软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(10):2674−2692 [doi: 10.13328/j.cnki.jos.005147]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

中文短文本聚合模型研究*

刘 震[1,3], 陈 晶[1], 郑建宾[2], 华锦芝[2], 肖淋峰[1]

[1](电子科技大学 计算机科学与工程学院 互联网科学中心,四川 成都 611731)

[2](中国银联股份有限公司 电子支付研究院,上海 201201)

[3](电子科技大学 大数据研究中心,四川 成都 611731)

通讯作者: 刘震, E-mail: quake@uestc.edu.cn

摘 要: 中文短文本聚合的目的是将两个数据集中属于同一对象的短文本信息进行匹配关联,同时要避免匹配不属于同一对象的短文本信息,这项研究对于多源异构的短文本数据资源整合具有重要的理论和现实意义.提出了一种有效的中文短文本聚合模型,通过快速匹配和精细匹配两个关键步骤可以大幅度降低匹配的候选对数量,并保证匹配的精度.针对传统短文本相似度算法的不足,提出了一种新颖的广义 Jaro-Winkler 相似度算法,并从理论上分析了该算法的参数特性.通过对不同数据集上的商户信息数据进行聚合实验,结果表明,新算法与传统算法相比,在匹配准确率和稳定性上具有最优的性能.

关键词: 中文短文本;聚合模型;文本相似度;广义 Jaro-Winkler 算法;快速匹配;精细匹配

中图法分类号: TP391

中文引用格式: 刘震,陈晶,郑建宾,华锦芝,肖淋峰.中文短文本聚合模型研究.软件学报,2017,28(10):2674−2692. http://www.jos.org.cn/1000-9825/5147.htm

英文引用格式: Liu Z, Chen J, Zheng JB, Hua JZ, Xiao LF. Research on aggregation model for Chinese short texts. Ruan Jian Xue Bao/Journal of Software, 2017,28(10):2674−2692 (in Chinese). http://www.jos.org.cn/1000-9825/5147.htm

Research on Aggregation Model for Chinese Short Texts

LIU Zhen[1,3], CHEN Jing[1], ZHENG Jian-Bin[2], HUA Jin-Zhi[2], XIAO Lin-Feng[1]

[1](Web Sciences Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

[2](Institute of Electronic Payment, China Unionpay Limited Liability Company, Shanghai 201201, China)

[3](Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Aggregation task for Chinese short texts is to associate a pair of similar short texts together. The pair needs to belong to same entity in two data sets. Such study has important theoretical and practical interests for data resource integration across different fields. In this article, an effective aggregation model is devised for Chinese short text. The model is able to decrease the volume of candidate pairs sharply for matching and ensure the matching accuracy via two key steps, namely fast matching and refined matching. Meanwhile, aiming to the deficiency of the traditional similarity algorithms for short text, an improved similarity algorithm, called generalized Jaro-Winkler is proposed. The aggregation experiments performed on different merchant data sets suggest that the new algorithm has the best performance both in matching accuracy and stability compared with those traditional algorithms.

Key words: Chinese short text; aggregation model; similarity of text; generalized Jaro-Winkler; fast matching; refined matching

随着互联网时代的来临,简洁而高效的短文本已成为适应人们快节奏工作与生活的一种重要信息交流载

* 基金项目: 国家自然科学基金(61300018); 中国银联-电子科技大学-金融大数据研究项目
Foundation item: National Natural Science Foundation of China (61300018); China Unionpay-UESTC-Project of Financial Big Data
收稿时间: 2016-03-03; 修改时间: 2016-05-13, 2016-09-07; 采用时间: 2016-10-01

体,这里的短文本通常是指少于 30 个字的文本消息.常见的短文本形式有手机短消息、搜索引擎中的用户查询,即时通信软件的对话消息、新闻标题、用户昵称、网络舆情信息等.伴随着短文本信息的广泛使用,近年来互联网上涌现出了很多针对短文本的聚合应用,比如聚合互联网上各大电子商务网站的商品信息用于商品的价格比较;通过从不同信息平台上聚合商户的相关信息,对商户进行多维度的属性标注,即所谓的用户画像[1];通过聚合不同地理位置(如:旅游景点)的信息数据以丰富地图信息等.这些短文本聚合应用给看似无用的短文本数据赋予了新的价值,因此受到了来自互联网行业专家、数据管理及挖掘领域专家的关注.短文本聚合通常围绕一个对象实体展开,比如以商品价格比较为例:一种手机商品名称为"小米手机 5",通过这个名称可以关联该商品在不同电商网站上的价格,从而可以将不同价格聚合在一起进行比较,方便消费者买到价格优惠的商品.

然而,实际的跨数据集的短文本聚合并没有上述例子那么简单,面临的主要挑战概括如下.

(a) 聚合对象的名称常常存在不一致的问题,比如手机商品在网站 *A* 被称为"小米手机 5",但在网站 *B* 却被称为"第五代小米手机",用简单的模糊匹配方法并不能保证匹配正确.这种聚合对象名称不一致的问题,在多源异构的短文本数据集中普遍存在.

(b) 从文本形式上来看,不同于一般形式比较规范的长文本,聚合对象的名称常常是包含有口语化的词或者简称缩写信息的短文本,这种不规范的特性,使得短文本聚合的难度比长文本更大.

(c) 从信息特征量来看,长文本包含的信息特征较多,可以用统计或机器学习的方法进行有效的聚合,而短文本包含的信息很少,缺乏上下文关系,所以采用常见结合文本特征的相似度算法计算短文本之间的相似度以实现聚合的效果并不理想.比如"小米手机"和"大米手机"在文本特征上看起来是十分相似的,只有一字只差,但其实它们属于两种手机.

(d) 从数据量上来看,互联网短文本的聚合数据量通常在百万数量级以上,所以,如何保证聚合效率也是短文本聚合需要解决的问题.

其中,上述(a)和(c)反映了当前短文本聚合研究面临的两难问题(dilemma issue),即在两个对象数据集中,名称有一定差异但应该配对的对象可能无法正确聚合;而名称非常相似的不同对象却可能被错误地聚合在一起.长期以来,由于不同行业和领域的数据存在封闭性问题,面向跨行业与跨邻域的数据聚合模型国内外研究相对较少,但近年来,随着大数据技术的蓬勃发展,开始逐渐受到研究者的关注[2],特别是 2013 年 3 月 18 日~22 日,在意大利热那亚召开的 EDBT 2013 会议专门举办了一个大数据聚合竞赛,有来自中国、美国、澳大利亚、英国等国的 11 支队伍参赛,其中,邓栋等人采用过滤+验证的框架,并融入"一对多"的快速验证算法、内容过滤机制以及硬件优化技术,在该项比赛中取得了优异成绩,但是,由于相关文献没有提供技术细节供参考,无法和本文提出的聚合模型进行对比[3].聚合任务在国外相关文献中通常被称为"命名实体归一化(named entity normalization)"[4,5]或者"记录连接(record linkage)"[6−8],最近也有国内外研究者将其称为"实体对齐(entity alignment)"[9,10],但这些研究主要针对英文文本.常见的聚合技术归纳起来主要可以分为两类:基于统计方法的聚合模型和基于机器学习的聚合模型.在基于统计的聚合模型中,代表性研究,如 PageRank 算法[11],通过侧重于信息源的权重计算来实现网络资源与主题词的聚合.在基于机器学习的聚合模型中,主要是利用相关的机器学习算法进行资源聚合与模式识别,如 Granitt 等人提出了一种改进的人工神经网络模型来实现信息的聚合[12].上述的数据聚合模型主要集中在多源网络资源的检索聚合上,侧重于数据级和特征级的聚合.本文实现的聚合模型可以将不同领域的中文短文本数据进行内容的聚合.中文短文本数据一般具有特征稀疏性和不规则性的特点,可供统计和提取的特征较少,并且待聚合的数据量大,所以,上述基于传统统计方法的聚合模型和基于机器学习的聚合模型并不适用[13].针对自动短文本聚合面临的技术挑战,本文创新性地提出了一种基于统计方法和中文短文本相似度计算相结合的排序聚合模型.排序聚合模型主要采用快速匹配和精细匹配两个步骤.在快速匹配中,通过基于词频的相似度算法得到初步的排序聚合结果,极大地缩小了匹配范围,再在精细匹配阶段,通过基于字符的相似度算法得到准确的聚合结果.

在中文短文本聚合模型的精细匹配中,通常需要用到文本相似度算法.目前通用的文本相似度算法主要分为两类:基于字符或分词的相似度算法和基于语义的相似度算法.基于字符或分词的相似度算法有 Jaro-Winkler

算法[14,15]、编辑距离(levenshtein distance)算法[16−18]、集合相似度算法[19]、最长公共子串(LCS)算法[20,21]、余弦相似度算法[22,23]等.其中,Jaro-Winkler 算法比较适合短文本相似度的计算[24],编辑距离算法通常被用于输入字符串的快速模糊匹配[25],LCS 算法除了常用于字符串的相似度计算外,近来还在分子生物学领域被应用于 DNA 分子间相似度的比较[26].基于语义的相似度算法一般需要依赖外部语义词典,常见的有基于 HowNet(知网)和基于 WordNet 两种.其中,HowNet 是针对中文的语义词典[27];而 WordNet 是基于英文的语义词典[28].另外,还有一些结合图结构的语义相似度算法,如可计算拓扑方法[29]、随机游走算法[30]等.语义相似度算法虽然有其自身的优势,但本文没有选择语义相似度算法进行中文短文本精细匹配,主要基于如下原因.

