# 问题重述

## 算法介绍

### 基于规则的标注器

基于规则的标注器是一种简单而有效的词性标注方法，它基于语法规则来标注文本中的词性。通常情况下，规则是由人工定义的，通常通过观察大量的已标注数据集来设计。例如，我们可以编写一个规则，指定一个单词后面紧跟着一个名词或代词时，将其标记为形容词。这些规则可以通过自然语言工具包（NLTK）中的 RegexpParser 类来定义和实现。规则可以根据需求进行修改，从而不断改进标注器的性能。与其他词性标注方法相比，基于规则的标注器的优点是速度较快、易于理解和修改，缺点是需要较多的人工工作来编写规则，并且可能无法应对所有的语言现象。

### 隐马尔可夫模型标注器

隐马尔可夫模型（Hidden Markov Model，HMM）是一种用于建立观测数据与某个未知的隐含状态之间关系的统计模型。在自然语言处理中，隐马尔可夫模型被广泛应用于词性标注任务中。

在基于HMM的词性标注中，我们假设每个单词的词性只依赖于其前面的N个单词的词性。对于一个给定的单词序列，我们可以通过计算给定词性序列的概率来找到最可能的词性序列。具体地，我们需要计算两个概率：转移概率和发射概率。

转移概率指的是一个词性转移到另一个词性的概率。例如，在一个以名词(N)作为句子开头的语料库中，接下来出现形容词(JJ)的概率可能比动词(V)更高。发射概率指的是一个词在给定词性的情况下出现的概率。例如，在一个名词(N)的上下文中，出现“cat”的概率可能比出现“run”的概率更高。

通过计算这些概率，我们可以使用维特比算法找到给定单词序列的最可能词性序列，从而完成词性标注的任务。

HMM标注器的优点是可以利用大量的训练数据进行训练，并且在训练数据充分的情况下可以取得很好的效果。缺点是需要大量的训练数据，并且在标注的结果中可能会出现一些不连续的错误

## treebank语料库介绍

Treebank语料库是一个广泛使用的英语句法树库，它包含了手工标注的英语文本、对应的句法树以及词性标注。该语料库最初由宾夕法尼亚大学与卡内基梅隆大学联合开发，目的是为了提供一个用于句法研究和自然语言处理的标准数据集。Treebank语料库的规模较大，包含了大量的新闻、杂志、广播新闻和会话等不同类型的文本，它已成为自然语言处理领域中的重要资源之一。

在Treebank语料库中，每个句子都被标注了其对应的句法树和词性标注。词性标注使用了Penn Treebank词性标记集，包括了大量的词性标记，如名词、动词、形容词、副词、介词、连词等。而句法树则使用了英语句法分析的一种常见形式，即短语结构语法（Phrase Structure Grammar, PSG），也称为上下文无关文法（Context-Free Grammar, CFG）。每个句子的句法树都表示为一个由父节点和子节点组成的树状结构，每个节点代表一个短语或单词，其中短语节点包含多个子节点，而单词节点不包含任何子节点。

Treebank语料库已经成为自然语言处理中的重要资源，广泛用于语言模型、文本分类、句法分析、语义分析、机器翻译等多个领域的研究和应用中。

# 实验分析

## 实验数据

text = "Hello, I'm Liao Shanchuan from Anhui province and I'm currently studying at Shanghai Normal University. " \  
 "I am a student who likes programming, music and guitar, and I also have the habit of exercising consistently. " \  
 "In terms of programming, my main working language is Java, and I like to use Java to develop various applications, including web applications, mobile applications, etc. " \  
 "I have studied the features of the Java language and its features. I study the features and application scenarios of the Java language in depth, and I am constantly learning new technologies to improve my skills and programming. " \  
 "In terms of research, I mainly use the Python language for data analysis and modeling. I am proficient in the Python language and various related data analysis tools, and I am able to process and analyze large amounts of data and predict future trends through data modeling. " \  
 "In addition, I am also a music lover and guitarist. I like to play guitar, play different styles of music, and relax and enjoy life through music. Overall, I am a person who loves to learn and improve myself constantly. I hope to continue to explore and develop my skills and contribute to society in my future career."

