第一章

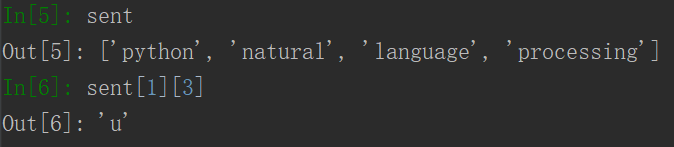
**2.26个字母可以组成26的10次方或者26\*\*10个10字母长的字符串。也就是141167095653376L（结尾处的L只是表示这是Python长数字格式）。100个字母长度的字符串可能有多少个？**

>>>26\*\*100

Out[3]:3142930641582938830174357788501626427282669988762475256374173175398995908420104023465432599069702289330964075081611719197835869803511992549376

**13.我们已经看到如何用词链表表示一个句子，其中每个词是一个字符序列。sent1[2][2]代表什么意思？为什么？请用其他的索引值做实验。**

sent1[2][2]代表sent1中第3个词的第3个字母。



**24.写表达式找出text6中所有符合下列条件的词。结果应该是词链表的形式： ['word`1', 'word2', ...]。**

**a. 以 ize 结尾**

**b. 包含字母 z**

**c. 包含字母序列 pt**

**d. 除了首字母外是全部小写字母的词（即titlecase）**

a.>>>[w for w in text6 if w.endswith(‘ize’)]



b.>>>[w for w in text6 if ‘z’ in w]



c.>>>[w for w in text6 if ‘pt’ in w]

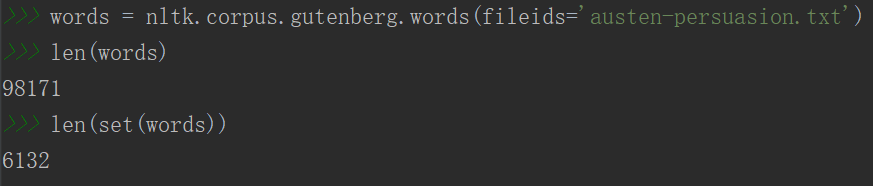


d.>>>[w for w in text6 if w.istitle()]



第二章

**2. 使用语料库模块处理austen-persuasion.txt。这本书中有多少词标识符？多少词类型？**



**13. 没有下位词的名词同义词集所占的百分比是多少？你可以使用wn.all\_synsets('n')得到所有名词同义词集。**

import nltk

from nltk.corpus import wordnet as wn

all\_noun = wn.all\_synsets('n')

all\_noun\_num = len(set(all\_noun))

noun\_have\_no\_hypon = filter(lambda ss: len(ss.hyponyms()) <= 0, wn.all\_synsets('n'))

noun\_have\_num = len(list(noun\_have\_no\_hypon))

print('There are %d nouns, and %d nouns without hyponyms, the percentage is %f' %(all\_noun\_num, noun\_have\_num, noun\_have\_num/all\_noun\_num\*100))



24.修改例2-1的文本生成程序，进一步完成下列任务：

a. 在一个词链表中存储 n个最相似的词，使用random.choice()从链表中随机选取一个词。（你将需要事先import random）

b. 选择特定的文体， 如： 布朗语料库中的一部分或者《 创世纪》 翻译或者古腾堡语料库中的文本或者一个网络文本。 在此语料上训练一个模型， 产生随机文本。 你可能要实验不同的起始字。文本的可理解性如何？讨论这种方法产生随机文本的长处和短处。

c. 现在使用两种不同文体训练你的系统， 使用混合文体文本做实验。讨论你的观察结果。

import nltk

cfdist = nltk.ConditionalFreqDist()

def generate\_model(cfdist,word,num=15):

for i in range(num):

print(word)

word = cfdist[word].max()

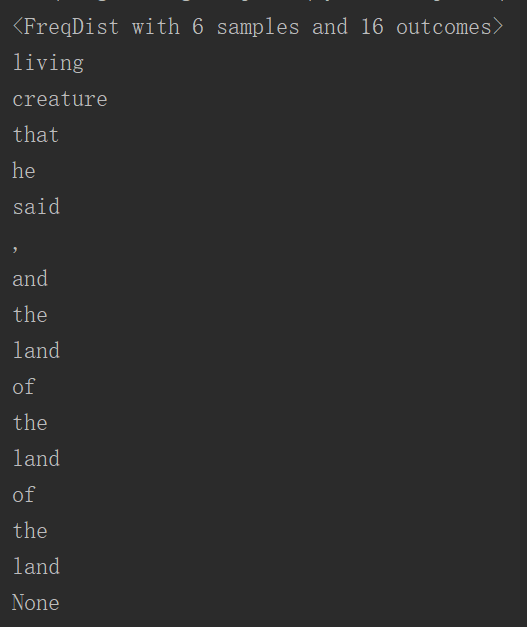
text = nltk.corpus.brown.words('english-kjv.txt')

bigrams = nltk.bigrams(text)

cfd = nltk.ConditionalFreqDist(bigrams)

print (cfd['living'])