(a) 语义相似度主要依据于词汇的预设概念,但目前依赖语言专家预设的概念库通常不能全面覆盖日新月异涌现的各种新词汇;概念相似度计算的准确性也受到语景多样性的影响,存在不稳定性,有时并不能准确反映客观事实,比如在知网的知识库中,“人”和“父亲”“经理”等词的相似度都为 1.

(b) 语义相似度计算通常需要对语义库做实时查询,存在较高的计算复杂度.比如知网中采用的是以义原为单位的树状层次结构,同义词林中也采用了树状的层次体系,计算时不仅需要考虑节点间的距离,还要考虑层次树的深度和区域密度.

(c) 区别于长文本,短文本的特征稀疏性以及上下文信息的缺失,也都会对语义相似度计算的准确性造成较大的负面影响.

本文尝试采用一些经典语义相似度算法实现短文本聚合,发现聚合效果和聚合效率都不能满足解决实际问题的需要.鉴于传统算法存在的不足,通过对基于字符或短语的相似度算法进行比较分析,并结合各种算法的优势和短文本数据的特点,本文提出了一种新颖的相似度算法,经过模型参数的分析,推导出了模型的参数范围并加以证明.最后通过实验验证了新相似度算法相比传统算法具有更高的匹配精确率和更好的稳定性(鲁棒性).

本文第 1 节对中文短文本聚合模型框架进行介绍.第 2 节对精细匹配中常用的相似度算法进行概括说明.第 3 节针对常用相似度算法的不足提出新的相似度算法并分析新算法的参数特性.第 4 节对所有算法的精确性和稳定性进行测试对比.第 5 节总结全文并对未来值得关注的研究方向进行初步探讨.

1 研究框架

本文提出的数据聚合策略主要包含了快速匹配和精细匹配两个步骤,通过这两个步骤可以达到兼顾聚合效率和聚合精度的目的.为了验证聚合效果,本文采用的数据主要来源于中国银联提供的商户数据(后文统称为内部商户)和通过互联网爬取的商户数据(后文统称为外部商户),两个数据集里的对应商户存在 95%以上名称不一致的情况.

1.1 快速匹配

由于本文获取的外部商户的数量在十万到百万之间,所以需要用快速过滤匹配的方式在较短的时间内产生可能的候选对计算集合,为精细匹配缩小匹配范围.王健楠等人提出的键树连接[31]和李国良等人提出的传递连接[32]两种方法以及其他基于编辑距离的变形算法[33]能够有效地实现英文文本的快速筛选配对,但由于中文构词的特殊性,这两种方法并不能有效适用于中文短文本.因此,针对中文短文本,本文提出了一种快速匹配方法,主要包括构建倒排索引[34,35]和选取候选对两个过程.倒排索引的构建过程如图 1 所示.

构建倒排索引需要先对每一个内部商户名称进行分词,得到内部商户名称的分词集合,然后针对分词集合中的每一项找到包含该项的所有外部商户.倒排索引中每一项都包括一个分词以及包含该分词对应的所有外部商户名称,同时,每个商户的编号和分词个数也被记录下来.

选取候选对是指对每一个内部商户名称(例如图 1 中的“红旗连锁超市”),快速得到与之比较相似的排名前 N 个外部商户.其中,外部商户的选取主要是根据词频相似度的大小进行排序得到的.假设 A 是内部商户名称,B_i 是 A 的倒排索引中第 i 个外部商户,统计在倒排索引中,B_i 出现的次数,记为 $count_i$.

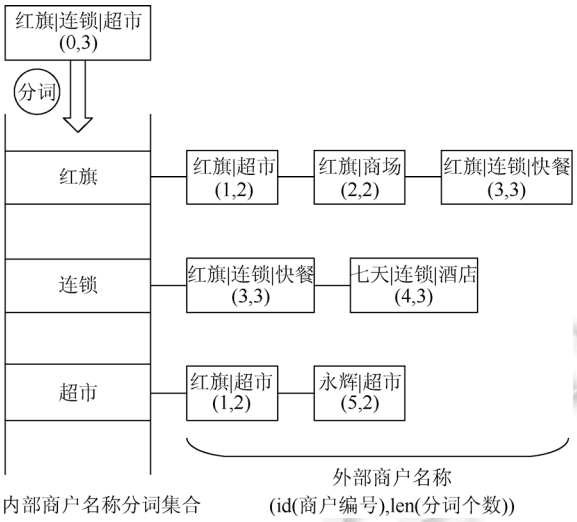

Fig.1 An example of inverted index for short texts

图 1 倒排索引的构建过程

定义 1. A 与 B_i 之间的词频相似度定义为

$$Sim(A,B_i)=\frac{count_i}{\sqrt{len(A)\times len(B_i)}} \tag{1}$$

其中,$len(A)$和 $len(B_i)$分别表示 A 和 B_i 的分词个数.

在图 1 所示的例子中,内部商户 A 的倒排索引总共包含 5 个外部商户,记为 B_1,B_2,B_3,B_4,B_5.根据式(1),可以得到 $Sim(A,B_1)=0.816$,$Sim(A,B_2)=0.408$,$Sim(A,B_3)=0.667$,$Sim(A,B_4)=0.333$,$Sim(A,B_5)=0.408$.最后对相似度按照从大到小排序,可以得到:

$$Sim(A,B_1)>Sim(A,B_3)>Sim(A,B_2)=Sim(A,B_5)>Sim(A,B_4) \tag{2}$$

然后取 top N(这里以 N=2 为例),即取 B_1(红旗超市)和 B_3(红旗连锁快餐)两个外部商户作为内部商户 A(红旗连锁超市)的配对候选.候选集合只保证了能够以极大概率包含外部正确匹配项,但是如果需要更为明确地判断候选集合中各条记录与内部商户是否属于同一个商户,那么还需要采用精细匹配算法.

在本文的聚合实验中,内外部的数据分别有 10 000 条和 370 000 条.如果直接进行精细匹配,需要进行 10000×370 000 次的相似度计算;而我们在快速匹配时,取 top N(N=1000),精细匹配过程中只需要 10000×1000 次的相似度计算.如果不计快速匹配的时间,精细匹配的时间仅为直接匹配的 1/370,并且由于候选对中的 1 000 个外部商户都是与当前内部商户名称最接近的商户,匹配的精确率也不会降低,因此快速匹配过程可以有效提高整个聚合模型的匹配效率.

1.2 精细匹配

在快速匹配获得候选对后,需要用精细匹配最终确定聚合的目标商户.在精细匹配中,主要用到了文本相似度算法.国内外文本相似度算法较多,常用的方法有 Jaro-Winkler 相似度计算、基于编辑距离相似度计算、最长公共子串(LCS)算法、余弦相似度算法等,本文将在第 2 节进行概括介绍.

2 常见的文本相似度算法

2.1 Jaro-Winkler 相似度计算

Jaro-Winkler 算法是 Jaro 算法的变种,是计算两个字符串之间相似度的一种算法.对于待匹配的字符串 s_1 和 s_2,计算匹配窗口 MW(matching window).当两个字符的距离(对应位置)不大于 MW 时,则可认为 s_1 与 s_2 相互

匹配.其中,MW 的计算公式如下式所示.

$$MW=\left\lfloor\frac{\max(|s_1|,|s_2|)}{2}\right\rfloor-1 \tag{3}$$

其中,$|s_1|$和$|s_2|$是两个待匹配字符串的长度.在匹配窗口的范围内,得到所有匹配字符的个数,记为 m.然后,按照每个匹配字符在 s_1 中的先后顺序,构成字符串 ms_1.最后,按照每个匹配字符在 s_2 中的先后顺序,构成字符串 ms_2.若 ms_1 和 ms_2 对应位置上的字符不同,则说明两个匹配的字符发生了换位操作.统计匹配字符中所有的换位操作次数,记为 t_j,则换位的字符数目 t 是发生换位操作的次数的一半,记为

$$t=\frac{t_j}{2} \tag{4}$$

则 Jaro-Winkler 相似度计算公式为

$$d_j=\begin{cases}\dfrac{1}{3}\left(\dfrac{m}{|s_1|}+\dfrac{m}{|s_2|}+\dfrac{(m-t)}{m}\right), & m>0\\ 0, & m=0\end{cases} \tag{5}$$

需要注意的是,对于起始部分存在相同的字符串对,Jaro-Winkler 算法的相似度为

$$d_w=d_j+(lp(1-d_j)) \tag{6}$$

式中,p 为常量,其范围为(0~0.25),默认值为 0.1;l 为字符串对的起始部分存在相同的长度.

2.2 基于编辑距离相似度计算

编辑距离,又称 Levenshtein 距离,是指在两个字串之间,由一个修改成另一个所需要的最少编辑操作次数.许可的编辑操作包括 3 种:将一个字符替换成另一个字符;插入一个字符;删除一个字符.例如,将 *campus* 一词转成 *compute*,需要 3 步.第 1 步将 *a* 替换为 *o*,*compus*($a\to o$);第 2 步将 *s* 替换为 *t*,*comput*($s\to t$);第 3 步插入字符 *e*,*compute*($\to e$).于是,*campus* 与 *compute* 两个字符串之间的编辑距离便为 3.

两个字符串 A,B 的 Levenshtein 距离 $d(A,B)$可以用动态规划算法计算得到[7],则基于编辑距离的相似度计算公式为

$$Sim(A,B)=1-\frac{d(A,B)}{\max(len(A),len(B))} \tag{7}$$

式中,$Sim(A,B)$为最终的相似度计算结果,A,B 为待计算的两个字符串,$len(X)$为 X 的字符个数.