## 实验环境

Pycharm

Python3.9

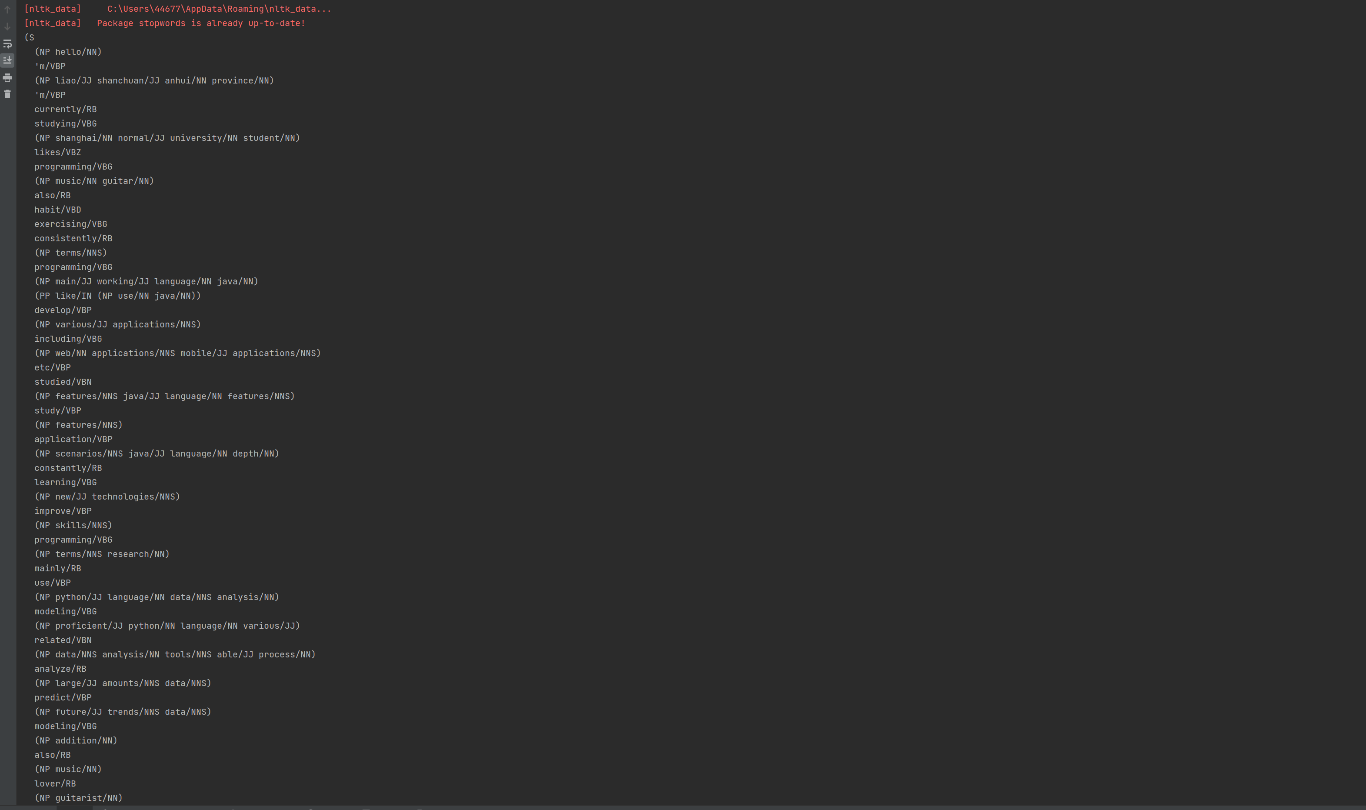
Windows11

## 实验结果与分析

### 基于规则标注器

首先调用pos\_tag\_default函数，函数如下：

def pos\_tag\_default(text):  
 *"""  
 使用基于规则的标注器进行词性标注 不需要训练* ***:param*** *text: 文本* ***:return****: cp为词性标注器(使用 NLTK 自带的 RegexpParser 类来定义匹配规则) tagged\_sentences是词性标注结果  
 """* words = nltk.word\_tokenize(text) # 分词  
 # 去掉标点符号  
 english\_punctuations = [',', '.', ':', ';', '?', '(', ')', '[', ']', '&', '!', '\*', '@', '#', '$', '%']  
 text\_list = [word for word in words if word not in english\_punctuations]  
 # 去掉停用词  
 stops = set(stopwords.words("english"))  
 text\_list = [word for word in text\_list if word not in stops]  
 # 构造基于规则的标注器  
 grammar = r"""  
 NP: {<DT|JJ|NN.\*>+} # 匹配名词短语  
 PP: {<IN><NP>} # 匹配介词短语  
 VP: {<VB.\*><NP|PP|CLAUSE>+$} # 匹配动词短语  
 CLAUSE: {<NP><VP>} # 匹配从句  
 """  
 cp = nltk.RegexpParser(grammar)  
  