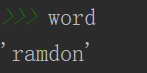
print(generate\_model(cfd,'living'))print(generate\_model(cfd,'tree'))



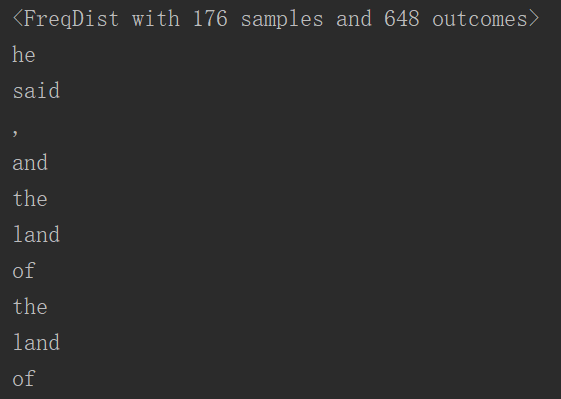
a.import random

words=['ramdom','rando','ramdon','ramdon']

word = random.choice(words)

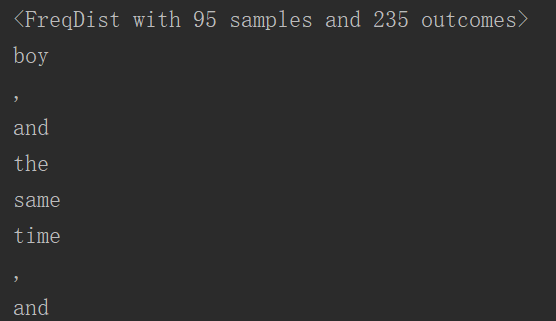


b. print(generate\_model(cfd,'he'))



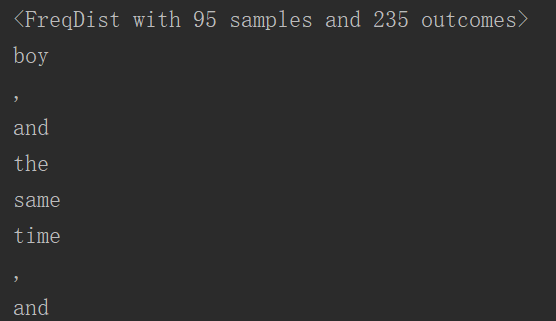
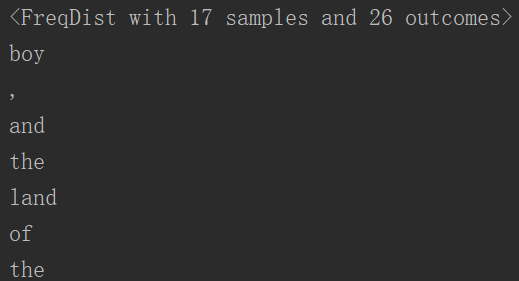
print (cfd['boy'])

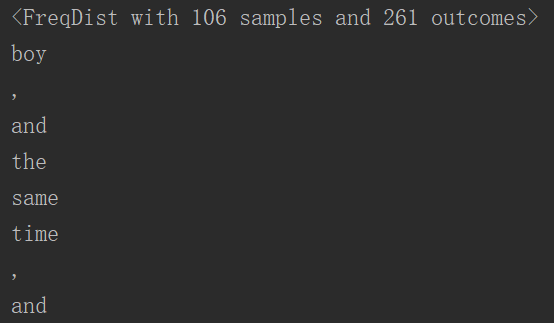
print(generate\_model(cfd,'boy'))



生成的随机文本很容易陷入“双连词循环”中，例如：ab；bc；ca。生成的文本可理解性不强。更换语料库也并没有消除掉这种现象，最后一个关键词‘boy’是在布朗语料库中产生随机文本的，但也出现了循环现象。

c．

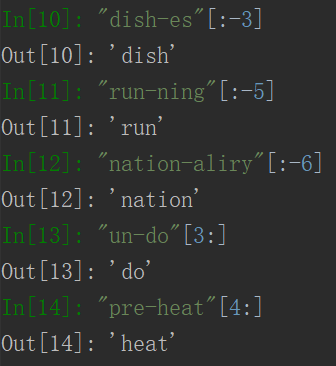




第一个是在genesis语料库中，第二个是在brown语料库中，第三个是在两个语料库的混合语料库中训练得出的结果。在不同的语料库中可能得到不同的结果，在混合语料库中结果一般与更大的语料库结果相似。

