(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105138514 A
(43)申请公布日 2015.12.09

(21)申请号 201510522091.X

(22)申请日 2015.08.24

(71)申请人 昆明理工大学
地址 650093 云南省昆明市五华区学府路253号

(72)发明人 彭艺 苏黎韡 邵玉斌 龙华 宋浩

(51)Int.Cl.
G06F 17/27(2006.01)

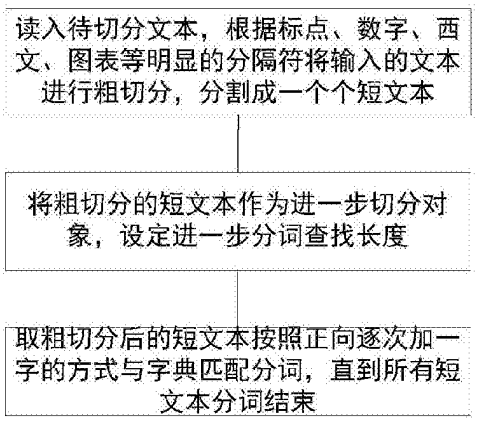
权利要求书1页 说明书7页 附图3页

(54)发明名称

一种基于词典的正向逐次加一字最大匹配中文分词方法

(57)摘要

本发明涉及一种基于词典的正向逐次加一字最大匹配中文分词方法，属于计算机中文文本处理技术领域。本发明包括步骤：首先读入待切分文本，根据标点、数字、西文、图表等明显的分隔符将输入的文本进行粗切分，分割成一个个短文本；将粗切分的短文本作为进一步切分对象，设定进一步分词查找长度；取粗切分后的短文本按照正向逐次加一字的方式与字典匹配分词，直到所有短文本分词结束。本发明避免了传统正向最大匹配分词速率—准确率难以平衡的缺点，在切词速度和分词准确率方面都比传统正向和逆向最大匹配分词算法有所提高。



