文章编号：2095-2163(2019)03-0180-04 中图分类号：TP391 文献标志码：A

基于约束的字符串相似度研究与应用

刘月锟

（青岛理工大学 信息与控制工程学院，山东 青岛 266520）

摘 要：为了提高计算字符串相似度的准确度，分析了字符串相似度计算中准确度难以提高的原因，研究了当前编辑距离计算中存在的问题，对编辑距离计算中替换操作代价进行修订，使编辑距离的计算更加符合实际应用，提出了相似字符串转换的不可逆，说明孤立的字符串难以做到精确匹配，挖掘与字符串密切相关的属性，提出了具有约束的字符串定义，在此基础上改进了莱文斯坦算法，通过对实例数据分析，验证了该方法在基于关系型数据库的应用系统中的有效性。

关键词：编辑距离；字符串相似度；莱文斯坦算法；约束字符串；转换不可逆

Research and application of string similarity based on constrained string

LIU Yuekun

（School of Information and Control Engineering, Qingdao University of Technology, Qingdao Shandong 266520, China）

【Abstract】 In order to improve the accuracy of calculating string similarity, this paper analyses the reasons why it is difficult to improve the accuracy of string similarity calculation, studies the problems existing in the current calculation of editing distance, and revises the cost of replacement operation in the calculation of editing distance. After that, the paper makes the calculation of editing distance more suitable for practical application, puts forward the irreversibility of similar string conversion. It is concluded that the isolated strings are difficult to match accurately. Therefore attributes closely related to strings are mined. A constrained definition of strings is proposed. On this basis, the Levenstein algorithm is improved. The validity of this method in application system based on relational database is verified by the analysis of case data.

【Key words】 edit distance; string similarity; Levenshtein algorithm; constrained string; irreversible transformation

0 引 言

随着数据规模的不断增长，数据共享是各单位间需要迫切解决的关键问题。各单位间数据库结构不同，数据代码差别很大，数据共享主要是基础数据共享，而基础数据中至关重要的就是单位信息。因此关于不同系统中单位名称的成功匹配即已成为一个颇具实用价值的研发项目。不同系统中单位的代码不同，单位名称存在很大差距，有的使用简称或当地约定俗成的名称，要进行单位名称的有效匹配，字符串相似度度量是有效的技术方法。通常，求解字符串相似度的方法可依据不同的特征将其划分为 3 类：基于字面相似的方法、基于统计关联的方法、基于语义相似的方法。其中，基于字面相似的方法中，比较典型的有基于编辑距离的方法[1]。编辑距离广泛应用于字符串近似搜索，与其他度量字符串相似性的算法，如：余弦距离、海明距离、Jaccard 距离等相比，编辑距离能够精确度量 2 个字符串的相似程度[2]。

本文对使用编辑距离度量字符串相似度的方法进行了深入剖析，研究提出了相似字符串转换的不可逆定义，同时证实了这一命题结论，即：将字符串作为孤立个体时，在一定程度上，难以明显提高匹配的准确度，故而本次研究中根据字符串的背景环境，考虑了与字符串密切相关的属性，进而提出了具有约束的字符串定义。在此基础上，参考关系型数据库的知识，通过改进 Levenshtein 算法，实现了单位名称的相似度对比。如此则不但提高了比对效率，而且也改善了比对精度。本文最后，通过社会保险与税务数据共享的实例，也验证了本文方法的良好应用效果。

1 字符串相似度的经典算法

Levenshtein 算法是经典的计算字符串相似度的算法。Levenshtein 算法使用编辑距离计算字符串的相似度，而利用编辑距离计算字符串的相似度，具有结果准确度高，操作简单等优点。对此将做出研究阐释如下。

1.1 编辑距离

编辑距离表示将一个字符串转换为另一个字符

作者简介：刘月锟（1999-），女，本科生，主要研究方向：计算机算法、数据挖掘。

收稿日期：2019-02-26

串所需要的最少编辑次数，编辑是指将字符串中的字符进行插入、删除或替换操作，这个概念由 Levenshtein 于 1966 年提出[3]，而由其一并提出的求编辑距离算法即称为 Levenshtein 算法，简称为 LD 算法。

设原字符串是 s，目标字符串为 t，编辑距离定义为：$distance(t,s)$。求取编辑距离的过程详见如下。

设字符串 t 长度为 n，字符串 s 长度为 m，$t[i]$ 表示字符串 t 的第 i 个元素，$s[i]$ 表示字符串 s 的第 i 个元素，编辑距离矩阵为 $\boldsymbol{ld}[n+1][m+1]$。

