从中文Web网页中获取实体简称的研究

丁远钧[1,2] 曹存根[2] 王 石[2] 符建辉[2]
(首都师范大学计算机科学联合研究院 北京 100037)[1]
(中国科学院计算技术研究所智能信息处理重点实验室 北京 100190)[2]

摘 要 简称是自然语言词汇的重要组成部分,其获取是自然语言处理中的一个基本而又关键的问题。提出了一种根据汉语全称从Web中获取对应汉语简称的方法。该方法包括获取和验证两个步骤。获取步骤通过选择查询模式从Web上获得候选简称集合。为了验证候选简称,定义了全简称关系约束,分别定性和定量地表示全称和对应简称之间的约束,构建了全简称关系图来表示所有全称和简称之间的联系,在验证过程中,先分别用约束公理和关系图对候选简称进行过滤,再用约束函数对候选简称分类,并以分类类别、语料标记和约束函数值作为属性构建决策树,利用决策树对候选简称进行验证。实验结果表明,获取方法的最终准确率为94.63%,召回率为84.09%,验证方法的准确率为94.81%。

关键词 自然语言处理,简称获取,约束公理,约束函数,关系图
中图法分类号 TP391.1 **文献标识码** B

Extracting Abbreviated Names for Chinese Entities from the Web

DING Yuan-jun[1,2] CAO Cun-gen[2] WANG Shi[2] FU Jian-hui[2]
(Joint Institute of Computer Science, Capital Normal University, Beijing 100037, China)[1]
(Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Abbreviations are the essential parts of the vocabularies in natural language, therefore, acquiring abbreviations is a basic and significant task of natural language processing. We proposed a method of extracting abbreviations for the given Chinese full names from the Web. The method has two phases: candidate abbreviations extraction and verification. In the extraction phase, we constructed query items to issue to Google and saved the results as the corpora, from which we extracted candidate abbreviations. In the verification phase, we defined a full names/abbreviations relations constraints, which includes a group of constraint axioms and a group of constraint functions. We built a relation graph to reflect the connection of all the full names and the abbreviations. In the process of verifying, the incorrect ones could be filtered out using the constraint axioms and the relation graph; the candidate abbreviations could be classified with the constraint functions, and the incorrect ones could be identified through a classifier, which was trained using the types of the candidate abbreviations, the values of the functions and the tags in the corpora. Comprehensive experiments show that the precision and recall rate of extracting abbreviation extraction are 94.63% and 84.09%, respectively, and the precision of candidate verification is 94.81%.

Keywords Natural language processing, Abbreviation acquisition, Constraint axioms, Constraint functions, Relation group

1 引言

自然语言的经济性原则和人们表达的简洁习惯共同导致了简称的出现,如"中国科学院"简称"中科院"。对于简称这个术语,学术界有不同的提法,如"缩略语"、"省语"等,与之相应,学术界对其定义也是说法各异。比较之后,我们认为袁晖的《现代汉语缩略语词典》一书中的定义比较好:"为了表达的简洁明快,将较长的词语进行简缩而形成的一个短小的词语称为缩略语。"[1]

在日常生活中,人们有时使用全称,有时使用简称,数据在数据库中的保存形式其全称和简称都有,因此在检索、查询或者数据处理时,往往需要将全称和简称对应起来,故获取已知全称对应的简称就很重要。

全称(记为 Fn)是对概念的完整称呼,简称(记为 An)是为了表达的简洁明快而对全称进行精简压缩后得到的称呼,同一概念的全称和简称构成一对全简称关系。若 Fn 和 An

到稿日期:2011-10-15 返修日期:2012-01-09 本文受国家自然科学基金(61173063,61035004),国家社科基金(10AYY003)资助。
丁远钧(1986-),男,主要研究方向大规模知识获取,E-mail:bestdyj@yahoo.com.cn;曹存根(1964-),男,研究员,博士生导师,主要研究方向为知识获取与共享、文本挖掘;王 石(1981-),男,助理研究员,主要研究方向为知识获取与共享、文本挖掘;符建辉(1985-),男,主要研究方向为知识获取。

具有全简称关系，则称 *Fn* 为 *An* 的全称，*An* 为 *Fn* 的简称，记作 *FA*(*Fn*,*An*)。由全称到简称，可以看作是一个信息量的压缩过程，由简称到全称，则可以看作是一个信息解压的过程，例如：对 *C*1="中国科学院计算技术研究所"进行压缩，得到 *C*2="中国科学院计算所"，再对 *C*2 进行压缩，得到 *C*3="中科院计算所"，对 *C*3 解压得到 *C*2，再对 *C*2 解压得到 *C*1。全称和简称都是相对的概念，例如在上述例子中，*C*2 相对于 *C*1 是简称，但相对于 *C*3 却是全称，单独讲 *C*2 是全称或简称都是没有意义的。

从 *Fn* 到 *An* 的压缩过程中一般都遵守一些缩略规则，因此，根据简称的形成规律和构词形式，简称可以分成缩节式、删节式、缩合式、数合式和特殊式。缩节式：选取全称中每个分词的一个或多个语素而形成的简称，如 *Fn*="中国科学院"，*An*="中科院"；删节式：选取保留全称中一个或多个分词的语素，删去其余次要分词形成的缩略语，如 *Fn*="第二次世界大战"，*An*="二战"；缩合式：对有共同语素的并列词语，缩减合并其不同成分的语素，并保留其共同语素形成的缩略语，如 *Fn*="工业、农业"，*An*="工农业"；数合式：全称中并列成分项数和有代表的共同语素组合形成的简称，如 *Fn*="最高人民法院、最高人民检察院"，*An*="两高"；特殊式：缩略方式和以上几类不同，如省名的简称、古代的数合式简称、音译名的简称等。

全简称关系获取作为文本知识获取(Knowledge Acquisition from Text，KAT)中一个基本而又关键的问题，其获取方法可以分为两大类：一类是基于模式的方法，主要利用语言学和自然语言处理技术，通过词法分析和语法分析提取关系模式，然后利用模式匹配获取全简称关系；另一类是基于统计的方法，主要基于语料库和统计语言模型，通过计算概念之间的关联度来获取全简称关系[2]。全简称关系的获取问题又可以从两个角度来看：一个是挖掘的角度，就是在没有外界输入的条件下获取全简称对；另一个是查找的角度，就是已知全称找简称或已知简称找全称。本文是从查找的角度来获取汉语简称的。

在关于简称获取的相关研究中，崔世起等提出一种在生语料中自动抽取中文缩略语的方法，其首先获取候选缩略语集和源短语库，然后利用语言模型和对齐模型等特征进行候选缩略语的对齐，最后得到一部粗糙的缩略语词典[3]。谢丽星等采用用户查询日志和锚文字文件，运用"同网站主题相关性"的思想进行初步的缩略语、源短语对的抽取，然后采用一系列过滤规则，结合分词按照缩略语的形成方式进行分类，最后调用搜索引擎采用多策略来识别缩略语、源短语对[4]。武子英等提出一种在生语料中自动抽取现代汉语缩略语的方法。首先获取候选缩略语的源短语候选集，然后利用基于上下文的源短语与缩略语配对方法，可自动生成一部缩略语词典[5]。以上 3 种方法都只限定获取缩节式和删节式的简称，而且获取的准确率都不高，不适合大规模的获取。田国刚等提出一种多特征约束的方法，从大规模的中文文本语料中获取同指关系[6]，该方法虽然能获取特殊式的简称，但获取对象是同指关系，而全简称关系只是同指关系中的一种，所以缺乏对全简称的获取进行进一步的研究。姜广等提出一种从 Web 网页中获取汉语全简称的方法，该方法先利用查找模式从 Google 中获取语料，再利用全简称提取算法从语料中提取出全简称，最后对获取的全简称对进行验证[7]。该方法相比之前的研究有了很大的进步，主要体现在：其一，获取的范围进一步放宽，不仅能获取缩节式、删节式的简称，而且能获取特殊式的简称；其二，该方法从 Google 搜索引擎中获取语料，语料资源非常丰富，适合多学科、大规模获取。但姜广的研究缺乏进一步的深入，准确率和召回率离实用尚有较大的差距，可改进和完善之处主要体现在：其一，查询模式不全面导致召回率偏低；其二，对语料缺乏足够的分析；其三，提取全简称的算法太笼统从而导致提取的结果不理想；其四，对获取到的全简称关系缺乏足够的验证和分析。