CN 105138514 A

1.一种基于词典的正向逐次加一字最大匹配中文分词方法，其特征在于：所述基于词典的正向逐次加一字最大匹配中文分词方法的具体步骤如下：

Step1、读入待切分文本，根据标点、数字、西文、图表等明显的分隔符将输入的文本进行粗切分，分割成一个个短文本；

Step2、将粗切分的短文本作为进一步切分对象，设定进一步分词查找长度L，其中L取小于词典里最大词长的长度；

Step3、取粗切分后的一个短文本的起始两个字，在词典里查找匹配；

若不存在当前输入的两个字，则表示第一个字是单字，将其切分出去；

若存在当前输入的两个字，则将查找文本的长度指针往后增加一个字，增加到三个字，继续在词典里进行匹配；

若此三字词不存在，则表明前两个字是一个词，将其切分出去，作为一次切分的结果；接着分词查找指针后移，取后面两个词进行新一轮的查找匹配；

若此三字词存在，则继续往后增加一个字，构成四字词，查找此四字词是否存在于词典里，以此类推，进行匹配查找，从而进行分词；

Step4、当查找到查找长度为L时，从L的下一个字符开始，重新按照步骤Step3中以此类推的方法进行查找匹配以及分词，直到所有短文本分词结束。

一种基于词典的正向逐次加一字最大匹配中文分词方法

技术领域

[0001] 本发明涉及一种基于词典的正向逐次加一字最大匹配中文分词方法，属于计算机中文文本处理技术领域。

背景技术

[0002] 随着科技的发展，人类社会已经进入了信息时代。让计算机“读懂”人类的自然语言，实现自由的人机交互已成为美好的愿景。对于人类语言来说，词是最小的、能独立活动的、有意义的语言单位。中文和英语、法语等西方语言存在着很大差异，西文的字和字之间有明显的空格作为分隔符，计算机很容易根据这些空格而理解一句话的含义；而中文句子中词和词紧密排在一起，计算机理解起来就要困难的多。中文分词是汉语信息处理的关键和前提，只有处理好中文分词，才能让计算机理解中文、进行后续的中文信息处理，并从海量的信息中提取有用信息为人类提供服务，实现计算机智能化。随着中文信息处理的发展，中文分词技术得到了广泛的应用，大体上主要在下面三个领域中深入应用，起着关键的作用。1) 计算机和人工智能领域：利用中文分词成果从事自然语言理解和处理研究，如语义分析，自动摘要，知识工程，机器翻译，专家系统和智能计算机等；2) 情报信息领域：在研究中文分词与自动标引、中文分词与情报检索和搜索引擎等技术的结合上，取得了许多可喜的成绩。3) 汉语语言学研究领域：利用中文分词来促进汉语言文字研究，如研究汉语言的特点，与其它语言的比较，汉语言的规范等。

[0003] 中文分词是中文信息处理的基础环节，也是制约其发展的一个严重“瓶颈”。近年来，中文分词技术引起了社会各界尤其是公司和高校的重视和研究，出现了各种各样的分词方法：双向最大匹配法、逐词遍历法、设立切分标志法、词频统计法、扩充转移网络法、双向 Markov 链法、模糊聚类法、专家系统法、最少分词法、神经元网络法等多种分词方法。不同分词方法模拟了人类分词行为的不同侧面，服务于不同用途的中文信息处理系统。总的来说，这些方法都是三个基本方法的扩展、延伸和改进。这三个基本方法分别是：基于词典的分词方法、基于统计的分词方法和基于理解的分词方法，它们分别代表了目前分词方法的三大发展方向。

[0004] 正向最大匹配法 (Forward Maximum Matching Method)，所谓“最大”是指该算法总是把以某一汉字开头的尽可能长的字串看作是一个词语，即体现出“长词优先”。当在词典中找不到该字串时（即匹配不成功时），再去掉最后一个汉字继续查找匹配。该方法一般简称为 FMM 法。其算法思想为：设 D 为词典，L 表示 D 中的最大词长，S 为待切分的字串。每次从 S 中取出长度为 L 的子串 M 与 D 中的词进行匹配。若匹配成功，则将该子串 M 作为一个词切分出来，同时指针后移 L 个字符继续匹配；否则将子串 M 的最后一个字去掉，再按相同的方法进行匹配，直到切分出所有的词。传统正向和逆向最大匹配分词算法，需要事先设定一个匹配长度 M，一般以分词词典中的最大词长作为匹配长度进行分词。它强调的是“长词优先”，每次都要从 M 个字符开始匹配。若 M 过长，要查找多次才能切分出一个词，造成不必要的时间浪费，分词速度不高。而 M 过短，有一些词长超过 M 的长词就不能被正确的切分

出来，无法保证分词的准确率。

[0005] 为了解决上述传统正向匹配算法出现的不足，本文基于正向匹配算法提出了正向逐次加一字最大匹配算法，较好地完善了传统算法的不足。

发明内容

[0006] 本发明提供了一种基于词典的正向逐次加一字最大匹配中文分词方法，以用于解决传统正向最大匹配分词方法造成的分词速度慢，分词结果不精确等问题，本方法不需要预先设定最大匹配词长，避免了传统的最大匹配法因设定的最大匹配词长过长，而进行多次无用匹配，分词速度较慢；最大匹配词长过短，又无法正确切分的情况。