2.3 最长公共子串(LCS)算法

最长公共子串 LCS 算法是将两个给定字符串分别删去 0 个或多个字符,但不改变剩余字符的顺序后得到的长度最长的相同字符序列.对于给定的字符串 A,B,X 是 A 和 B 的最长公共子序列,则是指对 A 和 B 的任意公共子序列 W,都满足$|W|\leqslant|X|$.其相似度计算公式为

$$Sim(A,B)=\frac{2\times len(X)}{len(A)+len(B)} \tag{8}$$

式中,$Sim(A,B)$为最终的相似度计算结果,A,B 为待计算的两个字符串.$len(X)$为 X 的字符个数.

例如:*A=abcdefghifvlhxyzw*,*B=ijkabclmdefgpqh*,它们所匹配到的最长公共子串为 *X=abcdefgh*.分析可知,LCS 算法得到的公共子序列都是严格有序的.由于两个串可以有多个长度相同的最大公共子串,则公共子序列可能是不唯一的.

2.4 短语相似度计算方法

短语相似度算法综合考虑了短语间相匹配文字的位置、匹配位置的偏移值、匹配文字长度等因素.为了便于算法实现,用 A="$A_0A_1\ldots A_{m-1}$"和 B="$B_0B_1\ldots B_{n-1}$"分别表示长度为 m 和 n 的短文本.基于相似度依赖于两个短文本中相同文字位置的关系,定义如下的变量:

$$C(A,i,B)=\{k\mid B_k=A_i,k=0,1,\ldots,m-1\} \tag{9}$$

当 A 的第 i 个文字在 B 中无相匹配文字时,$C(A,i,B)$为空集∅.同时,定义文字 A_i 的最小匹配偏移值为

$$d(A,i,B)=\begin{cases}\min\{|k-i| \mid k\in C(A,i,B)\}, & C(A,i,B)\neq\varnothing\\ n, & C(A,i,B)=\varnothing\end{cases} \tag{10}$$

其相似度计算公式为

$$Sim(A,B)=1-\frac{\sum_{i=1}^{m}d(A,i,B)+\sum_{j=1}^{n}d(B,j,A)}{2mn} \tag{11}$$

2.5 余弦相似度计算方法

用 $A=\{a_0,a_1,\ldots,a_{m-1}\}$,$B=\{b_0,b_1,\ldots,b_{n-1}\}$分别代表待匹配的短文本分词后的词序列,则 A 与 B 的并集记为

$$\mathrm{U}(A,B)=A\cup B \tag{12}$$

将 A 和 B 分别用向量 $\vec{a}$ 和 $\vec{b}$ 表示,如下式所示.如果词元素 a_i 或 b_i 属于集合 U(A,B),则对应向量中的 x_i 或 y_i 记为 1,否则,记为 0.其中,k 是 U(A,B)中元素的个数.

$$\vec{a}=(x_1,x_2,...,x_k) \tag{13}$$

$$\vec{b}=(y_1,y_2,...,y_k) \tag{14}$$

那么,A 和 B 之间的余弦相似度计算公式如下:

$$Sim(A,B)=\cos\langle\vec{a},\vec{b}\rangle=\frac{\sum_{i=1}^{k}x_iy_i}{\sqrt{\sum_{i=1}^{k}x_i^2}\sqrt{\sum_{i=1}^{k}y_i^2}} \tag{15}$$

总的来说,这 5 种算法都较适用于短文本相似度计算.其中,Jaro-Winkler 算法比较适合以字符为单位的计算;Levenshtein 算法、短语相似度算法、LCS 算法和余弦相似度算法比较适合以分词为单位的计算.

3 改进的文本相似度算法

3.1 匹配数据的格式和基本算法的优势和不足之处

在本文实验数据中,内外部商户名称和形式存在不一致问题,可能有以下几种典型情形.

(1) 内部商户是全称,外部商户是简称,比如“义利食品商业连锁有限公司”和“义利食品有限公司”;

(2) 外部商户包含了分店信息,比如“兰皓莎舞蹈培训中心”和“兰皓莎舞蹈培训东单店”;

(3) 内部商户有可以区分的类别词,外部商户没有,比如“都市丽人销售服装店”和“都市丽人”;

(4) 内外部商户名称很相似,但却不是同一商户,比如“乐行国际旅游有限公司”和“乐途国际旅游有限公司”;

(5) 内外部商户名称很相似,但类别词不同,比如“实浩隆超市”和“实浩隆电讯”;

(6) 内外部商户名称字面相似,但顺序不一样,比如“萨亚维瑜伽训练中心”和“维亚萨瑜伽训练中心”.

在上述 6 种情形中,情形 1~情形 3 是正例(名称有差异,但应该被匹配的内外部商户),情形 4~情形 6 是反例(名称相似,但不应该被匹配的内外部商户).

在进行内外部商户精细匹配时,需要针对内外部商户名称按照某一相似度算法计算相似度,然后将结果与一个选择设定的相似度阈值进行比较,如果大于阈值,则认为两个商户是同一商户,否则,判定为不同商户.为了便于研究,所有实验都对最后的相似度结果使用逻辑回归函数[36]进行了压缩映射.

定义 2. 假设内部商户 A 和外部商户 B 之间的相似度为 Sim,则压缩映射函数 y 定义为

$$y=\frac{1}{1+\mathrm{e}^{-Sim}} \tag{16}$$

通过压缩映射函数可以使相似度值区间从之前较大的[0,1]区间压缩到了[0.5,0.73]之间,因此可以缩小判定阈值的搜索范围.注意,压缩映射函数的引入对于提升算法的运算效率具有一定作用,但对于精细匹配的准确

率并没有直接帮助.5 种基本算法对以上 6 种典型情形的匹配情况可见表 1(其中,判别阈值是根据实验数据的匹配结果选取的经验最优阈值.关于最优阈值的讨论,详见第 4.5 节).

Table 1 Results of data matching based on the five algorithms

表 1 5 种算法的匹配结果

情形	算法														
	Jaro-Winkler			Levenshtein			LCS			短语相似度			余弦相似度		
	阈值	结果	是否匹配	阈值	结果	是否匹配	阈值	结果	是否匹配	阈值	结果	是否匹配	阈值	结果	是否匹配
1	0.71	0.71	是	0.66	0.62	否	0.67	0.66	否	0.67	0.67	是	0.68	0.66	否
2		0.72	是		0.63	否		0.66	否		0.66	否		0.66	否
3		0.71	是		0.62	否		0.66	否		0.67	是		0.66	否
4		0.71	是		0.66	是		0.66	否		0.66	否		0.66	否
5		0.69	否		0.67	是		0.67	是		0.67	是		0.67	否
6		0.72	是		0.66	是		0.66	否		0.70	是		0.73	是

观察表 1 可以发现:对于前 3 种正例,除了 Jaro-Winkler 算法以外,其他基本算法匹配效果都不理想;而对于后 3 种反例,所有算法都不能完全匹配正确.

从 Jaro-Winkler 算法的计算原理来看,Jaro-Winkler 算法没有考虑相同字符在原字符串中的间隔问题,所以对情形 4 中名称很相似的两个名称不能正确地拒绝匹配,而且 Jaro-Winkler 算法对位序变化减分权重较低(参见公式(4)中换位字符数目 t 的定义),所以对反例 6 中字符明显错位的情形依然匹配成正例.由于 Jaro-Winkler 算法对前缀相同的部分进行了加分,所以对正例的匹配效果比较好,并且整体得分较高.

从 Levenshtein 算法的计算原理来看,Levenshtein 算法对两个字符串的长短、位序等都很敏感,所以对极相似反例容易正确匹配,而对长度差异稍大的正例容易拒绝匹配.同时,其减分权重较大,导致整体得分偏低.

从 LCS 算法的计算原理来看,LCS 算法保证了两个字符串相同的部分相对有序,所以对位序有变化的反例(反例 6)识别效果较好;但字符串的长度对结果影响较大,所以对长度差异稍大的正例容易拒绝匹配.

从短语相似度算法的计算原理来看,短语相似度算法对两个字符串中相同字符的间隔进行了处理(参见公式(9)、公式(10)),所以对名称很相似的反例 4 的识别效果比较好.但没有考虑到相同字符的位序关系,因此对反例 6 的匹配效果不是很理想.

从余弦相似度算法的计算原理来看,余弦相似度算法考虑了两个字符串的整体长度和相同字符之间的间隔因素,所以对反例 4 有一定的识别能力.由于没有考虑到相同字符之间的位序关系,所以对反例 6 的匹配效果不理想.同时,余弦相似度算法考虑了两个字符串的并集,所以对名称差异较大的正例识别效果也不理想.

3.2 广义 Jaro-Winkler 相似度算法

上节提到,由于起始位置部分相同而加分,Jaro-Winkler 相似度算法对正例匹配效果比较理想.但因为对相同字符位序变化的减分权重较低,并且对相同字符之间的间隔没有处理,所以对反例匹配效果一般.因此,我们通过结合 Levenshtein 算法对相同字符位序变化的处理和短语相似度算法对相同字符之间的间隔的处理,提出一种广义 Jaro-Winkler 相似度算法,具体描述如下.

令 A="$a_0a_1\ldots a_{u-1}$",B="$b_0b_1\ldots b_{v-1}$",其中,$u\leqslant v$,A 代表较短的字符串,B 代表较长的字符串.

定义 3. 假设 A 中第 i 个字符在 B 中有相同字符与之匹配,则该字符在 B 中的相似匹配位置定义为

$$C(A,i,B)=\{k\,|\,b_k=a_i,k=0,1,...,v-1\} \tag{17}$$

注意到 A 中的字符只能在 B 中未匹配的字符中进行匹配,并只返回第 1 个匹配的位置.比如 A="aa",B="$abaa$",则 $C(A,0,B)=0$,$C(A,1,B)=2$.

