CSCI-8450

2018 Spring

Yuchen Wang

<http://www.nltk.org/book/ch05.html>

In all questions note how “The” and “the”, for instance, amount to the same word.

1. Train a unigram tagger on all of the sentences from the Brown corpus with the category *news* just as described in the “Unigram” Subsection of Section 5.1 of the textbook.
   1. Evaluate your tagger using “evaluate” function on all of the sentences from the Brown corpus with the category *lore.*
   2. How does this number compare to when this tagger is evaluate on all of the sentences from the Brown corpus with the category news.
   3. Provide the output of your tagger on the 200th sentence of the *lore* category of the Brown Corpus, (note how brown.sents(categories='lore')[199] produces the 200th sentence).

Code:

brown\_tagged\_sents = brown.tagged\_sents(categories=**'news'**)  
brown\_sents = brown.sents(categories=**'news'**)  
unigram\_tagger\_news = nltk.UnigramTagger(brown\_tagged\_sents)  
news\_evaluate = unigram\_tagger\_news.evaluate(brown\_tagged\_sents)  
print(**'news\_evaluate: '**, news\_evaluate)  
  
  
brown\_tagged\_sents = brown.tagged\_sents(categories=**'lore'**)  
lore\_evaluate = unigram\_tagger\_news.evaluate(brown\_tagged\_sents)  
print(**'lore\_evaluate: '**, lore\_evaluate)  
  
  
  
print(**'we can see that lore\_evalute is '**, lore\_evaluate, **'news\_evaluate is '**, news\_evaluate,**'lore is a lower than news'**)  
  
lore\_tagger = unigram\_tagger\_news.tag(brown\_sents[199])  
print(**'lore\_tagger\_200th: '**, lore\_tagger)

Result:

Exercise 1

news\_evaluate: 0.9349006503968017

lore\_evaluate: 0.7978585481282695

we can see that lore\_evalute is 0.7978585481282695 news\_evaluate is 0.9349006503968017. lore is a lower than news

lore\_tagger\_200th: [('Bellows', 'NP'), ('made', 'VBN'), ('the', 'AT'), ('request', 'NN'), ('while', 'CS'), ('the', 'AT'), ('all-woman', 'JJ'), ('jury', 'NN'), ('was', 'BEDZ'), ('out', 'RP'), ('of', 'IN'), ('the', 'AT'), ('courtroom', 'NN'), ('.', '.')]

1. Write code to search the Brown Corpus for particular words and phrases according to tags, to answer the questions a-d. Report your findings separately for the following categories of Brown corpus: *humor, romance, government*.
   * 1. Produce an alphabetically sorted list of the distinct words tagged as JJ. Report the number of distinct words tagged as JJ and the first five words in the sorted list.
     2. Identify words that can be plural nouns or third person singular verbs (e.g. *deals*, *flies*). Sort these words alphabetically. Report the first 10 elements of the sorted list.
     3. Identify three-word prepositional phrases of the form IN + AT + NN (eg. at *the house*). Report the 3 most frequent three-word prepositional phrases (break ties alphabetically).
     4. What is the ratio of masculine pronouns

*himself, his, him , he, he's, he'd, he'll*

to feminine pronouns

*herself, her, hers, she, she's, she'd, she'll,*

where both are used with tag PP?

What are the differences that you observe with respect to different categories of Brown corpus.

a)

Code:

print(**"Part 1: humor"**)  
tagged\_text = brown.tagged\_words(categories=**'humor'**)  
cfd = nltk.ConditionalFreqDist(tagged\_text)  
conditions = cfd.conditions()  
  
JJ\_words = [condition **for** condition **in** conditions **if** cfd[condition][**'JJ'**] != 0]  
JJ\_words.sort()  
print(**'JJ\_words\_len: '**, len(JJ\_words))  
print(**'JJ\_words\_humor: '**, JJ\_words[:5])  
  
print(**"Part 1: romance"**)  
tagged\_text = brown.tagged\_words(categories=**'romance'**)  
cfd = nltk.ConditionalFreqDist(tagged\_text)  
conditions = cfd.conditions()  
  
JJ\_words = [condition **for** condition **in** conditions **if** cfd[condition][**'JJ'**] != 0]  
JJ\_words.sort()  
print(**'JJ\_words\_len: '**, len(JJ\_words))  
print(**'JJ\_words\_romance: '**, JJ\_words[:5])  
  
print(**"Part 1: government"**)  
tagged\_text = brown.tagged\_words(categories=**'government'**)  
cfd = nltk.ConditionalFreqDist(tagged\_text)  
conditions = cfd.conditions()  
  
JJ\_words = [condition **for** condition **in** conditions **if** cfd[condition][**'JJ'**] != 0]  
JJ\_words.sort()  
print(**'JJ\_words\_len: '**, len(JJ\_words))  
print(**'JJ\_words\_government: '**, JJ\_words[:5])