[0007] 本发明的技术方案是：一种基于词典的正向逐次加一字最大匹配中文分词方法的具体步骤如下：

[0008] Step1、读入待切分文本，根据标点、数字、西文、图表等明显的分隔符将输入的文本进行粗切分，分割成一个个短文本；

[0009] Step2、将粗切分的短文本作为进一步切分对象，设定进一步分词查找长度 L，其中 L 取小于词典里最大词长的长度；

[0010] Step3、取粗切分后的一个短文本的起始两个字，在词典里查找匹配；

[0011] 若不存在当前输入的两个字，则表示第一个字是单字，将其切分出去；

[0012] 若存在当前输入的两个字，则将查找文本的长度指针往后增加一个字，增加到三个字，继续在词典里进行匹配；

[0013] 若此三字词不存在，则表明前两个字是一个词，将其切分出去，作为一次切分的结果；接着分词查找指针后移，取后面两个词进行新一轮的查找匹配；

[0014] 若此三字词存在，则继续往后增加一个字，构成四字词，查找此四字词是否存在于词典里，以此类推，进行匹配查找，从而进行分词；

[0015] Step4、当查找到查找长度为 L 时，从 L 的下一个字符开始，重新按照步骤 Step3 中以此类推的方法进行查找匹配以及分词，直到所有短文本分词结束。

[0016] 本发明的有益效果是：

[0017] 1、本方法基于词典的匹配查找机制，对输入的待切分文本进行查找匹配，来确定分词结果。分词时不预先设定最大匹配词长，而是根据词典里最大词条长度来设定一个略小于最大词长的相应的查找长度 L，避免了传统的最大匹配法因设定的最大匹配词长过长，而进行多次无用匹配，分词速度较慢；最大匹配词长过短，又无法正确切分的情况；

[0018] 2、本方法在分词响应时间以及分词准确性方面得到很好的改进。对于测试文本，利用本发明的正向逐次加一字匹配分词方法与传统的基于词典的正向最大匹配分词，以及逆向最大匹配分词方法在分词性能方面进行了比较，无论是准确度还是分词时间都得展现出了很好的优势。

附图说明

[0019] 图 1 为本发明的流程图；

[0020] 图 2 为本发明中实施例 1 正向逐次加一字匹配分词方法流程图；

[0021] 图 3 为本发明中基于词典的正向逐次加一字匹配分词方法与传统基于词典的分

词方法的精确度对比图。

具体实施方式

[0022] 实施例 1：如图 1-3 所示，一种基于词典的正向逐次加一字最大匹配中文分词方法，所述方法的步骤为：

[0023] 步骤一、粗切分；对待切分的文本进行剔除标点符号、空格、日期、数字、英文字母等标记，将待处理的文本设为 A，分成 N 个短文本序列 S_i 的集合 ($0 < i \leqslant N$)，即切分为 S_i 个短文本，$A = \{S_1, S_2, S_3, \ldots S_N\}$；

[0024] 步骤二、如图 2 所示，依次按顺序读入一个个粗切分后的短文本，记为 S_i，设每个句子序列 S_i 由 m 个字 W_{ij} ($0 < j \leqslant m$) 组成，即 $S_i = \langle W_{i1}W_{i2}W_{i3}\ldots W_{im}\rangle$；

[0025] 步骤三、将粗切分后的文本 S_i 进行分词。如图 2 所示，将文本进行分词处理。

[0026] 1) 设定一个略小于词典里最大词长的分词查找长度 L，L 一般略小于词典里最大词长；

[0027] 2) 在短文本 S_i 中顺序取起始前两个相邻的字符 $W_{ij}W_{i(j+1)}$，初始时为 $W_{i1}W_{i2}$，在词典中查找匹配，若当前输入的两个字 $W_{ij}W_{i(j+1)}$ 不是词典中的词，则转 (3)；否则，转 (4)；

[0028] 3) 若当前输入的两个字 $W_{ij}W_{i(j+1)}$ 在词典中不存在，则表明前两个字中的第一个字是一个词，将 W_{ij} 从句子 S_i 中切分出去。判断是否到 S_i 句尾，若是，则 S_i 分词结束；否则 j = j+1，再转 (2)；

[0029] 4) 若存在当前输入的两个字 $W_{ij}W_{i(j+1)}$，则将查找文本的长度指针往后增加一个字，即 $W_{ij}W_{i(j+1)}$ 后加一字，增加到三个字，得到 $S_k = W_{ij}W_{i(j+1)}W_{ik}$ ($o < k \leqslant L$)，继续在词典里进行匹配，判断新读入的词是否存在于词典中。若存在，则转 (5)，否则，转 (6)；