定义 4. 由定义 3 得到相同字符的匹配位置后,定义匹配成功的第 i 个字符和第 i+1 个字符之间的位置间隔为

$$\Delta(A,i+1,i,B)=C(A,i+1,B)-C(A,i,B) \tag{18}$$

其中,$\Delta(A,i+1,i,B)=1$ 表明相邻的相同字符之间没有间隔;$\Delta(A,i+1,i,B)>1$ 表明相邻的相同字符间至少有一个无关

字符的间隔;$\Delta(A,i+1,i,B)<1$ 表明相同字符之间存在反序.

定义 5. 假设 A 和 B 之间匹配成功的字符总个数为 N,那么 A 和 B 之间的相同字符匹配度 m 定义为

$$m=\begin{cases}1+\sum_{i=0}^{N-2}\dfrac{1}{\Delta(A,i+1,i,B)}, & N>1\\ N, & N=0,1\end{cases} \tag{19}$$

相同字符匹配度 m 综合考虑了相同字符的数量和相同字符之间的间隔两个因素,并且有 $m\leqslant N$ 成立.

定义 6. 针对 A 和 B 匹配成功的字符集,按照每个字符在 A 中的先后顺序,构成字符串 ms_1.同时,按照每个字符在 B 中的先后顺序,构成字符串 ms_2.那么,相同字符之间的序列无关度 t 定义为

$$t=d(ms_1,ms_2) \tag{20}$$

假设字符串 A 的长度为$|s_A|$,B 的长度为$|s_B|$,m 为 A 和 B 的相同字符匹配度,t 为相同字符间的序列无关度,则有如下定义.

定义 7. 字符串 A 和 B 的相似度定义为

$$d_{AB}=\begin{cases}\dfrac{1}{3}\left(\dfrac{m}{|s_A|}+\dfrac{m}{|s_B|}+\dfrac{(m-t)}{m}\right), & m>0\\ 0, & m\leqslant 0\end{cases} \tag{21}$$

我们注意到,如果字符串 A 和 B 的相似度比较高,并且前缀相同的字符越多,则是同一商户命名实体的几率会更大.所以,可通过前缀加分来修正相似度.

定义 8. 假设相同前缀的个数为 l,设置相似度阈值 *threshold* 和调整分数的常数 p.定义修正后的相似度分数为

$$d_w=\begin{cases}d_{AB}+(lp(1-d_{AB})), & d_{AB}\geqslant threshold\\ d_{AB}, & d_{AB}<threshold\end{cases} \tag{22}$$

其中,*threshold* 一般取值为 0.6;p 一般取值为 0.1.

广义 Jaro-Winkler 算法的匹配情况见表 2.

对比表 1 和表 2 可以发现:针对所有特殊情况,广义 Jaro-Winkler 算法都能正确匹配,尤其是对反例的拒绝匹配.由广义 Jaro-Winkler 算法的计算原理可以看出,它既保留了基本 Jaro-Winkler 算法由于起始位置部分相同对正例匹配的优势,又结合了 Levenshtein 算法对相同字符位序变化的处理和短语相似度算法对相同字符间隔处理的优势,所以,广义 Jaro-Winkler 算法更适合本文研究的短文本匹配.

Table 2 Results of data matching based on generized Jaro-Winkler algorithm

表 2 广义 Jaro-Winkler 算法的匹配结果

情形	阈值	结果	是否匹配
1	0.71	0.71	是
2		0.72	是
3		0.71	是
4		0.70	否
5		0.69	否
6		0.62	否

3.3 广义 Jaro-Winkler 相似度算法的参数分析

上一节提出的广义 Jaro-Winkler 算法综合考虑了字符长度、相同字符的间隔和位序之间的关系,使之对短文本的匹配相比其他算法更有优势.由于在实际计算中需要考虑到算法的效率问题,如果可以尽可能地缩小算法中各参数的范围,就能减少很多不必要的计算,从而尽可能地缩短算法的计算时间.基于这个考虑,我们从相同字符匹配度 m、相邻字符位置间隔Δ以及前缀长度 l 这 3 个方面对广义 Jaro-Winkler 算法的参数取值范围进行了分析和研究.

命题 1. 假设待匹配的两个字符串的长度分别为 a 和 $b(a\leqslant b)$,则相同字符匹配度 m 的取值范围为 $\frac{0.8ab}{a+b}\leqslant m\leqslant a$.

证明:广义 Jaro-Winkler 的相似度计算规则由定义 7 给出,根据 Winkler 在其论文中的描述[15],在 $d_{AB}\geqslant 0.6$ 时,两个字符串才有一定的相似性.当 $m>0$ 时,由定义 7 可以得到:

$$\frac{bm^2+am^2+ab(m-t)}{abm}\geqslant 1.8 \tag{23}$$

由 $t\geqslant 0$ 可以得到:

$$m\geqslant \frac{0.8ab}{a+b} \tag{24}$$

又由

$$m\leqslant \min(a,b)=a \tag{25}$$

得到

$$\frac{0.8ab}{a+b}\leqslant m\leqslant a \tag{26}$$

命题 1 得证. □

通过命题 1 对 m 范围的限制,在算法实现时,可以在得到 m 的时候就判断两个商户是否有相似的可能,如果 m 的值在命题 1 的范围内,则进行后续计算,否则,直接拒绝匹配.

引理 1. 假设两个字符串中较长的字符串长度为 b,相同字符数为 N,且相同字符间相对有序,令最大间隔为 Δ,则有 $N\leqslant b-(\Delta-1)$.

证明:令较长的字符串记为 B,由定义 3 可知,$C(A,i,B)$中每个值都是 B 中某个字符的位置,且都是相对有序(从小到大排序)的.假设匹配时从 B 的第 i 个字符开始匹配,中间共有 n 个间隔$\{\Delta_1,\Delta_2,...,\Delta_n\}$,并且一直匹配到 B 的第 j 个位置,则有

$$N=b-\sum_{k=1}^{n}(\Delta_k-1)-i-(b-(j+1)) \tag{27}$$

由式(27)可知,当 $i=0,j=b-1$ 时,N 能取得最大值.

所以有

$$N\leqslant b-\sum_{k=1}^{n}(\Delta_k-1) \tag{28}$$

令$\{\Delta_1,\Delta_2,...\Delta_n\}$中的最大值为$\Delta$,通过式(28)可以得到:

$$N\leqslant b-(\Delta-1) \tag{29}$$

所以引理 1 得证. □

命题 1 约束了 m 的取值范围,引理 1 反映了相同字符数 N 和最大间隔Δ的关系,而定义 5 中隐含了 m 和 N 的关系,结合命题 1、引理 1 和定义 5,我们可以得出最大间隔Δ的范围.

命题 2. 假设待匹配的两个字符串的长度分别为 a 和 $b(a\leqslant b)$,相同字符数为 N,且相同字符间相对有序,令最大间隔为Δ,则有 $\Delta\leqslant\begin{cases}\left\lfloor\frac{b^2+0.2ab}{a+b}\right\rfloor+1, & N>2\\ b-1, & 0\leqslant N\leqslant 2\end{cases}$.

证明:由引理 1 可知,

$$N\leqslant b-(\Delta-1) \tag{30}$$

可以得到:

$$\Delta\leqslant b-N+1 \tag{31}$$

同时,显然有:

$$\Delta\leqslant b-1 \tag{32}$$

所以有:

$$\Delta \leqslant \min(b-N+1, b-1) \tag{33}$$

(a) 当 $b–N+1<b–1$,即 $N>2$ 时,式(31)成立.

由定义 5 可知:$m \leqslant N$,结合引理 1,可以得到:

$$m \leqslant b-(\Delta-1) \tag{34}$$

结合命题 1 中的 $m \geqslant \frac{0.8ab}{a+b}$,可以得到:

$$\frac{0.8ab}{a+b} \leqslant b-\Delta+1 \tag{35}$$

所以可以得到:

$$\Delta \leqslant \frac{b^2+0.2ab}{a+b}+1 \tag{36}$$

而Δ必须是整数,所以有

$$\Delta \leqslant \left\lfloor \frac{b^2+0.2ab}{a+b} \right\rfloor +1 \tag{37}$$

(b) 当 $b–N+1 \geqslant b–1$,即 $N \leqslant 2$ 时,式(32)成立.

所以命题 2 得证. □

命题 1 和命题 2 分别约束了 m 和Δ的范围,而对于短文本的长度,广义 Jaro-Winkler 算法并不能给出明确的定义,但通过前缀的定义,可以对短文本长度进行近似的刻画.

命题 3. 两个字符串的相同前缀长度 l 的最大值为 10.

证明:广义 Jaro-Winkler 相似度算法的最终计算公式 d_w 如定义 8 所示.

由 $d_w \leqslant 1$,可得:

$$d_{AB}+(lp(1-d_{AB})) \leqslant 1 \tag{38}$$

所以有

$$lp \leqslant 1 \tag{39}$$

而 p 的取值为 0.1,所以有 $l \leqslant 10$. □

在这一节,通过对相同字符匹配度 m 和相邻字符位置间隔Δ的分析,我们发现直接计算这两个参数值就能够初步判定两个商户是否可能相似,从而决定是否终止算法.这可以减少大量不必要的计算,在大规模数据计算中具有显著意义.通过对前缀长度 l 的刻画,虽然不能对短文本的长度进行明确的限制,但从侧面大致确定了广义 Jaro-Winkler 算法适用的短文本的长度范围.