Result:

Part 1: humor

JJ\_words\_len: 661

JJ\_words\_humor: ['African', 'American', 'Anglo-American', 'Bogartian', 'Bolshevistic']

Part 1: romance

JJ\_words\_len: 1139

JJ\_words\_romance: ['African', 'American', 'Asian', 'Big', 'Bizarre']

Part 1: government

JJ\_words\_len: 1066

JJ\_words\_government: ['10-gallon', '10-year', '150-milliampere', '25-liter', '4-cell']

In this part, it is demonstrated that humor\_JJ\_words\_len is smaller than romance and government, while romance is similar as government. The findings of humor and romance are words, while government is the style of number-word

b)

Code:

print(**"Part 2: humor"**)  
tagged\_text = brown.tagged\_words(categories=**'humor'**)  
cfd = nltk.ConditionalFreqDist(tagged\_text)  
conditions = cfd.conditions()  
two\_words = [condition **for** condition **in** conditions **if** cfd[condition][**'NNS'**] **and** cfd[condition][**'VBZ'**]]  
two\_words.sort()  
print(**'two\_words\_len: '**, len(two\_words))  
print(**'two\_words\_humor:'**, two\_words[:10])  
  
print(**"Part 2: romance"**)  
  
tagged\_text = brown.tagged\_words(categories=**'romance'**)  
cfd = nltk.ConditionalFreqDist(tagged\_text)  
conditions = cfd.conditions()  
two\_words = [condition **for** condition **in** conditions **if** cfd[condition][**'NNS'**] **and** cfd[condition][**'VBZ'**]]  
two\_words.sort()  
print(**'two\_words\_len: '**, len(two\_words))  
print(**'two\_words\_romance:'**, two\_words[:10])  
  
print(**"Part 2: government"**)  
  
tagged\_text = brown.tagged\_words(categories=**'government'**)  
cfd = nltk.ConditionalFreqDist(tagged\_text)  
conditions = cfd.conditions()  
two\_words = [condition **for** condition **in** conditions **if** cfd[condition][**'NNS'**] **and** cfd[condition][**'VBZ'**]]  
two\_words.sort()  
print(**'two\_words\_len: '**, len(two\_words))  
print(**'two\_words\_government:'**, two\_words[:10])

Result:

Part 2: humor

two\_words\_len: 5

two\_words\_humor: ['contracts', 'dogs', 'lives', 'means', 'remains']

Part 2: romance

two\_words\_len: 16

two\_words\_romance: ['checks', 'concerns', 'cracks', 'falls', 'leaves', 'lies', 'lives', 'looks', 'plays', 'rings']

Part 2: government

two\_words\_len: 36

two\_words\_government: ['aims', 'amounts', 'answers', 'bases', 'calls', 'changes', 'claims', 'concentrates', 'concerns', 'contracts']

In this question, two words len of humor is the smallest, then romance, and last is government. For two words of humor, most of them are verb-s, one is noun-s. For romance, all of them are verb-s. For government, it is similar as humor; most of them are verb-s.

c)

code:

print(**"Part 3: humor"**)  
  
three\_words = []  
tagged\_text = brown.tagged\_words(categories=**'humor'**)  
trigrams = list(nltk.trigrams(tagged\_text))  
**for** (w1, t1),(w2,t2), (w3,t3) **in** trigrams:  
 **if** t1 == **'IN' and** t2 == **'AT' and** t3 == **'NN'**:  
 three\_words.append((w1, w2, w3))  
  
  
print( nltk.FreqDist(three\_words).most\_common(3))  
  
print(**"Part 3: romance"**)  
  
three\_words = []  
tagged\_text = brown.tagged\_words(categories=**'romance'**)  
trigrams = list(nltk.trigrams(tagged\_text))  
**for** (w1, t1),(w2,t2), (w3,t3) **in** trigrams:  
 **if** t1 == **'IN' and** t2 == **'AT' and** t3 == **'NN'**:  
 three\_words.append((w1, w2, w3))  
  
  
print( nltk.FreqDist(three\_words).most\_common(3))  
  
print(**"Part 3: government"**)  
  
three\_words = []  
tagged\_text = brown.tagged\_words(categories=**'government'**)  
trigrams = list(nltk.trigrams(tagged\_text))  
**for** (w1, t1),(w2,t2), (w3,t3) **in** trigrams:  
 **if** t1 == **'IN' and** t2 == **'AT' and** t3 == **'NN'**:  
 three\_words.append((w1, w2, w3))  
  
  
print( nltk.FreqDist(three\_words).most\_common(3))

Result:

Part 3: humor

[(('in', 'the', 'world'), 4), (('at', 'the', 'studio'), 3), (('to', 'a', 'stone'), 3)]