第三章

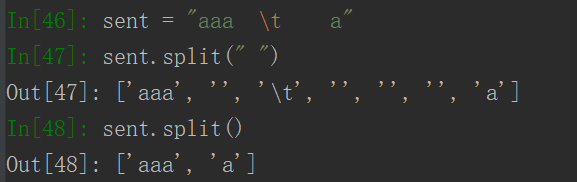
2. 我们可以使用切片符号删除词汇形态上的结尾。例如： 'dogs'[:-1]删除了 dogs的最后一个字符，留下 dog。使用切片符号删除下面这些词的词缀（我们插入了一个连字符指示词缀的边界，请在你的字符串中省略掉连字符）： dish-es, run-ning, nation-ality, un-do, pre-heat

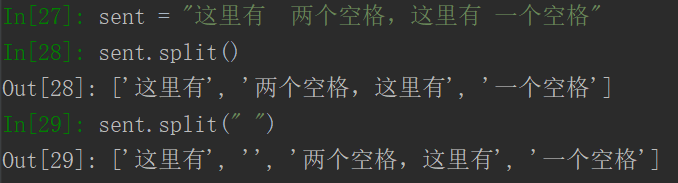


13. 不带参数的split与以' '作为参数的 split，即sent.split()与sent.split(' ')的区别是什么？

当被分割的字符串包含制表符、连续的空格或一个制表符与空格的序列会发生什么？

（在 IDLE中你将需要使用'\t'来输入制表符。）





我们可以看到，有“ “作为参数的切分会严格按照空格处切分并且制表符会以\t的形式出现，而没有参数的切分只会将空格断开的部分切开，并不会将制表符和空格单独切出来。

24. 尝试编写代码将文本转换成hAck3r， 使用正则表达式和替换， 其中 e→3， i→1， o→0， 1→|， s→5, .→5w33t!，ate→8。 在转换之前将文本规范化为小写。 自己添加更多的128替换。现在尝试将 s映射到两个不同的值：词开头的s映射为$，词内部的 s映射为5。

用正则表达式实现替换：

words = "I love Python NLP . This is a start of the test . ate"

word1=words.lower().split()

print(word1)

result=[]

for i in word1:

if 'e'in i:

result.append(i.replace('e','3'))

if 'i' in i:

result.append(i.replace('i','1'))

if 'o' in i:

result.append(i.replace('o','0'))

if '1' in i:

result.append(i.replace('1','|'))

if 's' in i:

result.append(i.replace('s','5'))

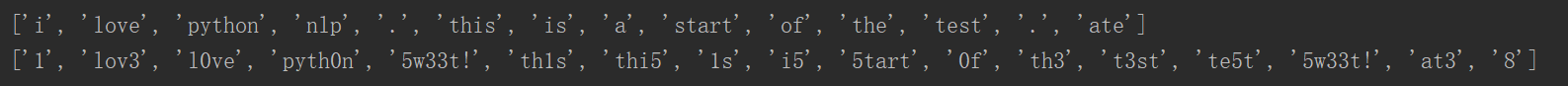
if '.' in i:

result.append(i.replace('.','5w33t!'))

if 'ate' in i:

result.append(i.replace('ate','8'))

print(result)



s映射到两个不同的值：

import re

text="I love Python NLP . This is a start of the test . ate"

print(text)

words=text.split()

for i in range(len(words)):

if words[i] in re.findall(r'\w\*\S\*s\w\*', text):

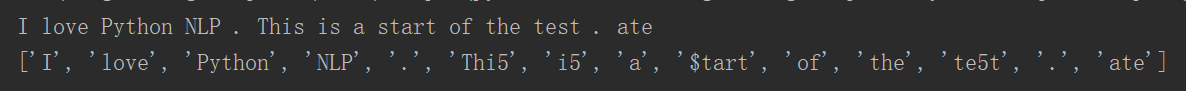
if words[i].startswith('s'):

words[i] = words[i].replace(words[i][0], '$')

else:

words[i] = words[i].replace('s', '5')

print(words)



35．阅读LanguageLog 中关于短语的as best as p can和 as best p can形式的帖子，其中p是一个代名词。 在一个语料库和3.5节中描述的搜索已标注的文本的 findall()方法的帮助下， 调查这一现象。 帖子在 http://itre.cis.upenn.edu/ myl/languagelog/archives/002733.htm

from nltk.corpus import webtext

import re

text = ' '.join(webtext.words())

text2 = 'as best as I can as best you can as best can'

text1 = ' '.join(text2.split())

