自然语言处理 Homework 2

宁晨然 17307130178

1.写程序处理布朗语料库，找到以下问题的答案： 哪些名词常以它们复数形式而不是它们的单数形式出现？（只考虑常规的复数形式，-s 后缀形式的）。b. 哪个词的不同词性标记数目最多？c. 按频率递减的顺序列出标记。前20 个最频繁的词性标记代表什么？d. 名词后面最常见的是哪些词性标记？这些标记代表什么?

ANS:

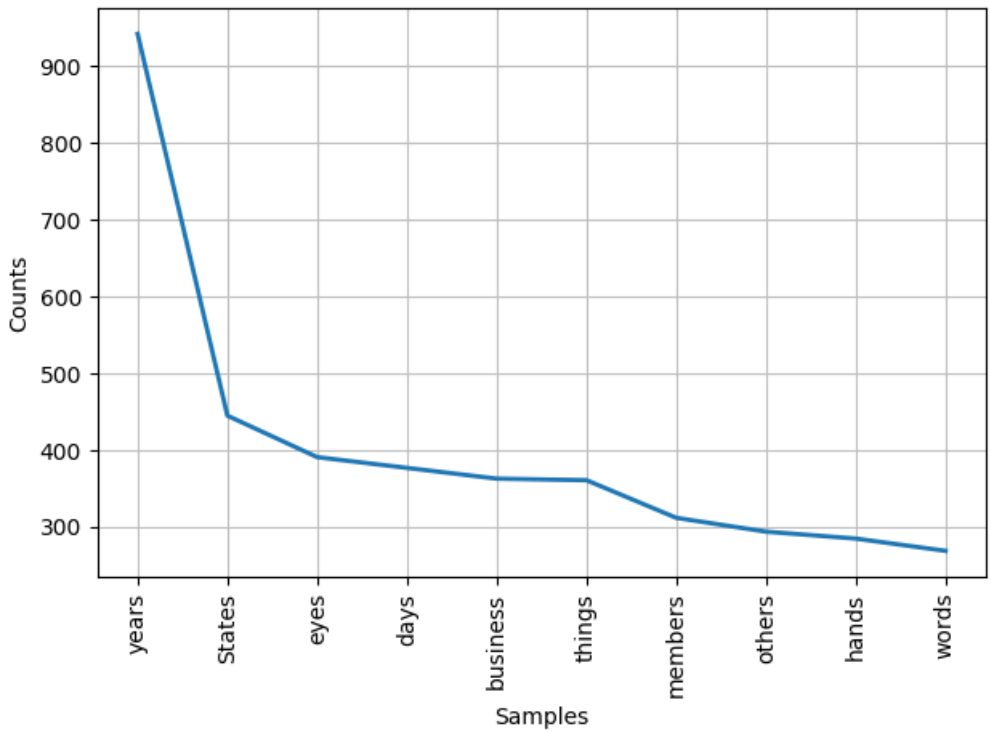
a.复数形式的词汇

```
from nltk.corpus import brown
import nltk

words = brown.tagged_words(tagset='universal')
word_double = nltk.FreqDist(word for (word,tag) in words if tag=='NOUN' and
word.endswith('s') )
word_double.plot(10)
```

首先获取布朗语料库中所有标好词性的词语，这里的`tagset`设置为`universal`，其中涵盖的词性较少比较简单，名词就是`NOUN`。词性为以下：`{'X', 'VERB', 'DET', '.', 'ADV', 'CONJ', 'PRON', 'ADP', 'PRT', 'NOUN', 'NUM', 'ADJ'}`。