初始化 $\boldsymbol{ld}$，$\boldsymbol{ld}[0][0]=0$，$\boldsymbol{ld}[i][0]=i(i=0,1,2,\cdots n)$，$\boldsymbol{ld}[0][j]=j(j=0,1,2,\cdots m)$。

计算矩阵中任意位置的值可表示为：

$\boldsymbol{ld}[i][j]=$

$\min(\boldsymbol{ld}[i-1][j-1]+cost$，// 替换操作

$\boldsymbol{ld}[i-1][j]+1$，// 删除操作

$\boldsymbol{ld}[i][j-1]+1)$// 插入操作

当 $t[i-1]=s[j-1]$ 时，$cost=0$，否则，$cost=1$。

运算后得到的 $\boldsymbol{ld}[n][m]$ 的值就是编辑距离[1-2]。

1.2　字符串相似度

设原字符串为 s，目标字符串为 t，s 的长度为 m，t 的长度为 n，定义 2 个字符串的相似度为：$sim(t,s)$。研究推得其计算公式为：

$$sim(t,s)=\frac{n+m-distance(t,s)}{n+m}. \tag{1}$$

相似度介于[0,1]之间，值越大表示 2 个字符串相似程度越大，值为 1 时表示 2 个字符串完全一致。

1.3　经典相似度算法存在的问题

考虑 2 个例子，设 $t=$“ab”，$s=$“cd”，按照 Levenshtein 算法计算 2 个字符串的相似度为 0.5；设 $t=$“临邑县迎曦大街 896 号”，$s=$“乐陵市湖滨西路 140 号”，按照 Levenshtein 算法计算 2 个字符串的相似度为 0.55；显然，按照字符串字面意义看，这 2 个相似度都不可能达到 0.5。

为了便于分析问题，现在简化字符串模型，假设字符串 t 长度为 n，字符串 s 长度为 m，$n>m$，$n-m=k$，t、s 中相同字符串在 m 长度中满足最大对应，事实上 k 长度的字符串中与 m 相同的字符，可以通过删除、插入操作变换到 t 字符串的前面，如：$t=$“abc”，$s=$“ac”，t 经过删除插入变为“acb”。由 Levenshtein 算法可得如下数学公式：

$$sim=\frac{i+m}{n+m}. \tag{2}$$

此公式的推导过程为：假设字符串 t 长度为 n，字符串 s 长度为 m，$n>m$，$n-m=k$，前 m 个字符中有相同字符 i 个，则字符串 t 和 s 前 m 个字符变为一致，需要 $m-i$ 次替换操作，然后，s 字符串经过 k 次插入，转换为与 t 一致，则编辑距离为：$m-i+k=m-i+n-m=n-i$，所以，$m+n-n+i=m+i$，则相似度即为公式(2)。

当 $n=m$ 且 $i=0$ 时，即 2 个字符串长度相等且没有相同字符时，相似度为 0.5，即使 $i=0$，$sim>0$，相似度永远大于零。

2　经典相似度算法改进

按照公式(2)计算的相似度永远大于零，究其原因就在于求编辑距离时插入、删除、替换为基本操作，地位等同，所产生的花费最大都是 1，但实际操作中，替换操作的花费与插入、删除是不等同的，替换可以分解为删除和插入两个操作，如将字符 a 替换为 b，操作为删除 a、插入 b，字符不同时，$cost$ 应为 2。

2.1　改进的相似度计算公式

改进的相似度计算公式可表示为：

$$sim\_imp=\frac{2i}{n+m}. \tag{3}$$

此公式的推导过程为：字符串模型仍然采用公式(2)的模型，则字符串 t 和 s 前 m 个字符变为一致需要 $2(m-i)$ 次替换操作，而后，s 字符串经过 k 次插入，转换为与 t 一致，则编辑距离为：$2(m-i)+k=2m-2i+n-m=n+m-2i$，所以，$m+n-n-m+2i=2i$，则相似度即为公式(3)。

当 $i=0$ 时，即 2 个字符串没有相同字符时，相似度为 0。

2.2　改进的字符串相似度算法

算法 1　2 个字符串的相似度算法改进

Input：string s and t

Output：$sim\_imp(t,s)$

字符串 s 长度 $m=len(s)$

字符串 t 长度 $n=len(t)$

定义二维数据 $dif[n+1][m+1]$

初始化 $dif[n+1][m+1]$

for $i=1$ to n

　for $j=1$ to m

```
if t[i] = s[j]
  cost = 0
else
  cost = 2
dif[i][j] = min(
  dif[i - 1][j] + 1, // 删除
  dif[i][j - 1] + 1,// 插入
  dif[i - 1][j - 1] + cost) // 替换
return (n + m - dif[n][m])/(n + m)
```