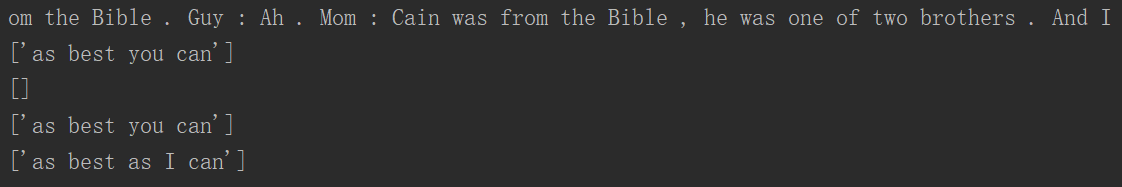
print (text)

print(re.findall(r'as best \w+ can',text))

print(re.findall(r'as best as \w+ can',text))

print(re.findall(r'as best \w+ can',text1))

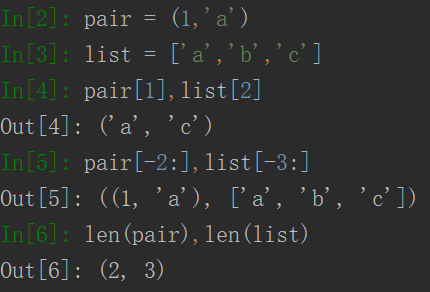
print(re.findall(r'as best as \w+ can',text1))



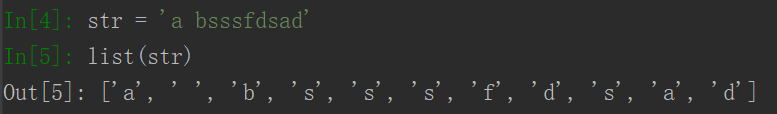
第四章

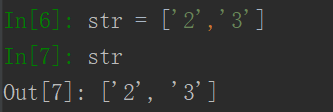
2. 确定三个同时在元组和链表上都可以执行的操作。确定三个不能在元组上执行的链表操作。命名一段使用链表替代元组会产生一个 Python错误的代码。

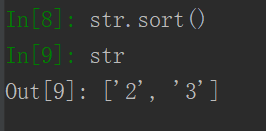
在元组和链表上都可以执行的操作：索引、切片、长度。



不能在元组上执行的链表操作：list、赋值、排序。







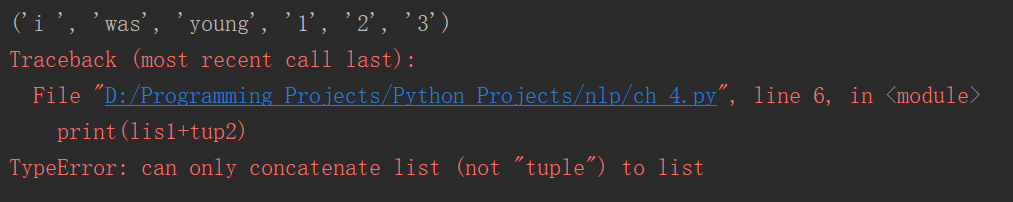
tup1 = ('i ', 'was', 'young')

tup2 = ('1', '2', '3')

print(tup1+tup2)

lis1 = list(tup1)

print(lis1+tup2)



13. 写代码初始化一个称为word\_vowels的二维数组的集合，处理一个词链表，添加每个词到word\_vowels[1][V]，其中l是词的长度， v是它包含的元音的数量。

from nltk.corpus import brown

words = brown.words(categories = 'news')[:10]

word\_vowels = [[[] for row in range(10)]for a in range(20)]

def count\_vowels( word ):

vowel = "aeiou"

num = 0

for char in word:

if char in vowel:

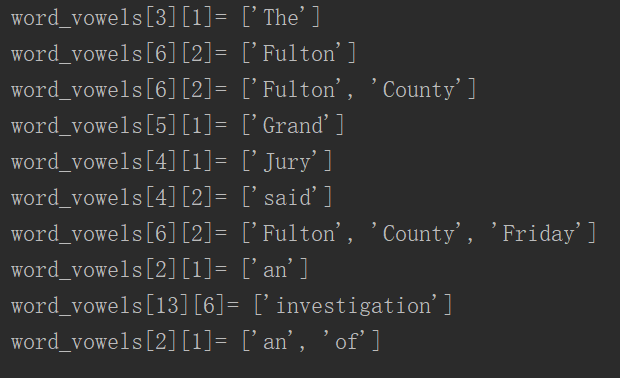
num += 1

return num

for word in words:

word\_vowels[len(word)][count\_vowels(word)].append(word)

print("word\_vowels[%d][%d]=" %(len(word),count\_vowels(word)),word\_vowels[len(word)][count\_vowels(word)])