筛选出其中名词以's'结尾的词汇，即名词复数形式。



但这样筛选出的名词只是名词复数，也可能是以s结尾的名词，并不是名词复数形式出现更多的名词。所以应该修改代码：

- 先找出名词
- 从名词列表中找出有名词复数的词汇

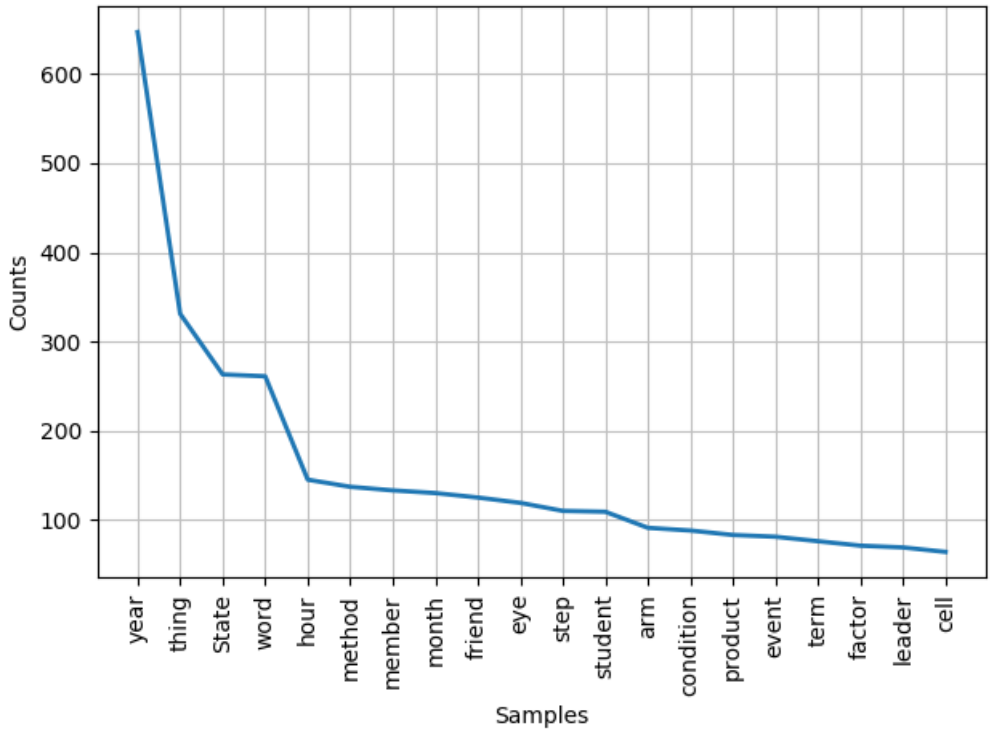
- 从有复数名词的词汇中找出名词复数多于名词单数形式的词汇
- 绘制这种词汇的频率图

```
from nltk.corpus import brown
from nltk import FreqDist

words = brown.tagged_words(tagset='universal')
word_noun = FreqDist(word for (word,tag) in words if tag=='NOUN')
word_has_double = [word for word in word_noun if (word+'s') in word_noun]
word_double_more = [word for word in word_has_double if word_noun[word+'s'] >
word_noun[word]]

word_double = FreqDist(word for (word,tag) in words if tag=='NOUN' and word in
word_double_more)
word_double.plot(20)
```

