Assignment 3: Parts-of-Speech Tagging

Name: Karan Patel

Categorizing and Tagging Words: http://www.nltk.org/book/ch05.html

In [1]:

```python
import nltk
from nltk.corpus import brown
from IPython.display import HTML, display
from tabulate import tabulate

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('brown')
nltk.download('universal_tagset')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\r631915\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\r631915\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\r631915\AppData\Roaming\nltk_data...
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package universal_tagset to
[nltk_data]     C:\Users\r631915\AppData\Roaming\nltk_data...
[nltk_data]   Package universal_tagset is already up-to-date!
```

Out[1]:

```
True
```

Q1. Search the web for 2 “spoof newspaper headlines”, to find such gems as: *British Left Waffles on Falkland Islands*, and *Juvenile Court to Try Shooting Defendant*. Manually tag these headlines to see if knowledge of the part-of-speech tags removes the ambiguity.

Answer:

Following are the two spoof newspaper headlines that I found from the web and my corresponding manual pos tagging:

1. Cows lose their jobs as milk prices drop

> Cows (noun) lose (verb) their (pronoun) jobs (noun) as(preposition) milk (pronoun) prices (noun) drop (verb).

1. Trump's Lawyers: Telling Armed Crazies to "Go to Capitol" and "Fight Like Hell" Was Just Metaphorical

> Trump's (noum) Lawyers (noun): Telling (adverb) Armed (verb) Crazies (noun) to (preposition) "Go (verb) to (preposition) Capitol (noun)" and (preposition) "Fight (verb) Like (adjective) Hell (noun)" Was (noun) Just (preposition) Metaphorical (noun)

Below, the headlines are tagged using `nltk`. It definitely helps remove a lot of ambiguity that I had when manually tagging the headlines.

```
In [2]:
tokens = nltk.word_tokenize("Cows lose their jobs as milk prices drop")
print("POS tags for headline #1:\n{}\n".format(nltk.pos_tag(tokens, tagset='universal')

tokens = nltk.word_tokenize('Trump\'s Lawyers: Telling Armed Crazies to "Go to Capitol"
print("POS tags for headline #2:\n{}".format(nltk.pos_tag(tokens, tagset='universal')))
```

```
POS tags for headline #1:
[('Cows', 'NOUN'), ('lose', 'VERB'), ('their', 'PRON'), ('jobs', 'NOUN'), ('as', 'ADP'),
('milk', 'NOUN'), ('prices', 'NOUN'), ('drop', 'NOUN')]

POS tags for headline #2:
[('Trump', 'NOUN'), ("'s", 'PRT'), ('Lawyers', 'NOUN'), (':', '.'), ('Telling', 'NOUN'),
('Armed', 'NOUN'), ('Crazies', 'NOUN'), ('to', 'PRT'), ('``', '.'), ('Go', 'VERB'), ('t
o', 'PRT'), ('Capitol', 'NOUN'), ("''", '.'), ('and', 'CONJ'), ('``', '.'), ('Fight', 'N
OUN'), ('Like', 'ADP'), ('Hell', 'NOUN'), ("''", '.'), ('Was', 'NOUN'), ('Just', 'NOU
N'), ('Metaphorical', 'NOUN')]
```

Q2. Tokenize and tag the following sentence: They wind back the clock, while we chase after the wind. What is the output?

Answer: The output is a list of tuples of words in sentence and their corresponding part of speech (pos) tag.

```
In [3]:
tokens = nltk.word_tokenize('They wind back the clock, while we chase after the wind.')
print("POS tags for sentence:\n{}".format(nltk.pos_tag(tokens)))
```

```
POS tags for sentence:
[('They', 'PRP'), ('wind', 'VBP'), ('back', 'RB'), ('the', 'DT'), ('clock', 'NN'), (',',
','), ('while', 'IN'), ('we', 'PRP'), ('chase', 'VBP'), ('after', 'IN'), ('the', 'DT'),
('wind', 'NN'), ('.', '.')]
```

Q3. Pick 2 words that can be either a noun or a verb (e.g., contest). Predict which POS tag is likely to be the most frequent in the Brown corpus, and compare with your predictions.

