基于实例的语法检查研究

谢坤武

(湖北民族学院 信息工程学院,湖北 恩施 445000)

摘要:随着计算机和互联网的迅速发展,成千上万的用户每天日常工作中都必须用英语写作、交流.对于母语不是英语的用户来说,英语写作是一大障碍.语法检查的技术起源于自然语言理解的应用,根据实用性和有效性原则出发,提出在系统实现中引入错误实例、负规则模式来实现语法检查,从而简化了分析算法,扩展了错误检查的覆盖面.

关键词:数据库;实例;模式匹配;语法检查

中图分类号: TP311 **文献标识码**: A **文章编号**: 1008 - 8423(2009)01 - 0040 - 04

An Grammar Check Research Based on Instance

XIE Kun - wu

(School of Information Engineering, Hubei University for Nationalities, Enshi 445000, China)

Abstract: With the rapid development of computers and Internet, tens of thousands of users tend to write and communicate in English in their daily work. For users whose mother tongue is not English, writing in Enshi is a major obstacle to them. Grammar check technology originated in the application of natural language understanding. This paper, according to the principle of practicality and effectiveness is to introduce the system error instantiation, the negative model to realize the rules of grammar check, which simplifies the analysis of algorithms and extends the coverage of error - check.

Key words: database; instantiation; pattern matching; grammar check

对于学习英语写作的人们而言,刚开始起步时,常常会犯一些常见的英语语法错误.如果在写作时能及时指出所犯的错误,并提供修改建议与范例,则可以更快的提高他们的英语水平和写作能力. Microsoft Word中的拼写检查可以检查出文档中的某些语法错误,但是检查出的语法错误类型有限,例如检查不出副词修饰名词这种常见的语法错误,这对刚学习英语写作的人帮助不大.目前,语法错误检查运用的方法主要有基于统计的方法[1],基于规则的方法[2,3]等,它们在提高整个校对系统的召回率和精确率方面起到了一定的作用.但这些系统的语法校对方法也存在着一些不足:统计方法中使用的词类邻接矩阵的 n元模型,只能反映局部的语法限制,而不包括长距离的语法限制;规则方法中句法的自底向上或自顶向下的分析方法不能发现特定词的搭配错误,同时耗费的系统开销也很大.文尝试利用挖掘语法错误的信息和特点,提出了一种模式匹配的语法错误检查的方法,在不遗漏有用语法信息的前提下,简化句法分析算法.

1 模式的表示与存储

1.1 实例和负规则模式

在机器翻译领域使用基于实例和规则相结合的方法取得了较大的成功.在语法检查系统中,实例的引入

收稿日期: 2008 - 12 - 10.
基金项目:湖北省教育厅教学研究项目 (2008297).
作者简介:谢坤武 (1970 -),男,硕士,副教授,主要从事数据挖掘、知识发现的研究.

有其可行性吗?答案是肯定的.首先从心理学角度来说有其可行性,因为人们通常都会有一定的思维惯性,同一个错误下次再犯的可能性会非常高.这样就可以用实例对这些错误用法进行模式化.其次,中国人书写的英语尤其是初学者所产生错误的特点是中文英语(Chinese English),即通常把中国人的思维和习惯用法套用在英文的使用中,这种错误并无规则可言. Awell and Elliott[4]研究发现 55%的错误都是局部语法错误,18%是全局语法错误,全局语法错误需要对整个句子的结构进行深入分析,27%的错误是因为语意不正确所产生的错误,这类错误可以根据其局部上下文的信息就可以做出判断,即通过有序的单词串或词类组合来判断.以下是从一些学生的英语练习中提取的一些典型的语法错误.

(1) He catch a cold after come to Hong Kong,　(2) I don't know when do they come;
(3) I can find good job opportunity,　(4) I can find really good opportunity,
(5) The world is getting more smaller and smaller,　(6) The level of my english has thus been improved,
(7) improve the living level for the people.

寓错误语法知识于巨量的单词序列串中,这种短语串称之为完全实例模式,简称实例.把这些模式都存储在数据库中,通过实例模式搜索引擎来完成句中语法错误的查找.用实例来描述语法错误可以无需使用复杂规则和约束,因为这些实例表达实际隐含着某些复杂的错误语法规则.但错误实例可以被认为有无限多的,如果都是完全实例化的错误就会造成数据库中数据量的膨胀,不利于错误模式的检索匹配,而且即使大量的实例模式对于错误语法的覆盖也仍是有限的.如下列的实例模式:

(1) * made the writer felt, (2) * makes the writer felt, (3) * make the writer felt,
(4) * made them felt,　(5) * makes her felt,　(6) * make him felt,
(7) * made us felt,　(8) * makes us felt,　(9) * make us felt,
(10) * made you felt,　(11) * makes you felt