4 性能对比测试

我们随机抽取了快速匹配结果中的 10 000 条商户候选对,构成样本测试集,然后针对本文介绍的 6 种相似度算法模型,进行精细匹配测试对比.通过人工检查这 10 000 个样本,定义了 5 类判别标签:X 表示数据异常样本,数量有 1 901 条;O 表示在互联网上找不到对应银联商户记录的样本,数量有 6 358 条;U 表示不确定匹配对错的样本,数量有 148 条;Y 表示正确配对的样本,有 1 217 条;N 表示错误配对的样本,有 376 条.在进行精细匹配测试之前,我们首先评估快速匹配的准确性.

4.1 快速匹配的准确性评估

要验证快速匹配的准确性,需要考察通过快速匹配得到的候选对集合是否覆盖到了目标匹配对象.换言之,如果目标匹配对象存在,则该匹配对象应该出现在候选对集合中.但在实际数据中,比如本文研究的商户数据中,内部数据和外部数据并不能完美匹配,原因是因为内部数据中有较大比例的商户实体在外部数据中并不存在对应的商户,比如内部数据中有一个商户名称为“实浩隆超市”,但我们通过互联网爬取的外部数据中可能并

不存在一个相应的“实浩隆超市”商户.这里,在抽样得到的 10 000 条测试数据中,经过人工检测后发现有多达 6 358 条商户记录没有外部商户可以与之匹配,即上文提到的 O 数据.

为了研究快速匹配的匹配效果,从测试数据集中取 $M(M=1000)$条人工标注为 Y 的内外部商户数据对,挑选这样的数据可以验证通过快速匹配得到的候选匹配数据集是否包含了对应的外部数据.对这 M 条内部数据通过快速匹配分别得到 $N=10,50,100,300,500,1000$ 共 6 个 Top-N 的外部候选数据集,分别记为 S_1,S_2,S_3,S_4,S_5,S_6;然后统计 M 条已知可匹配的外部数据中有多少分别包含在了这 6 个数据中,记为 M_1,M_2,M_3,M_4,M_5,M_6,即统计 Top-N 的外部数据是否覆盖到了需要配对的目标外部商户,而没有出现遗漏.显然,Top-N 的取值越大,越可能覆盖到目标外部商户.为了刻画快速匹配的匹配水平,我们提出匹配率计算公式,定义如下:

$$R_{mat}=\frac{M_i}{M}(i=1,2,3,4,5,6) \tag{40}$$

可以得到快速匹配的匹配率曲线如图 2 所示.

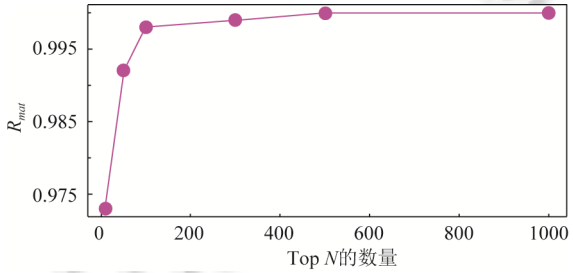

Fig.2 Curve of matching ratio for fast matching

图 2 快速匹配的匹配率曲线

通过图 2 可以发现,快速匹配的匹配水平还是非常高的,其中,在 $N=10$ 的情况下就可以达到 97%的匹配率;随着 N 的增加,匹配率也随之上升;在 $N=500$ 时,匹配率达到了 100%.这个实验结果表明,快速匹配是有效的.

4.2 精细匹配算法的 ROC 曲线

为了刻画 6 种算法精细匹配的准确性,实验进行了 6 种算法的 ROC 曲线的绘制.ROC(receiver operating characteristic curve)曲线[37]是指受试者工作特征曲线/接收器操作特性曲线,是反映敏感性和特异性连续变量的综合指标.在 ROC 曲线上,AUC(area under roc curve,即 ROC 曲线下的面积值)[28]是评价分类效果的一项直观指标.AUC 越大,匹配效果越好.

在实验中,将实例分成真正类(true positive,简称 TP)、假正类(false positive,简称 FP)、真负类(true negative,简称 TN)和假负类(false negative,简称 FN)这 4 种类型.其中,真正类(TP)的数量对应大于阈值的 Y 的样本数量;假正类(FP)的数量对应小于阈值的 Y 的样本数量;真负类(TN)的数量对应大于阈值的 O 和 N 的样本数量;假负类(FN)的数量对应小于阈值的 O 和 N 的样本数量.在此基础上,定义真正类率(true positive Rate,简称 TPR)为

$$TPR=\frac{TP}{(TP+FN)} \tag{41}$$

定义负正类率(false positive rate,简称 FPR)为

$$FPR=\frac{FP}{(FP+TN)} \tag{42}$$

本实验中,以负正类率(FPR)为横坐标,真正类率(TPR)为纵坐标进行 ROC 曲线的绘制.6 种算法的 ROC 曲线如图 3 所示.

观察图 3 可以发现:广义 Jaro-Winkler 算法效果最好,基本的 Jaro-Winkler 算法次之,Levenshtein 算法最差,LCS 算法、余弦相似度算法和短语相似度算法的效果位于两者之间.从曲线的走势来看,改进前后的 Jaro-

Winkler 算法更接近坐标轴的左上角,说明这两种算法有更高的敏感性和特异性.

4.3 精细匹配算法的泛化能力

前面在样本数为 10 000 的条件下,计算了各种算法的 AUC 值.为了研究所有算法的泛化性能[38,39],下面分别从这 10 000 个样本中随机选取 2 000,4 000,6 000,8 000 个样本,并分别计算 6 种算法在不同样本下的 AUC 值.重复以上过程 10 次,求出 10 次实验中 6 种算法的 AUC 均值和标准差,做出 AUC 的均值和标准差随样本规模变化的曲线,如图 4 所示.

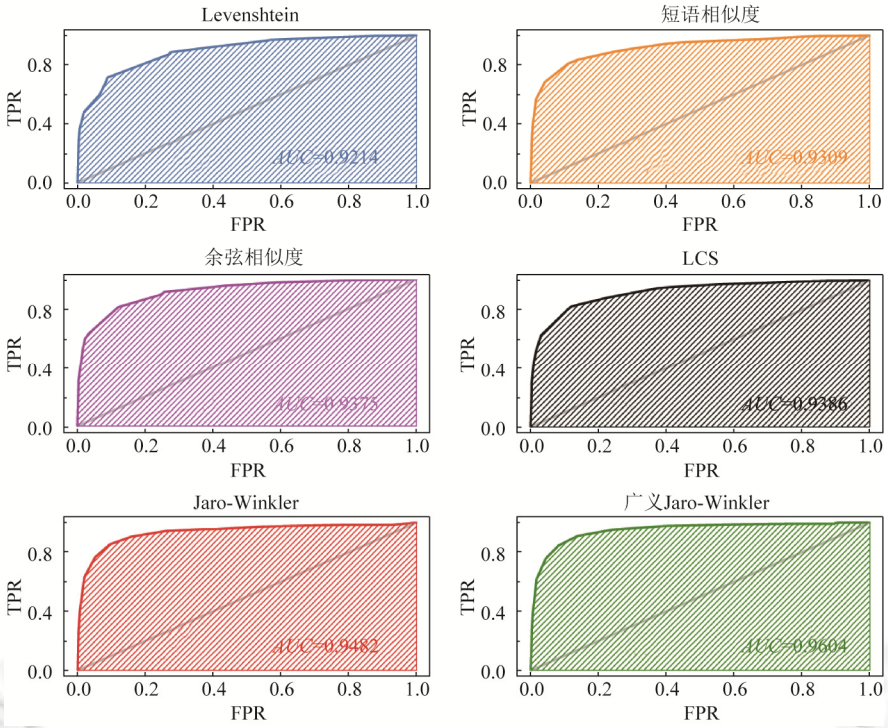

Fig.3 ROC curves for the six algorithms

图 3 6 种算法的 ROC 曲线的变化情况

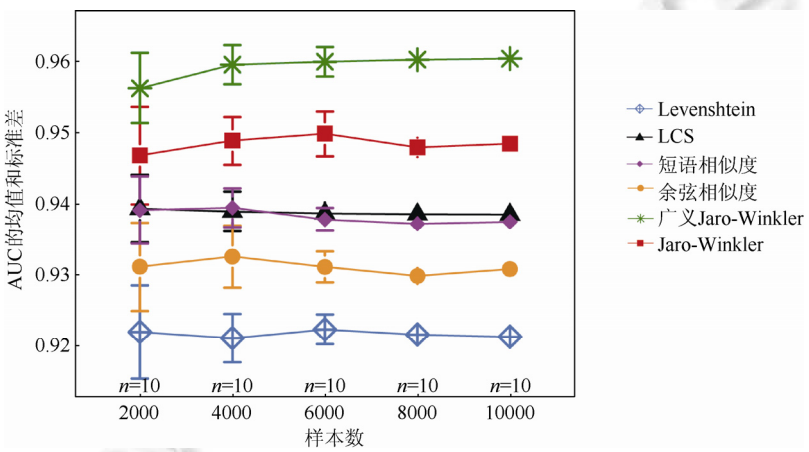

Fig.4 AUC values comparison for the six algorithms

图 4 6 种算法的 AUC 值随样本变化情况

观察图 4 可以发现:从曲线的走势来看,所有 6 种算法模型的 AUC 值随着新样本的增加没有出现太大的起伏,表现均比较稳定,表明这些算法都有较好的泛化性能,没有明显的过拟合问题.从 AUC 的均值来看,广义 Jaro-Winkler 算法在所有样本规模条件下,准确率均高于其他算法,说明该算法整体优于其他传统相似度算法.