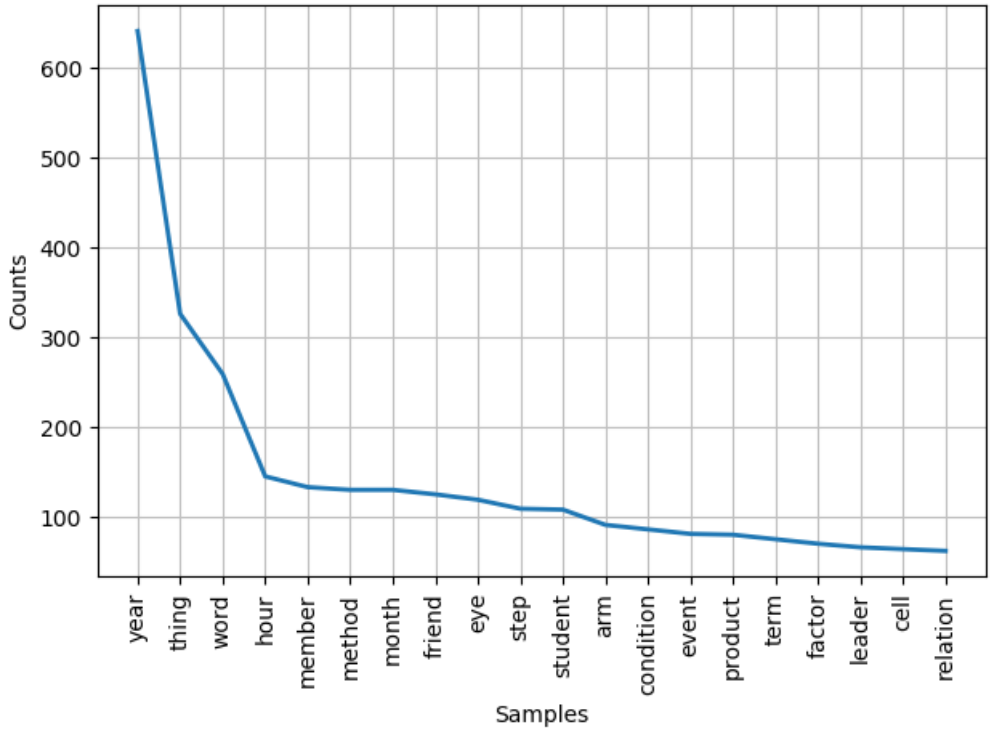
频率图如下：



其实也可以用`tagset=brown`来做，用`tag = 'NN'`和`tag = 'NNS'`来做

```
from nltk.corpus import brown
from nltk import FreqDist

words = brown.tagged_words()
word_noun = FreqDist(word for (word,tag) in words if tag=='NN')
word_double = FreqDist(word for (word,tag) in words if tag=='NNS')
word_has_double = [word for word in word_noun if word+'s' in word_double]
word_double_more = [word for word in word_has_double if word_double[word+'s'] >
word_noun[word]]
w = FreqDist(word for (word,tag) in words if tag=='NN' and word in
word_double_more)
w.plot(20)
```



b. 哪个词的不同词性标记数目最多?

```
from nltk.corpus import brown
words = brown.tagged_words()
words_differ = {}
for (word,tag) in words:
    if word not in words_differ.keys():
        words_differ[word] = [tag]
    elif tag not in words_differ[word]:
        words_differ[word].append(tag)
    else:
        continue

words_differ = [(word,len(words_differ[word])) for word in words_differ.keys()]
words_differ.sort(key=lambda x:x[1],reverse=True)

print(words_differ)
```

获取词性标注数目最多，最好的`tagset`应该是原来默认的`brown`，因为这个标注集的词性分类要细节很多。然后对于每个词语，统计其中`tag`的数目，输出结果。下面是部分结果输出

```
[('that', 12), ('A', 11), ('in', 10), ('to', 10), (':', 9), ('a', 8), ('for',
7), ('it', 7), ('well', 7), ('as', 7), ('fit', 7), ('home', 7), ('out', 7),
('I', 7), ('That', 7), ('Long', 7), ('Little', 7), ('Chinese', 7), ('of', 6),
('the', 6), ('and', 6), ('by', 6), ('many', 6), ('on', 6), ('one', 6), ('more',
6), ('near', 6), ('then', 6), ('can', 6), ('set', 6), ('down', 6), ('you', 6),
('still', 6), ('right', 6), ('cut', 6), ('More', 6), ('How', 6), ('open', 6),
('To', 6), ('Home', 6), ('beat', 6), ('Back', 6), ('B', 6), ('Light', 6),
('Right', 6), ('an', 5), ('no', 5), (',', 5), ('or', 5), ('best', 5), ('cost',
5),...]
```