[0030] 5) 若此三字词 $S_k = W_{ij}W_{i(j+1)}W_{ik}$ 存在，若此三字词存在，则继续将指针往 $S_k = W_{ij}W_{i(j+1)}W_{ik}$ 后增加一个字，构成四字词 $S_{k+1} = W_{ij}W_{i(j+1)}\ldots W_{ik}W_{i(k+1)}$，查找此四字词 $S_{k+1} = W_{ij}W_{i(j+1)}\ldots W_{ik}W_{i(k+1)}$ 是否存在于词典里，若是，则继续往后逐次加一字再判断，转 (7)；若不是，则把 S_k 切分出去，放入分词结果；

[0031] 6) 若此三字词 $S_k = W_{ij}W_{i(j+1)}W_{ik}$ 不存在，则表明前两个字 $W_{ij}W_{i(j+1)}$ 是一个词，将 $W_{ij}W_{i(j+1)}$ 从 S_i 中其切分出去，接着分词查找指针后移，使指针 j = j+2，再取后面两个词进行新一轮的查找匹配。若 $j \leqslant m$，表明当前短文本还未完全切分，转 (2)，若指针 j = m，则短文本 S_i 分词结束；

[0032] 7) 依此类推，每次移动分词指针时判断移动之后读入的当前词数 $k \leqslant L$ 是否成立，若成立，则继续在 $S_{k+1} = W_{ij}W_{i(j+1)}\ldots W_{ik}W_{i(k+1)}$ 后逐次加一字进行判断；否则从 $W_{i(k+1)}$ 处开始取两字字符进行下一轮查找匹配。

[0033] 步骤四、判断读入文本数 $i \leqslant N$ 是否成立，若成立，表明当前文本还未分词结束，则分词指针增加一，i = i+1，读入下一个句子重新按照上面的程序进行查找匹配以及分词，进行分词直到整个输入文本分词结束；否则，说明整个文本分词结束。

[0034] 实施例 2：如图 1-3 所示，一种基于词典的正向逐次加一字最大匹配中文分词方法，所述方法的步骤为：

[0035] 设定一个略小于词典里最大词长的分词查找长度 L；设待切分字符串为 $S = s_1s_2s_3s_4\ldots s_i$。从句头开始，取前两个字符 s_1s_2，判断 s_1s_2 是否是词典里的一个词，若不是，则

说明 s_1是单字词，将其切分出去，则将查找文本的长度指针往后增加一个字，增加到第三个字，取在词典中 s_2s_3进行新一轮的查找匹配；若 s_1s_2是词典中的词，则往后增加一个字，判断 $s_1s_2s_3$是否成词，若 $s_1s_2s_3$不是词典里的词，则表明 s_1s_2是一个词，将其切分出去；若 $s_1s_2s_3$是词典里的一个词，则继续往后增加一个字，查找 $s_1s_2s_3s_4$是否是词典里的词，若不是词，则将 $s_1s_2s_3$作为一个词切分出去，若是词典里的词，则继续往后增加一个词再来匹配。依此类推，直到整个句子 $S = s_1s_2s_3s_4...s_i$切分完毕。

[0036] 实施例 3：如图 1-3 所示，一种基于词典的正向逐次加一字最大匹配中文分词方法，所述方法的步骤为：

[0037] Step1、读入待切分文本，根据标点、数字、西文、图表等明显的分隔符将输入的文本进行粗切分，分割成一个个短文本；例如分成一个文本“今天天气特别的好”；

