针对舆情数据的去重算法①

张庆梅

(中国科学技术大学 软件学院, 苏州 215123)

摘 要: 针对在数据服务中舆情去重不可避免且缺乏理论指导的问题, 通过研究 SimHash、MinHash、Jaccard、Cosine Similarty 经典去重算法, 以及常见的分词和特征选择算法, 以寻求表现优异的算法搭配, 并对传统 Jaccard 和 SimHash 进行了改进分别产生新算法: 基于短文章的 Jaccard 和基于 Cosine Distance 的 SimHash. 针对比较对象众多实验效率低下的问题, 提出了先纵向比较筛选出优势算法, 然后横向比较获得最佳搭配, 最后综合比较的策略, 并结合 3000 舆情样本实验证明: 改进的 SimHash 比传统的 SimHash 具有更高的精度和召回率; 改进的 Jaccard 较传统 Jaccard, 召回率提高了 17%, 效率提高了 50%; MinHash+结巴全模式分词和 Jaccard+IKAnalyzer 智能分词在保持精度高于 96%的条件下, 都具有 75%以上的高召回率, 且稳定性很好. 其中 MinHash 去重效果略低于 Jaccard, 但特征比较时间较短, 综合表现最好.

关键词: 舆情数据; 去重算法; 相似度计算; 大数据服务

Duplicate Removal Algorithm for Public Opinion

ZHANG Qing-Mei

(School of Software Engineering, University of Science and Technology of China, Suzhou 215123, China)

Abstract: In big data services, duplicate removal of public opinion information is inevitable, and it lacks theoretical guidance. There is a research on the classical duplicate removal algorithm such as SimHash, MinHash, Jaccard, Cosine Similarty, as well as common segmentation algorithm and feature selection algorithm in order to seek excellent performance of the algorithm. The Jaccard based on short article and the SimHash algorithm based on Cosine Distance are proposed to improve the traditional algorithms. Aiming at the problem of the low efficiency of experiment on many research subjects, the strategy is adopted that filters out algorithm of obvious advantages by vertical comparison firstly, and gets the most appropriate algorithm collocation by horizontal comparison secondly, at last, makes a comprehensive comparison. The experiment of 3000 public opinion samples shows that improved SimHash has better effect than traditional SimHash; improved Jaccard increases the recall rate by 17% and improves the efficiency by 50% compared with traditional Jaccard. Under the condition that the accuracy is higher than 96%, MinHash+Jieba full pattern word segmentation and Jaccard+IKAnalyzer intelligent word segmentation has more than 75% recall rate and good stability. MinHash is a bit weak than Jaccard in the aspect of removal effect, yet has the best comprehensive performance and shorter feature comparison time.

Key words: public opinion data; duplicate removal algorithm; similarity computation; big data service

据中国互联网络信息中心统计, 截止到2015年12月, 我国社交网站、微博等社交应用的网民使用率达77.0%[1], 新媒体逐渐成为网民表达意见和看法、行使公民权利的重要渠道和方式[2], 是用户获取和分享“新闻热点”、“兴趣内容”、“专业知识”、“舆论导向”的重要平台[3]. 从社会学角度来看, 这些舆情信息反映了民众的社会政治态度, 有着强大的监督力度[4]. 而舆情信息的价值远远不止其传播性所带来的社会监督力度,

① 收稿时间:2016-08-28;收到修改稿时间:2016-09-27 [doi:10.15888/j.cnki.csa.005745]

在金融领域也广泛被使用. 由于舆情信息可以准确反映个人和企业的信用状况, 目前已有大数据服务公司采集舆情数据, 然后加工分析为金融机构在信用评定、风险评估方面提高参考. 然而随着大数据时代的到来, 抓取的舆情数据重复性冗余急剧增大[5], 这些重复的数据严重影响后期的加工处理和客户体验. 据调研, 目前的去重技术大多针对网页, 专门针对舆情数据的却很少. 因此对于舆情数据服务, 迫切需要针对舆情开展去重研究来解决数据重复带来的一系列问题. 本文通过对几种经典去重算法在舆情数据方面的表现进行研究, 并分析不同实现方式的去重算法之间的精度、召回率和效率的差异, 寻求在舆情去重上表现优异的算法, 为舆情数据服务在机器去重方面提供参考.