Part 3: romance

[(('in', 'the', 'morning'), 9), (('in', 'the', 'world'), 8), (('in', 'the', 'kitchen'), 7)]

Part 3: government

[(('of', 'the', 'state'), 17), (('in', 'the', 'year'), 12), (('on', 'the', 'basis'), 9)]

In this question, it is discovered that when category is government, we can get the largest frequency in ”of the state”, which 17 times. Romance,” in the morning”, 9 times, which is smaller than government. Humor, “in the world”, 4 times, which is the smallest in this three categories.

d)

**def** total\_count(cate):  
 pp\_list = []  
 humor\_tagged\_words = brown.tagged\_words(categories = cate)  
  
 **for** i **in** humor\_tagged\_words:  
 **if** i[1].startswith(**"PP"**) **and not** i[1] == **'PP$'**:  
 pp\_list.append(i[0])  
  
 male\_list = [**'himself'**, **'his'**, **'him'**, **'he'**, **"he's"**, **"he'd"**, **"he'll"**]  
 female\_list = [**'herself'**, **'her'**, **'hers'**, **'she'**, **"she's"**, **"she'd"**, **"she'll"**]  
  
 male\_count = 0  
 female\_count = 0  
  
 **for** i **in** pp\_list:  
 **if** i.lower() **in** male\_list:  
 male\_count = male\_count + 1  
 **if** i.lower() **in** female\_list:  
 female\_count = female\_count + 1  
  
 total\_count\_humor = male\_count / female\_count  
  
 print(**"the\_ratio\_of\_mas\_to\_fem: "**, cate,**" "**, total\_count\_humor)

print(**"Part 4: "**)  
total\_count(**'humor'**)  
total\_count(**'romance'**)  
total\_count(**'government'**)

Result:

Part 4:

the\_ratio\_of\_mas\_to\_fem: humor 2.495145631067961

the\_ratio\_of\_mas\_to\_fem: romance 1.3540925266903914

the\_ratio\_of\_mas\_to\_fem: government 201.0

Government is highest, romance is lowest, and humor is in the middle. If we check with the male\_count of government, we can see that it is 201, and female count is exactly 1, which is the reason of highest of these three categories.

1. Utilizing the complete Brown corpus report the following:
2. Analyze the tagged words to determine the number of distinct words that have exactly 5 possible tags. For example: The word *read* has 4 possible tags (NP, VB, VBN, VBD) found in the Brown corpus, whereas *debt* only has one possible tag (NN).

Code:

print(**"Part 1"**)  
count = 0  
brown\_news\_tagged = brown.tagged\_words()  
data = nltk.ConditionalFreqDist((word.lower(), tag)  
 **for** (word, tag) **in** brown\_news\_tagged)  
**for** word **in** sorted(data.conditions()):  
 **if** len(data[word]) == 5:  
 tags = [tag **for** (tag, \_) **in** data[word].most\_common()]  
 print(word, **' '**.join(tags))  
 count = count + 1  
print(**"count:"**, count)

Result:

count: 176

1. Determine which word(s) has the most distinct tags. Chose one of these word(s) and find sentences demonstrating the use of at least 5 distinct tags from the possible tags for the selected word.

Code:

print(**"Part 2"**)  
  
  
  
brown\_news\_tagged = brown.tagged\_words()  
top\_word = {}  
**for** i **in** brown\_news\_tagged:  
 **if** i[0].lower() **not in** top\_word:  
 top\_word[i[0].lower()] = [i[1]]  
 **else**:  
 tmp = top\_word[i[0].lower()]  
 tmp.append(i[1])  
 top\_word[i[0].lower()] = list(set(tmp))  
print(max((len(v), k) **for** k, v **in** top\_word.items()))  
  
tagged\_sents = []  
tmp = []  
**for** i **in** brown\_news\_tagged:  
 **if** i[1] == **'.'**:  
 tmp.append(i)  
 tagged\_sents.append(tmp)  
 tmp=[]  
 tmp.append(i)  
five\_tags = [**'CS'**,**'DT'**,**'WPS'**,**'QL'**,**'NIL'**]  
**while** (len(five\_tags) > 0):  
 **for** s **in** tagged\_sents:  
 **for** (w, t) **in** s:  
 **if** w == **'that' and** t **in** five\_tags:  
 print(s)  
 five\_tags.remove(t)

Result:

(15, 'that')

[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-TL'), ('Jury', 'NN-TL'), ('said', 'VBD'), ('Friday', 'NR'), ('an', 'AT'), ('investigation', 'NN'), ('of', 'IN'), ("Atlanta's", 'NP$'), ('recent', 'JJ'), ('primary', 'NN'), ('election', 'NN'), ('produced', 'VBD'), ('``', '``'), ('no', 'AT'), ('evidence', 'NN'), ("''", "''"), ('that', 'CS'), ('any', 'DTI'), ('irregularities', 'NNS'), ('took', 'VBD'), ('place', 'NN'), ('.', '.')]

