基于 python 的中文分词的实现及应用

刘新亮 严姗姗
(北京工商大学计算机学院，100037)

摘 要 中文分词的实现及应用属于自然语言处理范畴，完成的是中文分词在 Python 语言环境下的实现，以及利用这个实现的一个应用程序接口和一个中文文本处理的应用。设计共分为五个部分，分别是：分词模块、包装模块、应用程序接口、Nonsense 模块，这个项目是为了下一步开放源代码的中文搜索引擎提供中文分词功能，同时通过表现代码的娱乐性达到促进公开源代码的发展。

关键词 中文分词；Python 语言；程序接口

1 引言

自然语言处理是研究实现人与计算机之间用自然语言进行有效通信的各种理论和方法的一个领域，是计算机科学领域和人工智能领域的重要发展方向之一。它大体包括自然语言理解和自然语言生成两个部分，前者是使计算机理解自然语言文本，后者是使计算机以自然语言文本来表达给定的意图、思想等。

自然语言处理是目前比较前沿的学科，拥有广阔的前景，但同时是非常有难度的。现在已经出现了能够针对一定应用的实用系统，但是通用的、高质量的自然语言系统仍然是较长期的努力目标。这些实用系统已经具有相当的自然语言处理能力，其中有些已经商品化、甚至产业化。典型的例子有：各种数据库和专家系统的自然语言接口、各种机器翻译系统、全文信息检索系统、自动文摘系统等[1]。

2 中文分词技术

中文分词技术属于自然语言处理技术范畴，对于一句话，人可以通过自己的知识来明白哪些是词，哪些不是词，但如何让计算机也能理解，其处理过程就是分词算法。现有的分词算法可分为三大类：基于字符串匹配的分词方法、基于理解的分词方法和基于统计的分词方法[2]。

（1）基于字符串匹配的分词方法：这种方法又叫做机械分词方法，它是按照一定的策略将待分析的汉字串与一个“充分大的”机器词典中的词条进行匹配，若在词典中找到某个字符串，则匹配成功（识别出一个词）。按照扫描方向的不同，串匹配分词方法可以分为正向匹配和逆向匹配；按照不同长度优先匹配的情况，可以分为最大（最长）匹配和最小（最短）匹配；按照是否与词性标注过程相结合，又可以分为单纯分词方法和分词与标注相结合的一体化方法。

（2）基于理解的分词方法：这种分词方法是通过让计算机模拟人对句子的理解，达到识别词的效果。其基本思想就是在分词的同时进行句法、语义分析，利用句法信息和语义信息来处理歧义现象。它通常包括三个部分：分词子系统、句法语义子系统、总控部分。在总控部分的协调下，分词子系统可以获得有关词、句子等的句法和语义信息来对分词歧义进行判断，即模拟人对句子的理解过程。

（3）基于统计的分词方法：从形式上看，词是稳定的字的组合，因此在上下文中，相邻的字同时出现的次数越多，就越有可能构成一个词。

到底哪种分词算法的准确度更高，目前并无定论。对于任何一个成熟的分词系统来说，不可能单独依靠某一种算法来实现，都需要综合不同的算法。笔者了解，海量科技的分词算法就采用“复方分词法”。所谓复方，相当于用中药中的复方概念，即用不同的药材综合起来去医治疾病。同样，对于中文词的识别，需要多种算法来处理不同的问题。

3 分词中的难题

有了成熟的分词算法，是否就能容易的解决中文分词的问题呢？事实远非如此。中文是一种十分复杂的语言，让计算机理解中文语言更是困难。在中文分词过程中，有两大难题一直没有完全突破。

（1）歧义识别：歧义是指同样的一句话，可能有两种或者更多的切分方法。例如：表面的，因为“表面”和“面的”都是词，那么这个短语就可以分成“表面 的”和“表 面的”。这种称为交叉歧义。