1 相关技术和实现方法

本文将整个去重分为三个步骤: 首先是分词, 将一篇文章转化为词语列表; 然后是对文章进行特征选择, 实现文章特征属性的提取; 最后是基于相似度计算的去重算法进行去重. 因此关键技术包括分词、特征选择、相似度计算. 每种技术中本文都有多种候选算法, 相关技术的研究对象如表 1 所示.

表 1 相关技术的研究对象

分词算法	特征选择	相似度计算
结巴分词	词频 TF	Jaccard
IKAnalyzer 分词	TF-IDF	Cosine Similarity
HanLP 分词	TextRank	SimHash
		MinHash

1.1 分词

本文的分词具体指中文分词, 目的是将汉字序列切分成由词语组成的序列[6]. 分词算法的不同将直接影响去重效果. 本文尝试通过比较不同分词算法对舆情去重效果的影响, 来获得最适合舆情去重的分词方法. 本文选用中文分词中比较常用的 3 种分词方法: 结巴分词、IKAnalyzer 分词和 HanLP 分词, 其中结巴分词包含 3 种模式: 精确模式、全模式和搜索引擎模式. IKAnalyzer 包含 2 种分词模式: 细粒度模式和智能模式. HanLP 包含 8 个分词器: 标准分词、NLP 分词、索引分词、N 最短路径分词、最短路径分词、CRF 分词、极速词典分词和繁体分词. 由于本文的舆情样本全部是简体中文, 因此本文只将前 7 种分词纳入此后的研究中去.

1.2 特征选择

选用较小维度的特征代表整个文本正文的过程就是特征选择. 在本文中将几种常见的特征选择纳入研究范围, 分别是: 词频、TF-IDF 和 TextRank. 这三种特征选择都是权重特征, 适合与 Cosine Similarity 和 SimHash 算法结合使用.

1.2.1 词频

词频是指词语出现的次数, 词频统计通常不单独被使用, 一般是结合其他算法一起使用, 应用范围涉及中文分词、研究热点分析、文本分析等诸多方面[7-9]. 常用词频的计算方式是获取某个词在文章中出现的次数, 但这种计算方式忽略了文章有长短之分. 当文章篇幅差距很大, 将不能准确体现文章内容之间的差异性. 因此在本文采用的是相对词频, 它对的计算公式如式(1)所示.

$$词频 = \frac{某词在文章中出现的次数}{文章总词数} \tag{1}$$

1.2.2 TF-IDF

TF-IDF 和词频同样都是常用的加权技术, 但相比于词频, TF-IDF 能够反映整个词在一个文本集合或者语料库中的“重要程度”, 词频仅仅在一定程度上反映一个词在一篇文章的重要程度, 没有将整个文本库的大小考虑进去. TF-IDF 广泛应用于自动关键词提取、文本摘要提取等[10,11]. TF-IDF 的主要思想是词语的重要性随着这个词在文本出现的次数成正比, 同时随着它在整个文本集合中出现的频率成反比, 某个词在文章中的重要程度越大, TF-IDF 的值就越大. 了解 TF-IDF 首先了解逆文档频率, 词频和逆文档频率的乘积就是 TF-IDF, 逆文档频率(IDF)的计算公式如式(2)所示.

$$逆文档频率 = log\left(\frac{文档库中的文档总数}{包含某词的文档数+1}\right) \tag{2}$$

1.2.3 TextRank

TextRank 是受启发于 PageRank, PageRank 最开始是用于网页相关性和重要性的评估, 获取网页排序, 提高用户对搜索引擎检索结果的满意度, 此算法由 Google 的创始人谢尔盖•布林和拉里•佩奇在 1998 年提出[12]. PageRank 的计算公式如式(3)所示.

$$S(V_i) = (1-d) + d \times \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{3}$$

$S(V_i)$表示网页 i 的重要性, d 是阻尼系数, 通常设为 0.85. $In(V_i)$是指向网页 i 的链接集合, $Out(V_i)$表示网页 i 指向的网页集合, $|Out(V_i)|$表示网页 i 指向的网页集合的元素个数. 整个计算需要经过多次迭代, 初始设置网页重要性为 1.