这些还仅仅是对 make这个单词的错误使用所产生的部分实例.所以很自然就会想到使用词性和句子成分类对实例中某些部分进行抽象.如果把上述实例称为实例模式的话,引入一个称为负规则模式:单词与词性或者语段的顺序组合串模式.通过这种模式来消除大量的信息冗余,从而减少实例模式的数量.这种模式不仅利用了浅层分析的结果而且还利用了词汇本身所具有的信息,也就是说,一些需要全局分析才能检查到的语法错误也能由这种模式所覆盖.对于上面的实例模式可以通过负规则模式来表示:

* make NX(名词词组)VBN (动词完成式), * makes NX(名词词组) VBN (动词完成式)

在上述设定条件下,语法错误的检查首先根据词性和浅层分析后的语段标识,通过调用负规则模式搜索引擎在模式库中查找匹配,如果匹配则找到语法错误.但如果完全用词类或者语段标识作为负规则模式的符号描述,就有可能把一些非错误的情况包含进来,如上例负规则 * make NX VBN, 若将 make也使用其相应的词类"动词"来代替,形成新的负规则模式:VB(动词)NX VBN, 则会出现通用化过度的问题.因为有些动词就会允许有这样的表达如 have my hair cut 因为词类是个高度概括的特征信息,从某种意义来说是一组等价类的集合,但等价类中的一些单词在英语语法的使用中会出现特殊性质.所以这涉及到在负规则模式中哪一部分应该使用词类或者语段类型.

系统实现中采用对词性进行子分类和具体单词化相结合的方法来解决这种通用化过渡的问题.首先对那些不具有通用语法规律的封闭词类或语段进行具体化,如介词,对于一些属于开放词类,且经常出现的错误的语法使用,也将该部分进行具体化为词汇表达.如上例中的 make用法,这里主要是因为动词的用法并不都具有这种形式,而对该词的错误用法的统计,发现出现的频率很高,那么将预先设定的负规则模式进行部分实例化:VB(动词) NX VBN.

取其中的 VB(动词)具体化为 make等.对词性的子分类方式可以提供更明晰的词汇特征信息,有利于判断单词的用法,如对动词类子分为及物动词类和不及物动词类,这为判断动词相关的语法错误时提供了依据.其次对于因词性标记造成的不正确词性标识,选取它的对应单词来代替,也就是在负规则模式中对相应的词类部分用单词来表示.

1.2　模式的存储

在数据库中对实例和负规则模式采用统一的处理方式进行存储.具体的数据字典如下:

1.2.1　用户端

(1) Category(categoryId, category, explanation, hyperlink, example)

(2) Error(errorid, error, suggestion, length, categoryid)

(3) Time(time)

说明:Category表存储的是各类语法错误的类别名(category)、对错误的解释(explanation)、举例(example)、对该错误类别更详细的指导的网络链接地址(hyperlink);Error表中 error存储的是由错误字符串所提取的实例字符串或负规则模式串,suggestion是对该实例提供的更正建议. length的值即标识串长度,categoryid使得 Error表与 category表联系起来;Time表维护当前数据库中各个表数据的最新更新时间.

1.2.2 服务器端

(1) privilege(Id, password, prior)

(2) Category(categoryId, category, explanation, hyperlink, example, time)

(3) Error(errorid, error, suggestion, length, categoryid, time)

(4) Time(time)

(5) DeleteData(id, tableid, time)

说明:privilege表存储了登陆用户的验证信息. (2),(3)表结构在客户端对应表结构的基础上再增加一个更新时间段 time.

DeleteData表中的数据是记录(2),(3)表中的删除信息:id记录 tableid所指的表中已经删除的模式索引(errorid). tableid指向(2),(3)这两类表. time记录该实例删除的时间. (4),(5)这两个表及(2),(3)中的time字段的设定都是为了保持客户端与服务器端数据的一致性.

2 错误检查引擎

定义 1 前缀子串匹配:设 X为主标记串,Y为任意标记串,若 Y与 X的前 k个标记串相同,其中 $|Y| = k \leq |X|$, $|x|$表示标记串 x的标记数(标记指单词符号或者语段符号),则称 Y为 X的前缀子串,即 Y前缀子串匹配 X.

错误检查流程图见图 1,具体的描述如下:

(1)对文本首先切分为句,然后对句子的每一个单词进行词性的标识. 结果是单词和词性一一对应的字符串序列. 如: the/DT horse/NN raced/VBD past/JJ the/DT barn/NN fell/VBD ./.

(2)对每一句使用基于分层的有限状态自动机进行语段的分析. 下面的例子显示了一个分析的结果,其中分割符"[ ]"包括的部分即句子中解析出的语段:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only $1.8 billion] [PP in] [NP September].