24. 阅读关于“ 关键字联动”的内容（ (Scott & Tribble, 2006)的第5章）。从 NLTK的莎士比亚语料库中提取关键字，使用 NetworkX包，画出关键字联动网络

找不到这本书。。。

35. 写一个程序实现发现一种n×n的四方联词的蛮力算法：纵横字谜，它的第n行的词

与第n列的词相同。讨论见http://itre.cis.upenn.edu/ myl/languagelog/archives/002679.ht

ml

word = ["The", "moment", "you", "think", "about", "giving", "up"]

def zonghengzimi(word):

# 生成以word为基础的，10X10的10阶乘

import pandas as pd

import numpy as np

import random

raw = []

for i in range(100):

raw.append(chr(random.randint(97, 122)))

data = pd.DataFrame(np.array(raw).reshape(10, 10))

print(len(data))

print(data)

# 生成了10X10的阶乘随机字母，需要把七个词语放进去，并把一部分改成空白元素进行填写

for i in range(len(word)):

a = random.randint(1, 10)

b = random.randint(1, 10)

for j in range(len(word[i])):

if a > b and b + len(word[i]) < 10:

data.ix[a, b + j] = word[i][j]

if a > b and b + len(word[i]) > 10:

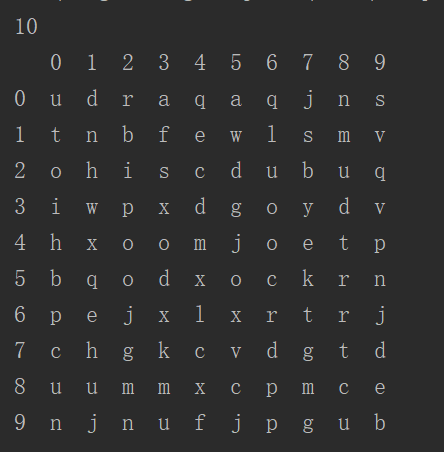
b = 0

data.ix[a, b + j] = word[i][j]

return data

# 这样就简单生成了一个纵横字谜,尝试在这些子母中找出相应的word当中的单词

zonghengzimi(word)



第五章

2. 找个同伴，轮流挑选一个既可以是名词也可以是动词的词（如： contest）；让对方预测哪一个可能是布朗语料库中频率最高的。检查对方的预测，为几个回合打分。

#两个词（he，she）在布朗语料库中出现的频率

from nltk.corpus import brown

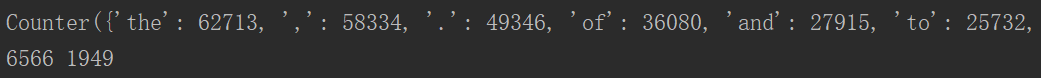
import collections

word = brown.words()

words = collections.Counter(word)

print (words)

print (words['he'],words['she'])

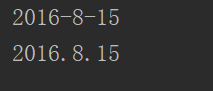


13. 我们可以使用字典指定由一个格式化字符串替换的值。 阅读关于格式化字符串的Python库文档（ http://docs.python.org/lib/typesseq-strings.html），使用这种方法以两种不同的格式显示今天的日期。

d={'year':2016,'month':8,'day':15}

print ('%s-%s-%s' %( d['year'],d['month'],d['day']))

print ('%s.%s.%s' %( d['year'],d['month'],d['day']))



24. 数据稀疏问题有多严重？调查n-gram标注器当 n从 1增加到 6时的性能。为准确性得分制表。 估计这些标注器需要的训练数据， 假设词汇量大小为 10⁵而标记集的大小为10²。

import nltk

from nltk.corpus import brown

brown\_tagged\_sents = brown.tagged\_sents()

brown\_sents = brown.sents()

size = int(len(brown\_tagged\_sents)\*0.9)

train\_sents = brown\_tagged\_sents[:size]

text\_sents = brown\_tagged\_sents[size:]

bigram\_tagger = nltk.BigramTagger(train\_sents,cutoff=1)

print ("cutoff=1,正确性得分为：",bigram\_tagger.evaluate(text\_sents))

bigram\_tagger = nltk.BigramTagger(train\_sents,cutoff=2)

print ("cutoff=2,正确性得分为：",bigram\_tagger.evaluate(text\_sents))

bigram\_tagger = nltk.BigramTagger(train\_sents,cutoff=3)

print ("cutoff=3,正确性得分为：",bigram\_tagger.evaluate(text\_sents))

bigram\_tagger = nltk.BigramTagger(train\_sents,cutoff=4)