Answer: Following are the 2 words (that can be either a noun or a verb) that I picked:

1. increase
   - My prediction of most likely POS tag between Noun or Verb: Verb
   - Actual: Noun
2. attack
   - My prediction of most likely POS tag between Noun or Verb: Verb
   - Actual: Noun

```
In [4]:
word_1 = 'increase'
tag_fd = nltk.FreqDist(tag for (word, tag) in brown.tagged_words(tagset='universal') if
print('Counts of when "{}" is noun = {}'.format(word_1, tag_fd['NOUN']))
print('Counts of when "{}" is verb = {}'.format(word_1, tag_fd['VERB']))

word_2 = 'attack'
tag_fd = nltk.FreqDist(tag for (word, tag) in brown.tagged_words(tagset='universal') if
print('\nCounts of when "{}" is noun = {}'.format(word_2, tag_fd['NOUN']))
print('Counts of when "{}" is verb = {}'.format(word_2, tag_fd['VERB']))
```

```
Counts of when "increase" is noun = 112
Counts of when "increase" is verb = 82

Counts of when "attack" is noun = 78
Counts of when "attack" is verb = 24
```

Q4. Use sorted() and set() to get a sorted list of tags used in the Brown corpus, removing duplicates.

In [5]:
```
sorted(set(tag for (word, tag) in brown.tagged_words(tagset='universal')))
```

Out[5]:
```
['.',
 'ADJ',
 'ADP',
 'ADV',
 'CONJ',
 'DET',
 'NOUN',
 'NUM',
 'PRON',
 'PRT',
 'VERB',
 'X']
```

Q5. Write programs to process the Brown Corpus and find answers to the following questions:

1. Which nouns are more common in their plural form, rather than their singular form? (Only consider regular plurals, formed with the -s suffix.)

Answer: My program found a total of 822 nouns that are more common in their plural form, rather than their singular form. Some of the examples are shown by the output of the code. For example, noun "painting" (with a singlualr count of 27) is more common in its plural form "paintings" (with a plural count of 34).

In [6]:
```
# Q5.1

# Note: foreign word noun tags are excluded from sets below since plural foreign words 
singular_noun_tags = {'NN', 'NN$', 'NN+BEZ', 'NN+HVD', 'NN+HVZ', 'NN+IN', 'NN+MD', 'NN+
plural_noun_tags = {'NNS', 'NNS$', 'NNS+MD', 'NPS', 'NPS$', 'NRS'}

singular_noun_freq_dist = nltk.FreqDist(word for (word, tag) in brown.tagged_words() if
all_singular_nouns = set(singular_noun_freq_dist.keys())
print(f'Found a total of {len(all_singular_nouns)} singular nouns in the corpus.\n')

all_regular_plural_nouns = {singular_noun + 's' for singular_noun in all_singular_nouns
plural_noun_freq_dist = nltk.FreqDist(word for (word, tag) in brown.tagged_words() if t

total_count = 0
for singluar_noun in all_singular_nouns:
    plural_noun = singluar_noun + 's'

    singular_noun_count = singular_noun_freq_dist[singluar_noun]
    plural_noun_count = plural_noun_freq_dist[plural_noun]

    if plural_noun_count > singular_noun_count:
        if total_count <= 10:
            print(f'(Singluar) noun "{singluar_noun}" is more common in its plural form
        total_count += 1

print(f'\nFound a total of {total_count} nouns that are more common in their plural for
Found a total of 23187 singular nouns in the corpus.

(Singluar) noun "painting" is more common in its plural form "paintings". Singular form
count = 27, plural form count = 34
(Singluar) noun "pledge" is more common in its plural form "pledges". Singular form coun
t = 2, plural form count = 3
(Singluar) noun "bound" is more common in its plural form "bounds". Singular form count
= 3, plural form count = 10
(Singluar) noun "customer" is more common in its plural form "customers". Singular form
count = 23, plural form count = 37
(Singluar) noun "cart" is more common in its plural form "carts". Singular form count =
4, plural form count = 5
(Singluar) noun "requirement" is more common in its plural form "requirements". Singular
form count = 27, plural form count = 78
(Singluar) noun "alloy" is more common in its plural form "alloys". Singular form count
= 2, plural form count = 3
(Singluar) noun "loyalist" is more common in its plural form "loyalists". Singular form
count = 1, plural form count = 2
(Singluar) noun "2-year-old" is more common in its plural form "2-year-olds". Singular f
orm count = 2, plural form count = 3
(Singluar) noun "cuff" is more common in its plural form "cuffs". Singular form count =
1, plural form count = 2
(Singluar) noun "striving" is more common in its plural form "strivings". Singular form
count = 1, plural form count = 3

Found a total of 822 nouns that are more common in their plural form, rather than their
singular form. Only showing a few above.
```

Q5.2. List tags in order of decreasing frequency. What do the 20 most frequent tags represent?