4.4 精细匹配算法的稳定性分析

下面我们进一步讨论本文研究的 6 种算法的稳定性.这里主要通过比较匹配算法 F_1 值标准差的信息熵展开稳定性研究.

4.4.1 算法的 F_1 值

与 AUC 指标类似,F_1 值[40]也是评价分类算法有效性的一个常用指标,是精确率和召回率的调和均值.在已知 TP、FN、FP、TN 的前提下,可以定义

精确率:

$$P=\frac{TP}{TP+FP} \tag{43}$$

召回率:

$$R=\frac{TP}{TP+FN} \tag{44}$$

于是可以定义 F_1 值为

$$F_1=2\Big/\left(\frac{1}{P}+\frac{1}{R}\right) \tag{45}$$

即:

$$F_1=\frac{2TP}{2TP+FP+FN} \tag{46}$$

由公式(42)~公式(45)可知,当精确率和召回率都比较高时,F_1 值也较大.

4.4.2 最大熵原理[41]

在信息论和概率统计中,熵(entropy)是表示随机变量不确定性的度量.最大熵原理认为,学习概率模型时,在所有可能的概率模型(分布)中,熵最大的模型是概率分布最均匀的模型.

假设离散型随机变量 X 的概率分布用 $P(x)$表示,其概率分布为

$$P(X=x_i)=p_i,i=1,2,...,n \tag{47}$$

则其熵定义为

$$H(p)=-\sum_{i=1}^{n}p_i\log p_i \tag{48}$$

这里,为了研究相似度算法 F_1 值的稳定性,定义概率 p_i 的取值为不同样本规模条件下进行匹配实验的 F_1 值的归一化标准差.归一化标准差定义为

$$SSD_i=SD_i\Big/\sum_{i}^{n}SD_i \tag{49}$$

其中,SD_i 是第 i 组实验的 F_1 值标准差(每组实验进行 10 次).标准差的熵值越大,表明标准差的分布越均匀,模型也越稳定.并且从熵定义可以验证:

$$0\leqslant H(SSD)\leqslant\log n \tag{50}$$

4.4.3 F_1 值标准差的热力图和熵值曲线图

本文通过分别随机选取 2 000,4 000,6 000,8 000 个样本,并计算 6 种算法在不同样本规模和不同阈值条件时的 F_1 值.重复以上过程 10 次,得到 6 种算法 F_1 值的均值和标准差,F_1 值的均值结果显示,广义 Jaro-Winkler 算法优于其他所有算法,与 AUC 的均值结果一致.这里,为了节约篇幅,仅绘制出标准差分布的热力图如图 5 所示.为了直观地比较 6 种算法的稳定性,进一步分别绘制了在不同阈值和不同样本规模条件下的熵值曲线图.根据最大熵原理,熵值曲线的下面积越大,表明算法的稳定性越好.

6种算法的F_1值的标准差随样本数大小和阈值变化的热力图和F_1值的标准差随阈值和样本规模变化的熵值曲线图如图5所示.

观察图5,6种算法F_1值的标准差热力图分布,可以发现:在样本规模较小时,方差分布的差别较大,但随着样本规模的扩大,所有算法的方差分布都趋于稳定.

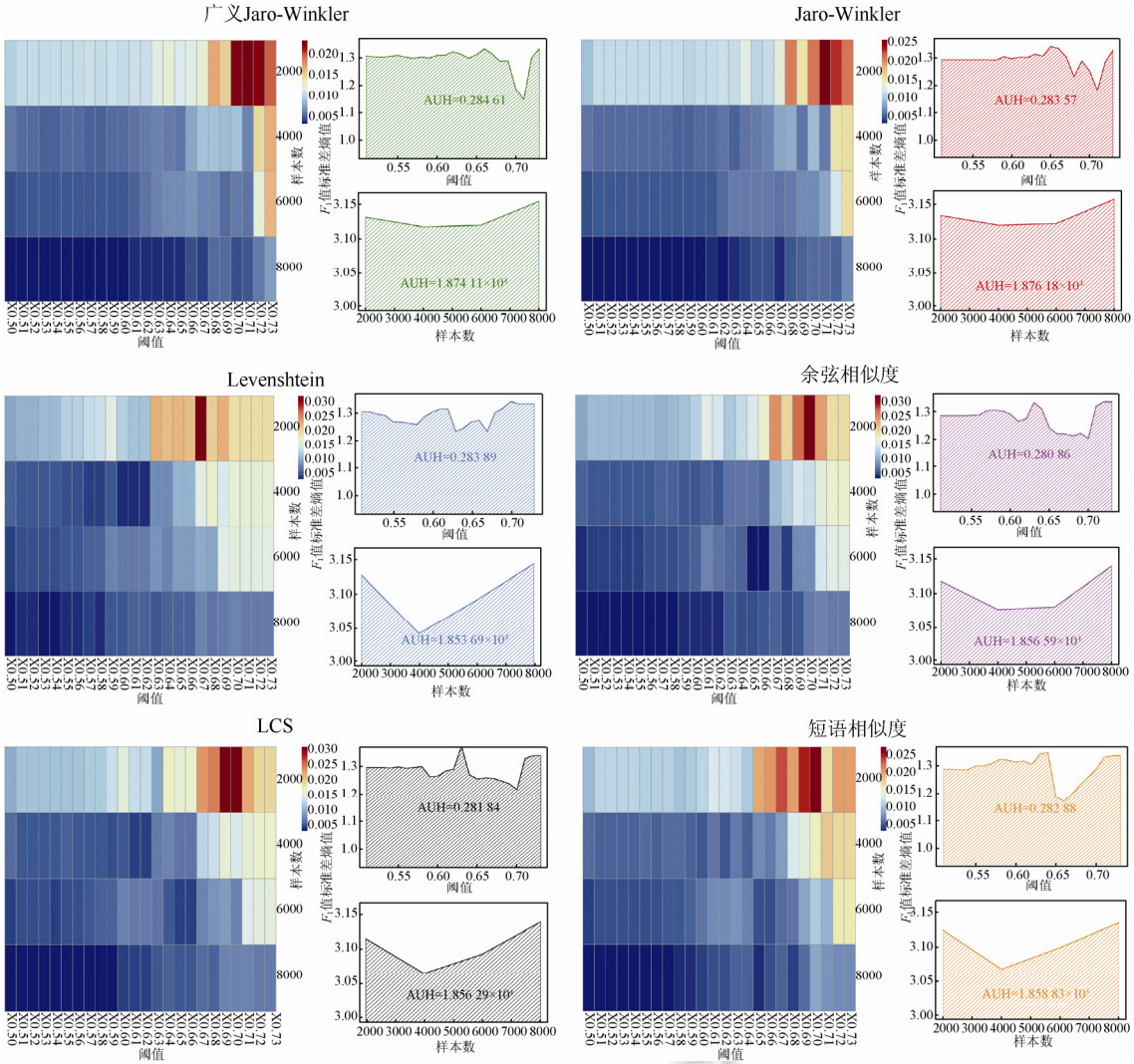

Fig.5 Standard deviation distribution of F_1 value for the six algorithms and their corresponding entropy curves

图5 6种算法的F_1值标准差分布以及熵值随阈值变化情况

为了进一步比较6种算法的稳定性,通过分别计算6种算法F_1值方差的熵值曲线下面积(AUH),可以发现:

(1) 不同阈值条件下,广义Jaro-Winkler>Levenshtein>Jaro-Winkler>短语相似度>LCS>余弦相似度.

(2) 不同样本规模条件下,Jaro-Winkler>广义Jaro-Winkler>短语相似度>余弦相似度>LCS>>Levenshtein.

广义Jaro-Winkler算法在两种情况下的AUH值均排名靠前,根据最大熵原理可以判断,广义Jaro-Winkler算法的稳定性总体要优于其他5种算法.

4.5 精细匹配算法的最优阈值

在实际应用中,判断一个模型的好坏需要结合精确率和召回率两个方面,比如本文测试的商户聚合应用中

通常需要在不低于一定召回率(60%)的基础上具有尽可能高的精确率.基于这个标准,我们发现每种算法的最优判定阈值都会有所不同.

本文的 6 种相似度算法的精确率和召回率随阈值变化曲线如图 6 所示.

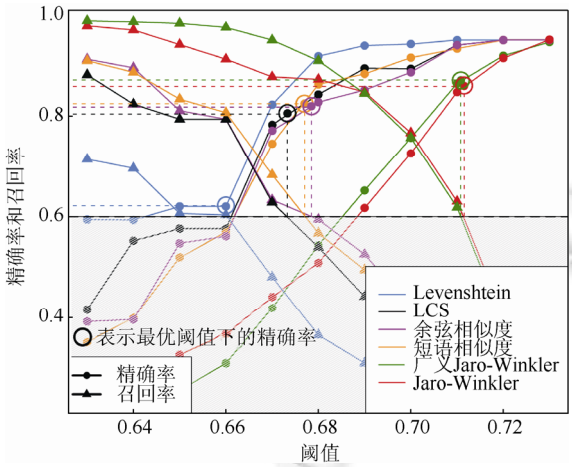

Fig.6 Precison and recall curves for the six algorithms

图 6 6 种算法的精确率和召回率随阈值变化情况

观察图 6 可以发现:在匹配召回率达到 60%时,6 种算法的精确率对比排名次序与 AUC 的对比结果有一定差别,但广义 Jaro-Winkler 算法的表现依然最好.其中,当广义 Jaro-Winkler 算法的最优阈值为 0.711 时,精确率最高(精确率约为 87.1%),紧接着依次是基本 Jaro-Winkler 算法的最优阈值为 0.709(精确率约为 85.9%),短语相似度算法的最优阈值为 0.677(精确率约为 82.5%),余弦相似度算法的最优阈值为 0.679(精确率约为 81.8%),LCS 算法的最优阈值为 0.674(精确率约为 80.4%),最后,Levenshtein 算法的最优阈值为 0.662(精确率约为 62.2%).

