广西科学院学报
Journal of Guangxi Academy of Sciences

2009,25(4):248~250,256
Vol. 25,No. 4 November 2009

平行语料库的相似语句去重算法*

Algorithm for Removing Similar Sentence on Parallel Corpus

申文明[1],黄家裕[2],刘连芳[1,2]
SHEN Wen-ming[1],HUANG Jia-yu[2],LIU Lian-fang[1,2]

(1.广西大学计算机与电子信息学院,广西南宁 530004;2.南宁平方软件新技术有限公司,广西南宁 530003)
(1. School of Computer, Electronic and Information, Guangxi University, Nanning, Guangxi, 530004, China; 2. Pingsoft New Technology Co. Ltd. of Naning, Nanning, Guangxi, 530004, China)

摘要:尝试对平行语料库中需要去重的中文句子相似情况作分类,利用整体相似因子和局部相似因子计算句子的相似度,并借鉴KMP算法的匹配跳跃思想,提出中文字符串匹配的类KMP算法,并对算法进行实验验证。结果表明,算法具有较好的效果,能够实现平行语料库中相似句子的去重。算法开放测试的召回率达94%,去重准确率达到84%。算法可以应用于任何长度的语句比对,适用范围广。

关键词:去重 相似句子 平行语料库 类KMP
中图法分类号:TP391.3 **文献标识码**:A **文章编号**:1002-7378(2009)04-0248-03

Abstract: The similarity of Chinese sentence is classified and duplicated sentence is removed. Sentence similarity depends on similarity of unitary factor and partial factor. According to the idea of KMP's jump, the simular KMP in chinese sentence is used. The experiment results show that the algorithm is effective, the recall rate of duplicate removal reach 94%, and the precision rate reach 84% in large scale testing.

Key words: duplicate removal, similar sentence, parallel corpus, similar KMP

平行语料库是语料库的一种,是由原文本及其对应的翻译文本构成的语料库[1]。平行语料库将原文和译文经过对齐处理,可以提取出翻译对应语,因而广泛应用于基于实例的机器翻译(EBMT)。平行语料库不仅可以用于机器翻译,还可以在人机交互翻译中给译员提供翻译范例,帮助译员快速、高质量地完成翻译工作。

平行语料库的语料对齐单位根据粒度的大小可以分为篇章、段落、句子、短语、词等多个档次,其中最为重要而且较为成熟的是基于句子的对齐。平行语料库中句子的质量关系着整个机器翻译系统的效果和效率,所以对原始语料句的筛选是保证语料库质量的重要前提。

当前,语料库语料的重要收集方法之一是通过网络自动获取,但是在网络中存在着大量重复或相似的句子。基于平行语料库的机器翻译系统所需要的双语句对一般在百万级以上,如果把这些冗余的句子放入平行语料库,不但会浪费存储资源,而且还会影响翻译系统的工作效率和翻译质量。因此在构建语料库的前期工作中,根据中文语句去掉大量重复或相似的句对是一项具有实际意义的工作。

句子去重就是在收集语料的时候把重复的或者极其相似的句子去除。对于完全重复的字串,陈桂林等通过字串的首字符和字串的长度进行Hash运算,将首字符或长度相等的字串聚成一个子类,然后对每个子类进行快速排序,对每个子类进行去重[2]。但是我们在语料库的收集中发现:不仅要去掉完全重复的句子,还需要去掉极其相似的句子。例如当一个句子中的某个分句或短语整体发生移动之后,它

收稿日期:2009-10-10
作者简介:申文明(1984-),男,硕士研究生,主要从事信息检索和自然语言处理方面的研究工作。
* 南宁市人才小高地基金项目(No. 2007007)资助。

与原来的句子仍很相似,此类相似句子往往对应相同或相近的译文,在语料库中同时保留这两个句子对翻译的贡献不大,所以对句子去重还要对相似的句子进行去重,而不只限于完全重复的句子。本文实现一种中文相似句子的去重算法,对平行语料库句对的去重具有较好的效率和效果。