TextRank 计算对象从网页转化为文本中的词语或者句子, 每个词语或句子根据此算法会得到相应的权重. 具体计算公式如式(4)所示.

$$WS(V_i)=(1-d)+d\times\sum_{V_j\in In(V_i)}\frac{1}{\sum_{V_k\in Out(V_j)}w_{jk}}WS(V_j) \quad (4)$$

本文利用此特征选择主要是获取不同词语的权重值, 即把每个词语看成一个节点(V_i). 当计算对象是词语时, 因为 w_{jk} 取值都为 1, TextRank 就蜕变成 PageRank. 不过式 4 中的变量含义有所变化, $S(V_i)$表示文本中词语 i 的重要性, $In(V_i)$是文章中指向词语 i 的词语集合, $|Out(V_i)|$表示文章中词语 i 指向的词语集合的元素个数. 词语之间的相邻关系, 依赖于窗口大小的设置, 一个窗口中的任意两个词语之间都是相邻的, 并且边都是无向无权的. 由于 TextRank 需要经过多次迭代, 因此特征获取的时间复杂度很高.

1.3 相似度计算

相似度计算是指是在特征选择的基础上通过去重算法来求取文章之间相似度的过程, 是自然语言处理和数据挖掘中常用的操作. 本文参考网页去重的经典算法, 将 Jaccard、Cosine Similarity、SimHash 和 MinHash 纳入研究范围, 对于传统实现方式, MinHash 有两种: 基于单 Hash 函数的 MinHash 算法和基于多 Hash 函数的 MinHash 算法, 其余的各有一种. 本文除了实现传统的算法之外, 还对传统 Jaccard 和 SimHash 进行改进分别产生新的算法: 基于短文章的 Jaccard 和基于 Cosine Distance 的 SimHash.

2 基于相似度计算的去重算法

对于不同的应用场景, 考虑到数据规模、时间开销, 去重算法的选择会有所不同. 本文在此分析不同算法的去重原理以及时间开销, 从理论上分析不同算法的优缺点, 并给出具体的实现步骤. 为不同需求的应用场景在去重算法的选择上提供参考.

2.1 Jaccard 算法

Jaccard 系数, 又称 Jaccard 相似度系数, 用来评估两个集合之间的相似度和分散度[13], Jaccard 系数越大表明两篇文章的相似度越大. 利用 Jaccard 去重, 首先将文章通过分词转化为由词语构成的特征集合, 通过检查两个集合的 Jaccard 系数是否超过指定的阈值来判断文章是否重复.

1) 传统的 Jaccard

传统的 Jaccard, 基于 Merge 算法, 通过求取两个文章的特征集合交集和并集的长度比例来衡量文章之间的距离. 计算公式如式(5)所示.

$$Jaccard(A,B)=\frac{|A\cap B|}{|A\cup B|} \quad (5)$$

从实现的原理上看, 传统的 Jaccard 算法, 并没有将两篇文章的长度差异考虑进去, 假设两篇文章重复的文章长度差异很大, 例如一个包含 1500 个单词, 一个包含 500 个单词, 两篇文章的单词交集长度是 500, 利用传统的 Jaccard 计算两篇文章距离, 结果是: 0.25, 传统 Jaccard 的阈值一般在 0.5 以上, 在这种情况下, 就很容易漏判长度差异大的重复文章. 此外 Merge 算法的时间复杂度是 O(m+n)(m 和 n 是两个集合的长度), 不是很高, 但当文章篇幅很长, 数据规模很大时, 这个时间开销将会非常庞大. 因此 Jaccard 算法不适应文章篇幅普遍较长、数据规模较大的业务场景.

2) 基于短文章的 Jaccard

针对传统 Jaccard 对属于包含关系重复的文章识别能力低的问题, 本文提出一种基于短文章的 Jaccard, 通过求取两个特征集合交集占短文章集合长度的比例来衡量两文章的距离. 以下简称改进的 Jaccard, 计算公式如式(6)所示.

$$Jaccard(A,B)=\frac{|A\cap B|}{min(|A|,|B|)} \quad (6)$$

在这种改进下, 属于包含关系的重复文章, 即使文章长度差异很大, 求取的文章 Jaccard 系数也会随文章相似程度的增大而增大. 对于传统 Jaccard 中的例子, 使用改进的 Jaccard 计算, 两篇文章的距离就是 1, 即完全重复, 符合实际情况. 改进的 Jaccard 的时间复杂度和传统 Jaccard 相同, 但是相比传统的 Jaccard 少了求并集的过程, 因此时间消耗要少.