[0038] Step2、将粗切分的短文本作为进一步切分对象，设定进一步分词查找长度 L = 7，其中 L 取小于词典里最大词长的长度，其中最大词长为 12；

[0039] Step3、取粗切分后的一个短文本的起始两个字“今天”，在词典里查找匹配；经匹配“今天”存在于词典中，那么查找文本的长度指针往后增加一个字，增加到三个字“今天天”，继续在词典里进行匹配；经匹配“今天天”不存在，则表明“今天”是一个词，那么把“今天”切分出去，作为一次切分的结果；接着分词查找指针后移，取后面两个词“天气”进行新一轮的查找匹配；经匹配“天气”存在，那么查找文本的长度指针往后增加一个字，增加到三个字“天气特”，继续在词典里进行匹配；经匹配“天气特”不存在，则表明“天气”是一个词，那么把“天气”切分出去，作为一次切分的结果；依次类推，进行匹配查找，从而进行分词，分词的结果为 / 今天 / 天气 / 特别 / 的 / 好 /；具体分词的过程见表 1 所示；

[0040] 表 1 正向逐次加一字最大匹配分词过程

[0041]

匹配字段	匹配经过	匹配结果
今天	词典中存在	今天
天气	词典中存在	天气
特别	词典中存在	特别
的好	词典中不存在	
的	单字词	的
好	单字词	好

[0042] 为了验证本方法的有益效果，用本方法与传统的正向最大匹配分词方法、逆向最大匹配分词方法（一次最大匹配字符长度为 4）进行对比，传统的正向最大匹配分词方法、逆向最大匹配分词方法的分词过程如表 2、表 3 所示；

[0043] 1）正向最大匹配分词方法：

[0044] 表 2 正向最大匹配分词过程

[0045]

匹配字段	匹配经过	匹配结果
今天天气	词典中不存在	
今天天	词典中不存在	
今天	词典中存在	今天
天气特别	词典中不存在	
天气特	词典中不存在	
天气	词典中存在	天气
特别的好	词典中不存在	
特别的	词典中不存在	
特别	词典中存在	特别
的好	词典中不存在	
的	单字词	的
好	单字词	好

[0046] 正向最大匹配的结果是：/ 今天 / 天气 / 特别 / 的 / 好 /

[0047] 2) 逆向最大匹配分词方法：由右至左从待切分字符串中取子串进行匹配；

[0048] 表 3 逆向最大匹配分词过程

[0049]

匹配字字段	匹配经过	匹配结果
特别的好	词典中不存在	
别的好	词典中不存在	
的好	词典中不存在	
好	单字词	好
气特别的	词典中不存在	
特别的	词典中不存在	
别的	词典中不存在	

的	单字词	的
天气特别	词典中不存在	
气特别	词典中不存在	
今天天气	词典中不存在	
天天气	词典中不存在	
天气	词典中存在	天气
今天	词典中存在	今天

[0050] 逆向最大匹配的结果是：/ 今天 / 天气 / 特别 / 的 / 好 /

[0051] 从上述三种方法的分词过程可以看出，虽然最终的分词结果都是相同的、正确的，但是从分词的过程上可以清楚的看到传统的基于词典的正向、逆向最大匹配方法的分词过程都出现了读入词不存在的重复匹配步骤，浪费了分词的时间，造成分词后词典匹配、歧义判断的工作量。而本发明提出的正向逐次加一字最大匹配方法，几乎每个两字词都得到了一步分词的快速、准确分词，这样分词的整体效率就得到了很大的提高，试验仿真的结论也证明了这一点，如下表 4 所示。

[0052] 表 4 三种分词方法的平均切分速率比较

[0053]

分词方法	平均切分速度（字 /s）
传统的正向最大匹配法	52000
传统的逆向最大匹配法	103000
正向逐次加一字匹配法	113000

[0054] 将三种方法应用到本发明的试验环境中去，以一个完整的包含 27 万个词条的词库作为分词词典，在硬件采用计算机内存 1G 及以上，软件为 Windows7，使用 JAVA 开发语言，My Eclipse 8.5 开发工具的运行环境下进行模拟实验。选取了经济、科技、社会新闻、军事四个方面大小均为 0.02M 左右的文章，利用三种不同的分词算法进行分词，得到的结果如图 3 所示，纵坐标表示分词准确率，横坐标表示分词的领域，可以看到在这三种分词方法中，本文提出的正向逐次加一字匹配方法和传统的正向、逆向最大匹配分词方法相比，准确率均得到了提高。