1 相似句子类型

目前,对相似度还没有统一的定义,在不同的具体应用中,相似度的含义也有所不同。Dekang Lin给出了一个统一的、与应用领域无关的非形式化定义:A与B之间的相似度一方面与它们的共性相关,共性越多,相似度越高;另一方面与它们的区别相关,区别越大,相似度越低[3]。我们在本文中两个句子的相似度指两个句子的字符重复程度。根据句子相似的程度把确定需要去掉的句子分成4种类型:(1)句子完全重复。两个句子长度相等,而且每个位置上的字符也相等,两个句子完全一致。(2)句子包含。一个句子完全被另一个句子包含。例如:A:越南被视外国投资者的乐土。B:近几年来,越南被视为外国投资者的乐土。(3)句型变换。句子中短语位置调换,句型发生变化。例如:A:越南因为有大量廉价的劳动力才能吸引外国的投资。B:越南能吸引外国的投资是因为有大量廉价的劳动力。(4)小部分同义转换。句子中部分短语被同义词替换。例如:A:中国的上海在吸引外资方面独占鳌头。B:中国的上海在吸引外资方面首屈一指。

2 相似句子去重算法设计

2.1 相似句子去重算法

从待检测的文本中取一个句子 s_i 与当前所得的句子集合 $S(s_0,s_1,\cdots\cdots,s_{i-1})$ 中的句子逐个进行相似度计算,如果存在某一个句子 s_j 与 s_i 相似度超过某一个阈值,则 s_i 不能作为一个新的句子加入到 S,否则将 s_i 添加到 S 中。算法的过程如下:

(1)$k=0$;

(2)计算 s_i 和 s_k 的相似度,如果判定相似则退出循环并输出,否则转到(3);

(3)$k=k+1$,如果 k 小于 t 则转到(2),否则转到(4);

(4)将 s_i 加入 S 中,$t=t+1$ 并退出。

算法最为关键的是句子相似度的计算,王莹莹[4]通过计算短语间相匹配字符的位置和匹配位置的偏移值来计算中文短语的相似度。但是两个句子间存在短语的多次匹配和字符的连续匹配现象,显然针对短语的相似度计算方法并不适用于长度较长的句子。本算法利用字符串匹配的方法统计两个句子中字符重复的总个数和两个句子的最大相等字串,分别得到整体相似因子和局部相似因子,然后综合两个相似因子得到句子的相似度。算法可以应用于任何长度的语句比对,适用范围广。

2.2 句子相似度计算

2.2.1 定义

在下列定义中,s_i(i 为正整数)为待处理的完整语句。

定义1 句子整体相似因子 $EN_{ij}(s_i,s_j)=2\cdot PN_{ij}/(sl_i+sl_j)$。$EN_{ij}$ 为两个句子 s_i、s_j 中相同字符占两个句子的比例,取值范围[0,1];sl_i 为句子 i 的长度;sl_j 为句子 j 的长度;PN_{ij} 是两个句子 s_i、s_j 匹配时,短句子中的字符在长句子中出现的个数。

因为句子相对比较短,汉字出现重复的现象比较少,所以如果字符在长句中出现则仅统计1次。

考虑如下现象:一个短句的所有字符都在长句出现,但是语义完全不相关。例如:"天气好热"和"天上好像飞过一个热气球"。为了防止这种情况的出现,本文引入了局部相似因子。

定义2 句子局部相似因子 $SEN_{ij}(s_i,s_j)=PSN_{ij}/min(sl_i,sl_j)$。$SEN_{ij}$ 为两个句子 s_i、s_j 最大相同字符串占短句的比例,取值范围[0,1];PSN_{ij} 为两个句子 s_i、s_j 匹配时对应字符连续相等的最大个数;$min(sl_i,sl_j)$ 为两个句子中短句的长度。

定义3 句子相似度 $Sim_{ij}(s_i,s_j)=\lambda_1\cdot EN_{ij}(s_i,s_j)+\lambda_2\cdot SEN_{ij}(s_i,s_j)$,其中:$\lambda_1$ 和 λ_2 是属于[0,1]的常数,且满足 $\lambda_1+\lambda_2=1$。

两个句子 s_i、s_j 的相似度由整体相似因子 EN_{ij} 和局部相似因子 SEN_{ij} 来决定,通过参数 λ_1 和 λ_2 来调配它们在整个相似度计算中所占的比例。显然 $0\leqslant Sim_{ij}(s_i,s_j)\leqslant 1$。

2.2.2 算法步骤

判断两个句子是否相似的步骤如下。

(1)根据经验设定相似度阈值 t;

(2)提取两个句子 s_i、s_j;