在关于简称分类的相关研究中，语言学家们对简称的本体研究较为充分，提出了很多不同的分类体系，但这些研究学术性强，而且是针对人的，不太适合计算机自动分类。在关于计算机可以实现的简称自动分类的研究中，支流等在研究简称还原的实验中，设置了一套适合计算机识别和处理的缩略语的分类框架[8]。鲍明凌等从类型、缩略方式、构成方式、语法结构几个方面对缩略语进行了分类，并用数据库详细统计了缩略语在音节、类型、缩略方式、语法结构类型、构成方式几方面的特征，以期为中文信息处理服务[9]。但以上研究只是初步探索，均未给出用计算机对简称进行分类的具体算法。

本文给出了一种根据汉语全称从 Web 中获取对应汉语简称的方法。该方法包括获取和验证两个步骤。获取步骤通过选择查询模式来构造查询项，将查询项提交到 Google 搜索引擎中获取全简称语料，再利用提取算法从全简称语料中提取出候选简称，从而形成全称对应的候选简称集合。为了验证候选简称，本文定义了全简称关系约束，包括一组约束公理和一组约束函数，其中，约束公理定性地表示全称和对应简称之间的约束，约束函数定量地表示全称和对应简称之间的约束，文中还构建了全简称关系图来表示所有全称和简称之间的联系。在验证过程中，先分别用约束公理和关系图对候选简称进行过滤，再用约束函数对候选简称分类，并以分类类别、语料标记和约束函数值作为属性，利用 J48 算法[10]构建决策树，利用决策树对候选简称进行验证。

本文第 2 节介绍从 Web 网页中抽取候选简称；第 3 节介绍候选简称的验证；第 4 节介绍实验与分析；最后是总结及未来的工作。

2 从 Web 网页中抽取候选简称

2.1 获取锚语料

我们经常会在提到一个概念词时紧接着给出它的简称来解释说明，比如，"……河北省科学技术协会(简称河北省科协)，……"。从上一句话中，可以比较容易地获取到一对全简称关系 FA(河北省科学技术协会，河北省科协)，值得庆幸的是，这种类型的语料在网页中大量出现，为我们获取全简称关系提供了丰富的语料资源。为了获取这种类型的语料来构成语料库，本文中构造了 3 种查询模式(见表 1)。

表 1 查询模式

模式编号	具体内容
查询模式 1	"Fn 简称"
查询模式 2	"Fn * 简称"
查询模式 3	"全称 Fn"

查询模式 1 和查询模式 2 之间的区别是查询模式 2 中多

加了一个“*”,这个“*”能匹配一个词。有一种语言现象:书面表达时,人们喜欢在“简称”之前加上一些例如“下面”、“下文”之类的方位词。这种情况下基于查询模式1就无法查找到这些结果,例如:要查找 *Fn*=“校学位评定委员会下设学位办公室”的简称,查询模式1就没有搜索结果,而查询模式2却有5条结果。

对于任一给定的 *Fn*,我们可以基于查询模式来构造具体的查询项(注意:查询项必须包含一对双引号,以使引号内的内容在搜索结果中能紧挨着出现),比如“河北省科学技术协会简称”,再将查询项提交到 Google 搜索引擎中搜索,然后将搜索结果的前100条的锚文本保存下来作为该查询项的锚语料(若搜索结果少于100条则全部保存下来)。

我们以3910个汉语 *Fn* 做实验,其中用查询模式1能获取到 *An* 的占64.65%,用查询模式2能获取到 *An* 的占61.18%,用查询模式3能获取到 *An* 的占21.02%,用查询模式1或查询模式2能获取到 *An* 的占82.51%,用查询模式1−3能获取到 *An* 的占84.27%。因此,为了兼顾效率和召回率,优先选择查询模式1,其次查询模式2,最后查询模式3。

2.2 获取全简称句子

对于每个查询项的每个锚文本,我们将其中包含查询项的句子称为全简称句子,对于每个查询模式,我们相应地分别构造一种抽取模式,从锚文本中抽取全简称句子。

抽取模式1:(对应查询模式1)“Prefix+*Fn*+Pun? + Ref+Pun? + Co-referent+Pun”

抽取模式2:(对应查询模式2)“Prefix+*Fn*+Pun? + Ori+Ref+Pun? + Co-referent+Pun”

抽取模式3:(对应查询模式3)“Co-referent+Pun? + Eng? + Pun? + Ref^{-1}+Pun? + Fn+Pun? + Suffix”

其中,Ref 指全简称关系词“简称”,Ref^{-1}指全简称关系词“全称”,Fn 指已知全称,Prefix 指出现在 *Fn* 前的短语或短句(可为空),Pun 指标点符号,Pun? 指标点符号或空,Ori 指方位词(“下”、“以下”、“下面”……),Co-referent 指待提取的短语或短句,Eng? 指英文单词串(可为空),Suffix 指出现在 *Fn* 之后的短语或短句(可为空)。

关于获取语料和抽取关键句子的例子见表2。

表2 获取语料和抽取关键句子的例子

查询模式	查询模式1:“Fn 简称”
查询项	“科学技术协会简称”
锚文本	河北省科学技术协会(简称河北省科协),英文全称:Hebei Association for Science and Technology (HAST)是河北省科学技术工作者的群众组织,是中共河北省委领导下的...
全简称句子	河北省科学技术协会(简称河北省科协)

2.3 提取候选简称集

由一个已知的 *Fn* 从 Web 上获取对应的 *An*,往往能获取到几个结果,这些结果称为 *Fn* 对应的候选简称 *Can*,*Fn* 对应的所有 *Can* 构成 *Fn* 的候选简称集 *CanSet*。针对不同的查询模式,本文构造了两种不同的简称提取算法来提取 *CanSet*。候选简称集提取算法1适用于查询模式1和查询模式2的情形,候选简称集提取算法2适用于查询模式3的情形。

2.3.1 候选简称集提取算法1

通过对查询模式1和查询模式2所获得的全简称语料进行分析,我们发现全简称句子有一定的结构,故在候选简称集提取算法1中根据结构的不同将全简称句子分为6种类型:半标号型、后部分型、多合一型、标号对型、无前缀型和有前缀型。从这6种类型的全简称句子中提取出的候选简称,其语料类型(tag)为相应的全简称句子的类型,参见算法 CANSEA1。

CANSEA1 的输入为利用查询模式1和查询模式2所获取的 Web 预料。下面针对算法中涉及的几种类型做详细解释。

半标号型:*Can* 的左右两边只有一边有配对符号,说明该句子很可能不包含完整的 *An*。例如,利用查询模式1查询 *Fn*=“最高人民检察院”,得到全简称句子:最高人民检察院(简称“高〈*b*〉。产生这种错误的原因是获取锚语料时没有完整地获取整个句子。

后部分型:在全简称句子中,*Fn* 是另一全称“*Fn*”的后部分,故 *Can* 也是“*Fn*”对应的简称“*Can*”的后部分,由于过度缩减,*Can* 很可能不是 *Fn* 的简称。例如,利用查询模式1查询 *Fn*=“胸腔积液”,得到全简称句子:化脓性胸腔积液(简称脓胸)。在上一条全简称句子中,“脓胸”是“化脓性胸腔积液”的简称,但由于过度缩减,“胸”不是“胸腔积液”的简称。但有些情况下不存在过度缩减的问题,例如,利用查询模式1查询 *Fn*=“最高人民检察院”,得到全简称句子:中华人民共和国最高人民检察院(简称中国高检)。在上一条全简称句子中,“中国高检”是“中华人民共和国最高人民检察院”的简称,但其中的“高检”也是“最高人民检察院”的简称。所以,我们需要进一步研究如何判断有没有过度缩减。