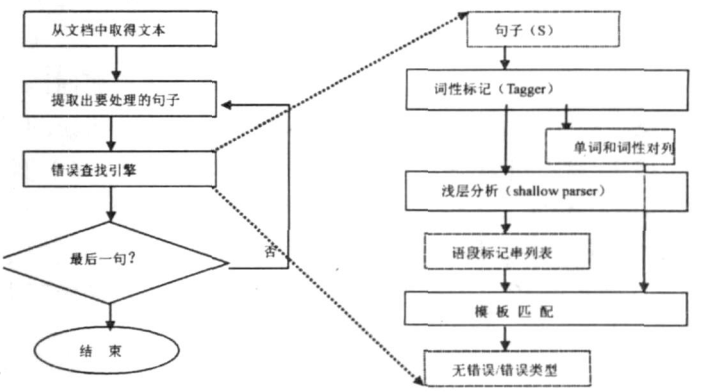

图 1 检查引擎流程图

Fig 1 Check engine flow chart

系统中所采用的浅层分析只是运用了低层次的有限状态自动机来分析句子的主要语段,对于那些无法用语段标记表示的部分用词类来代替,即单词语段. 对于有限状态自动机所使用的词类,采用 Penn Treebank[5]标记集中的词类为标准,这个标记集不但体现了词汇的类型属性,还包含有词汇的其他特征属性信息. 如对动词的分类:

* VB:动词原型, * VBD:动词过去式, * VBG:动词现在分词形式, * VBN:动词过去分词形式,

* VBP:动词现在时、非第三人称单数形式, * VBZ:动词现在时、第三人称单数形式.

因为在系统的浅层分析中并不需要产生一个覆盖全句的成分 S,所以最终会是一些预先定义的语段的顺序连接列表. 如上面的句子分析结果就是一个这样的列表:[NP] [VP] [NP] [VP] [PP] [NP] [PP] [NP]

(3)模式匹配查找

模式匹配查找需要处理两种情况,一种是完全实例模式的匹配查找,一种是负规则模式的匹配查找.

以句为处理单元的匹配算法描述如下:

* 根据句子字符串 S获得 WS="$W_1/T_1\ W_2/T_2\cdots W_n/T_n$"; CS="$C_1\ C_2\cdots C_m$"

其中 WS是指单词和词性所形成的串,W_i为单词串,T_i为词性标注符号;

CS是指句子语段标识符号串列表,$C_i \rightarrow T_{i1}\ T_{i2}\cdots T_{ik}$;

Tr(C_i)="$W_{i1}\ W_{i2}\cdots W_{ik}$", $i_1 \geq 1$ and $ik \leq n$, n为该句所包含的单词数.

* 定义错误范围值 span; 错误发现标志 F = false;

* while i < m do

{ // i初始为 1.

调用查找函数 FindError(Tr(Ci),m, i, true);

若 F = false,则调用 FindError(Ci,m, i, true);

若 F = true,则认为发现一错误,提示相关信息,由用户选择相应错误修改操作, i = i + span;否则 i = i + 1.

}

其中 FindError函数定义如下:

FindError(string pattern, int m, int j, bool flag)

/* pattern将要匹配的子串, m为句子切分的语段数, j为当前处理语段在 CS串中的索引, flag控制标记, 定义字符串 subStp =Φ (Φ 为空白符号); */{

将 pattern与数据库中模式进行前缀子串匹配 Match(pattern);

If匹配成功 then {

j = j + 1;当 j < m时: {

subStp = pattern;

pattern = pattern + Tr(Cj);

FindError(pattern, m, j, false);

If F! = true then

pattern = subStp + Cj; FindError(pattern, m, j, true); }

}

Else {

if flag! = true then

if | pattern| = = 错误模式主串长度 + 1 then

保存错误范围值 span ∈[0,m], F = true,函数返回;

}

}

3 总结

本文在实例匹配的基础上引入分析的技术,这种方法同时考虑了语法的短距离和长距离的限制信息,提高了系统的语法检查能力,克服了单纯采用基于实例匹配方法的不足,同时又避免了现有的检查系统所使用的复杂分析技术,扩展了错误检查的覆盖面.进一步研究将集中在错误模式的自动获取,随着错误模式的增加,手工收集和编写错误规则的工作量会越来越大,将机器学习功能增加进来,可以更好地发现错误规则.

参考文献:

[1] De Marcken G G. Parsing the LOB Corpus[C]//28th Annual Meeting of the ACL. Proceeding of the conference, June, 1990.

[2] Eric Brill A simple rule - based part of speech tagger[C]//Proceedngs of the Third Conference on Applied Natural Language Processng, 1992.

[3] 曹文君,费晓明.一个基于规则的程序语法错误诊断系统[J].计算机研究与发展, 1989(1): 24 - 30.

[4] Atwell E, Elliott S. Dealing with Ill - Formed English Text[C]//The Computational Analysis of English: a Corpus - Based Approach, Longman, 1987.

[5] Merialdo M, Santorini B, Marcinkiewicz M. Building a large annotated corpus of English: The Penn Treebank[J]. Computational Linguistics, 1993, 19(2): 313 - 330.