（2）新词识别：新词，专业术语称为未登录词。也就是那些在字典中都没有收录过，但又确实能称为词的那些词。最典型的是人名，人可以很容易理解句子“王军虎去广州了”中，“王军虎”是个词，因为是一个人的名字，但要是让计算机去识别就困难了。如果把“王军虎”作为一个词收录到字典中去，全世界有那么多名字，而且每时每刻都有新增的人名，收录这些人名本身就是一项巨大的工程。即使这项工作可以完成，还是会存在问题，例如：在句子“王军虎头虎脑的”中，“王军虎”还能不能算词？新词中除了人名以外，还

有机构名、地名、产品名、商标名、简称、省略语等都是很难处理的问题，而且这些又正好是人们经常使用的词，因此对于搜索引擎来说，分词系统中的新词识别十分重要。目前新词识别准确率已经成为评价一个分词系统好坏的重要标志之一。

4 软件结构

如图 1 所示，软件实现过程大体是：首先由分词模块用 C 语言实现分词，这部分主要运用了中国科学院的 ICTCLAS，对它进行研究及处理；然后用 SWIG 对分词模块进行包装，使其能够被 Python 语言调用，在 Python 语言环境下实现中文分词

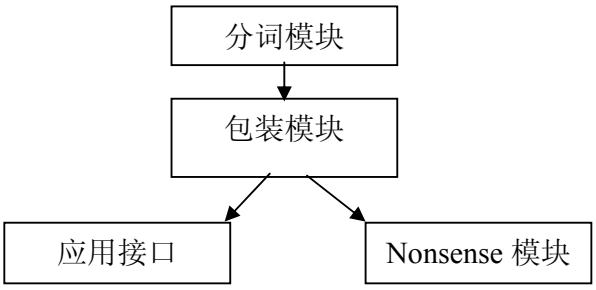

图 1 软件结构图

4.1 分词模块结构

分词模块包括两个部分，其一是中国科学院的公开源代码软件 ICTCLAS(Institute of Computing Technology, Chinese Lexical Analysis System) 。该系统的功能有：中文分词；词性标注；未登录词识别。分词正确率高达 97.58%(973 专家组评测结果)，未登录词识别召回率均高于 90%，其中中国人名的识别召回率接近 98%，处理速度为 31.5Kbytes/s。ICTCLAS 的特色还在于：可以根据需要输出多个高概率结果，有多种输出格式，支持北大词性标注集，973 专家组给出的词性标注集合。该系统得到了专家的好评，并有多篇论文在国内外发表。[3][4][5]

ICTCLAS 的功能有：中文分词，词性标注，未登录词识别。中文分词，即输入“我是中国人。”，得到处理结果为“我/是/中国/人/。/”。词性标注，即对每个切分开的词进行词性说明，支持北大词性标注集。如上例可得到结果为：“我/r 是/v 中国/n 人/n 。/w ”。未登录词识别即解决新词识别问题。分词模块还可以根据需要输出多个高概率结果，例如，用户输入“我是中国人。”

分词模块的第二个部分是对 ICTCLAS 的处理。通过对源代码中 ICTCLAS_WinDlg.cpp 的函数 On_button_run ( ) 的修改，得到 seg.cpp,以便下一步对 ICTCLAS 进行包装。

Seg.h:

```
class Segment{
    CResult m_ICTCLAS;
public: char* process_paragraph(char* sSource,int type,int format);
};
```

seg.cpp:

```
//#include <ICTCLAS.h>
#include <string.h>
//#include "StdAfx.h"
//#include <iostream.h>
#include "seg.h"
char* Segment::process_paragraph(char* sSource,int type,int format)
{
    char* sResult;//,*sSource;
    //CResult m_ICTCLAS;
    m_ICTCLAS.m_nOutputFormat=format;
    m_ICTCLAS.m_nOperateType=type;
    //clock_t start, finish;
    if(format!=2)
        sResult=new char [(strlen(sSource)+13)*3];
    else
        sResult=new char [(strlen(sSource)+13)*50];
    if (!m_ICTCLAS.ParagraphProcessing(sSource,sResult)){
        return NULL;
     }
    else{
        return sResult;
     }
```