[0055] 以上实施例的实验结论表 4，图 3 均能表明本发明的一种基于词典的正向逐次加一字最大匹配分词方法较传统的基于词典的分词方法在分词切分速度，分词准确率方面都有很显著的改进。

[0056] 上面结合附图对本发明的具体实施方式作了详细说明，但是本发明并不限于上述实施方式，在本领域普通技术人员所具备的知识范围内，还可以在不脱离本发明宗旨的前

提下作出各种变化。

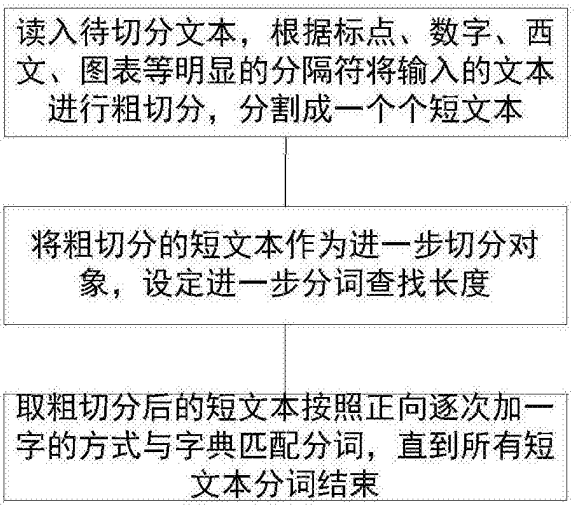

读入待切分文本，根据标点、数字、西文、图表等明显的分隔符将输入的文本进行粗切分，分割成一个个短文本
将粗切分的短文本作为进一步切分对象，设定进一步分词查找长度
取粗切分后的短文本按照正向逐次加一字的方式与字典匹配分词，直到所有短文本分词结束