多合一型:*Fn* 作为整体的成分与另外的全称一起出现,整体的简称是几个全称的合并式简称。例如,利用查询模式1查询 *Fn*=“最高人民检察院”,得到全简称句子:最高人民法院与最高人民检察院(简称两高)。在上一条全简称句子中,“最高人民检察院”与“最高人民法院”组成一个整体,“两高”是整体的简称。这种语料的结构有一个明显特征:*Fn* 是整体的最后部分且在 *Fn* 前有“和”、“与”、“及”等连接词。

标号对型:*Fn* 前面无汉字,且 *Can* 被配对符号标出,无需利用算法再确定 *Can* 的边界,直接提取。例如,利用查询模式1查询 *Fn*=“最高人民检察院”,得到全简称句子:最高人民检察院(简称“高检”)。

无前缀型:*Fn* 前面无汉字,且 *Can* 未被配对符号所标出,*Can* 无需确定左边界,但需要确定右边界。例如,利用查询模式1查询 *Fn*=“最高人民检察院”,得到全简称句子:最高人民检察院简称高检成立于1954年。

有前缀型:*Fn* 前面有汉字,*Can* 需要确定左边界和右边界。例如,利用查询模式1查询 *Fn*=“最高人民检察院”,得到全简称句子:贾春旺当选为最高人民检察院(简称高检)检察长。

候选简称提取算法1的具体内容如下:

候选简称提取算法1(candidate abbreviation set extract algorithm CANSEA1)

输入:Fn 对应的全简称句子集合 Sent(利用查询模式1和查询模式2获取的)

输出:Fn 对应的带语料类型 tag 的候选简称集 CanSet

Step1 从 Sent 中任取一全简称句子 fa_sent,利用2.2节中的抽取模式1或2抽取出 fa_sent 中的 Fn、Prefix 和 Co-referent,从 Sent 中删除 fa_sent

Step2 记 Co-referent 的单字表示为 Co-referent = $P_1P_2\cdots P_n$,其中

P_i 代表一个汉字，定义候选简称 Can←null，定义 Can 的语料类型 tag←null，定义 Can 在 Co-referent 中的左边界 left←1 和右边界 right←n

Step3 if Co-referent 左边是配对标号 and 右边不是对应的配对标号
then tag←半标号型
end if

Step4 if Prefix = null
if tag = null
then tag←无前缀型
end if
转 Step7
end if
if Prefix! = null and tag = null
then tag←有前缀型
end if

Step5 if Prefix 的最后一个字是“和”或“与”或“及”
then for each $P_i \in \{P_1P_2\cdots\cdots P_n\}$
if P_i 不在 Fn 中出现
then tag←多合一型；转 Step6
end if
end for each
end if

Step6 for each $P_i \in \{P_1P_2\cdots P_n\}$
if P_i 不在 Fn 中出现 and P_i 在 Prefix 中出现
then left←i+1
end if
if P_i 在 Fn 中出现
then break；
end if
end for each
if left>1 and tag=有前缀型
then tag←后部分型
end if

Step7 if Co-referent 被标号对标出 and tag =无前缀型
then tag←标号对型
end if

Step8 for each $P_i \in \{P_{left}P_{left+1}\cdots P_{n-1}\}$
if P_i 在 Fn 的最后一个分词中出现 and P_{i+1}不在 Fn 中出现
then right←i
end if
end for each

Step9 Can←$P_{left}P_{left+1}\cdots\cdots P_{right}$
CanSet←Can

Step10 if Sent 非空
then 转 Step1
else return CanSet

CANSEA1 是从用查询模式 1 和查询模式 2 获取的全简称句子集合中逐条取出全简称句子，再从全简称句子中提取出候选简称加入候选简称集合，故算法的关键部分是从相应的全简称句子中提取出候选简称。

2.3.2 候选简称集提取算法 2

针对查询模式 3 所获得的预料，我们提出了候选简称集提取算法 2(CANSEA2)，其具体内容如下：

候选简称提取算法 2(candidate abbreviation set extract algorithm CANSEA2)

输入：Fn 对应的全简称句子集合 Sent（利用查询模式 3 获取的）

输出：Fn 对应的带语料类型 tag 的候选简称集 CanSet

Step1 从 Sent 中任取一全简称句子 fa_sent，利用 2.2 节中的抽取模式 3 抽取出 fa_sent 中的 Fn、Suffix 和 Co-referent，从 Sent 中删除 fa_sent

Step2 对 Co-referent 和 Suffix 分别分词并且标注词性，分词结果分别为：$\{P_1P_2\cdots P_k\}$和$\{R_1R_2\cdots\cdots R_n\}$，定义候选简称 Can←null，定义 Can 在 Co-referent 中的一级左边界下标 left1←1，二级左边界下标 left2←1，左边界下标 left←1 和右边界下标 right←k，定义动词可截取标志 flag_v←0，右边界根据词性可截取标志 flag_right←0

Step3 for each $P_i \in \{P_1P_2\cdots P_k\}$
if P_i 和 Fn 有相同的字 and flag_v =0
then flag_v←1；
//P_i 之后的动词都不可以作为左边界
end if
if P_i 和 Fn 有相同的字 and left2 = 1
then left2←i；
//P_i 可能是 Can 的第一个分词
end if
if P_i 的词性为“连词”或“介词”或“助词”或“代词”
then left1←i+1；
end if
if P_i 的词性为“动词”and flag_v = 0
then left1←i+1；
end if
end for each

Step4 for each $P_j \in \{P_kP_{k-1}\cdots P_1\}$
if P_j 和 Fn 有相同的字 and flag_right=0
then flag_right←1；
//P_j 可能是 Can 的一个分词
end if
if P_j 的词性为“连词”或“介词”或“助词”或“动词”and flag_right=0
then right←j−1；
end if
if P_j 和 Suffix 有相同的字 and P_j和 Fn 无相同的字
then right←j−1；
end if
end for each

Step5 if left2 <= right
then left←left2
end if
if left1 <= right
then left←left1
end if

Step6 Can ←{$P_{left}P_{left+1}\cdots P_{right}$}
CanSet←Can

Step7 if Sent 非空
then 转 Step1
else return CanSet

CANSEA2 是从用查询模式 3 获取的全简称句子集合中逐条取出全简称句子，再从全简称句子中提取出候选简称加入候选简称集合，故算法的关键部分是从相应的全简称句子中提取出候选简称。

3 对候选简称的验证

3.1 全简称关系约束

全称和简称之间存在着一定的约束，这些约束的强弱程度不同，我们从语义和统计两个角度来分析，定义约束函数来

定量地表示全称和简称之间的约束，定义语义公理来定性地表示全称和简称之间的约束。

定义 1 全简称关系约束是一个四元组 $R=(Fn,An,F,A)$，其中，Fn 是全称，An 是简称，F 是 Fn 和 An 之间的约束函数集，A 是 Fn 和 An 必须满足的约束公理集。