Answer: The 20 most frequent tags represent top 20 part of speech tags that appeared the most in the brown corpus. My code below prints out the counts associated with each pos tag. Words with those pos tags are the most frequent in the corpus.

Looking at the 20 most frequent tags, it looks like singluar nouns and prepositions are the most common POS tags. Period (.) and comma (,) also are relatively very frequent.

In [7]:
```
# Q5.2

tags_frequency_dist = nltk.FreqDist(tag for (word, tag) in brown.tagged_words())
print(tags_frequency_dist.most_common(20))
```

```
[('NN', 152470), ('IN', 120557), ('AT', 97959), ('JJ', 64028), ('.', 60638), (',', 5815
6), ('NNS', 55110), ('CC', 37718), ('RB', 36464), ('NP', 34476), ('VB', 33693), ('VBN',
29186), ('VBD', 26167), ('CS', 22143), ('PPS', 18253), ('VBG', 17893), ('PP$', 16872),
('TO', 14918), ('PPSS', 13802), ('CD', 13510)]
```

Q6. Generate some statistics for tagged data to answer the following questions:

1. What proportion of word types are always assigned the same part-of-speech tag?
   Answer: 84.42834971546819 % of the words are always assigned the same part-of-speech tag.
2. How many words are ambiguous, in the sense that they appear with at least two tags?
   Answer: 8729 words are ambiguous.
3. What percentage of word tokens in the Brown Corpus involve these ambiguous words?
   Answer: % of word tokens that involve ambiguous words = 78.64892283102192 %

In [8]:
```
num_word_tokens = len(brown.words())
```

```
print(f'Number of work tokens = {num_word_tokens}')

all_word_types = {word for word in brown.words()}
print(f'Found a total of {len(all_word_types)} word types / unique words.')

all_pos_tags = {tag for (word, tag) in brown.tagged_words()}
print(f'Found a total of {len(all_pos_tags)} POS tags.\n')

word_types_with_unique_pos_tag_assignment_count = 0
num_ambiguous_words = 0
ambiguous_word_tokens_count = 0

word_type_to_pos_tags_dict = {word_type:set() for word_type in all_word_types}
for word_token, tag in brown.tagged_words():
    word_type_to_pos_tags_dict[word_token].add(tag)

for word_type, pos_tags_set in word_type_to_pos_tags_dict.items():
    if len(pos_tags_set) == 1:
        word_types_with_unique_pos_tag_assignment_count += 1
    elif len(pos_tags_set) > 1:
        num_ambiguous_words += 1

for word_token in brown.words():
    if len(word_type_to_pos_tags_dict[word_token]) > 1:
        ambiguous_word_tokens_count += 1

print(f'% of word types that are always assigned the same part-of-speech tag = {(word_t
print(f'Number of words that are ambiguous (i.e. they appear with at least two tags) =
print(f'% of word tokens that involve ambiguous words = {(ambiguous_word_tokens_count /
```

```
Number of work tokens = 1161192
Found a total of 56057 word types / unique words.
Found a total of 472 POS tags.

% of word types that are always assigned the same part-of-speech tag = 84.42834971546819
%
Number of words that are ambiguous (i.e. they appear with at least two tags) = 8729
% of word tokens that involve ambiguous words = 78.64892283102192 %
```

Q9. There are 264 distinct words in the Brown Corpus having exactly three possible tags.

1. Print a table with the integers 1..10 in one column, and the number of distinct words in the corpus having 1..10 distinct tags in the other column.
2. For the word with the greatest number of distinct tags, print out sentences from the corpus containing the word, one for each possible tag.

In [9]:

```
all_word_types = {word for word in brown.words()}
word_type_to_pos_tags_dict = {word_type:set() for word_type in all_word_types}
for word_token, tag in brown.tagged_words():
    word_type_to_pos_tags_dict[word_token].add(tag)

distinct_pos_tag_count_to_distinct_words_count = {i + 1:0 for i in range(10)}
word_to_distinct_pos_tags_count = (None, 0)  # word, distinct pos tags count
for word, pos_tags_set in word_type_to_pos_tags_dict.items():
    num_unique_tags = len(pos_tags_set)

    if num_unique_tags > 0 and num_unique_tags <= 10:
        distinct_pos_tag_count_to_distinct_words_count[num_unique_tags] = distinct_pos_
    if num_unique_tags > word_to_distinct_pos_tags_count[1]:
        word_to_distinct_pos_tags_count = word, num_unique_tags

display(HTML(tabulate([(pos_int, count) for (pos_int, count) in distinct_pos_tag_count_


print(f'\n"{word_to_distinct_pos_tags_count[0]}" is the word with the greatest number o

for tag in word_type_to_pos_tags_dict[word_to_distinct_pos_tags_count[0]]:
    for tagged_sentence in brown.tagged_sents():
        if (word_to_distinct_pos_tags_count[0], tag) in tagged_sentence:
            sentence = ' '.join([word for word, tag in tagged_sentence])
            print(f'\nFor {tag} tag, sentence = {sentence}')
            break
```