2.2 Cosine Similarity 算法

Cosine Similarity 又称 Cosine Distance, 与几何中的向量余弦夹角很相似. 当把一篇文章的特征抽象成一个向量时, 可以使用这种方式计算文章之间的相似

度, 计算公式如式(7)所示.

$$Cosine\ Similarity(A,B)=\frac{A*B}{\|A\|\|B\|} \quad (7)$$

具体实现步骤如下:

Step 1. 对文章 *A*, *B* 进行分词操作, 得到文章 *A*, *B* 的词语集合 *sa*, *sb*.

Step 2. 分别对集合 *sa* 和 *sb* 进行特征选择, 形成两组由加权的词语构成的向量 *da*, *db*. 如果不选择使用特征选择, 则形成两组由加权都为 1 的词语构成的向量 *da*, *db*.

Step 3. 获取集合 *sa* 和 *sb* 的并集 *unionS* 分别将 *da*, *db* 转化为以 *unionS* 为坐标向量 *ca*, *cb*.

Step 4. 求得 *Cosine Similarity(ca,cb)*为 *k*.

Step 5. 对于给定的阈值 *t*, 若 *k*>*t* 返回 *True*; 若 *k*<*t*, 返回 *False*.

对于 Step 3 向量坐标的转化, 需要遍历集合 unionS 中的元素, 并依次判断每个元素在待转化向量中的存在情况, 因此整个相似度计算的时间复杂度平均为 O(n*m)(n 为并集的长度, m 为待转化向量的长度), 相比于 Jaccard, 时间开销更大.

2.3 SimHash 算法

SimHash 是由 Charikar 在 2002 年提出的去重算法, 主要用于海量文本的去重工作[14]. SimHash 对文章进行相似度计算, 需要两步, 首先特征提取形成指纹, 然后根据指纹进行特征比较, 计算相似度.

1) 传统的 SimHash

传统的 SimHash 首先将一篇文章转化为由 k 位 0/1 构成的指纹(k 通常取 32 或 64), 然后利用 Hamming Distance(海明距离)来对两篇文章的指纹进行相似计算. 海明距离是指两串二进制编码对应比特位取值不同的比特数目, 海明距离越大则相似度越小. 由于 SimHash 能将一篇文章转化为 k 位的字符, 相比于 Jaccard 和 Cosine Similarity, 能大大降低特征比较的维度. 虽然多了特征提取的步骤, 但对于大数据服务, 一篇文章只需在入库时进行一次特征提取, 然后将形成的指纹保存下来, 而特征比较会在每次去重时都要基于指纹进行多次. 因此对于大规模的数据去重, SimHash 具有绝对优势的去重效率. 传统的 SimHash 的具体实现步骤如下:

Step 1. 首先对文章 *A*, *B* 进行特征获取, 形成指纹码 *fa* 和 *fb*. 形成指纹码的过程如下:

Step 1.1. 对文章进行分词, 得到文章的词语集合 *s*.

Step 1.2. 对集合 *s* 进行特征选择, 形成一组由加权的词语构成的向量 *d*. 如果不选择使用特征选择, 则形成由加权都为 1 的词语构成的向量 *d*.

Step 1.3. 初始化一个 32 维的向量 *v*, 将向量中的每个元素初始值设置为 0.

Step 1.4. 对于文章的集合 *s* 中的每个词语进行如下运算: 将词语 *word* 利用 Hash 函数计算后得到一个 32 位的签名 *f*. 对于一个 32 位的签名 *f*, 遍历每一比特位, 如果第 *i* 位上为 0, 从向量 *v* 的第 *i* 维中减去这个词语的权值 *d[word]*, 否则加上该词语的权值 *d[word]*. 完成 *s* 全部词语的计算后, 一篇文章将被映射成一个 32 维向量 *g*.

Step 1.5. 如果 *g* 的第 *i* 维大于 0, 则将 32 位指纹的第 *i* 位(从左数)置为 1, 否则置为 0. 最终一篇文章被映射成一个 32 位的指纹码.

Step 2. 初始化变量 *count*=0, *i*=0.