通过图 6 还可以发现,这 6 种算法模型的最优阈值的取值相差较大.广义 Jaro-Winkler 算法和基本 Jaro-Winkler 算法的最优阈值靠近坐标轴右侧,在一个较高的区域(0.71 左右);Levenshtein 算法的最优阈值则靠近坐标轴左侧,在相对偏低的区域(0.66 左右);其他 3 种算法的最优阈值则在 0.67 与 0.68 之间且比较接近.

4.6 精细匹配算法的性能分析

为了研究精细匹配算法的性能,我们从两个方面进行了分析和验证.首先从算法复杂度方面进行讨论.假设两条待匹配文本串 A 和 B 的长度分别为 m 和 n,满足 $m\leqslant n$.由于 Levenstein 算法和 LCS 算法均采用了标准动态规划算法,其算法时间复杂度均为 $O(mn)$;余弦相似度算法涉及并集运算表示两个向量,算法的复杂度更高,为 $O((m+n)^2)$;短语相似度算法涉及到两个偏移量的计算,总的计算量为 $2mn$,所以时间复杂度仍然为 $O(mn)$;Jaro-Winkler 算法总的计算量为 $m\times([n/2]-1)+3m+n$,其复杂度依然近似为 $O(mn)$;本文提出的广义 Jaro-Winkler 算法由于引入了限制参数 m(关于参数 m 的讨论,请参见第 3.3 节),复杂度讨论起来比较复杂,主要分两种情况:当参数 m 限制而省略了编辑距离计算时,其算法复杂度为 $O(mn)$;否则,需要计算编辑距离,算法复杂度为 $O(mn+k^2)$,k 为 A 和 B 的公共子串长度.

为了进一步验证本文提出的精细匹配算法的匹配性能,我们进行了匹配时间的对比实验.测试数据集为 8 098 条内部数据和 371 779 条外部数据,对内部数据和外部数据进行快速匹配,分别得到 N=200,400,600,800,1000 这 5 个 Top-N 的数据集,分别记为 S_1,S_2,S_3,S_4,S_5,然后测试 6 种算法在 5 个数据集上的精细匹配处理时间.同时,统计广义 Jaro-Winkler 算法通过限制参数 m(关于参数 m 的讨论,请参见第 3.3 节)在第 i 数据集上提前终止处理的外部数据条数记为 S_i^*,即通过剔出部分无关数据而缩短算法处理时间.这里,剔除率计算公式定义为

$$R_{rej}=\frac{S_i^*}{S_i}(i=1,2,3,4,5) \tag{51}$$

6 种算法在不同数据集上的运行时间以及广义 Jaro-Winkler 算法在不同数据集上的数据剔除率如图 7 所示.

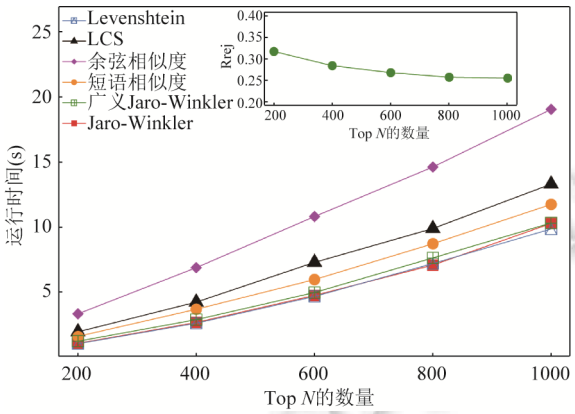

Fig.7 Running-Time curves for the six algorithms vs. the top-N sample size and curve of rejection ratio for the generalized Jaro-Winkler algorithm vs. the top-N sample size

图 7 6 种算法的运行时间随 top N 样本数量变化曲线以及针对广义 Jaro-Winkler 算法的拒绝率随 top 样本数据量变化曲线

通过图 7 可以发现,广义 Jaro-Winkler 算法在运行时间上略高于 Levenshtein 算法和 Jaro-Winkler 算法,但明显优于余弦相似度、LCS 和短语相似度这 3 种算法,说明广义 Jaro-Winkler 算法虽然比传统算法更复杂,但在运行时间上并不明显弱于传统算法.从图 7 也可以看到,广义 Jaro-Winkler 算法通过限制参数 m 的取值,可以有效减少数据集中无关数据的计算量,被剔除的数据量在 5 个数据集的占比均超过了 25%.可见,参数 m 的引入对于提升广义 Jaro-Winkler 算法的计算效率起到了明显的作用.

5 总 结

本文建立了一个完整的数据聚合模型来对内外部商户名称进行实体关联,并在人工标注的实验数据上验证了模型的有效性和精确性.主要工作和贡献体现在以下 3 个方面.

(1) 本文面向中文商户名称一类短文本数据,提出了中文短文本聚合的完整模型框架,设计和实现了快速匹配和精细匹配两个聚合步骤.通过快速匹配,可以有效减小匹配候选对的规模,将匹配对象锁定在一个较小的范围内,比如设定为 Top-1000 条;通过精细匹配,再从 1 000 条候选匹配对象中,准确地识别出存在的目标匹配商户.如果 1 000 条候选匹配对象中不存在目标匹配商户,精细匹配过程也能够准确地拒绝掉非目标匹配商户.因此,本文通过快速匹配和精细匹配过程,实现了聚合效率和聚合准确率的兼顾.

(2) 本文针对短文本的相似度算法进行了总结和归纳,通过具体案例,分析了常见算法的优势及不足之处.本文的讨论对于短文本相似度算法的研究具有一般的指导意义.

(3) 通过结合各种算法的优势,提出了一种广义 Jaro-Winkler 相似度算法,并分析了该算法的参数特性,通过严格推导的参数范围,可以简化算法的实现,提升算法执行效率.在商户数据集上进行的测试和对比的结果表明,广义 Jaro-Winkler 算法相比其他算法具有更好的中文短文本匹配精确度和稳定性.但必须指出,本算法仅在中文商户名称数据上进行了较系统的测试,其扩展性还有待在其他数据集上作进一步验证.

对于本研究工作的未来展望,我们认为可以从以下 4 个方面展开.

(1) 本文的短文本聚合实验针对的是两个相关数据集,而互联网上可关联的数据集可能非常多,比如多个

电商网站的商品数据聚合问题.两两匹配的方法虽然直观,但不一定是效率最好的方法,如何在多个数据集中快速筛选出匹配候选对是一个值得研究的问题.

(2) 针对中文短文本命名实体,如果可以识别和抽象出命名规则,也可以从规则匹配的角度实现中文短文本的快速、准确聚合.因此如何有效地进行细粒度的命名规则建模,是未来值得研究的一个重要课题.

(3) 当面向互联网的应用时, 待聚合的数据规模通常非常庞大,传统单机算法程序已不能适应海量数据的处理要求.因此,基于本文提出的中文短文本聚合模型框架的分布式改造是未来的一个重要研究方向;如何利用目前流行的 Map/Reduce 框架[42−44]实现这一目标是一个近期值得研究的课题.

(4) 在实际应用中,相关数据集可能会随时更新,重新更新倒排表再进行快速匹配和精细匹配的过程显然是一个极为耗时的过程,如何适应数据集的动态更新(增量学习)[45]也是一个值得研究的重要问题.

致 谢 感谢匿名审稿人对本文提出的修改意见,这些宝贵意见对于完善本文具有非常大的帮助.

References:

[1] Lin Y, Liu Z, Sun M, Liu Y, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In: Proc. of the AAAI. Palo Alto: Association for the Advancement of Artificial Intelligence, 2015. 2181–2187.

[2] Wang S, Wang HJ, Qin XP, Zhou X. Architecting big data: Challenges, studies and forecasts. Jisuanji Xuebao (Chinese Journal of Computers), 2011,34(10):1741–1752 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01741]

[3] Deng D, Jiang Y, Wang JN. Aggregation practice for big data. Communications of the CCF, 2013,9(8):44–47 (in Chinese with English abstract).

[4] Khalid MA, Jijkoun V, De Rijke M. The impact of named entity normalization on information retrieval for question answering. In: Proc. of the European Conf. on Information Retrieval. Berlin, Heidelberg: Springer-Verlag, 2008. 705–710. [doi: 10.1007/978-3-540-78646-7_83]

[5] Jijkoun V, Khalid MA, Marx M, Rijke MD. Named entity normalization in user generated content. In: Proc. of the 2nd Workshop on Analytics for Noisy Unstructured Text Data. ACM, 2008. 23–30. [doi: 10.1145/1390749.1390755]

[6] Brizan DG, Tansel AU. A. survey of entity resolution and record linkage methodologies. Communications of the IIMA, 2015,6(3):5.

[7] Koudas N, Sarawagi S, Srivastava D. Record linkage: Similarity measures and algorithms. In: Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of Data. ACM, 2006. 802–803. [doi: 10.1145/1142473.1142599]

[8] Christen P. A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. on Knowledge and Data Engineering, 2012,24(9):1537–1555. [doi: 10.1109/TKDE.2011.127]

[9] Lee S, Hwang S. ARIA: Asymmetry resistant instance alignment. In: Proc. of the 28th AAAI Conf. on Artificial Intelligence. Palo Alto: Association for the Advancement of Artificial Intelligence, 2014. 94–100.