在对约束函数集和约束公理集进行详细说明前，列出在本章中使用到的基本符号：

- Fn 表示全称；
- Can 表示 Fn 的候选简称；
- An 表示 Fn 的简称；
- $GoogleArchSet(Fn)$ 表示 Fn 的 Google 锚文本集，即从 Google 中查找 Fn 对应的简称时所返回的前 100 条锚文本的集合，若返回的锚文本总数 N 少于 100，则 $GoogleArchSet(Fn)$ 只包含仅有的 N 条锚文本；
- $CanSet(Fn)$ 表示 Fn 的候选简称集，即从 $GoogleArchSet(Fn)$ 中提取出的 Fn 对应的候选简称组成的集合；
- $N\_CanSet(Fn)$ 表示 $CanSet(Fn)$ 中所含候选简称的个数；
- $FnSet(Can)$ 表示候选简称 Can 对应的全称集，即 $FnSet(Can)$ 中的每一个 Fn 的候选简称集中都含有 Can；
- $N\_FnSet(Can)$ 表示 $FnSet(Can)$ 中所含全称的个数；
- $FA(Fn,An)$ 表示 Fn 和 An 具有全简称关系；
- $length(str)$ 表示汉字串 str 中所含汉字的个数；
- $n\_word(Fn,An)$ 表示同时出现在 Fn 和 An 中的汉字个数；
- $N\_Clas(Fn)$ 表示 Fn 经过分词后，出现的分词个数；
- $N\_Cover(Fn,An)$ 表示 Fn 中被 An 覆盖到的分词个数；
- $CoverSet(Fn,An)$ 表示 Fn 中被 An 覆盖到的分词的集合；
- p_i：表示全称中的第 i 个分词；
- $p_1/p_2/\cdots/p_m$：表示由分词 $p_1,p_2,\cdots,p_m$ 组成的分词序列，其中/表示分词间的分隔符；
- $centre(Fn)$ 表示 Fn 的分词中心点的位置，即 Fn 经过分词后，最中间的那个分词的位置，或最中间的那两个分词的平均位置，$centre(Fn)=(N\_Clas(Fn)+1)/2$；
- $d_i(Fn)$ 表示 Fn 的第 i 个分词 p_i 的中心偏移量，即 Fn 的分词中心点的位置与 Fn 的第 i 个分词的位置之间的位移，$d_i(Fn)=i-centre(Fn)$；
- (Fn) 表示 Fn 的最大中心偏移量，即 Fn 的所有分词的中心偏移量的最大值，$(Fn)=(N\_Clas(Fn)-1)/2$；
- $Len_i(Fn,An)$ 表示第 i 个未被覆盖分词串所含的分词数。对 Fn 进行分词后，未被 An 覆盖到的那些分词，如果在 Fn 中相联则组成未被覆盖分词串，如果不相联则单独成串，第 i 个未被覆盖分词串所含的分词个数记为 $Len_i(Fn,An)$；
- $freq(Fn,An)$ 表示从 $GoogleArchSet(Fn)$ 中提取出的 An 的个数；
- ε 表示一个无穷小的数；
- $loca(Fn,Can)$ 表示 Can 在 $CanSet(Fn)$ 中的频度次序，即对 $CanSet(Fn)$ 中的元素按 $freq(Fn,Can)$ 的大小降序排序后，Can 在其中的次序；
- $NoInclude(s_1,Set)$ 表示汉字串的集合 Set 中的任何汉字串都不是汉字串 s_1 的子串；
- $Interrogative$ 表示疑问词集合，包含“什么”、“怎么”、“啥”、“吗”等疑问词；
- $concat(s_1,s_2)$ 表示汉字串 s_1 和汉字串 s_2 连接后的汉字串；
- $NumIn(s,c)$ 表示汉字 c 在汉字串 s 中出现的次数。

下面对约束函数集中的约束函数的具体含义进行说明。

约束函数 1　An 的字来自 Fn 中的比率

一般情况下，全称包含简称所包括的所有汉字。例如，Fn=“北京大学”，An=“北大”，An 中的每个汉字都来自于 Fn 中。在候选简称集中，出现在 Fn 中的字的比率越高的候选简称的优先级越高。

约束函数 1 的形式定义和计算如下，该函数直接引用自文献[12]：

$$f_1(Fn,Can)=\frac{n\_word(Fn,Can)}{length(Can)}$$

例如，Fn=“孔子庙”，Can_1=“孔庙”，Can_2=“文庙”。根据约束函数 1，有 $f_1(Fn,Can_1)>f_1(Fn,Can_2)$，所以 Can_1 的优先级比 Can_2 的优先级高。

约束函数 2　Fn 与 An 的语序

在缩略过程中，绝大多数简称保持着全称中的字序。例如，Fn=“奥林匹克运动会”，An=“奥运会”，An 中的 3 个字的顺序严格按照 Fn 中出现的顺序排列。

约束函数 2 的形式定义和计算如下，该函数直接引用自文献[12]：

$$f_2(Fn,Can)=\begin{cases}1, & Fn\text{与}Can\text{语序一致}\\ 0, & \text{否则}\end{cases}$$

注意：Fn 与 An 语序相同蕴含着 An 中的所有字都出现在 Fn 中，若 An 中有不出现在 Fn 中的字，则约束函数 2 的值为 0。

约束函数 3　An 对 Fn 的分词覆盖率

全称通常由多个分词组成，有的情况下全称的一个或多个分词在简称中可以被省略，但一般被省略的分词个数不会超出全称分词数的二分之一，候选简称覆盖全称的分词越多，就越可能成为简称。

约束函数 3 的形式定义和计算如下：

$$f_3(Fn,Can)=\begin{cases}1-\log_{10}\log_{10}\left[\dfrac{N\_Clas(Fn)}{N\_Cover(Fn,Can)}\right] & \text{条件 1}\\ 0 & \text{条件 2}\end{cases}$$

其中：

条件 1：$1\leqslant\dfrac{N_{Clas(Fn)}}{N_{Cover(Fn,Can)}}<10$

条件 2：$1\leqslant\dfrac{N_{Clas(Fn)}}{N_{Cover(Fn,Can)}}<10$

该函数来自对发明专利[12]的改进，原公式为：

$$f(Fn,Can)=1-\log\left[\frac{N\_Clas(Fn)}{N\_Cover(Fn,Can)}\right]$$

约束函数 4　An 对 Fn 的分词覆盖重心

全称通常由多个分词组成，有的情况下全称中的一个或多个分词可以在简称中被省略，但是被省略的分词应该均匀地分布在全称中，而不应该都集中在全称的前部分或后部分。例如，Fn=“中国/贵州/航空/工业/集团/公司”，An=“贵航集团”，Fn 中省略的分词“中国”、“工业”、“公司”分别在 Fn 的

前部分、中间部分和后部分。

约束函数 4 的形式定义和计算如下：

$$f_4(Fn,Can)=\frac{\sum_i d_i(Fn)}{N\_Cover(Fn,Can)\times l(Fn)}$$

式中，d_i 对应的 $p_i\in CoverSet(Fn,Can)$。

约束函数 5　Fn 中未被 An 覆盖到的最长连续分词数

全称通常由多个分词组成，有的情况下全称中的一个或多个分词可以在简称中被省略，但是被省略的分词在全称中通常不会连续出现，即全称中的分词连续在简称中省略的概率比较小。

约束函数 5 的形式定义和计算如下：

$$f_5(Fn,Can)=1-\frac{\max\{Len_1(Fn,Can),\cdots,Len_N(Fn,Can)\}}{N\_Clas(Fn)}$$

式中，N 表示 Fn 中所含有的未被覆盖分词串的个数。

约束函数 6　Fn 和 An 的长度关系

通常规范的候选简称不会过度缩减，以保证多数人能见名知意。因而多数简称对应的全称长度在一个范围内，一般是简称长度的 1.5～5 倍，全称长度超出这一范围的概率较小。

约束函数 6 的形式定义和计算如下：

$$f_6(Fn,Can)=\begin{cases}0 & \text{条件 1}\\ 0.5 & \text{条件 2}\\ 0.8 & \text{条件 3}\\ 1-\log_{10}\log_{10}\left[\dfrac{length(Fn)}{2\times length(Can)}\right] & \text{条件 4}\end{cases}$$