Step 3. *i*=*i*+1, 对于 *fa* 和 *fb*, 比较 *fa* 的第 *i* 个比特位与 *fb* 的第 *i* 比特位是否相同.

Step 4. 如果不相同则 *count*=*count*+1, 对于给定的阈值 *t*, 若 *count*>*t* 返回 *False*; 否则判断 *i* 是否小于 32, 如果 *i*<32, 跳到 3., 如果 *i*>=32, 返回 *True*.

(2) 基于 Cosine Distance 的 SimHash

在对 Cosine Distance 和传统 SimHash 研究的基础上, 本文提出基于 Cosine Distance 的 SimHash, 以下简称 SimHashCosine. 该 SimHash 特征提取只保留传统 SimHash 实现步骤的 Step 1.1-1.4, 然后利用 Cosine Distance 来计算指纹之间的相似度, 最后通过判断是否超过给定的阈值来判定是否重复. 两种 SimHash 的时间开销差异主要体现在是特征比较上, 若 *n* 为指纹码的长度, *m* 为阈值(*n*>*m*), 传统的 SimHash 相似度计算利用 Hamming Distance, 时间复杂度最坏情况是 O(*n*), 最小只有 O(*m*), 而 SimHashCosine, 相似度计算利用 Cosine Distance, 时间复杂度至少 O(*n*), 且时间频度至少是传统 SimHash 的 3 倍, 因此在特征比较效率上传统的 SimHash 更高一点.

2.4 MinHash 算法

MinHash 和 SimHash 一样, 能对文章进行很好的降维, 适用于大规模的网页去重工作[15]. MinHash 经过特征提取, 将一篇文章最终转化为 *n* 个最小 Hash 函

数值构成的特征集合, 然后基于 Hash 函数值集合获取 Jaccard 距离来衡量相似度.

1) 基于单 Hash 函数的 MinHash

基于单 Hash 函数的 MinHash, 以下简称 MinOneHash, 在进行特征提取仅使用了一个 Hash 函数, 然后使用传统的基于 Merge 算法的 Jaccard 计算相似度, 具体的实现步骤如下:

Step 1. 首先对文章 *A*, *B* 进行特征集合获取, 形成特征集合 *va* 和 *vb*. 形成特征集合的过程如下:

Step 1.1. 文章进行分词, 得到文章的词语集合 *s*.

Step 1.2. 对于 *s* 中的每个词语都使用指定的一种 Hash 函数进行 Hash 操作, 获得 Hash 值集合 *h*.

Step 1.3. 从集合 *h* 中挑选出最小的 *n* 个 Hash 值, 形成包含 *n* 个元素的集合 *v*, *v* 即是指纹集合.

Step 2. 获取集合 *va* 和 *vb* 的交集 *interS*

Step 3. 若 *interS*=空集, 返回 *False*; 否则获取集合 *va* 和 *vb* 的并集 *unionS*. 然后求得集合 *va* 和 *vb* 交集和并集长度的比值 *k*.

Step 4. 对于给定的阈值 *t*, 若 *k*>*t* 返回 *True*; 若 *k*<*t*, 返回 *False*.

2) 基于多 Hash 函数的 MinHash

基于多 Hash 函数的 MinHash, 以下简称 MinMutilHash, 使用 *n* 个 Hash 函数进行特征提取(*n*>1), 特征提取的步骤: 对于事先确定的 *n* 个 Hash 函数, 对于每个 Hash 函数, 按照约定的顺序都对文章的词语集合 *s* 中的所有词语进行 Hash 操作, 形成各自的 Hash 函数值集合, 然后各自从各自的 Hash 函数值集合中筛选出最小 Hash 值, *n* 个 Hash 函数最终获得 *n* 个最小值. 由于特征提取计算维度的扩大, 相对于 MinOneHash, 时间复杂度较高. 但 MinMutilHash 相似度计算法是根据 Broder 提出的最小独立置换概念, 通过求得两个 Hash 函数值集合中对应位置 Hash 值相同的元素数目来评估相似度, 特征比较的时间复杂度是 O(*n*), 相比于 MinMutilHash 的 O(*m*+*n*), 特征比较效率要高.