 # 进行标注  
 tagged\_words = nltk.pos\_tag(text\_list)  
 tagged\_sentences = cp.parse(tagged\_words)  
  
 return cp, tagged\_sentences

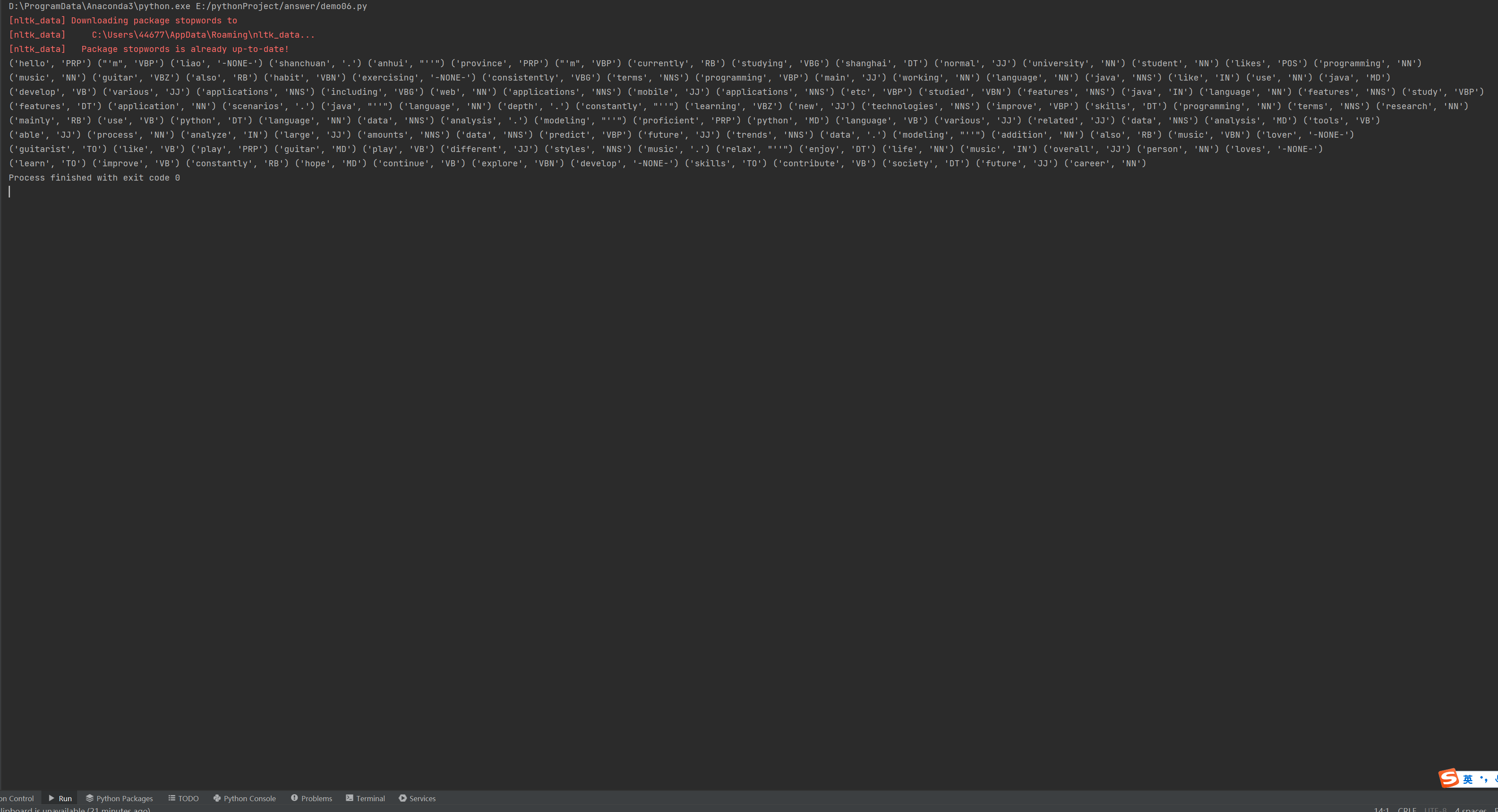
之后传入实验数据(见标题实验数据)然后打印结果即可得到

### 隐马尔可夫模型标注器

首先调用函数pos\_tag\_hmm() 函数具体如下：

def pos\_tag\_hmm(text):  
 *"""  
 使用隐马尔可夫模型标注器进行词性标注 使用treebank语料库进行训练* ***:param*** *text: 训练文本* ***:return****: 第一个返回值是词性标注器，第二个是标注的结果  
 """* words = nltk.word\_tokenize(text) # 分词  
 # 去掉标点符号  
 english\_punctuations = [',', '.', ':', ';', '?', '(', ')', '[', ']', '&', '!', '\*', '@', '#', '$', '%']  
 text\_list = [word for word in words if word not in english\_punctuations]  
 # 去掉停用词  
 stops = set(stopwords.words("english"))  
 text\_list = [word for word in text\_list if word not in stops]  
  
 train\_sents = nltk.corpus.treebank.tagged\_sents()  
 hmm\_tagger = nltk.HiddenMarkovModelTagger.train(train\_sents)  
 return hmm\_tagger, hmm\_tagger.tag(text\_list)