按照算法1计算1.3中2个例子的字符串的相似度,分别为0和0.09,这是与实际情况相符的。

2.3 改进字符串相似度算法的不足

改进后的算法仍然存在不足,在一定地域环境中人们对于单位名称习惯使用单位简称或约定俗成的叫法,这样一来就极容易造成误判,如:“高邑县第一中学”和“赵县第一中学”在本县域范围内可能都习惯使用“第一中学”作为简称,但与“第一中学”计算相似度时,前者的相似度是0.73,而后者的相似度为0.8,如何提高比对准确度即已成为本次课题的研究焦点。

针对字符串相似度度量中提高准确率问题,多年来,一些学者对字符串的相似度发表了很多研究成果。文献[4]根据简称与全称的构成关系,构造简称规则,从而与全称对照,这样虽能缓解一定的问题不足,但同样面临很多困难,如:“西南大学”、“西北大学”都简称“西大”,根据全称到简称规则没有差别,前两个字都代表方位,后两个字都代表专用名词;文献[5]采用分词的方式计算相似度,虽然有一定改进,但也同样遇到了文献[4]的难题;文献[6]基于词向量研究了句子的相似度。文献[7]综合编辑距离和分词等方法,提出了面向用户查询意图的相似度分析方法。文献[8]提出了使用块编辑距离计算字符串相似度的方法,上述方法都试图从字符串本身,寻找提高相似度计算的方法,没有考虑与其他因素的关联关系,但由于相似字符串转换不可逆,总是存在一定的不确定性,导致相似度度量精确度难获实质性提升;文献[9]对中文简称和全称设置置信度,需要大量基础数据源,计算工作量较大。

定义1 相似字符串转换的不可逆性 由全称根据一定规则能转换为简称,用同样规则由简称转换为全称是不可逆的。同样,相似字符串之间的转换也是不可逆的。

显然全称到简称是多对一的关系,就像哈希函数一样,所以其过程是不可逆的,即不能简单由简称推出全称。同样适用字符串相似度比较,虽然从不同研究角度出发,能寻找出提高计算字符串相似度的方法,但这些方法具有不确定性。

3 计算约束字符串相似度的算法

全称-简称转换是不可逆的,但事物并不是孤立存在的,如:在陕西说起“西大”,人们第一印象是“西北大学”,在重庆提到“西大”,人们首先想起的是“西南大学”,简称必然有其密切关联的属性。在企业基础信息的对比中,企业名称、企业地址、所在区域是联系紧密的3个属性。

定义2 具有约束的字符串 将需要比对的元素和与其有紧密关系的属性,组成一个元组,这个元组就是有约束关系的字符串,记为 $T(d_1,d_2,\cdots\cdots,d_n)$,$d_1$ 为被约束字符串,$d_2,\cdots\cdots,d_n$ 为约束属性。

在中文环境的多数情况下,句子只有在一定语义环境中,才有具体含义,句子就是被约束字符串,语义环境就是约束属性。具有约束的元组求相似度时,对应属性分别求相似度,然后合成求出复合相似度,求相似度应用算法2,研究给出设计代码如下。

算法2 具有约束的字符串复合相似度

```
Input:元组 T and S
Output: com_sim(T,S)
com_rat = 1
for i to n
   t = T[i]
   s = S[i]
   rat = sim_imp(t,s)
   com_rat = com_rat * rat
return com_rat
```

4 实例分析

将税务登记信息与社保的参保企业信息作为分析实例,按照算法2设计相似度算法,寻找单位名称的相互匹配数据,使用Python3.6编程,数据库使用MySQL5.7。

需要匹配的字符串为企业名称,名称相似度阈值设为0.75,约束属性为单位地址和行政区划代码,单位地址相似度阈值为0.5,行政区划代码相似度阈值为1,即必须为同一行政区内企业匹配,不是同一行政区内,相似度为0;具有约束的中文字符串复合相似度阈值为0.375。约束字符串记为 $T(name,addr,div)$,其中 $name$ 表示企业名称,$addr$ 表示单位