3 实验测试及分析

3.1 测试方案设计

由于涉及算法众多, 以排列组合的形式进行组合测试需要耗费大量时间. 因此本文针对表 1 所列算法, 先纵向比较剔除明显劣势的算法, 然后横向比较获得各个去重算法最适宜的分词算法和特征选择, 最后对去重表现良好的候选算法, 进行进一步优化后再综合测试比较的策略.

本文以精度、召回率、计算时间来衡量算法的去重效果. 精度是衡量算法准确性的指标, 公式如式(8)所示. 召回率是衡量算法查全程度的指标. 公式如式(9)所示

$$\text{精度}=\frac{\text{检索出的实际重复的文章数}}{\text{检索出的重复的总文章}} \tag{8}$$

$$\text{召回率}=\frac{\text{检索出的实际重复的文章数}}{\text{文章库中所有重复的文章数}} \tag{9}$$

考虑到大数据服务对数据准确性的要求, 去重效果的衡量标准以精度优先, 精度越高表示去重效果越好; 其次是召回率, 召回率越高去重效果越好; 在精度相差不大时, 优先选择召回率高的算法, 相差不大的标准是正负差值不超过 1%; 计算时间最后考虑. 计算时间中包括两部分: 特征提取时间, 特征比较时间. 在大数据服务的舆情去重中, 对一篇文章特征提取只需要进行一次, 特征比较则会进行很多次, 因此对于不同的去重算法, 算法特征比较时间要优于特征提取时间考虑. 测试样本统一使用包含 3000 真实舆情文章的数据集.

3.2 纵向比较

3.2.1 分词算法的比较

为了保证实验结果不受特征选择的影响, 在本实验中对词语都不进行特征选择, 为了保证实验结果不受去重算法的影响, 在本实验中去重算法统一使用传统的 SimHash. 测试结果如表 2 所示.

表 2　基于结巴分词不同模式的去重测试结果

分词方法	精度/%	召回率/%	时间/s
结巴精确模式	87.94	52.83	87.05
结巴全模式	90.81	55.83	62.59
结巴搜索引擎模式	85.88	52.65	86.51
IKAnalyzer 细粒度	86.25	53.18	409.89
IKAnalyzer 智能	95.10	58.30	344.41
HanLP 标准	90.17	55.12	182.74
HanLP 索引	87.78	54.59	197.97
HanLP NLP	83.70	54.42	177.44
HanLP N 最短路径	89.49	52.65	203.61
HanLP 最短路径	88.79	51.77	182.59
HanLP CRF	91.35	56.01	209.35
HanLP 极速词典	84.18	55.48	198.73

注: 表中时间是指整个计算时间.

由表 2 可得, 精度: IKAnalyzer 智能>HanLP

CRF >结巴全模式分词>90.5%, 召回率: IKAnalyzer 智能>HanLP CRF >结巴全模式分词>55.5%, 因此保留 IKAnalyzer 智能、HanLP CRF 和结巴全模式.

3.2.2 特征选择算法的比较

本文继续使用 SimHash 算法, 分词算法选用 IKAnalyzer 智能分词, 以无加权为参照, 观察不同特征选择下去重效果的差. 实验结果如表 3 所示.

表 3 基于不同特征选择的去重测试结果

特征选择	精度(%)	召回率(%)	时间(s)
无加权	96.77	60	54.83
词频	78.87	56	51.84
TF-IDF	93.02	40	54.71
TextRank	96.77	60	4308.42

注: 表中时间是指整个计算时间.

由表 3 可得, 无加权和 TextRank 去重表现最好, 但是根据实验发现 TextRank 特征提取时间很长导致总计算时间太长, 且更换其他分词算法时, 结合 TextRank 的去重效果都有所降低, 因此舆情去重在此只保留无加权.

3.2.3 去重算法比较

去重算法的比较研究部分主要任务是从 Jaccard、SimHash、MinHash 中各筛选出一种, 然后和 Cosine Similarity 进行比较. 测试结果如表 4 所示.

表 4 不同去重算法的测试结果

去重算法	精度(%)	召回率(%)	时间(s)
传统 Jaccard	98.12	73.85	629.52
改进 Jaccard	95.52	90.46	341.68
传统 SimHash	90.81	55.83	72.51
SimHashCosine	98.77	67.49	82.29
MinOneHash	98.09	54.17	53.52
MinMultiHash	98.03	61.66	33.32
Cosine Similarity	94.71	82.16	22832.8

注: 表中时间是指特征比较时间.