(3)分别计算长度 sl_i,sl_j;

(4)计算 PN_{ij} 和 PSN_{ij};

(5)根据两个句子的比例设定平衡参数 λ_1 和 λ_2,计算两个句子的相似度 Sim_{ij},当 Sim_{ij} 大于设定的阈值 t 则句子 s_i、s_j 被认定相似,否则不相似。

2.3 中文字符串匹配的类 KMP 算法

为了提高计算两个句子的 PN 和 PSN 效率，我们借鉴 KMP 匹配跳跃的思想[5]，提出针对中文字符串匹配的类 KMP 算法，在匹配中也实现查找的跳跃。算法核心是把当前字符匹配查找循环中得到的字符连续相等的个数作为下一次循环的跳跃因子，减少循环次数，从而提高匹配查找的速度。在类 KMP 算法中，跳跃因子 *Jump* 取值为：

$$Jump=\begin{cases}1;x<=1\\x;x>1\end{cases}\quad x:\text{上一次循环时字符连续相等的个数}$$

类 KMP 算法的核心算法如下。

```
Similarity(String s1,String s2)
//类 KMP 算法用于计算 PN 和 PSN,其中 s1 为短句,s2 为长句
{
  for(i=0;i<s1.length;i+=Jump)
  {
    int flag=1;//是否出现匹配过的标志
      int Jump=1;//跳跃因子默认为 1
      for(j=0;j<s2.length;j++)
       if(s1[i]==s2[j]) //如果出现字符匹配
       {
          if(flag)
          {
            PN++;
            flag=0;//单个字符匹配只统计 1 次
          }
          icount=0;//统计连续匹配的个数,初值为 0
           While(i+icount<s1.length&&j+icount<s2.length&&s1[i+icount]==s2[j+icount])
          {
            icount++;
          }
           Jump=icount;//跳跃因子为连续匹配字串长度 icount
          PN=Jump-1+PN;
          if(icount>PSN)
             PSN=icount;//PSN 为连续匹配子串的最大值
          icount=0;//icount 重新置 0
       } //end if
  }  //end for
}
```

在最坏的情况下，两个句子没有连续相等的子串，计算时间是 $O(n^2)$；在最好情况下，两个句子完全重复，计算时间是 $O(n)$。算法的时间复杂度是 $O(n^2)$，不需要预处理。

3 算法试验和分析

为了验证算法的效率和效果，我们设计了算法程序进行算法试验。开发语言为 Java，硬件环境为：CPU PIV 2.4GB，内存 512MB。

3.1 算法效率分析

取一个长度为 20 个字符的句子和现有的所有句子做相似性计算，验证现有句子总量变化对算法效率的影响结果如表 1 所示。

表 1 现有句子总量变化对算法效率的影响

句子总数(万个)	总时间(ms)	*I/O* 消耗时间(ms)
0.2	188	
1	969	78
2	2078	109
5	4640	203
10	5247	688

从表 1 可以看出，*I/O* 的消耗时间随着句子总量的增加成线性增长。如果预先把当前的句子一次性写入内存，就不用每次从文件中取出句子，可以将大部分的 *I/O* 消耗转移到预处理阶段。

3.2 算法的有效性

有效性是算法的生命，这里用去重的召回率和准确率来衡量算法的有效性。我们将去重的重点放在短句和长句的比例在 0.2～1 之间的句子。经过多次试验将平衡参数 λ_1 和 λ_2 设置如表 2 所示。相似度的阈值默认设置为 0.5。第 1 组数据是笔者根据相似情况自己设计的样本 100 句；第 2 组、第 3 组、第 4 组数据是从互联网获得，分别是关于广西交通规划（7540 句）、中国足球（6840 句）、中国电影报道（4200 句）的句子。从实验结果（表 3）可以看出，算法的召回率较好，平均召回率为 91.5%，但是准确率平均只有 84.5。我们分析误判主要出现在以下几种情况：(1)当同义替换出现比较多；(2)句子太短；(3)句子很长而且包含不同语义的子句多。所以，在具体实践中为提高准确率，可以在进行相似检测的时候生成检测报表，报表中记录被系统认定为相似的句子编号和内容，然后人工判断是否相似，剔除误判的结果。