地址，*div* 代表行政区划代码，*addr* 和 *div* 为约束属性。有约束字符串与无约束字符串总体效率情况见表 1。

表 1　有约束与无约束字符串整体效率

Tab. 1　Constrained and unconstrained string overall efficiency /%

名称	匹配数据量下降	数据准确率
无约束	0	54
有约束	41	73

由表 1 可知，有约束字符串与无约束字符串相比，无效的匹配数据下降了 41%。而匹配的数据准确率由 54%提高到 73%。

4.1　简称相同情况对比分析

选取社保数据中单位名称是“房产管理中心”的单位，无约束相似度匹配结果见表 2，名称相似度用 *name_rat* 表示。

表 2　无约束名称相似度

Tab. 2　Unconstrained name similarity

原名称	匹配名称	*name_rat*
房产管理中心	平原县房产管理中心	0.80
房产管理中心	临邑县房产管理中心	0.80
房产管理中心	齐河县房产管理中心	0.80
房产管理中心	乐陵市房产管理中心	0.80

由表 2 可知，名称的相似度都是 0.80，不能准确地匹配到正确数据。

有约束相似度情况见表 3，名称相似度用 *name_rat* 表示，地址相似度用 *addr_rat* 表示，行政区划相似度用 *div_rat* 表示，复合相似度用 *com_rat* 表示。

表 3　有约束单位名称相似度

Tab. 3　Constrained unit name similarity

原名称	匹配名称	*name_rat*	*addr_rat*	*div_rat*	*com_rat*
房产管理中心	平原县房产管理中心	0.80	0.32	0	0
房产管理中心	临邑县房产管理中心	0.80	0.61	1	0.488
房产管理中心	齐河县房产管理中心	0.80	0.16	0	0
房产管理中心	乐陵市房产管理中心	0.80	0.09	0	0

可以看出具有约束的字符串可以实现精确匹配。

4.2　同一行政区划内对比情况

选取社保数据中单位名称是“宁津县重达汽车配件有限公司”的企业，有约束相似度匹配结果见表 4。

表 4　有约束同一区划内单位名称相似度

Tab. 4　Constrained unit name similarity within the same zone

原名称	匹配名称	*name_rat*	*addr_rat*	*div_rat*	*com_rat*
宁津县重达汽车配件有限公司	宁津县伊高汽车配件有限公司	0.85	0.55	1	0.46
宁津县重达汽车配件有限公司	山东重达汽车配件有限公司	0.80	1	1	0.80

可以看出“山东重达汽车配件有限公司”与“宁津县重达汽车配件有限公司”具有更高的复合相似度，反映了实际情况。但是没有地址约束时，相似度最高的是“宁津县伊高汽车配件有限公司”，这是不正确的。

5　结束语

本文分析了当前计算字符串相似度的 Levenshtein 算法存在的不足，提出了改进方法，给出了相似字符串转换的不可逆定义，指出现在求字符串相似度的局限性，进而得出具有约束属性字符串的概念，不再把字符串看做孤立的个体，而把字符串、约束属性视为密切联系的整体，基于约束字符串改进 Levenshtein 算法，由于关系型数据库中字符串约束属性容易获得，该方法在关系型数据库中求解字符串相似度方面具有很好效果。

参考文献

[1] 姜华，韩安琪，王美佳，等. 基于改进编辑距离的字符串相似度求解算法[J]. 计算机工程，2014，40(1)：222-227.

[2] 张润梁，牛之贤. 基于基本操作序列的编辑距离顺序验证[J]. 计算机科学，2016，43(6A)：51-54.

[3] 马立东. 编辑距离算法及其在英语易混词自动抽取中的应用[J]. 智能计算机与应用，2013，3(1)：47-51.

[4] 沈嘉懿，李芳，徐飞玉，等. 中文组织机构名称与简称的识别[J]. 中文信息学报，2007，21(6)：17-21.

[5] 黄林晟，邓志鸿，唐世渭，等. 基于编辑距离的中文组织机构名简称-全称匹配算法[J]. 山东大学学报(理学版)，2012，47(5)：43-48.

[6] 田星，郑瑾，张祖平. 基于词向量的 Jaccard 相似度算法[J]. 计算机科学，2018，45(7)：186-189.

[7] 李景玉，张仰森，陈若愚. 面向用户查询意图的句子相似度分层计算[J]. 计算机科学，2015，42(1)：227-231.

[8] 王斌，郭庆，李中博，等. 支持块编辑距离的索引结构[J]. 计算机研究与发展，2010，47(1)：191-199.

[9] 郭晖，董源，周钢. 基于属性关联相似度的中文简称匹配算法研究[J]. 计算机与数字工程，2018，46(9)：1726-1730.