[10] Zhuang Y, Li GL, Feng JH. A survey on entity alignment of knowledge base. Journal of Computer Research and Development, 2016,53(1):165–192 (in Chinese with English abstract).

[11] Ishii H, Tempo R, Bai E, Dabbene F. Distributed randomized PageRank computation based on Web aggregation. IEEE Trans. on Automatic Control, 2010,55(9):1987–2002. [doi: 10.1109/CDC.2009.5399514]

[12] Granitto PM, Verdes PF, Ceccatto HA. Neural network ensembles: Evaluation of aggregation algorithms. Artificial Intelligence, 2005,163(2):139–162. [doi: 10.1016/j.artint.2004.09.006]

[13] Islam A, Inkpen D. Semantic text similarity using corpus-based word similarity and string similarity. ACM Trans. on Knowledge Discovery from Data, 2008,2(2):1–25. [doi: 10.1145/1376815.1376819]

[14] Nikolov A, Uren V, Motta E, Roeck A. Integration of semantically annotated data by the KnoFuss architecture. In: Knowledge Engineering: Practice and Patterns. Berlin, Heidelberg: Springer-Verlag, 2008. 265–274. [doi: 10.1007/978-3-540-87696-0_24]

[15] Herzog TH, Scheuren F, Winkler WE. Record linkage. Wiley Interdisciplinary Reviews: Computational Statistics, 2010,2(5): 535–543. [doi: 10.1002/wics.108]

[16] Jiang H, Han AQ, Wang MJ, Wang Z, Wu YL. Solution algorithm of string similarity based on improved Levenshtein distance. Computer Engineering, 2014,40(1):222–227 (in Chinese with English abstract).

[17] Okuda T, Tanaka E, Kasai T. A method for the correction of garbled words based on the Levenshtein metric. IEEE Trans. on Computers, 1976,100(2):172–178. [doi: 10.1109/TC.1976.5009232]

[18] Su Z, Ahn BR, Eom KY, Kang MK, Kim JP, Kim MK. Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm. In: Proc. of the 3rd Int'l Conf. on Innovative Computing Information and Control (ICICIC 2008). IEEE, 2008. 569. [doi: 10.1109/ICICIC.2008.422]

[19] Deng D, Li G, Wen H, Feng JH. An efficient partition based method for exact set similarity joins. Proc. of the VLDB Endowment, 2015,9(4):360–371. [doi: 10.14778/2856318.2856330]

[20] Tasi CS, Huang YM, Liu CH, Huang YM. Applying VSM and LCS to develop an integrated text retrieval mechanism. Expert Systems with Applications, 2012,39(4):974–3982. [doi: 10.1016/j.eswa.2011.09.039]

[21] Hunt JW, Szymanski TG. A fast algorithm for computing longest common subsequences. Communications of the ACM, 1977,20(5):350–353. [doi: 10.1145/359581.359603]

[22] Ye J. Cosine similarity measures for intuitionistic fuzzy sets and their applications. Mathematical and Computer Modelling, 2011,53(1):91–97. [doi: 10.1016/j.mcm.2010.07.022]

[23] Kumar S, Rana JL, Jain RC. Text document clustering based on phrase similarity using affinity propagation. Int'l Journal of Computer Applications, 2013,61(18):38–44. [doi: 10.5120/10032-5077]

[24] Gomaa WH, Fahmy AA. A survey of text similarity approaches. Int'l Journal of Computer Applications, 2013,68(13):13–18. [doi: 10.5120/11638-7118]

[25] Jupin J, Shi JY, Obradovic Z. Understanding cloud data using approximate string matching and edit distance. In: Proc. of the High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion. IEEE, 2012. 1234–1243. [doi: 10.1109/SC.Companion.2012.149]

[26] Namiki Y, Ishida T, Akiyama Y. Fast DNA sequence clustering based on longest common subsequence. In: Emerging Intelligent Computing Technology and Applications. Berlin, Heidelberg: Springer-Verlag, 2012. 453–460. [doi: 10.1007/978-3-642-31837-5_66]

[27] You B, Yan YS, Sun YG, Liu J. Method of information content evaluating semantic similarity on HowNet. Computer Systems and Applications, 2013,22(1):129–133 (in Chinese with English abstract).

[28] Pedersen T, Patwardhan S, Michelizzi J. WordNet: Similarity: Measuring the relatedness of concepts. In: Demonstration Papers at HLT-NAACL 2004. Palo Alto: Association for Computational Linguistics, 2004. 38–41.

[29] Wagner H, Dlotko P, Mrozek M. Computational topology in text mining. In: Wagner H, Dłotko P, Mrozek M, eds. Proc. of the Computational Topology in Image Context of the 4th Int'l Workshop. Berlin, Heidelberg: Springer Int'l Publishing, 2012. 68–78.

[30] Ramage D, Rafferty AN, Manning CD. Random walks for text semantic similarity. In: Proc. of the 2009 Workshop on Graph-Based Methods for Natural Language Processing. Stroudsburg: ACM, 2009. 23–31.

[31] Wang J, Feng J, Li G. Trie-Join: Efficient trie-based string similarity joins with edit-distance constraints. Proc. of the VLDB Endowment, 2010,3(1-2):1219–1230. [doi: 10.14778/1920841.1920992]

[32] Li G, Deng D, Wang J, Feng J. Pass-Join: A partition-based method for similarity joins. Proc. of the VLDB Endowment, 2011,5(3):253–264. [doi: 10.14778/2078331.2078340]

[33] Jiang Y, Li G, Feng J, Li WS. String similarity joins: An experimental evaluation. Proc. of the VLDB Endowment, 2014,7(8):625–636. [doi: 10.14778/2732296.2732299]

[34] Wang D, Zhang H. Inverse-Category-Frequency based supervised term weighting schemes for text categorization. Journal of Information Science and Engineering, 2013,29(2):209–225.

[35] Cambazoglu BB, Kayaaslan E, Jonassen S, Aykanat C. A term-based inverted index partitioning model for efficient distributed query processing. ACM Trans. on the Web (TWEB), 2013,7(3):15. [doi: 10.1145/2516633.2516637]

[36] Yalcin A, Reis S, Aydinoglu AC, Yomralioglu T. A GIS-based comparative study of frequency ratio, analytical hierarchy process, bivariate statistics and logistics regression methods for landslide susceptibility mapping in Trabzon, NE Turkey. Catena, 2011,85(3):274–287. [doi: 10.1016/j.catena.2011.01.014]

[37] Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Müller M. pROC: An open-source package for R and S+ to analyze and compare ROC curves. BMC bioinformatics, 2011,12(1):77. [doi: 10.1186/1471-2105-12-77]

[38] Sebastiani F. Machine learning in automated text categorization. ACM Computing Surveys, 2002,34(1):1–47. [doi: 10.1145/505282.505283]

[39] Agarwal A, Duchi JC. The generalization ability of online algorithms for dependent data. IEEE Tran. on Information Theory, 2013, 59(1):573–587. [doi: 10.1109/TIT.2012.2212414]

[40] De Weerdt J, De Backer M, Vanthienen J, Baesens B. A robust *F*-measure for evaluating discovered process models. In: Proc. of the 2011 IEEE Symp. on Computational Intelligence and Data Mining (CIDM). IEEE, 2011. 148–155. [doi: 10.1109/CIDM.2011.5949428]

[41] Pressé S, Ghosh K, Lee J, Dill KA. Principles of maximum entropy and maximum caliber in statistical physics. Reviews of Modern Physics, 2013,85(3):1115. [doi: 10.1103/RevModPhys.85.1115]

[42] Dean J, Ghemawat S. MapReduce: A flexible data processing tool. Communications of the ACM, 2010,53(1):72–77. [doi: 10.1145/1629175.1629198]

[43] Liu Z, Liu M. Logistic regression parameter estimation based on parallel matrix computation. In: Theoretical and Mathematical Foundations of Computer Science. Berlin, Heidelberg: Springer-Verlag, 2011. 268–275. [doi: 10.1007/978-3-642-24999-0_38]

[44] Deng D, Li G, Hao S, Wang J, Feng J, Li WS. Massjoin: A mapreduce-based method for scalable string similarity joins. In: Proc. of the 30the IEEE Int'l Conf. on Data Engineering. IEEE, 2014. 340–351. [doi: 10.1109/ICDE.2014.6816663]

[45] Deng D, Li G, Wen H, Jagadish HV, Feng J. META: An efficient matching-based method for error-tolerant autocompletion. Proc. of the VLDB Endowment, 2016,9(10). [doi: 10.14778/2977797.2977808]

附中文参考文献:

[2] 王珊,王会举,覃雄派,周烜.架构大数据:挑战,现状与展望.计算机学报,2011,34(10):1741–1752. [doi: 10.3724/SP.J.1016.2011.01741]

[3] 邓栋,姜禹,王健楠.大数据融合实战.中国计算机学会通讯,2013,9(8):44–47.

[10] 庄严,李国良,冯建华.知识库实体对齐技术综述.计算机研究与发展,2016,53(1):165–192.

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

[26] 游彬,严岳松,孙英阁,刘靖.基于 HowNet 的信息量计算语义相似度算法.计算机系统应用,2013,22(1):129–133.

刘震(1976－),男,吉林省吉林市人,博士,副教授,CCF 专业会员,主要研究领域为数据挖掘与算法,大数据分析技术.

华锦芝(1979－),男,高级工程师,主要研究领域为电子支付,大数据,信息安全,人工智能.

陈晶(1991－),女,硕士,主要研究领域为文本挖掘技术.

肖淋峰(1993－),男,硕士,主要研究领域为自然语言处理.

郑建宾(1983－),男,工程师,主要研究领域为电子支付,大数据,人工智能.