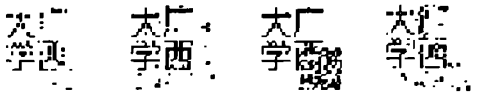
表 2 平衡参数 λ_1 和 λ_2 的设置

短句和长句之比	λ_1	λ_2
0.2～0.6	0.3	0.7
0.6～1.0	0.8	0.2

（下转第 256 页）

2.2 水印攻击检测

对已嵌入水印的图像进行JPEG压缩、添加椒盐噪声、剪切和中值滤波攻击，提取出来的水印如图5所示。



(a) (b) (c) (d)

图5 含水印图像受攻击后提取的水印

(a)QF=50%的JPEG压缩提取的水印(NC=0.8288)；(b)添加椒盐噪声提取的水印(NC=0.9965)；(c)裁剪右下角1/4提取的水印(NC=0.7755)；(d)3×3中值滤波提取的水印(NC=0.4998)。

从图5可以看出，本文所实现的数字水印算法对JPEG压缩、椒盐噪声和载剪有很好的鲁棒性，对中值滤波的鲁棒性一般，但是肉眼也能识别出来。

3 结束语

本文实现了一种基于DWT的非自适应数字水印算法，算法的MATLAB实现结果表明，该算法简单易懂，有一定的抗攻击能力。但是在实验中发现该算法对高斯噪声的鲁棒性很差，添加高斯噪声后水印基本检测不出来，还需要对该算法做进一步完善。

参考文献：

[1] 杨义先. 数字水印基础教程[M]. 北京：人民邮电出版社，2007.

[2] 董长虹，赖志国，余啸海. Matlab图像处理与应用[M]. 北京：国防工业出版社，2004.

[3] 金聪. 数字水印理论与技术[M]. 北京：清华大学出版社，2008.

[4] 黄达人，刘九芬，黄继武. 小波变换域图像水印嵌入对策和算法[J]. 软件学报，2002，13(7)：1290-1295.

(责任编辑：韦廷宗 邓大玉)

(上接第250页)

表3 算法的召回率与准确率试验结果

实验编号	相似句子数(个)	召回率	准确率
1	55	0.92	0.87
2	16	0.94	0.84
3	18	0.88	0.82
4	46	0.92	0.85

3.3 阈值对算法效果的影响

使用3.2中的第2组数据，改变阈值的大小来观察对算法召回率和准确率的影响结果如图1所示。从图1的实验结果表明，阈值在0.5～0.6的时候召回率和准确率达到一个较好的平衡，这和吴平博等[6]的试验结果基本一致。

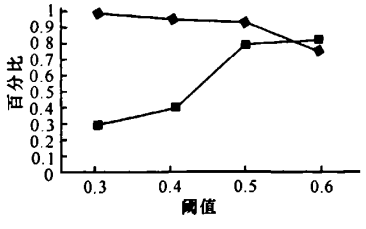


图1 阈值对算法效果的影响

-◆-：召回率；-■-：准确率。

4 结束语

在语料库收集过程中进行相似句子的去重是非常必要的，但是，目前国内尚缺少这类研究。本文尝试利用整体相似因子和局部相似因子计算句子的相似度，并借鉴KMP算法的匹配跳跃思想，提出中文字符串匹配的类KMP算法，对于完全重复的和极其相似的句子去重有很好的效果。由于本文的相似句子去重算法没有考虑到语义上的相似，因此还需要进一步改进算法，以提高去重的准确率。

参考文献：

[1] 黄俊红，范云. 双语平行语料库对齐技术述评[J]. 外语电教化，2007(118)：21-25.

[2] 陈桂林，王永成. 字串去重的快速算法研究[J]. 情报学报，2000，19(3)：254-258.

[3] Dekang Lin，Patrick Pantel. DIRT-Discovery of inference rules fromtext [J]. Journal of Natural Language Engineering，Fall-Winter，2001，1：1-6.

[4] 王莹莹. 中文短语相似度计算方法研究及应用[D]. 长沙：长沙理工大学硕士学位论文，2008：24-26.

[5] 严蔚敏，吴伟民. 数据结构(C语言版)[M]. 北京：清华大学出版社，1997：80-81.

[6] 吴平博，陈群秀，马亮. 基于特征串的大规模中文网页快速去重算法研究[J]. 中文信息学报，2003，17(2)：28-35.

(责任编辑：邓大玉)