Num. of distinct tags	Num. of distinct words
1	47328
2	7186
3	1146
4	265
5	87
6	27
7	12
8	1
9	1
10	2

```
"that" is the word with the greatest number of distinct tags (12). Here are all distinct
pos tags found: {'WPO', 'WPO-NC', 'NIL', 'WPS', 'CS-HL', 'DT-NC', 'CS', 'QL', 'WPS-NC',
'CS-NC', 'DT', 'WPS-HL'}


For WPO tag, sentence = He was able to smell a bargain -- and a masterpiece -- a contine
nt away , and the Museum of Modern Art's Alfred Barr said of him : `` I have never menti
oned a new artist that Thompson didn't know about '' .

For WPO-NC tag, sentence = Thus to has light stress both in that was the conclusion that
I came to and in that was the conclusion I came to .

For NIL tag, sentence = Thus , as a development program is being launched , commitments
and obligations must be entered into in a given year which may exceed by twofold or thre
efold the expenditures to be made in that year .

For WPS tag, sentence = Regarding Atlanta's new multi-million-dollar airport , the jury
recommended `` that when the new management takes charge Jan. 1 the airport be operated
in a manner that will eliminate political influences '' .

For CS-HL tag, sentence = According to the official interpretation of the Charter , a me
mber cannot be penalized by not having the right to vote in the General Assembly for non
payment of financial obligations to the `` special '' United Nations' budgets , and of c
ourse cannot be expelled from the Organization ( which you suggested in your editorial )
, due to the fact that there is no provision in the Charter for expulsion .

For DT-NC tag, sentence = He has his own system of shorthand , devised by abbreviations
: `` humility '' will be `` humly '' , `` with '' will be `` w '' , and `` that '' will
```

```
be `` tt '' .

For CS tag, sentence = The Fulton County Grand Jury said Friday an investigation of Atla
nta's recent primary election produced `` no evidence '' that any irregularities took pl
ace .

For QL tag, sentence = While the city council suggested that the Legislative Council mig
ht perform the review , Mr. Notte said that instead he will take up the matter with Att
y. Gen. J. Joseph Nugent to get `` the benefit of his views '' .

For WPS-NC tag, sentence = In of all the suggestions that were made , his was the sillie
st the possessive his represents his suggestion and is stressed .

For CS-NC tag, sentence = But when to represents to consciousness in that was the moment
that I came to , and similarly in that was the moment I came to , there is much stronger
stress on to .

For DT tag, sentence = `` Actually , the abuse of the process may have constituted a con
tempt of the Criminal court of Cook county , altho vindication of the authority of that
court is not the function of this court '' , said Karns , who is a City judge in East S
t. Louis sitting in Cook County court .

For WPS-HL tag, sentence = Factors that inhibit learning and lead to maladjustment
```

Q8: How serious is the sparse data problem? Investigate the performance of n-gram taggers as n increases from 1 to 6. Tabulate the accuracy score.

Answer: Based on the accuray score, the sparse data problem seems to be pretty serious for when n is larger than 2. The accuracy scores drop off significantly for n-gram taggers after n is larger than 2.

The size of an n-gram table increases significantly with increasing `n` variable with the large table having lots of unfilled/empty array cells. Also, as n gets larger, the specificity of the contexts increases further impacting the accuracy.

In [10]:
```python
brown_tagged_sents = brown.tagged_sents()

size = int(len(brown_tagged_sents) * 0.9)

train_sents = brown_tagged_sents[:size]
test_sents = brown_tagged_sents[size:]

table = []
for i in range(6):
    n = i + 1
    n_gram_tagger = nltk.NgramTagger(n, train=train_sents)

    row = n, n_gram_tagger.evaluate(test_sents)
    table.append(row)

display(HTML(tabulate(table, headers=['N-Grams', 'Accuracy Score'], tablefmt='html')))
```

N-Grams	Accuracy Score
1	0.884935
2	0.351575
3	0.202971
4	0.152511

N-Grams	Accuracy Score
5	0.1402
6	0.138367