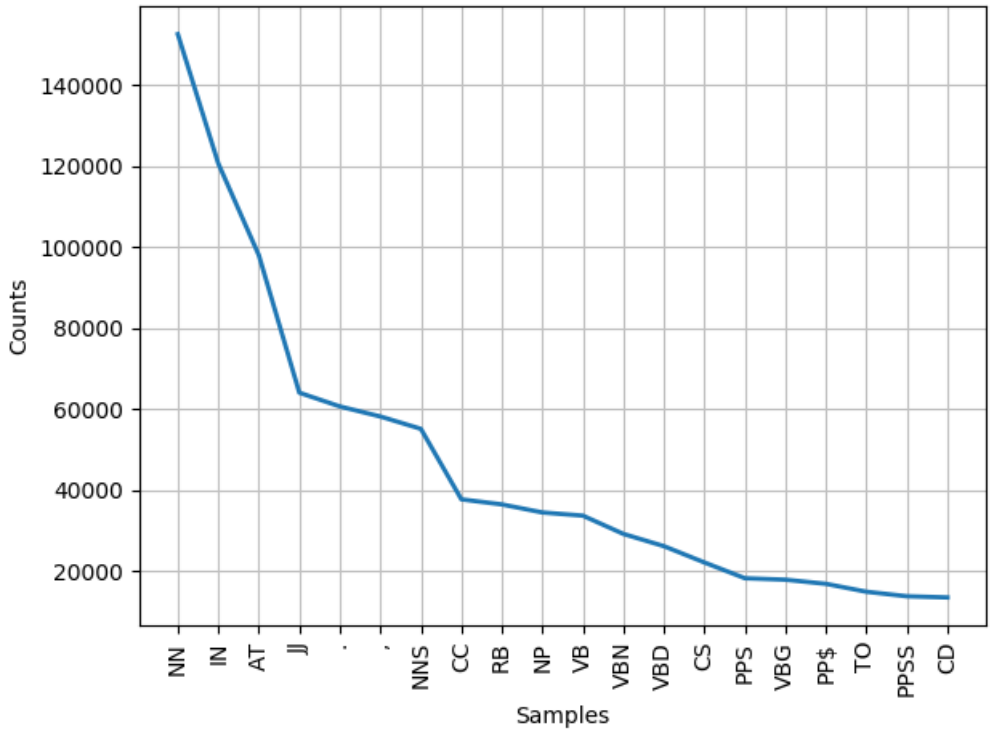
可见最多词性的是`that`。

c. 按频率递减的顺序列出标记。前20 个最频繁的词性标记代表什么?

```
from nltk.corpus import brown
from nltk import FreqDist

words = brown.tagged_words()
w = FreqDist(tag for (word,tag) in words)
w.plot(20)
```

根据`FreqDist`画出标记词性（`tagset=brown`），其中前20个`tag`为：



前20个最频繁的词性标记代表，句子成分中最常用的词性。比如`NN`是第一个，显然名词是我们语句中最常用的词性。

d. 名词后面最常见的是哪些词性标记？这些标记代表什么？

名词后面最常见的标记有`IN AT JJ . , NNS CC RB NP`等。

`IN`介系词，`AT`冠词，`JJ`形容词.....

2.使用`nltk`提供的任意分类器，以及你能想到的特征，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：400 个词为测试集，400个词为开发集，剩余的个词为训练集。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？

```
from nltk.corpus import names
import nltk
import random

name_male = [(name,'male') for name in names.words('male.txt')]
name_female = [(name,'female') for name in names.words('female.txt')]
samples = name_female + name_male
random.shuffle(samples)

def gender_feature(word):
    return {'last_letter':word[-1]}
```

```
def gender_feature2(word):
    return {'first_letter': word[0]}
def gender_feature3(word):
    return {'second_letter': word[1]}
def gender_feature4(word):
    return {'last_2letter': word[-2:]}
def gender_feature5(word):
    return {'last_2letter': word[-2:],'last_letter':word[-1]}

def set_data(data, function):
    return [(function(name),gender) for (name,gender) in data]

def test(feature):
    test_data = samples[:400]
    devtest_data = samples[400:800]
    train_data = samples[800:]
    test_set = set_data(test_data,feature)
    devtest_set = set_data(devtest_data,feature)
    train_set = set_data(train_data,feature)
    classifier = nltk.NaiveBayesClassifier.train(train_set)
    print("Using feature",feature)
    print(nltk.classify.accuracy(classifier, devtest_set))
    print(nltk.classify.accuracy(classifier, test_set))

test(gender_feature)
test(gender_feature2)
test(gender_feature3)
test(gender_feature4)
test(gender_feature5)
```

首先选取名词的特征，我分了四个特征，都是首尾字母的特征，企图寻找最佳的表示特征。

其次是分类器选择了朴素贝叶斯模型，因为这个名字的分类，基本可以根据贝叶斯的统计得出特征对应的分类。并且这是二分类问题也比较简单。将样本分为三个集合，测试集，训练集，开发集，输出每个特征对应的训练结果如下。

```
Using feature <function gender_feature at 0x037C3198>
0.78
0.7925
Using feature <function gender_feature2 at 0x0F81E930>
0.625
0.6575
Using feature <function gender_feature3 at 0x0F81E978>
0.625
0.6225
Using feature <function gender_feature4 at 0x0F81E9C0>
0.8025
0.79
Using feature <function gender_feature5 at 0x0F81EA08>
0.7875
0.7975
```

可以知道末尾字母的特征比较明显，所以可以使用末尾字母。但是当把末尾两个字母和一个字母结合起来，并没有实质性的改变，所以最终选择**末尾一个字母**作为名字特征即可。

最终测试集准确率效果达到`0.7925`，效果较好。