暨 南 大 学
本科生课程论文

论文题目：用“类”建立数据接口并数据可视化研究

学　　院：智能科学与工程学院

学　　系：人工智能

专　　业：人工智能

课程名称：Python 程序设计

学生姓名：邓宙京

学　　号：2022101819

指导教师：林聪

2022 年 12 月 28 日

用“类”建立数据接口并数据可视化研究——以比较两本不同类型小说的词性分布为例

[摘 要]

分析《活着》和《神雕侠侣》两本类型不同的现代小说的词性分布。用一个类读入整本小说, 用 **Jieba 分词**自然语言处理工具, 初始化过程分析内容, 分解词语并获取词性，类对象取索引返回词和词性两项主要信息。在调用类对象的函数中, 实现词性的统计, 用饼状图可视化个主要词性频率，并对比两本小说的饼状图。

[关键词]

Python；类；API；自然语言处理；结巴分词；词性分布；数据可视化

1. 绪论

1.1 文献综述

自然语言处理（Natural Language Processing，NLP）是计算机科学领域与人工智能领域中的一个重要方向。它研究人与计算机之间用自然语言进行有效通信的理论和方法，目的是让计算机处理或“理解”自然语言，以执行自动翻译、文本分类和情感分析等，是人工智能中最为困难的问题之一。当前国内外已有很多自然语言处理技术，如 Anaconda，Scikit-learn，TensorFlow，Keras，Gensim，NLTK，Jieba 等，其中 Jieba 是比较受欢迎的的中文分词工具。在此，笔者将主要通过结巴分词进行分词并完成词性分布统计与数据可视化等工作。

1.2 术语说明

①结巴分词：当前效果较好的一种中文分词器，支持中文简体、中文繁体分词，同时还支持自定义词库。

②API（Application Programming Interface）：应用程序接口，又称应用编程接口，就是软件系统不同组成部分衔接的约定。

2.正文

2.1 实验动机

本实验涉及 API、结巴分词、matplotlib 库绘图等技术，能延伸成为文本分析等类型的应用程序，能够应用于自然语言处理方面，给自动翻译、文本分类和情感分析等工作提供技术支持，让计算机处理或“理解”自然语言，是人工智能自然语言处理的基础，其发展有利于人工智能的进一步发展。