其中：

条件 1：$length(Fn)/length(Can)\leqslant 1$

条件 2：$1<length(Fn)/length(Can)\leqslant 1.5$

条件 3：$1.5<length(Fn)/length(Can)<2$

条件 4：$length(Fn)/length(Can)\geqslant 2$

该函数来自对发明专利[12]的改进，原公式为：

$$f(Fn,Can)=\begin{cases}0 & \text{条件 1}\\ 0.5 & \text{条件 2}\\ 1-\log\log\left[\dfrac{length(Fn)}{2\times length(Can)}\right] & \text{条件 3}\end{cases}$$

其中：

条件 1：$length(Fn)/length(Can)\leqslant 1$

条件 2：$1<length(Fn)/length(Can)<2$

条件 3：$length(Fn)/length(Can)\geqslant 2$

约束函数 7　An 在 $GoogleArchSet(Fn)$ 中出现的频度

由全称到 Google 上查找简称时，在 $GoogleArchSet(Fn)$ 中出现频度越高的候选简称的优先级越高。

约束函数 7 的形式定义和计算如下：

$$f_7(Fn,Can)=1-\left(\frac{freq(Fn,Can)-100}{100}\right)^2$$

例如，Fn ＝“锂离子电池”，Can_1 ＝“锂电池”，Can_2 ＝“锂电”，$freq(Cfn_1)=42$，$freq(Cfn_2)=12$，根据约束函数 7，$f_7(Fn,Can_1)>f_7(Fn,Can_2)$，所以 Can_1 的优先级比 Can_2 的优先级高。

由 Fn 查找 An 时，有时会得到几个候选简称，它们构成候选简称集 $CanSet(Fn)$，对于 $CanSet(Fn)$ 中的任意一个候选简称 Can_i，分析 $FA(Fn,Can_i)$ 时可以类比 $CanSet(Fn)$ 中其它候选简称的指标值。

下面的 4 个约束函数是基于候选简称集定义的。

约束函数 8　Can 的字来自 Fn 中的相对比率

与约束函数 1 相比，约束函数 8 强调 Can 在 $CanSet(Fn)$ 中的相对性，比如，有些外来音译词汇的简称和全称就没有相同的字，有些简称还原成全称时进行了一些同义转化等等。

约束函数 8 的形式定义和计算如下：

$$f_8(Fn,Can)=\frac{n\_word(Fn,Can)+\varepsilon}{\max\{n\_word(Fn,Can_j)\}+\varepsilon}$$

式中，$Can,Can_j\in CanSet(Fn)$。

例如，Fn ＝“夫子庙”，Can_1 ＝“孔庙”，Can_2 ＝“文庙”，虽然 $f_1(Fn,Can_1)$ 只有 0.5，但是 $f_1(Fn,Can_2)$ 也只有 0.5，所以不能因为 $f_1(Fn,Can_1)$ 的值低就判定 Cfn_1 不是正确的候选简称。

约束函数 9　Can 对 Fn 的相对覆盖率

与约束函数 3 相比，约束函数 9 强调 Can 在 $CanSet(Fn)$ 中的相对性，比如，有些候选简称对全称的覆盖率都不高，那么覆盖率相对高的候选简称的优先级更高。

约束函数 9 的形式定义和计算如下：

$$f_9(Fn,Can)=\frac{N\_Cover(Fn,Can)+\varepsilon}{\max\max\{N\_Cover(Fn,Can_j)\}+\varepsilon}$$

式中，$Can,Can_j\in CanSet(Fn)$。

约束函数 10　Can 在候选简称集中的频率

由 Fn 查找 Can 时，有时候选简称集中所有的候选简称的频度都非常低，那么约束函数 7 的约束作用就被淡化，所以约束函数 9 考虑各候选简称的相对频度，在候选简称集中，频度相对高的候选简称的优先级更高。

约束函数 10 的形式定义和计算如下：

$$f_{10}(Fn,Can)=\frac{freq(Fn,Can)}{\sum_j freq(Fn,Can_j)}$$

式中，$Can,Can_j\in CanSet(Fn)$。

例如，Fn ＝“自治区扶贫开发领导小组办公室”，Can_1 ＝“自治区扶贫办”，Can_2 ＝“扶贫办”，$freq(Cfn_1)=3$，$freq(Cfn_2)=1$，虽然根据约束函数 7，Cfn_1 和 Cfn_2 的频度都比较低，但是根据约束函数 10，Cfn_1 和 Cfn_2 在候选简称集中的频率都比较高。

约束函数 11　候选简称集中的元素按照频度升序排序后，Can 在其中的相对位置

当候选简称集中的元素比较多时，频度比较低的候选的重要性相对比较低。

约束函数 11 的形式定义和计算如下：

$$f_{11}(Fn,Can)=\frac{N\_CanSet(Fn)-loca(Fn,Can)+1}{N\_CanSet(Fn)\times\log_{10}(loca(Fn,Can)+9)}$$

约束函数 11 的值越低的候选简称的重要性越低。

以上对约束函数集中的约束函数的具体含义进行了说明，它们定量地表示了 Fn 和 Can 之间的约束，而约束公理则定性地表示 Fn 和 An 之间的约束，下面对约束公理进行具体说明：

约束公理 1　词长不等公理

形式表示：

$FA(Fn,An)\rightarrow length(Fn)>length(An)$

直观意义：在全简称关系中，Fn 的字数必须大于 An 的字数。

约束公理 2　陈述语气公理

形式表示：$FA(Fn,An)\rightarrow NoInclude(Fn,Interrogative)\wedge NoInclude(An,Interrogative)$

直观意义：Fn 和 An 中都不包含疑问词“什么”、“怎么”、“啥”等。

约束公理 3　形式不重复公理

形式表示：$FA(Fn,An)\rightarrow \exists s[(Fn=concat(s,s))\vee(An=concat(s,s))]$

直观意义：在全简称关系中，Fn 和 An 都不可以是 ss 形式的汉字串，其中 s 是汉字串。

约束公理 4　语义不重复公理

形式表示：$FA(Fn,An)\rightarrow \exists c[(NumIn(Fn,c)\geqslant 1)\wedge(NumIn(Fn,c)\geqslant NumIn(An,c))]$

直观意义：所有出现在 Fn 中的汉字，在 Fn 中出现的次数必须要不小于在 An 中出现的次数。

约束公理 5　不泛指公理

形式表示：$FA(Fn,An)\rightarrow N\_FnSet(An)\leqslant 5$

直观意义：候选简称所对应的全称应该小于等于 5 个。

3.2　全简称关系图

由一批全简称对可以构成一个二分图，具体方法是：所有的全称构成全称集 $F=\{Fn_1,Fn_2,Fn_3,\cdots,Fn_n\}$，所有的简称构成简称集 $A=\{An_1,An_2,An_3,\cdots,An_m\}$。$F$ 和 A 构成图的顶点集 $V=F\cup A$，$\forall Fn\in F$，$\forall An\in A$，若 Fn 和 An 构成一对全简称，则构造一条连接 Fn 和 An 的无向边。

定义 2　全简称关系图 FAG(Fullname and Abbreviation Graph)是一个四元组，即 $FAG=(F,A,E,f)$，其中，$F=\{Fn_1,Fn_2,\cdots,Fn_n\}$ 是全称集，$A=\{An_1,An_2,\cdots,An_m\}$ 是简称集，$F\cup A$ 是顶点集，$E=\{e_1,e_2,\cdots,e_l\}$ 是无向边集，f 是 E 到 $F\times A$ 上的映射，即 $\forall e_k\in E$，总存在顶点 $Fn_i\in F$ 和 $An_j\in A$，使得 $f(e_k)=\langle Fn_i,An_j\rangle$ 成立，也就是说 e_k 是连接 Fn_i 和 An_j 的无向边。