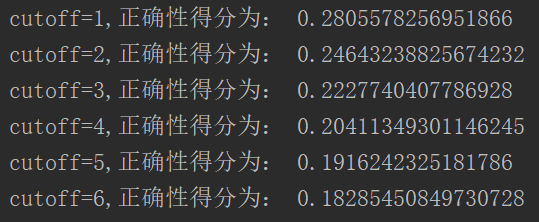
print ("cutoff=4,正确性得分为：",bigram\_tagger.evaluate(text\_sents))

bigram\_tagger = nltk.BigramTagger(train\_sents,cutoff=5)

print ("cutoff=5,正确性得分为：",bigram\_tagger.evaluate(text\_sents))

bigram\_tagger = nltk.BigramTagger(train\_sents,cutoff=6)

print ("cutoff=6,正确性得分为：",bigram\_tagger.evaluate(text\_sents))



35. 写一个程序，按照词must后面的词的标记为它的上下文分类。这样可以区分 must

的“ 必须” 和“ 应该” 两种词意上的用法吗？

import nltk

from nltk.corpus import brown

from nltk.util import ngrams

brown\_sents = ''.join('%s' %id for id in list(brown.sents()) if id not in [',',' ','.',",","."])

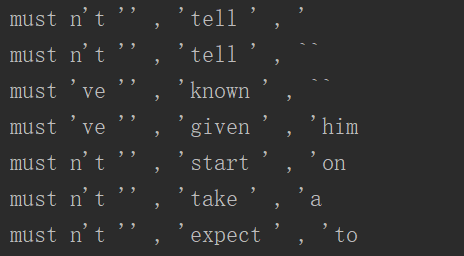
token=nltk.word\_tokenize(brown\_sents)

bigrams=ngrams(token,8)

for (i,j,k,d,a,b,c,f) in bigrams:

if i=="must":

print(i,j,k,d,a,b,c,f)



好像没有办法判断“必须”和“应该”的区别。

第六章

2. 使用任何本章所述的三种分类器之一， 以及你能想到的特征， 尽量好的建立一个名字性别分类器。从将名字语料库分成 3个子集开始： 500个词为测试集， 500个词为开发测试集，剩余 6900个词为训练集。然后从示例的名字性别分类器开始，逐步改善。使用开发测试集检查你的进展。一旦你对你的分类器感到满意， 在测试集上检查它的最终性能。相比在开发测试集上的性能，它在测试集上的性能如何？这是你期待的吗？

from nltk.corpus import names

import nltk

import random

def gender\_features(word):

return {'suffix1':word[-1],'suffix2':word[-2],'first\_letter':word[0],'length':len(word)}

print(gender\_features('John'))

names = ([(name,'male') for name in names.words('male.txt')]+[(name,'female') for name in names.words('female.txt')])

random.shuffle(names)

train\_names,devtest\_names,test\_names = names[6900:],names[6400:6900],names[:500]

train\_set= [(gender\_features(n),g) for (n,g) in train\_names]

devtest\_set= [(gender\_features(n),g) for (n,g) in devtest\_names]

test\_set= [(gender\_features(n),g) for (n,g) in test\_names]

classifier = nltk.NaiveBayesClassifier.train(train\_set)

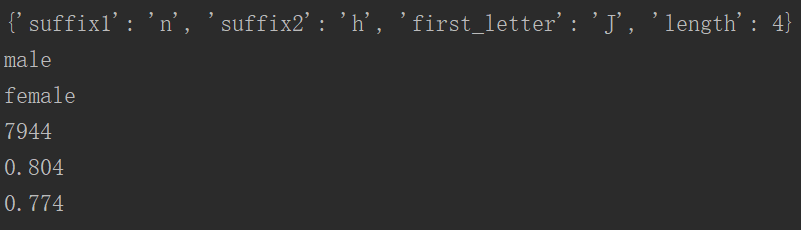
print(classifier.classify(gender\_features('Neo')))

print(classifier.classify(gender\_features('Trinity')))

print(len(names))

print(nltk.classify.accuracy(classifier,devtest\_set))

print(nltk.classify.accuracy(classifier,test\_set))



每次运行这个程序得出的准确率都有差别但基本在0.75-0.80之间

第七章

2.写一个标记模式匹配包含复数中心名词在内的名词短语， 如many/JJ researchers/NNS, two/CD weeks/NNS, both/DT new/JJ positions/NNS。尝试使用泛化处理单数名词短语的标记模式。

import nltk

sentence = [("the","DT"),("little","JJ"),("yellow","JJ"),("dogs","NNS"),("barked","VBD"),("at","NN"),("the","DT"),("cat","NN")]

grammar = r"""

