# 问题重述：

基于Assignment #1的自我介绍（或者重新写一段不少于350字的内容），使用任意编程语言、任意语料库，任意选择**两种词性标注方法**（注意：与Assignment #1选择的算法不同），将自我介绍的内容做词性标注，计算出自我介绍的内容中封闭类的比例和开放类的比例，并**比较两种算法的实验结果**（包括：Precision，Recall，F1-score）。

# 代码如下：

# -\*- coding: utf-8 -\*-  
*"""  
 @Time : 2023/4/24 15:14  
 @Author : liao.sc  
 @File : demo06  
 @Contact : 446773160@qq.com  
"""*import nltk  
from nltk.corpus import stopwords  
  
from sklearn.metrics import precision\_score, recall\_score, f1\_score  
  
nltk.download('stopwords')  
  
# 自我介绍  
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.".lower()  
  
  
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  
  
  
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)  
  
  
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  
  
  
def compare\_pos\_taggers(rule\_based\_tagger, hmm\_tagger, test\_sents):  
 *"""  
 基于规则的标注器进行词性标注和隐马尔可夫模型标注器进行词性标注 两个词性标注器的比较器，比较内容包含Precision，Recall，F1-score* ***:param*** *rule\_based\_tagger:规则的标注器* ***:param*** *hmm\_tagger:马尔可夫模型标注器* ***:param*** *test\_sents:标注文本* ***:return****:  
 """* rule\_based\_preds = [rule\_based\_tagger.tag(sent) for sent in test\_sents]  
 rule\_based\_true = [[(w, t) for w, t in sent] for sent in test\_sents]  
 hmm\_preds = [hmm\_tagger.tag(sent) for sent in test\_sents]  
 hmm\_true = [[(w, t) for w, t in sent] for sent in test\_sents]  
  
 rule\_based\_flat\_preds = [t for sent in rule\_based\_preds for \_, t in sent]  
 rule\_based\_flat\_true = [t for sent in rule\_based\_true for \_, t in sent]  
 hmm\_flat\_preds = [t for sent in hmm\_preds for \_, t in sent]  
 hmm\_flat\_true = [t for sent in hmm\_true for \_, t in sent]  
  
 precision\_rule\_based = precision\_score(rule\_based\_flat\_true, rule\_based\_flat\_preds, average='weighted')  
 recall\_rule\_based = recall\_score(rule\_based\_flat\_true, rule\_based\_flat\_preds, average='weighted')  
 f1\_score\_rule\_based = f1\_score(rule\_based\_flat\_true, rule\_based\_flat\_preds, average='weighted')  
 precision\_hmm = precision\_score(hmm\_flat\_true, hmm\_flat\_preds, average='weighted')  
 recall\_hmm = recall\_score(hmm\_flat\_true, hmm\_flat\_preds, average='weighted')  
 f1\_score\_hmm = f1\_score(hmm\_flat\_true, hmm\_flat\_preds, average='weighted')  
  
 print('Results for rule-based tagger:')  
 print(f'Precision: {precision\_rule\_based:.3f}')  
 print(f'Recall: {recall\_rule\_based:.3f}')  
 print(f'F1-score: {f1\_score\_rule\_based:.3f}')  
 print('Results for HMM tagger:')  
 print(f'Precision: {precision\_hmm:.3f}')  
 print(f'Recall: {recall\_hmm:.3f}')  
 print(f'F1-score: {f1\_score\_hmm:.3f}')