给定全简称关系图 $FAG=(F,A,E,f)$，$\forall e_k\in E$，总存在 $Fn_i\in F$ 和 $An_j\in A$，使得 $f(e_k)=\langle Fn_i,An_j\rangle$，称顶点 Fn_i 和 An_j 与边 e_k 关联，顶点 Fn_i 和 An_j 相邻。

给定全简称关系图 $FAG=(F,A,E,f)$，$\forall v_i\in F\cup A$，与 v_i 相邻的所有顶点组成 v_i 的相邻点集，记为 $Adj(v_i)$，与 v_i 相邻的所有顶点的个数称为 v_i 的度数，记为 $d(v_i)=|Adj(v_i)|$。

图 1 是一个全简称关系图，全称集 $F=\{Fn_1,Fn_2,Fn_3,Fn_4,\cdots,Fn_n\}$，简称集 $A=\{An_1,An_2,An_3,An_4,\cdots,An_m\}$。

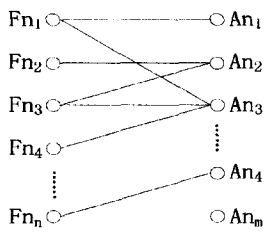

图 1　全简称关系图示例

3.3　基于约束函数集的简称分类

在引言中我们曾根据简称的形成规律和构词形式，将简称分为缩节式、删节式、缩合式、数合式和特殊式。由于缩合式和数合式大部分都含有标点符号，而本文中的获取算法只能获取不含标点符号的全称的简称，故将获取到的少数不含标点符号的缩合式和数合式简称分到其它 3 种类型中。本小节提出了一种基于约束函数集的简称自动分类方法，分类框架见表 3。

表 3　候选简称的类型 type

- 候选简称
  - 缩节式
    - 绝对缩节式
      - 高频绝对缩节式
      - 低频绝对缩节式
    - 近似缩节式
      - 高频近似缩节式
      - 低频绝对缩节式
  - 删节式
    - 前向型删节式
    - 居中型删节式
    - 后向型删节式
  - 特殊式
    - 异字型
    - 异序型
  - 待定

表 3 中的待定类型是指该 Can 无法划归到缩节式、删节式和特殊式中的任意一种，该 Can 很有可能是错误的候选简称，需要在验证过程中区别对待。

具体的分类标准和各类候选简称需要满足的条件见表 4，其中的 f_i 表示 3.1 节中的第 i 个约束函数的值。

表 4　候选简称的分类标准

类别 type	需要满足的条件
高频绝对缩节式	$f_1=1\wedge f_2=1\wedge f_3=1\wedge f_{11}=1$
低频绝对缩节式	$f_1=1\wedge f_2=1\wedge f_3=1\wedge f_{11}<1$
高频近似缩节式	$f_1=1\wedge f_2=1\wedge 0.823<=f_3<1\wedge f_9=1\wedge f_{11}=1$
低频近似缩节式	$f_1=1\wedge f_2=1\wedge 0.823<=f_3<1\wedge f_9=1\wedge f_{11}<1$
前向型删节式	$f_1=1\wedge f_2=1\wedge f_3<1\wedge f_4<=0.5$
居中型删节式	$f_1=1\wedge f_2=1\wedge -0.5<f_4<0.5\wedge(f_3<0.823\vee f_9<1)$
后向型删节式	$f_1=1\wedge f_2=1\wedge f_3<1\wedge f_4>=0.5$
异序型	$f_1=1\wedge f_2=0\wedge f_{11}=1$
异字型	$f_1<1\wedge(f_7>=0.5\vee f_{10}>=0.5\vee(f_7>0.05\wedge f_9=1\wedge f_{11}=1))$

表 4 中，高频绝对缩节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中的每个分词在 Can 中都有语素对应，且 Can 在候选简称集中频度最高。例如，Fn=“中国/科技/大学”，Can=“中科大”。

表 4 中，低频绝对缩节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中的每个分词在 Can 中都有语素对应，且 Can 在候选简称集中频度不是最高。例如，Fn=“中国/科技/大学”，Can=“中国科大”。

表 4 中，高频近似缩节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中的绝大部分分词在 Can 中都有语素对应，且 Can 在候选简称集中频度最高。例如，Fn=“重庆/国际/信托/投资/有限公司”，Can=“重庆国托公司”。

表 4 中，低频近似缩节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中的绝大部分分词在 Can 中都有语素对应，且 Can 在候选简称集中频度不是最高。例如，Fn=“重庆/国际/信托/投资/有限公司”，Can=“重庆国投公司”。

表 4 中，前向型删节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中只有部分分词在 Can 中有语素对应，且这些分词大多出现在 Fn 的前半部分。例如，Fn=“联想/集团/有限公司”，Can=“联想”。

表 4 中，居中型删节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中只有部分分词在 Can 中有语素对应，且这些分词均匀出现在 Fn 的前后部分。例如，Fn=“海尔/集团/有限公司”，Can=“海尔公司”。

表 4 中，后向型删节式的直观意义：Can 中的所有字都出现在 Fn 中，Can 与 Fn 保持语序一致，Fn 中只有部分分词在

Can 中有语素对应，且这些分词大多出现在 *Fn* 的后半部分。例如，*Fn*="中华/人民/共和国/共产主义/青年/团"，*Can*="共青团"。

表 4 中，异序型的直观意义：*Can* 中的所有字都出现在 *Fn* 中，*Can* 与 *Fn* 语序不一致，且 *Can* 在候选简称集中频度最高。例如，*Fn*="哈尔滨第六制药厂"，*Can*="哈药六厂"。

表 4 中，异字型的直观意义：*Can* 中的部分字不出现在 *Fn* 中，但 *Can* 在候选简称集中的相对频度很高或相对覆盖率比较高。例如，*Fn*="娲皇圣母庙"，*Can*="女娲庙"。

3.4 候选简称的预过滤

3.4.1 基于约束公理集的预过滤

对于候选简称集中的每个候选简称 *Can*，验证 *Fn* 与 *Can* 是否同时满足公理 1—4 的约束要求，如果不满足则该候选简称是错误的，但满足并不表示该候选简称就是正确的，还需要经过其它方法进一步验证。

例如：*Fn*="娲皇圣母庙"，*Can*="女娲庙庙"，*Can* 符合公理 1—3，但不符合公理 4 的约束要求，因为汉字"庙"在 *Can* 中出现了两次，而在 *Fn* 中仅出现了一次，故 *Can* 不正确。之所以会出现这种现象是因为在语料中 *Can* 之后没用标点符号与下文隔开。

3.4.2 基于全简称关系图的预过滤

利用全简称关系图进行验证的方法只适用于输入的全称个数比较多的情况，按照第 3.2 小节中介绍的全简称关系图的构图方法来构造全简称关系图 $FAG=(F,A,E,f)$。

利用全简称关系图进行验证的具体方法如下：$\forall v_i \in A$ 如果 $d(v_i)$ 则 $\forall v_k \in Adj(v_i)$，如果 v_i 的简称类型不是缩节式，则对于全称 v_k 而言该候选简称 v_i 是错误的。

例如：在实验的 3910 个全称中有 24 个 *Fn* 的候选简称集中有"公司"，所以 *Can*="公司"是一个泛指的候选简称，如果对于 *Fn* 而言，*Can*="公司"是缩节式，则认为 *Can* 是 *Fn* 的正确简称，否则认为 *Can* 是 *Fn* 的错误简称。

3.5 基于决策树的验证

由候选简称的语料类型 tag、分类类型 type 以及各个约束函数作为决策属性，利用 J48 算法生成决策树，经过剪枝修正，最终的决策树见图 2，利用决策树对候选简称集中的候选简称进行类别标注，"F"表示错误，"T"表示正确，去除类别是"F"的候选简称，保留类别是"T"的候选简称。