图 1

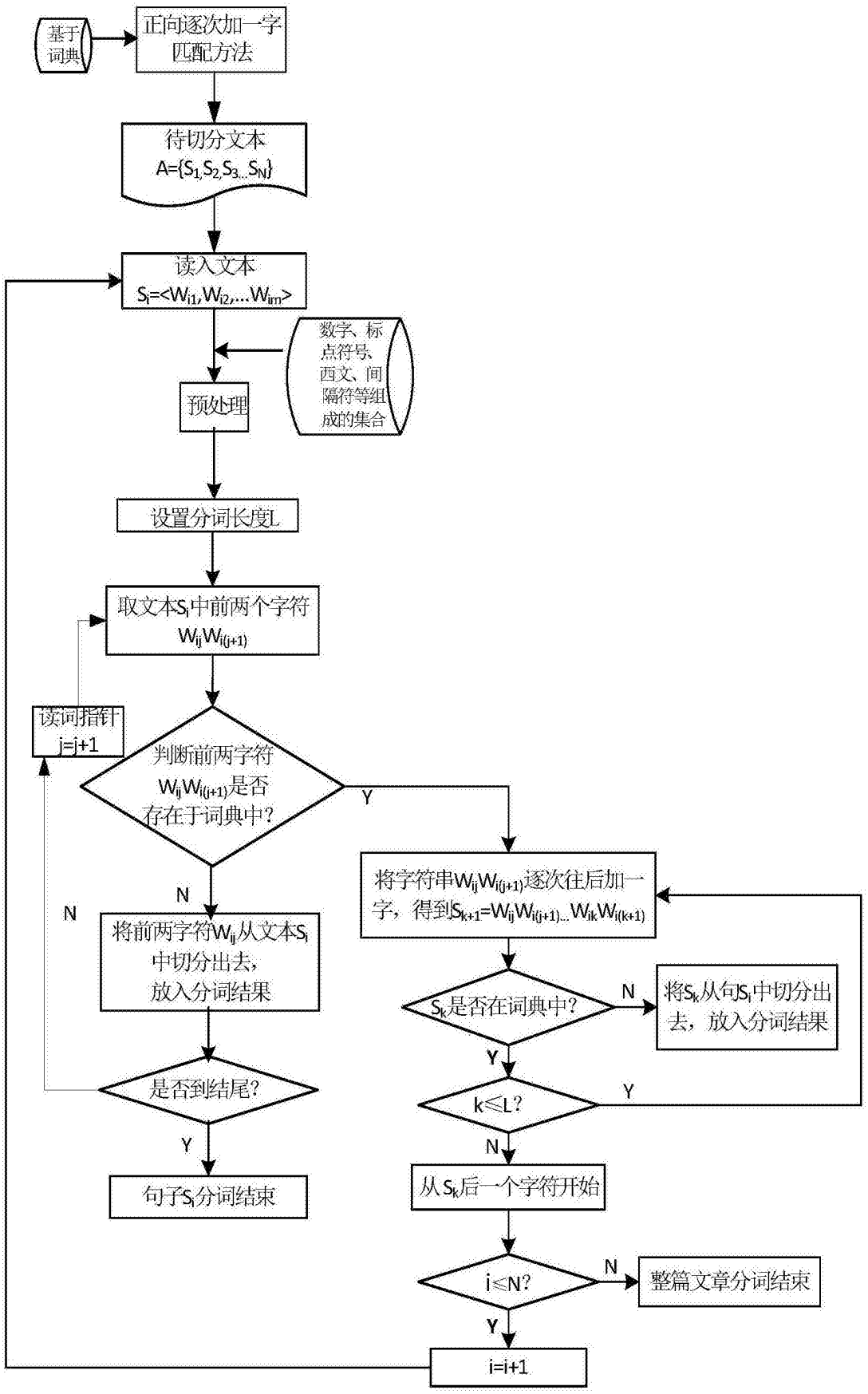

基于词典
正向逐次加一字匹配方法
待切分文本 A={S1,S2,S3...SN}
读入文本 Si=<Wi1,Wi2,...Wim>
数字、标点符号、西文、间隔符等组成的集合
预处理
设置分词长度L
取文本Si中前两个字符 WijWi(j+1)
读词指针 j=j+1
判断前两字符 WijWi(j+1)是否存在于词典中?
Y
N
N
将前两字符Wij从文本Si中切分出去，放入分词结果
是否到结尾?
Y
句子Si分词结束
将字符串WijWi(j+1)逐次往后加一字，得到Sk+1=WijWi(j+1)...WikWi(k+1)
Sk是否在词典中?
N
将Sk从句Si中切分出去，放入分词结果
Y
k≤L?
Y
N
从Sk后一个字符开始
i≤N?
N
整篇文章分词结束
Y
i=i+1

图 2

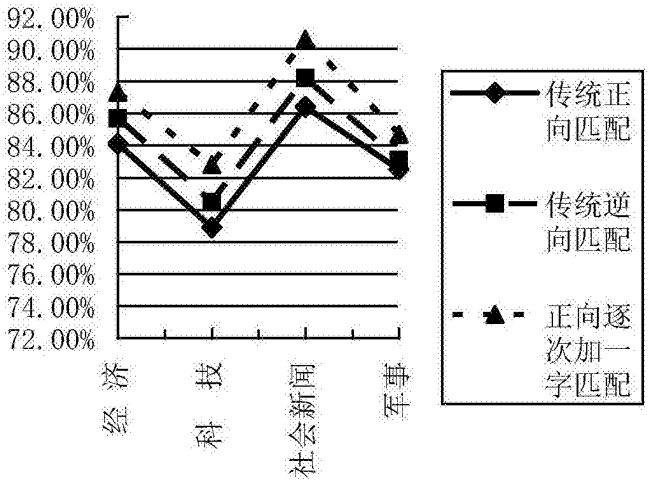

92.00%
90.00%
88.00%
86.00%
84.00%
82.00%
80.00%
78.00%
76.00%
74.00%
72.00%
经济
科技
社会新闻
军事
传统正向匹配
传统逆向匹配
正向逐次加一字匹配

图 3