4.2 包装模块

具体过程是：① 根据 seg.cpp 编写 seg.i 文件;②通过 swig 用 seg.i 生成 seg.py 和 seg_wrap.cpp,命令行为：swig c++ –python cpp1.i;③ 通过 setup.py 把 seg.cxx 和 seg_wrap.cxx 编译到一起，生成_seg1.pyd，a.编写 setup.py,b.输入命令行 setup.py build.ext;④ 把_seg.pyd 拷贝到与其他文件同一目录下;⑤ 在 seg.py 中 import _seg.pyd;⑥ 用 python 调用此程序时 import seg.py;

具体算法：

seg.i：

```
 %module seg
%{
#include "seg.h"
%}
extern class Segment{
public: char* process_paragraph(char* sSource,int type,int format);
};
```

整个分词模块相当于一个黑盒子，seg.py 相当于是唯一与其联系的线，我们通过 seg.py 使用分词模块，体现包装的

作用。process_paragraph 是 seg.cpp 中的函数，通过调用它，可以完成分词工作。函数有三个参数，分别是中文文本，标示类型和输出标准。

setup.py：

```
from distutils.core import setup, Extension

setup(name="_seg", version="1.0",
        ext_modules=[Extension("_seg",
["seg.cpp","Result\\Result.cpp","Segment\\Segment.cpp","Utility
\\Dictionary.cpp","Segment\\DynamicArray.cpp","Segment\\NSh
ortPath.cpp","Segment\\Queue.cpp","Segment\\SegGraph.cpp","
Tag\\Span.cpp","StdAfx.cpp","Unknown\\UnknowWord.cpp","U
tility\\Utility.cpp","Utility\\ContextStat.cpp","seg_wrap.cxx"],

include_dirs=["Result\\","Segment\\","Dictionary\\","DynamicAr
ray\\","NShortPath\\","Queue\\","SegGraph\\","Span\\","StdAfx\\
","UnkownWord\\","Utility\\","ContextStat\\"])])
```

4.3 应用程序接口

forexoweb.py:

```
import seg
import sys
import string
a=seg.Segment()
def tokenizer(filecontent):
    (type=1 一级标注；format=0 北大标准)
    b=a.process_paragraph(filecontent,1,0)
    list=b.split()
    （删去标点，助词，叹词）
    list = filter(lambda x: (not (x[-2:] in ('/w','/u','/e'))),
                list)
    list = map(lambda x: x.split('/')[0],
list)
```

定义函数 tokenizer，参数 filecontent 表示需要被分词的中文文本。调用函数 process_paragraph，选择一级标注，并且以北大标准输出。然后将分次结果转换成列表形式，并删去标点符号，助词和叹词，删除标注，以方便之后开发搜索引擎之用。

例如，需要被分词的中文文本为“我是中国人吗？”，对其处理过程如下：

分词：“我/r 是/v 中国/n 人/n 吗/y ？/w ”

转换成列表形式：

['我',’/r’,'是',’/v’,'中国',’/n’,'人',’/n’,'吗',’/y’,'?',’/w’]

删除标点符号，助词和叹词,删除标注：

['我', '是', '中国', '人']