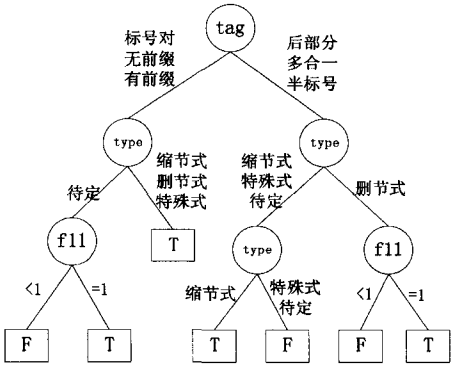

图 2 验证简称的决策树

3.6 简称集的优先级排序

通过上述验证之后得到已知全称的简称集，本小节讨论如何对简称集中的简称进行排序。本文采用优先级综合函数 *PRI*(*Fn*,*An*) 对简称集中同一类的简称进行排序。

PRI(*Fn*,*An*) 的定义如下：

$$PRI(Fn,An)=\sum_{1}^{11}F_i * \rho_i$$

式中，$F_i=\begin{cases} f_i(Fn,An), & i\neq 4 \\ 1-|f_i(Fn,An)|, & i=4 \end{cases}$ 为每个函数在综合评价时采取的权重，F_i 与 ρ_i 间的对应关系见表 5。ρ_i 的大小根据各函数对全简称关系的约束程度通过实验得到。

表 5 函数与函数权重的对应关系

编号	函数内容	ρ_i
F_1	An 的字来自 Fn 中的比率	0.12
F_2	Fn 与 An 的语序	0.08
F_3	An 对 Fn 的分词覆盖率	0.06
F_4	An 对 Fn 的分词覆盖重心	0.08
F_5	Fn 中未被 An 覆盖到的最长连续分词数	0.04
F_6	Fn 和 An 的长度关系	0.06
F_7	An 在 GoogleArchSet(Fn) 中出现的频度	0.10
F_8	An 的字来自 Fn 中的相对比率	0.12
F_9	An 在简称集中的相对覆盖率	0.10
F_{10}	An 在简称集中的频率	0.12
F_{11}	简称集中的元素按照频度升序排序后，An 在其中的相对位置	0.14

4 实验和分析

用以下评价指标来评价汉语简称的获取：

汉语简称获取的正确率：

$$AP=\frac{\text{正确获取的汉语简称个数}}{\text{所有获取的汉语简称个数}}$$

汉语简称获取的召回率：

$$AC=\frac{\text{正确获取到对应简称的全称个数}}{\text{所有要获取对应简称的全称个数}}$$

正确候选简称验证的正确率：

$$TP=\frac{\text{正确验证的正确候选简称个数}}{\text{所有验证为正确的候选简称个数}}$$

正确候选简称验证的召回率：

$$TC=\frac{\text{正确验证的正确候选简称个数}}{\text{总共获取的正确候选简称个数}}$$

错误候选简称验证的正确率：

$$FP=\frac{\text{正确验证的错误候选简称个数}}{\text{所有验证为错误的候选简称个数}}$$

错误候选简称验证的召回率：

$$FC=\frac{\text{正确验证的错误候选简称个数}}{\text{总共获取的错误候选简称个数}}$$

田国刚博士对同指关系的获取进行了比较深入的研究[11]，本实验的测试数据选取田国刚博士实验结果中的 3910 个汉语全称，这批数据具有多学科、多领域的特点。测试结果见表 6。

表 6 汉语简称获取和验证的实验结果

AP	AC	TP	TC	FP	FC
94.63%	84.09%	95.87%	97.98%	88.75%	79.11%

如表 6 所列，简称的获取方面，正确率比较高，但是召回率不是很高，主要原因是测试数据中有些全称比较偏，在 Google 搜索引擎中没有相关的简称信息，这不是获取方法的问题，而是语料的稀缺。候选简称的验证方面，本文中的验证方法对正确候选验证的正确率和召回率都比较高，但是对错误候选验证的正确率和召回率要差一点，主要原因是特殊式简称和错误简称比较难区别。

总之，全简称关系获取的性能除了受获取方法的影响外，还受语料本身和分词工具的影响，语料质量的高低和分词工具的性能也是影响获取性能的关键因素。因此，尽管本文在获取后进行了预过滤和验证，但获取的最终结果中仍然还存在着一些错误，具体分为以下几类。

1) 概念词提取错误

这类错误的特征是其本应得到正确的简称，但是由于从全简称句子中获取简称时边界没有定好，导致获取错误，这类错误的简称由于很容易归为异字型简称，因此在验证时很难被验证出来，例如：Fn="武汉航空公司"，获取到的错误简称 An="原武航"。

2) 锚文本获取不全

这类错误的特征是其本应得到正确的简称，但是由于简称只有前部分出现在锚语料中，导致获取到的结果只是简称的前部分，例如：Fn="中国银联股份有限公司"，获取到的错误简称 An="中国"。造成这类错误的原因是获取锚语料时，将 Google 返回的包含查询项的摘要保存下来作为锚语料，所以有时会出现简称只有前部分出现在摘要中，后部分在摘要中被省略了。

3) 非全简称关系

这类错误的特征是两个概念词不满足全简称关系，但是在语料中能够被匹配模式和提取算法提取出来，例如：从全简称句子"南京信息工程大学(原南京气象学院，以下简称南信大)"中提取出 Fn="南京气象学院"，An="南信大"。造成这类错误的原因是句子本身并不表达全简称关系，但能够与全简称句子的提取模式相匹配，且能够被抽取简称的提取算法提取出简称。

4) 部分全简称关系

这类错误的特征是获取了全简称关系的局部，例如，用提取算法 1 从全简称句子"红旗业余大学-简称红大"中提取"业余大学"的简称，得到"大"。造成这类错误的原因是从全简称关系中截取一部分来作为新的全简称关系，但是全简称关系的局部不一定也能构成全简称关系。

5) 有错别字的简称

这类错误的特征是获取的简称中含有错别字，这类错误的简称由于很容易归为异字型简称，因此很难在验证时去除，例如：Fn="玉树藏族自治州"，获取到的错误简称 An="王树州"。

5 与相关工作的比较

本节将本文中的方法与之前相关研究的几种方法进行了比较。表 7 是将获取的准确率和召回率进行比较的结果。

表 7 比较几种方法的准确率

方法	指标		
	全称的个数	准确率	召回率
崔世起的方法[3]	148	51.4%	—
谢丽星的方法[4]	200	68.3%	—
武子英的方法[5]	1806	74.1%	—
姜广的方法[7]	300	87.1%	71.7%
本文中的方法	3910	94.63%	84.09%

在表 7 中只和姜广的方法比较了召回率，这是因为别的方法是从挖掘的角度来获取全简称关系的，故在测试集中不存在没有语料的全称，而本文中的方法和姜广的方法是从查找的角度来获取的，故有的全称获取不到语料，从而召回率比较低。

下面从简称的获取范围来对几种方法进行比较，见表 8。

表 8 比较几种方法的获取范围

方法	简称类型				
	缩节式	删节式	缩合式	数合式	特殊式
崔世起的方法[3]	√	—	—	—	—
谢丽星的方法[4]	√	√	—	—	—
武子英的方法[5]	√	√	—	—	—
姜广的方法[7]	√	√	△	△	√
本文中的方法	√	√	△	△	√

表 8 中"√"表示可以获取，"—"表示不可以获取，"△"表示可以部分获取，例如本文中的方法可以获取不含标点符号的缩合式和数合式简称，但不可以获取含有标点符号的简称。

结束语 本文针对目前汉语简称获取研究中获取范围受限、获取准确率较低等现状，提出了一种根据汉语全称从 Web 中获取对应汉语简称的方法，该方法包括获取和验证两个步骤。获取步骤通过选择查询模式来构造查询项，将查询项提交到 Google 中获取全简称语料，再利用提取算法从全简称语料中提取出候选简称，从而形成全称对应的候选简称集合。为了验证候选简称，本文定义了全简称关系约束，包括一组约束公理和一组约束函数，分别定性和定量地表示全称和对应简称之间的约束，构建了全简称关系图来表示所有全称和简称之间的联系，在验证过程中，先分别用约束公理和关系图对候选简称进行过滤，再用约束函数对候选简称分类，并以分类类别、语料标记和约束函数值作为属性构建决策树，利用决策树对候选简称进行验证。实验结果表明，本方法适合多学科、大规模、高准确率的汉语简称获取。后续，仍有一些问题有待我们进一步研究：