2.2 关键技术分析

①课程知识点：变量与字符串

函数

类和对象

文件和异常

一些文件类型输入输出

for 循环进阶

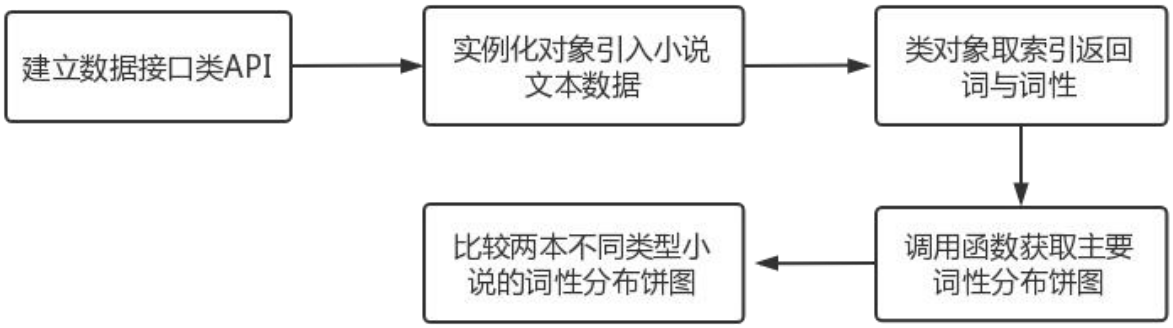
序列型数据类型与可迭代

②

模块/包	主要用途
jieba	切割分词中文文本，进行词性标注
matplotlib	绘制饼图，统计数据

2.3 设计思路

①简单流程图



②文字描述代码的主要结构，主函数，关键功能函数/方法的调用（如图所示）

```
###利用 Python 数据模型的 API 建立一个数据接口类并实现数据可视化

import jieba                        #引入所需库
import jieba.posseg as pseg
import matplotlib.pyplot as plt

seg_list = []
class Cut():
    """分词并统计类"""

    def __init__(self,name):
        """初始化属性"""
        self.name = name
        self.labels = ['n','v','d','a','p']  #词性标签,主要对名词，动词，副词，形容词及介词进行统计
        self.num_list = [0,0,0,0,0]  #数量列表

    def __getitem__(self,index):
        """实现类对象取索引返回词和词性两项主要信息"""
        with open(self.name,"r",encoding='utf-8') as f:
            for line in f.readlines():
                for w in pseg.cut(line):  #使用jieba对line分词及词性标注
                    seg_list.append(w)
            return seg_list[index]

    def show_pie(self):
        """运用jieba分词实现词性统计并用饼图可视化"""
        with open (self.name,'r',encoding='utf-8') as f1:
            for line in f1.readlines():
                words = pseg.cut(line)  #使用jieba对line分词及词性标注
                for x in words:
                    if x.flag  in self.labels:  #判断x的词性是否存在词性列表中
                        if x.flag == 'n':
                            index1 = 0
                        if x.flag == 'v':
                            index1 = 1
                        if x.flag == 'd':
                            index1 = 2
                        if x.flag == 'a':
                            index1 = 3
                        if x.flag == 'p':
                            index1 = 4
                        self.num_list[index1] += 1  #统计各类词性总数
        rate = []
        for i in range(len(self.num_list)):
            result = self.num_list[i]/sum(self.num_list)  #获取词性频率
            rate.append(result)
        plt.rcParams['font.sans-serif'] = ['SimHei']  #引入绘图数据
        plt.pie(x=rate,autopct='%1.2f%%',labels=self.labels)
        plt.title(self.name+"主要词性分布统计图")
        plt.show()  #输出饼图
```

```python
###分析两本类型不同的现代小说的词性分布

from api import Cut  #从api模块中引入Cut类

novel1 = r"C:\Users\86178\Desktop\Python程序设计\《活着》.txt"
novel2 = r"C:\Users\86178\Desktop\Python程序设计\《神雕侠侣》.txt"

cut1 = Cut(novel1)  #实例化对象1
print(cut1[99])  #类对象取索引
cut1.show_pie()  #获取词性分布统计饼图

cut2 = Cut(novel2)  #实例化对象2
print(cut2[99])
cut2.show_pie()
```

2.4 实验过程

①数据输入截图

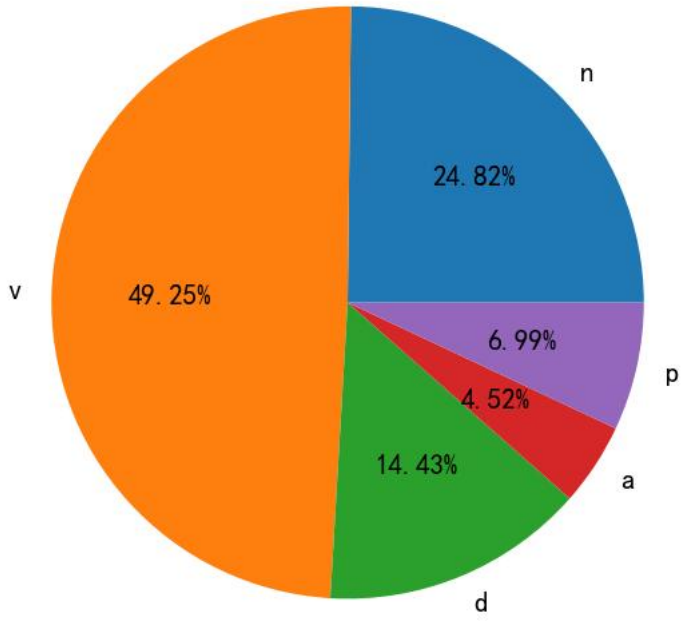
```python
###分析两本类型不同的现代小说的词性分布其一

from api import Cut  #从api模块中引入Cut类

novel1 = r"C:\Users\86178\Desktop\Python程序设计\《活着》.txt"
cut1 = Cut(novel1)  #实例化对象1
print(cut1[99])  #类对象取索引
cut1.show_pie()  #获取词性分布统计饼图
```

```python
###分析两本类型不同的现代小说的词性分布其二

from api import Cut  #从api模块中引入Cut类

novel2 = r"C:\Users\86178\Desktop\Python程序设计\《神雕侠侣》.txt"
cut2 = Cut(novel2)  #实例化对象2
print(cut2[99])
cut2.show_pie()
```

②结果截图

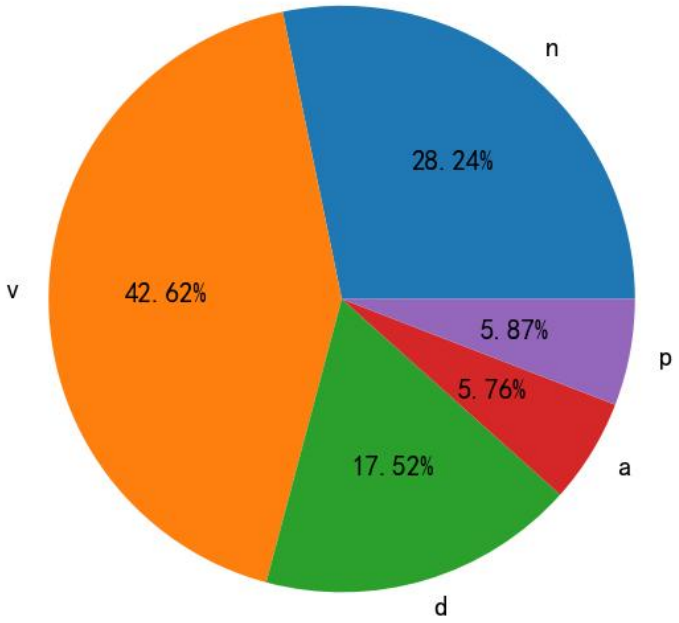
```
PS C:\Users\86178> & C:/Users/86178/.conda/envs/GAN/python.exe c:/Users/86178/Desktop/Python程序设计/实例化对象1.py
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\86178\AppData\Local\Temp\jieba.cache
Loading model cost 0.376 seconds.
Prefix dict has been built successfully.
起/v
```