NP: {<CD|DT|PRP\$>?<JJ>\*<NNS>}

{<NN>}

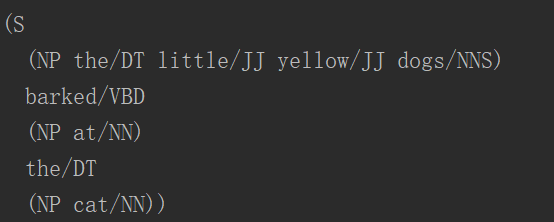
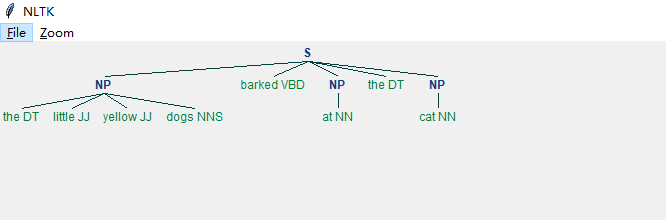
"""

cp = nltk.RegexpParser(grammar)

result = cp.parse(sentence)

print(result)

result.draw()

13. 挑选CoNLL分块语料库中三种块类型之一。写一个函数为你选择的类型做以下任务：

a. 列出与此块类型的每个实例一起出现的所有标记序列。

b. 计数每个标记序列的频率， 并产生一个按频率减少的顺序排列的列表； 每行要包含一个整数（频率）和一个标记序列。

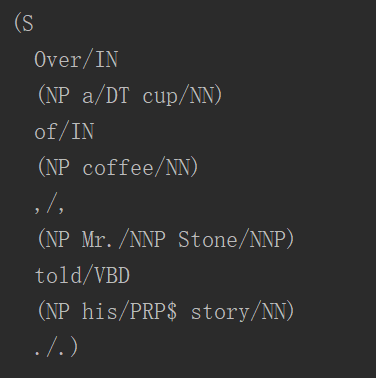
c. 检查高频标记序列。使用这些作为开发一个更好的分块器的基础。

from nltk.corpus import conll2000

temp = conll2000.chunked\_sents('train.txt',chunk\_types=['NP'])[99]

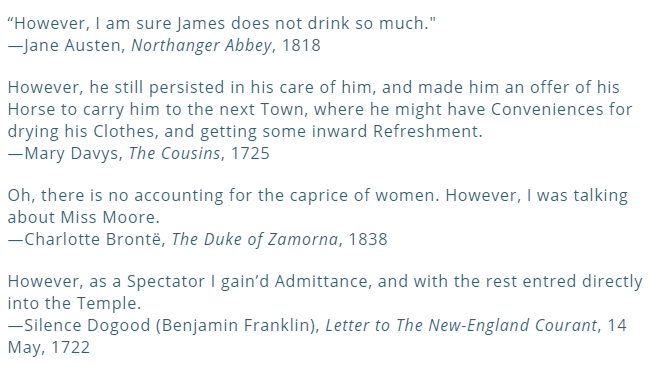
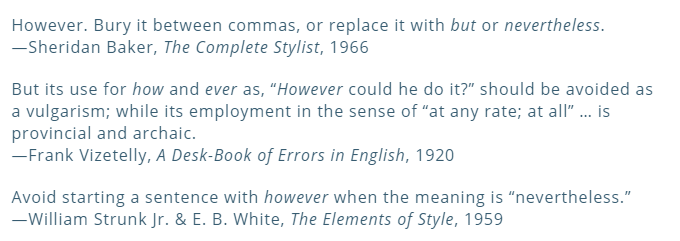
print(temp)

print(type(temp))



第八章

2回想一下Strunk和 White 禁止在句子开头使用however表示“ although” 的意思。在网上搜索句子开头使用however的范围有多广？



13. 思考词序列： Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo。如 http://en.wikipedia.org/wiki/Buffalo\_buffalo\_Buffalo\_buffalo\_buffalo\_buffalo\_Buffalo\_buffalo解释的， 这是一个语法正确的句子。 思考此维基百科页面上表示的树形图， 写一个合适的文法。 正常情况下是小写， 模拟听到这句话时听者会遇到的问题。 你能为这句话找到其他的解析吗？当句子变长时分析树的数量如何增长？（ 这些句子的更多的例子可以在http://en.wikipedia.org/wiki/List\_of\_homophonous\_phrases找到。

import nltk

groucho\_grammar = nltk.CFG.fromstring('''

S -> NP VP

NP -> NP RC | PN N

VP -> V NP

RC -> NP V

PN -> "Buffalo"

N -> "buffalo"

V -> "buffalo"

''')

sent = ['Buffalo','buffalo','Buffalo','buffalo','buffalo','buffalo','Buffalo','buffalo']