由表 4 可知:

① 在精度和召回率上, SimHashCosine 同时高于 SimHashHamming, 保留 SimHashCosine.

② MinMultiHash 精度略低于 MinOneHash, 但两者相差不大, 且在召回率和特征比较时间上, MinMultiHash 相比于 MinOneHash 具有绝对优势, 因此保留 MinMultiHash.

③ Cosine Similarity 时间花费太大, 确定舍去.

④ 传统的 Jaccard 精度明显高于改进的 Jaccard, 但改进的 Jaccard 召回率和特征效率明显高于传统的 Jaccard, 各具明显优势, 实际使用时可以根据场景需求进行选择, 在面向金融行业的大数据服务中, 以精度优先保留传统的 Jaccard.

3.3 横向比较

在算法横向比较部分, 分词算法保留 IKAnalyzer 智能、HanLP CRF 和结巴全模式, 排除使用特征选择, 因此在横向比较部分主要研究保留的分词算法对去重算法的影响. 便于表示在此将 IKAnalyzer 智能、HanLP CRF、结巴全模式分词分别简称为智能、CRF、全模式. 横向比较结果如表 5 所示.

表 5 横向比较结果

去重算法	分词	精度(%)	召回率(%)
SimHashCosine	全模式	98.77	67.54
	智能	98.82	69.43
	CRF	97.70	70.49
MinMultiHash	全模式	97.88	65.19
	智能	97.04	63.60
	CRF CRF	98.06	62.37
Jaccard	全模式	97.69	74.56
	智能	98.15	74.91
	CRF	98.36	74.03

由表 5 可知:

① 精度优先原则, SimHashCosine 与 IKAnalyzer 智能结合效果最高.

② MinMultiHash 与三种分词方法结合时, 全模式和 CRF 精度最高且相差很小, 考虑全模式的召回率明显高于 CRF, 确定 MinMultiHash 和全模式结合.

③ Jaccard 与三种分词方法结合时, 召回率和精度都相差不大, 但特征比较时间, 全模式: 1018.42s, 智能: 638.54s, CRF: 861.57s, 其中 IKAnalyzer 智能模式最短, 因此选择智能模式和 Jaccard 结合.

3.3 综合比较

算法横向比较后筛选出这 3 种算法: MinMultiHash+结巴全模式、Jaccard+IKAnalyzer 智能、SimHashCosine+IKAnalyzer 智能. 阈值的不同, 会导致去重结果有很大差异, 此处研究这 3 种算法去重效果随着阈值的变化情况. 此外本文认为一个好的去重算法, 应当在保持较高精度时召回率也很高, 算法的特征比较时间短, 算法的稳定性较好. 这个稳定性主要体现在在整个阈值取值范围内, 精度和召回率随阈值的整体变化是否比较平稳. 本文以折线图的形式展示每种算法随着阈值的改变, 精度和召回率的变

化趋势. 精度随阈值的变化折线图如图 1 所示, 召回率随阈值变化折线图如图 2 所示. 如果一个算法的某个阈值精度少于 80%或召回率低于 40%, 相应阈值下的精度和召回率都不再被显示.