4.4 Nonsense 模块

基本算法：

① 对中文文本进行分词处理

② 确定新文本第一个词

③ 确定新文本第二个词

④ 循环确定下一个词，直到新文本字数等于原文本

```
import seg
import sys
import string
import random
class Myclass:
        def __init__(self):
        self.a = seg.Segment()
       def tokenizer(self,allLines):
        b = self.a.process_paragraph(allLines,0,0)
        words = b.split()
        self.first = self.find_firstword_dict(words)
        (self.word_dict,  self.sum_dict)  =  self.change_
to_dict(words)
        self.word_length = len(words)
        #return [ unicode(word, 'GBK') for word in words ]
    def find_firstword_dict(self,words):
            sentences = []
        current_sentence = []
        for word in words:
          if word in ["。","？","！"]:
            sentences.append(current_sentence)
                current_sentence = []
            else:
              current_sentence.append(word)
        if len(current_sentence) > 0:
            sentences.append(current_sentence)
        sum=0
        a=0
        first={}
        for current_sentence in sentences:
          a = current_sentence[0]
          first[a] = first.setdefault(a,0) + 1
          sum += 1
          for key in first.keys():
            first[key] = float(first[key]) / sum
        return first
        (first={'我':0.33,'你':0.33,'她':0.33}
        假设:first={'我':0.5,'你':0.2,'她':0.3})
    def find_first_word(self):
        first = self.first
```

```
        h = random.random()
        m=0
        firstword = ''
       values = []
        firstkeys = first.keys()
        start = 0
        for eachkey in firstkeys:

values.append((eachkey,start,start+self.first[eachkey]))
            start += self.first[eachkey]
        (values = [('我',  0,0.5),
                   ('你',0.5,0,7),
                  ('她',0.7,  1)])
       for value in values:
          word,start,end = value
            if h >= start and h <= end:
                firstword = word
                break
        return firstword
    def change_to_dict(self,words):
        word_dict = {}
        sum_dict = {}
        sum = 0
        curr_word = words[0]
        for next_word in words[1:]:
            if word_dict.has_key(curr_word):
                if word_dict[curr_word].has_key(next_word):
                    word_dict[curr_word][next_word] += 1
                else:
                    word_dict[curr_word][next_word] = 1
                sum_dict[curr_word] += 1
            else:
                word_dict[curr_word] = {next_word:1}
                sum_dict[curr_word] = 1
            curr_word = next_word
            sum_dict = {'我': 1,'是':3})
        return (word_dict, sum_dict)
    def get_next_word(self,next_words,sum):
        h = random.randint(1,sum)
        m=0
        nextword = ''
        values = []
        nextkeys = next_words.keys()
        (nextkeys = ['我','你','她'])
        start = 0
            for eachkey in nextkeys:

values.append((eachkey,start,start+next_words[eachkey]))
            start += next_words[eachkey]
        for value in values:
            word,start,end = value
            if h >= start and h <= end:
                nextword = word
                break
        return nextword
```

5 结论

在这个软件中，我们在 Python 语言环境下实现了中文分词，并进行了中文信息处理的应用。这对 Python 语言和中文信息处理来说，是双向的扩展，都具有积极的意义：Python 语言的优势可以给中文信息处理带来更好的效果，同时中文信息处理的广阔前景也给 Python 语言以及与其相关的 Zope 等提供了更大的发展空间。有关中文分词的应用，还可以继续往人工智能方向发展，例如利用 Python 语言编写的贝叶斯分类器，可以获得更有实际意义的应用，比如垃圾邮件分类等。

参考文献

[1]《python 核心编程》，（美）Wesley J.Chun（陈仲才）著，杨涛 王建桥 杨晓云 高文雅 等译，机械工业出版社，2001．8

[2]徐飞 孙劲光，基于一种粗切分的最短路径中文分词研究[J] 计算机与信息技术，2007（11）

[3]Hua-Ping ZHANG Qun LIU Xue-Qi CHENG Hao Zhang Hong-KuiYu. Chinese Lexical Analysis Using Hierarchical Hidden Markov Model.

[4]Remi Delon, CherryPy Tutorial, 12 May 2004, Release 0.10

[5]Python Tutorial,Guido van Rossum Fred L. Drake, Jr., editor,PythonLabs ,Release 2.3.4,May 20, 2004

收稿日期：9 月 10 日　　修改日期：9 月 12 日

作者简介：刘新亮（1972-），男，北京工商大学计算机学院讲师，研究方向：数据库应用。