(1) 对于 Google 中没有对应语料的全称，怎么获取其对应的简称。有些全称在 Web 网页中没有全简称语料，但是可以通过类比的方法来获得对应的简称。比如，$Fn1$="北京市舞蹈家协会"在 Web 中没有全简称语料，但是 $Fn2$="广东省舞蹈家协会"有对应的简称 $An2$="广东舞协"，我们可以通过类比的方法来求得 $Fn1$ 对应的简称 $An1$="北京舞协"。

(2) 考虑全称中含有标点符号时其对应简称的获取，主要是缩合式、数合式的简称获取。例如：Fn="最高人民法院、最高人民检察院"时获取对应的简称 An="两高"。

(3) 过滤获取不全的锚文本，进一步提高全简称语料的质量。因为在获取锚语料时，是将 Google 返回的包含查询项的摘要保存下来作为锚语料，所以有时会出现简称只有前部分出现在摘要中，后部分在摘要中被省略了，这种锚文本语料应该被过滤掉。

参 考 文 献

[1] 袁晖，阮显忠. 现代汉语缩略语词典[M]. 北京：语文出版社，2002

[2] 刘磊，曹存根，张春霞，等. 概念空间中上下位关系的意义识别研究[J]. 计算机学报，2009，32(8)：1651-1661

[3] 崔世起，刘群，林守勋，等. 中文缩略语自动抽取初探[C]// 全国第八届计算语言学联合学术会议论文集，2005. 南京：清华大学出版社，2005：53-58

(下转第 195 页)

因此准确率和召回率的微平均是相同的。根据式(10)可知，对于同一个数据集，这3个指标均相同。

实验中，我们采用 Baseline，PCA-kNN，LDA-kNN 在测试集上的实验结果与本文提出的 MLD-RLSC 算法进行比较，具体的实验结果如表2所列。

表2　4种算法的分类性能比较

Method	宏平均			微平均
	Precision	Recall	F	F
MLD-RLSC	0.94421	0.9534	0.9474	0.9527
PCA-*kNN*	0.9080	0.9213	0.9122	0.9403
LDA-*kNN*	0.8668	0.8595	0.8568	0.8744
Baseline	0.7739	0.7522	0.6917	0.7438

从表2可以看出，MLD-RLSC 方法在准确率、召回率和 F 值等各项评价指标上均高于其他3种分类算法。原因在于本算法能利用训练样本的标签信息，在降维过程中保持类内的紧凑性和类间的分离性，估计样本几何流形结构，从而保证了很好的分类性能。

此外，为了说明本文所提出的 MLD-RLSC 算法的高效性，我们通过实验进一步对这4种分类算法的运行时间进行了比较，实验结果如表3所列。

表3　4种算法的运行时间比较

算法	运行时间
MLD-RLSC	36.7348s
LDA-kNN	37.1256s
PCA-kNN	321.9318s
Baseline	241.1980s

从表3可以看出，MLD-RLSC 算法的运行时间与 LDA-kNN 算法相当，远低于其他两种分类算法。原因在于：MLD-RLSC 能够有效地避免高维文档引起的“维数灾难”问题，缩小了分类器的搜索范围，从而保证了算法具有很快的运行速度。

结束语　Web 文档具有高维度、样本稀疏和特征不太明显的特点，传统的特征提取方法通常是保存文档空间的全局结构，实现从高维到低维的线性映射。针对传统文档分类算法在处理高维数据集存在的测试时间过长、分类准确率不高的问题，本文提出基于流形正则化的文档分类算法 MLD-RLSC，该算法能够充分利用训练样本的类别信息帮助学习样本空间的局部几何结构，以提取有效的低维特征并进行分类。实验表明，本文方法具有很好的分类性能和较快的测试运行速度。

参考文献

[1] Ian T J. Principal Component Analysis [J]. Chemometrics and Intelligent Laboratory Systems, 1987, 2(1-3): 37-52

[2] 杨健，杨静宇，叶晖. Fisher 线性鉴别分析的理论研究及其应用[J]. 自动化学报，2003(04)

[3] Martinez A M, K A C. PCA versus LDA[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(2): 328-340

[4] 李波. 基于流形学习的特征提取方法及其应用研究[D]. 合肥：中国科学技术大学，2008

[5] Roweis S T, Saul L K. Nolinear Dimensionality Reduction by Locally Linear Embedding[J]. Science, 2000, 290(5500): 2323-2326

[6] He Xiao-fei. *Locality Preserving Projections* [M]. Cambridge, USA: MIT Press, 2003

[7] He Xiao-fei. Neighborhood Preserving Embedding [C]// Tenth IEEE International Conference on Computer Vision. 2005, 2: 1208-1213

[8] Chen H T. Local Discriminant Embedding and Its Variants[C]// IEEE. 2005

[9] Yang Y, Liu X. A re-examination of text categorization methods [C]// ACM. USA, 1999

[10] Duda R O. Pattern Classification [M]. New York: Wiley-Interscience, 2000

[11] Ye Jie-ping. Least squares linear discriminant analysis[C]// ICML'07. Corvalis, Oregen: ACM Press, 2007

[12] Belkin M. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples[J]. The Journal of Machine Learning Research, 2006, 7: 399-434

[13] Gerald S, Wong A, Yang C S. A vector space model for automatic indexing[J]. Communications of the ACM, 1975, 18(11): 613-620

[14] 林永民，吕震宇，赵爽，等. 文本特征加权方法 TF·IDF 的分析与改进[J]. 计算机工程与设计，2008(11)

[15] Yang Y. Noise reduction in a statistical approach to text categorization[C]// ACM, Proc. of SIGIR. 1995

(上接第182页)

[4] 谢丽星，孙茂松，佟子健，等. 基于用户查询日志和锚文字的汉语缩略语识别[C]// 中国计算机语言学研究前沿进展，2009. 烟台：清华大学出版社，2009：551-556

[5] 武子英，郑家恒. 现代汉语缩略语自动识别的方法研究[J]. 计算机工程与设计，2007，28(16)：4052-4054

[6] Tian Guogang, Cao Cungen, Liu Lei, et al. MFC: A Method of Co-referent Relation Acquisition from Large-Scale Chinese Corpora[C]//Proceedings of the ICNC'06-FSKD'06. Xi'an, China, 2006: 1256-1261

[7] Guang Jiang, Cao Cungen, Sui Yuefei, et al. A General Approach to Extracting Full Names and Abbreviations for Chinese Entities from the Web[C]//IFIP Advances in Information and Communication Technology. Manchester, UK, 2010: 271-280

[8] 支流，段慧明，朱学锋，等. 中文缩略语知识库建设[C]// 第三届学生计算语言学研讨会论文集，2006. 沈阳：中文信息学会，2006：316-320

[9] 鲍明凌，亢世勇. 基于数据库的现代汉语新词语缩略语的研究[C]// 第一届学生计算语言学研讨会论文集，2002. 北京：中文信息学会，2002：233-238

[10] Han Jia-wei, Kamber M. 数据挖掘概念与技术(第2版)[M]. 范明，孟小峰，译. 北京：机械工业出版社，2007：188-200

[11] 田国刚. 受限中文语料的自监督文本知识获取研究[D]. 北京：中国科学院计算技术研究所，2007

[12] 卢汉，曹存根，岳小莉. 一种根据实体的汉语简称识别出实体全称的方法和系统[P]. ZL200710119513. 中国，2007