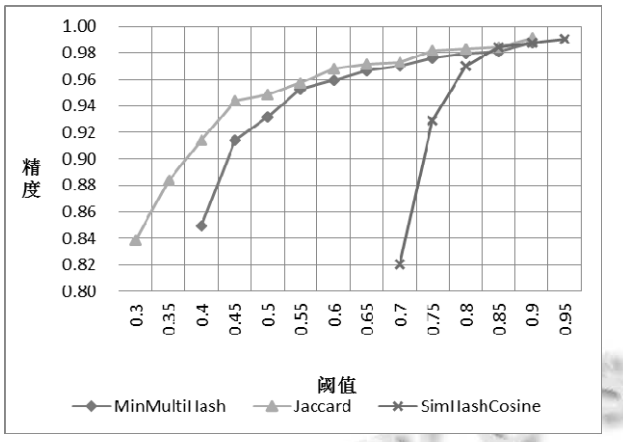

图 1　精度随阈值的变化折线图

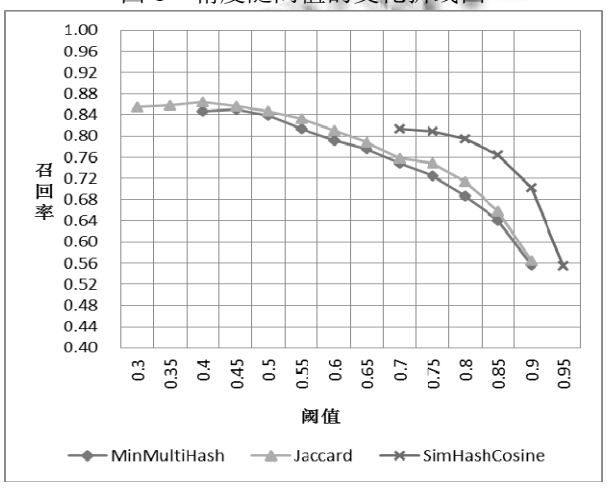

图 2　召回率随阈值的变化折线图

由图 1 和 2 很明显可以看出:

① Jaccard 和 MinMultiHash 在很大的阈值变化范围内, 都能同时保证较高的精度和较高的召回率.

② Jaccard 始终以微弱的优势, 在精度和召回率上高于 MinMultiHash.

③ 算法的稳定性排序: Jaccard>MinMultiHash>SimHashCosine.

④ 结合表 4 观察, MinMultiHash 特征比较时间远小于 Jaccard.

因此在舆情去重场景中, 对算法精度和召回率非常高, 推荐 Jaccard; 追求较高的精度和召回率, 同时对时间的要求也很高的情况, 推荐 MinMultiHash.

4　结语

舆情是大数据服务中一种重要的数据产品, 但随着大数据时代的来临, 舆情服务必须解决重复严重的问题才能提供更高质量的数据. 本文通过对分词算法、特征选择和去重算法进行实验研究, 并对传统的 Jaccard 和 SimHash 进行了改进. 提出了先纵向比较, 后横向比较, 最后综合比较的实验策略, 通过此实验策略筛选出了舆情去重表现突出的算法搭配. 随着舆情研究的深入, 在今后可将 Hadoop 算法纳入研究范围, 以提高算法的去重效率.

参考文献

1 中国互联网信息中心.2016 年第 37 次中国互联网络发展状况统计报告.http://www.cnnic.net.cn/gywm/xwzx/rdxw/2016/201601/t20160122_53293.htm.[2016].
2 魏超.新媒体技术发展对网络舆情信息工作的影响研究.图书情报工作,2014,58(1):30–34.
3 胡洋,刘秀荣,魏娜,张么九,刘婉行,钮文异.北京健康教育微博体系初建参与者网络及微博使用习惯的现状分析.中国健康教育,2014,30(8):706–708.
4 吴绍忠,李淑华.互联网络舆情预警机制研究.中国人民公安大学学报,2008,14(3):38–42.
5 贺知义.基于关键词的搜索引擎网页去重算法研究[硕士学位论文].武汉:华中师范大学,2015.
6 龙树全,赵正文,唐华.中文分词算法概述.电脑知识与技术,2009,5(10):2605–2607.
7 刘洪波.词频统计的发展.情报科学,1991,12(6):69–73.
8 朱小娟,陈特放.基于 SVM 的词频统计中文分词研究.微计算机信息,2007,23(30):205–207.
9 华秀丽,朱巧明,李培峰.语义分析与词频统计相结合的中文文本相似度量方法研究.计算机应用研究,2012,29(3):833–836.
10 王景中,邱铜相.改进的 TF-IDF 关键词提取方法.计算机科学与应用,2013,35(10):2901–2904.
11 Cho J, Shivakumar N, Garcia-Molina H. Finding Replicated Web Collections. Acm Sigmod Record, 2000, 29(2): 355–366.
12 黄德才,戚华春.PageRank 算法研究.计算工程,2003,32(4):145–146.
13 Real R, Vargas JM. The Probabilistic Basis of Jaccard's Index of Similarity. Systematic Biology,1996,45(3):380–385.
14 Sood S, Loguinov D. Probabilistic Near-Duplicate Detection Using SimHash. Acm Conference on Information, New York, 2011: 1117–1126.
15 Rao BC, Zhu E. Searching Web Data using MinHash LSH. International Conference on Management of Data, New York, 2016: 2257–2258.