C:\Users\86178\Desktop\Python程序设计\《活着》.txt主要词性分布统计图




```
PS C:\Users\86178> & C:/Users/86178/.conda/envs/GAN/python.exe c:/Users/86178/Desktop/Python程序设计/实例化对象2.py
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\86178\AppData\Local\Temp\jieba.cache
Loading model cost 0.371 seconds.
Prefix dict has been built successfully.
离愁/v
```

C:\Users\86178\Desktop\Python程序设计\《神雕侠侣》.txt主要词性分布统计图



结论

综上所诉，比较《活着》和《神雕侠侣》两本不同类型现代小说的饼图可知，它们相同之处在于两者都以动词（v)居多，频率接近 50%，其次名词（n）、副词（d）介词（p），形容词（a）最少；不同之处在于前者动词与介词比后者更多，后者名词、副词、形容词更多。结巴分词能够很好地对中文文本进行分词和词性标注等工作。

参考文献

[1] 余华.活着[M].北京十月文艺出版社,2021.

[2] 金庸.神雕侠侣[M].广州出版社,2017.

[3] https://matplotlib.org/
[4] http://t.csdn.cn/P8xdL
[5] http://t.csdn.cn/W9G7U

附录

```
###利用 Python 数据模型的 API 建立一个数据接口类并实现数据可视化


import jieba                        #引入所需库
import jieba.posseg as pseg
import matplotlib.pyplot as plt
```

```
seg_list = []
class Cut():
    """分词并统计类"""
```

```
    def __init__(self,name):
        """初始化属性"""
        self.name = name
        self.labels = ['n','v','d','a','p']  #词性标签,主要对名词，动词，副词，形容词及介词进行统计
        self.num_list = [0,0,0,0,0]  #数量列表
```

```
    def __getitem__(self,index):
        """实现类对象取索引返回词和词性两项主要信息"""
        with open(self.name,"r",encoding='utf-8') as f:
            for line in f.readlines():
                for w in pseg.cut(line):  #使用 jieba 对 line 分词及词性标注
                    seg_list.append(w)
            return seg_list[index]
```

```
    def show_pie(self):
        """运用 jieba 分词实现词性统计并用饼图可视化"""
        with open (self.name,'r',encoding='utf-8') as f1:
            for line in f1.readlines():
                words = pseg.cut(line)  #使用 jieba 对 line 分词及词性标注
                for x in words:
                    if x.flag  in self.labels:  #判断 x 的词性是否存在词性列表中
                        if x.flag == 'n':
                            index1 = 0
                        if x.flag == 'v':
                            index1 = 1
                        if x.flag == 'd':
                            index1 = 2
                        if x.flag == 'a':
                            index1 = 3
                        if x.flag == 'p':
                            index1 = 4
                        self.num_list[index1] += 1  #统计各类词性总数
        rate = []
        for i in range(len(self.num_list)):
```

```
                        index1 = 4
                    self.num_list[index1] += 1  #统计各类词性总数
        rate = []
        for i in range(len(self.num_list)):
            result = self.num_list[i]/sum(self.num_list)  #获取词性频率
            rate.append(result)
        plt.rcParams['font.sans-serif'] = ['SimHei']  #引入绘图数据
        plt.pie(x=rate,autopct='%1.2f%%',labels=self.labels)
        plt.title(self.name+"主要词性分布统计图")
        plt.show()  #输出饼图
```

```
###分析两本类型不同的现代小说的词性分布其一


from api import Cut  #从 api 模块中引入 Cut 类
```

```
novel1 = r"C:\Users\86178\Desktop\Python 程序设计\《活着》.txt"
cut1 = Cut(novel1)  #实例化对象 1
print(cut1[99])  #类对象取索引
cut1.show_pie()  #获取词性分布统计饼图


###分析两本类型不同的现代小说的词性分布其二


from api import Cut  #从 api 模块中引入 Cut 类
```

```
novel2 = r"C:\Users\86178\Desktop\Python 程序设计\《神雕侠侣》.txt"
cut2 = Cut(novel2)  #实例化对象 2
print(cut2[99])
cut2.show_pie()
```