[('.', '.'), ('Regarding', 'IN'), ("Atlanta's", 'NP$'), ('new', 'JJ'), ('multi-million-dollar', 'JJ'), ('airport', 'NN'), (',', ','), ('the', 'AT'), ('jury', 'NN'), ('recommended', 'VBD'), ('``', '``'), ('that', 'CS'), ('when', 'WRB'), ('the', 'AT'), ('new', 'JJ'), ('management', 'NN'), ('takes', 'VBZ'), ('charge', 'NN'), ('Jan.', 'NP'), ('1', 'CD'), ('the', 'AT'), ('airport', 'NN'), ('be', 'BE'), ('operated', 'VBN'), ('in', 'IN'), ('a', 'AT'), ('manner', 'NN'), ('that', 'WPS'), ('will', 'MD'), ('eliminate', 'VB'), ('political', 'JJ'), ('influences', 'NNS'), ("''", "''"), ('.', '.')]

[('.', '.'), ('``', '``'), ('Actually', 'RB'), (',', ','), ('the', 'AT'), ('abuse', 'NN'), ('of', 'IN'), ('the', 'AT'), ('process', 'NN'), ('may', 'MD'), ('have', 'HV'), ('constituted', 'VBN'), ('a', 'AT'), ('contempt', 'NN'), ('of', 'IN'), ('the', 'AT'), ('Criminal', 'JJ-TL'), ('court', 'NN-TL'), ('of', 'IN-TL'), ('Cook', 'NP'), ('county', 'NN'), (',', ','), ('altho', 'CS'), ('vindication', 'NN'), ('of', 'IN'), ('the', 'AT'), ('authority', 'NN'), ('of', 'IN'), ('that', 'DT'), ('court', 'NN'), ('is', 'BEZ'), ('not', '\*'), ('the', 'AT'), ('function', 'NN'), ('of', 'IN'), ('this', 'DT'), ('court', 'NN'), ("''", "''"), (',', ','), ('said', 'VBD'), ('Karns', 'NP'), (',', ','), ('who', 'WPS'), ('is', 'BEZ'), ('a', 'AT'), ('City', 'NN-TL'), ('judge', 'NN'), ('in', 'IN'), ('East', 'JJ-TL'), ('St.', 'NP-TL'), ('Louis', 'NP-TL'), ('sitting', 'VBG'), ('in', 'IN'), ('Cook', 'NP-TL'), ('County', 'NN-TL'), ('court', 'NN-TL'), ('.', '.')]

[('.', '.'), ('While', 'CS'), ('the', 'AT'), ('city', 'NN'), ('council', 'NN'), ('suggested', 'VBD'), ('that', 'CS'), ('the', 'AT'), ('Legislative', 'JJ-TL'), ('Council', 'NN-TL'), ('might', 'MD'), ('perform', 'VB'), ('the', 'AT'), ('review', 'NN'), (',', ','), ('Mr.', 'NP'), ('Notte', 'NP'), ('said', 'VBD'), ('that', 'QL'), ('instead', 'RB'), ('he', 'PPS'), ('will', 'MD'), ('take', 'VB'), ('up', 'RP'), ('the', 'AT'), ('matter', 'NN'), ('with', 'IN'), ('Atty.', 'NN-TL'), ('Gen.', 'JJ-TL'), ('J.', 'NP'), ('Joseph', 'NP'), ('Nugent', 'NP'), ('to', 'TO'), ('get', 'VB'), ('``', '``'), ('the', 'AT'), ('benefit', 'NN'), ('of', 'IN'), ('his', 'PP$'), ('views', 'VBZ'), ("''", "''"), ('.', '.')]

[('.', '.'), ('Thus', 'NIL'), (',', ','), ('as', 'NIL'), ('a', 'NIL'), ('development', 'NIL'), ('program', 'NIL'), ('is', 'NIL'), ('being', 'NIL'), ('launched', 'NIL'), (',', ','), ('commitments', 'NIL'), ('and', 'NIL'), ('obligations', 'NIL'), ('must', 'NIL'), ('be', 'NIL'), ('entered', 'NIL'), ('into', 'NIL'), ('in', 'NIL'), ('a', 'NIL'), ('given', 'NIL'), ('year', 'NIL'), ('which', 'NIL'), ('may', 'NIL'), ('exceed', 'NIL'), ('by', 'NIL'), ('twofold', 'NIL'), ('or', 'NIL'), ('threefold', 'NIL'), ('the', 'NIL'), ('expenditures', 'NIL'), ('to', 'NIL'), ('be', 'NIL'), ('made', 'NIL'), ('in', 'NIL'), ('that', 'NIL'), ('year', 'NIL'), ('.', '.')]