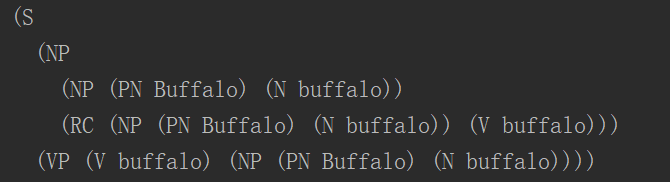
parser = nltk.ChartParser(groucho\_grammar)

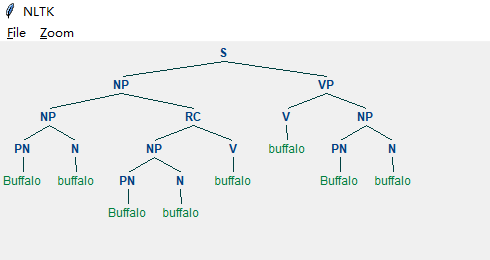
trees = parser.parse(sent)

for tree in trees:

print(tree)

tree.draw()





24. 编写一个递归函数生成嵌套了括号形式的一棵树，去掉叶节点并在子树之后显示非终结符。 于是8.6节中的关于 Pierre Vinken的例子会产生： [[[NNP NNP]NP ,[ADJP [CD NNS]NP JJ]ADJP ,]NP-SBJ MD [VB [DT NN]NP [IN [DT JJ NN]NP]PP-CLR[NNP CD]NP-TMP]VP .]S。连续的类别应用空格分隔。

35. 修改函数init\_wfst()和complete\_wfst()，当一个非终结符添加到WFST中的单元时，它包括了它所派生的单元的记录。实现一个函数，将一个分析树的 WFST转换成这种形式。

第九章

2. 对例9-1中文法进行改变，使用特征COUNT来区分下面的句子：

(56) a. The boy sings.

b. \*Boy sings.

(57) a. The boys sing.

b. Boys sing.

(58) a. The water is precious.

b. Water is precious.

% start S

# ###################

# Grammar Productions

# ###################

# S expansion productions

S -> NP[NUM=?n,COUNT=?c] VP[NUM=?n,COUNT=?c]

S -> NP[NUM=?n,COUNT=?c] V[NUM=?n,COUNT=?c] ADJ

# NP expansion productions

NP[NUM=?n,COUNT=?c] -> N[NUM=?n,COUNT=?c]

NP[NUM=?n,COUNT=?c] -> Det[NUM=?n,COUNT=?c] N[NUM=?n,COUNT=?c]

NP[NUM=pl,COUNT=yes] -> N[NUM=pl,COUNT=yes]

# VP expansion productions

VP[NUM=?n,COUNT=?c] -> V[NUM=?n,COUNT=?c]

# ###################

# Lexical Productions

# ###################

Det -> 'the'

N[NUM=sg,COUNT=yes] -> 'boy'

N[NUM=sg,COUNT=no] -> 'water'

N[NUM=pl,COUNT=yes] -> 'boys'

V[NUM=sg,COUNT=no] -> 'is'

V[NUM=sg,COUNT=yes] -> 'sing'

V[NUM=pl,COUNT=yes] -> 'sings'

ADJ -> 'precious'

13. 形态范例很少是完全正规的，矩阵中的每个单元的意义有不同的实现方式。例如： 词 walk的现在时态词性变化只有两种不同形式：第三人称单数的walks和所有其他人称和数量组合的 walk。一个成功的分析不应该额外要求 6个可能的形态组合中有 5个有相同的实现方式。设计和实施一个方法处理这个问题。

第十章

2. 翻译下面的句子为一阶逻辑的谓词参数公式。

a. Angus likes Cyril and Irene hates Cyril.

b. Tofu is taller than Bertie.

c. Bruce loves himself and Pat does too.

d. Cyril saw Bertie, but Angus didn’ t.

e. Cyril is a four-legged friend.

f. Tofu and Olive are near each other

a．

b.

c.

*d.*

e.

f.

13. 以(Warren & Pereira, 1982)为出发点， 开发一种技术， 将自然语言查询转换为一种可以更加有效地在模型中评估的形式。例如：给定一个(P(x) & Q(x))形式的查询，将它转换为(Q(x) & P(x))，如果Q的范围比P小。

第十一章

2. 编写一个函数，从一个词汇条目删除指定的字段。（我们可以在把数据给别人之前用它做些清洁，如：删除包含无关或不确定的内容的字段。）

with open('test.txt', 'r') as fpr:

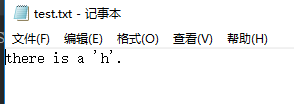
content = fpr.read()

content = content.replace('h', '')

print(content)

with open('test1.txt', 'w') as fpw:

fpw.write(content)

13. 获取CSV格式的比较词表，写一个程序，输出相互之间至少有三个编辑距离的同源词。