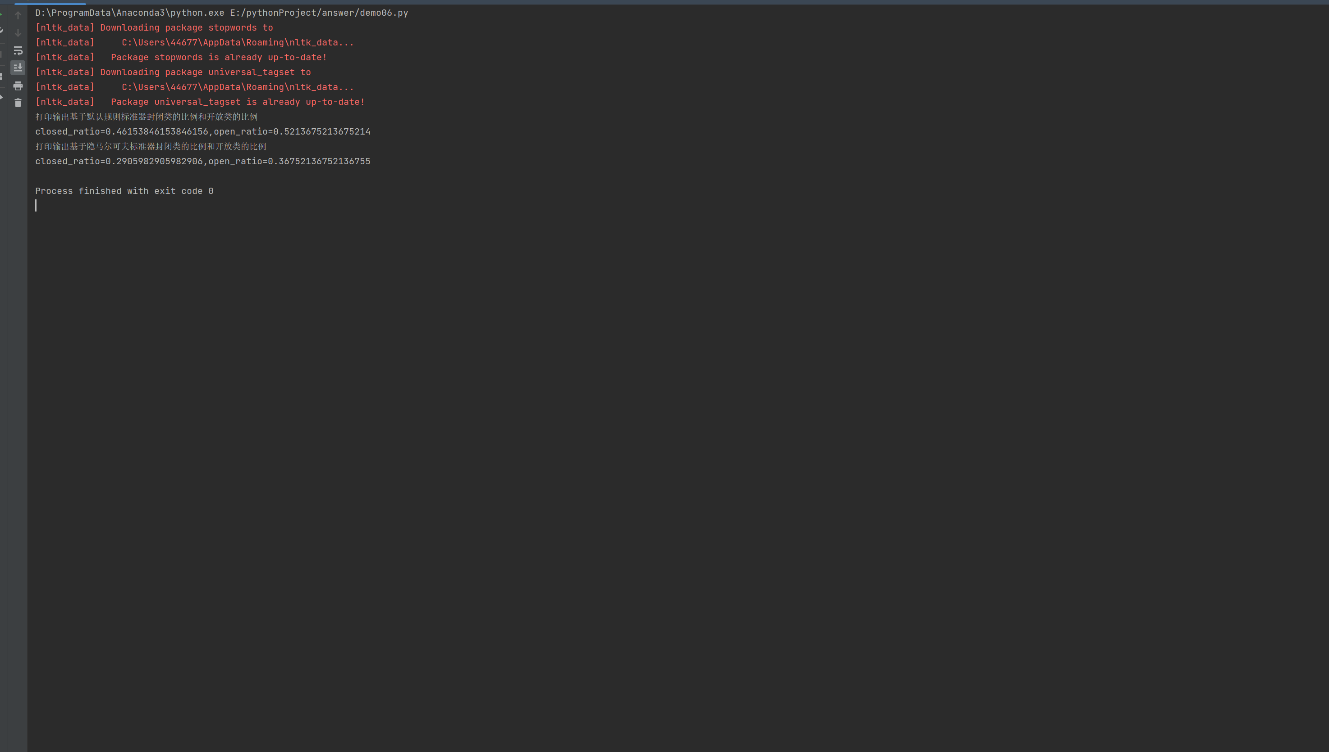
之后传入实验数据(见标题实验数据)然后打印结果即可得到



### 封闭类的比例和开放类的比例

调用函数get\_closed\_open\_ratio，函数如下：

def get\_closed\_open\_ratio(tagged\_words):  
 # 定义封闭类和开放类的词性标签  
 closed\_class\_tags = ['NN', 'NNS', 'NNP', 'NNPS']  
 open\_class\_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'JJ', 'JJR', 'JJS', 'RB', 'RBR', 'RBS']  
  
 # 统计封闭类和开放类的出现频率  
 closed\_count = sum(1 for word, tag in tagged\_words if tag in closed\_class\_tags)  
 open\_count = sum(1 for word, tag in tagged\_words if tag in open\_class\_tags)  
  
 # 计算封闭类和开放类的比例  
 closed\_ratio = closed\_count / len(tagged\_words)  
 open\_ratio = open\_count / len(tagged\_words)  
  
 return closed\_ratio, open\_ratio

传入两个算法标注好的词性实验结果如下

### 两种算法的实验结果

**Precision:**

pos\_tag\_default: 0.7273

pos\_tag\_hmm: 0.8462

**Recall:**

pos\_tag\_default: 0.8077

pos\_tag\_hmm: 0.8462

**F1-score:**

pos\_tag\_default: 0.7659

pos\_tag\_hmm: 0.8462