronics, 2016, 38(11): 2567-2571.

[7] FAN H B, YAO N M. A Fast and Exact Single Pattern Matching Algorithm [J]. Journal of Computer Research and Development, 2009, 46(8): 1341-1348.

[8] LIANG J Y, BAI L, et al. K-Modes Clustering Algorithm Based on a New Distance Measure [J]. Journal of Computer Research and Development, 2010, 47(10): 1749-1755.

[9] REN J, LI C P. Improved minimum distance classifier-weighted minimum distance classifier [J]. Journal of Computer Applications, 2005, 25(5): 992-994.

[10] YUAN Y, MA L B. Affine Translation Surfaces in Minkowski 3D-Space [J]. Journal of Northeastern University (Natural Science), 2013, 34(10): 1517-1520.

（下转第 83 页）

（续表）

影响力值排序	因素序号	影响力值	影响力值排序	因素序号	影响力值	影响力值排序	因素序号	影响力值
13	94	0.0111	46	59	0.01	79	66	0.0094
14	14	0.011	47	69	0.01	80	3	0.0093
15	25	0.011	48	78	0.01	81	54	0.0093
16	84	0.011	49	82	0.01	82	55	0.0093
17	35	0.0109	50	9	0.0099	83	65	0.0093
18	89	0.0109	51	11	0.0099	84	70	0.0093
19	15	0.0108	52	13	0.0099	85	90	0.0093
20	31	0.0108	53	24	0.0099	86	12	0.0092
21	38	0.0108	54	77	0.0099	87	45	0.0092
22	19	0.0107	55	80	0.0099	88	61	0.0092
23	72	0.0107	56	97	0.0099	89	96	0.0092
24	1	0.0106	57	33	0.0098	90	29	0.0091
25	23	0.0106	58	43	0.0098	91	49	0.0091
26	32	0.0106	59	47	0.0098	92	76	0.0091
27	74	0.0106	60	60	0.0098	93	91	0.0091
28	6	0.0105	61	62	0.0098	94	99	0.0091
29	7	0.0105	62	81	0.0098	95	51	0.009
30	39	0.0105	63	2	0.0097	96	20	0.0089
31	93	0.0105	64	4	0.0097	97	86	0.0089
32	37	0.0104	65	26	0.0097	98	21	0.0088
33	85	0.0104	66	44	0.0097	99	34	0.0088

结束语　本文将影响网站服务质量的各个因素之间存在的内部关系用矩阵进行表示，迭代计算影响力值，最终得到影响网站服务质量的因素重要性排序，避免了以往层次分析法等方法主观性较强的缺点。实验结果表明，依据影响因素间内部关系计算得到的重要性排序，能反映出各因素对网站的真实影响，对网站服务质量的维护有一定的参考价值。在进行重要性排序时，本文的初始值设为相同，而事实上很多因素的影响初值并不相同。在下一步的研究中，可以结合工程经验对初始值进行设定，避免因为影响力传入数目多而提高其重要性。

参 考 文 献

[1] LANDRUM H, PRYBUTOK V R. A service quality and success model for the information service industry[J]. European Journal of Operational Research, 2004, 156(3): 628-642.

[2] 赵杨. 行业信息中心网站信息服务质量评价[J]. 情报资料工作，2009(6)：32-37.

[3] 邓仲华，汪宣晟，李志芳，等. 信息资源云服务的质量评价指标研究[J]. 图书与情报，2012(4)：12-15.

[4] 刘志亮. 企业 IT 服务质量评价模型及其应用研究[D]. 武汉：华中科技大学，2013.

[5] 卢军. 非量化质量管理体系评价方法初探[J]. 轻工科技，2015(7)：147-148.

[6] PAGE L, BRIN S, MOTWANI R, et al. The PageRank citation ranking: Bringing order to the Web[J]. Stanford Digital Libraries Working Paper, 1998, 9(1): 1-14.

[7] 宫秀文，张佩云. 基于 PageRank 的社交网络影响最大化传播模型与算法研究[J]. 计算机科学，2013(S1)：136-140.

[7] 庞红美，刘宏志. 基于 PageRank 算法的信息工程安全监理风险评估研究[J]. 计算机安全，2014(8)：17-20.

[9] 邵晶晶. 基于 PageRank 排序算法改进的若干研究[D]. 武汉：华中师范大学，2009.

（上接第 60 页）

[11] KE J J, HU J Z. Fault feature extraction method based on Manhattan distance and stochastic neighbor embedding [J]. Application Research of Computers, 2015, 32(10): 2992-2995.

[12] WANG L F, WANG Y, et al. Application of Chebyshev local collocation method to trajectory optimization [J]. Journal of Harbin Institute of Technology, 2013, 45(5): 95-100.

[13] XIE J Y, XIE W X. Several Feature Selection Algorithms Based on the Discernibility of a Feature Subset and Support Vector Machines [J]. Chinese Journal of Computers, 2014, 37(8): 1704-1718.

[14] YU Y Y. Multi-model Estimation Based on Jaccard Distance and Conceptual Clustering [J]. Computer Engineering, 2012, 38(10): 22-26.

[15] YANG H F, LI G J. Novel antenna selection algorithm based on Tanimoto similarity [J]. Journal of Systems Engineering and Electronics, 2008, 19(3): 624-627.

[16] CHEN D L, SHEN Y T, et al. A Measure Model of Similarity for Finding the Best Coach [J]. Journal of Northeastern University (Natural Science), 2014, 35(12): 1697-1700.

[17] WU D, TENG Y P. Word Segment and Search Techniques for Chinese Information Search Engines [J]. Journal of Computer Applications, 2004, 24(7): 128-131.

[18] XIAO W, TANG D K, et al. Knowledge push based on Lucene and collaborative filtering algorithm [J]. Journal of Changchun University of Technology (Natural Science Edition), 2016, 37(5): 503-506.

[19] HE W. The Research for Fast Exact String Matching Algorithm [D]. Hefei: Hefei University of Technology, 2010.

[20] http://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F/192?sefr=enterbt.

[21] WANG Z. Analysis of producer and consumer problem algorithm [J]. Journal of Jilin Province Economic Management Cadre College, 2008, 22(3): 78-81.