

Problem 19

The `evaluate()` method works out how accurately the tagger performs on this text. For example, if the supplied tagged text was `[('the', 'DT'), ('dog', 'NN')]` and the tagger produced the output `[('the', 'NN'), ('dog', 'NN')]`, then the score would be 0.5. Let's try to figure out how the evaluation method works:

1. A tagger `t` takes a list of words as input, and produces a list of tagged words as output. However, `t.evaluate()` is given correctly tagged text as its only parameter. What must it do with this input before performing the tagging?

Strip the tags from the gold standard text.

2. Once the tagger has created newly tagged text, how might the `evaluate()` method go about comparing it with the original tagged text and computing the accuracy score?

For each work in the tagged output, it would compare whether or not the tag of that word matches the tag of the same word from the gold standard input.

3. Now examine the source code to see how the method is implemented. Inspect `nltk.tag.api.__file__` to discover the location of the source code, and open this file using an editor.

Source Code for `evaluate()`:

```
def evaluate(self, gold):
    """
    Score the accuracy of the tagger against the gold
    standard.
    Strip the tags from the gold standard text, retag it
    using
    the tagger, then compute the accuracy score.

    :type gold: list(list(tuple(str, str)))
    :param gold: The list of tagged sentences to score
                  the tagger on.
    :rtype: float
```

```

"""

tagged_sents = self.tag_sents(untag(sent) for sent in
    gold)
gold_tokens = list(chain(*gold))
test_tokens = list(chain(*tagged_sents))
return accuracy(gold_tokens, test_tokens)

```

Problem 20

Write code to search the Brown Corpus for particular words and phrases according to tags, to answer the following questions:

1. Produce an alphabetically sorted list of the distinct words tagged as MD.

Code:

```

def sorted_tags():
    """
    Produce an alphabetically sorted list of the distinct
    words tagged as MD.
    """
    tagged_words = nltk.corpus.brown.tagged_words()
    words_of_interest = []
    for tag_pair in tagged_words:
        if tag_pair[1] == 'MD':
            words_of_interest.append(tag_pair[0].lower())
    distinct_words_of_interest =
        list(set(words_of_interest))
    distinct_sorted_words_of_interest =
        sorted(distinct_words_of_interest, key=str.lower)

    print("Distinct words tagged as MD: %s" %
        distinct_sorted_words_of_interest)
    print("Number of distinct_words_of_interest: %s" %
        len(distinct_sorted_words_of_interest))

```

Output:

```
$ python3 expanded_tagger.py
Distinct words tagged as MD: ["c'n", 'can', 'colde',
    'could', 'dare', 'kin', 'maht', 'mai', 'may',
    'maye', 'mayst', 'might', 'must', 'need', 'ought',
    'shall', 'should', 'shuld', 'shulde', 'wil', 'will',
    'wilt', 'wod', 'wold', 'wolde', 'would']
Number of distinct_words_of_interest: 26
```

2. Identify words that can be plural nouns or third person singular verbs (e.g. deals, flies).

Code:

```
def multi_tags():
    brown_tagged = brown.tagged_words()
    data = nltk.ConditionalFreqDist((word.lower(), tag)
        for (word, tag) in brown_tagged)
    words_of_interest = []
    for word in sorted(data.conditions()):
        if len(data[word]) > 2:
            tags = [tag for (tag, _) in
                data[word].most_common()]
            if ('VBZ' in tags and 'NNS' in tags):
                words_of_interest.append(word)
    print("Words categorized as both VBZ and NNS: ")
    print(words_of_interest)
```

Output:

```
$ python3 expanded_tagger.py
Words categorized as both VBZ and NNS:
['accounts', 'acts', 'addresses', 'aids', 'appeals',
    'associates', 'attacks', 'attempts', 'attributes',
    'backs', 'bangs', 'banks', 'bars', 'bellows',
    'benefits', 'boards', 'bridges', 'bugs', 'burns',
    'calls', 'cares', 'causes', 'centers', 'champions',
    'changes', 'charges', 'checks', 'claims', 'clouds',
```

```
'colors', 'comments', 'contacts', 'contracts',  
'controls', 'costs', 'courts', 'dances', 'designs',  
'dies', 'dishes', 'dogs', 'doubles', 'dreams',  
'drifts', 'drinks', 'exercises', 'exhibits',  
'faces', 'factors', 'falls', 'fashions', 'fears',  
'features', 'feeds', 'fields', 'figures', 'flies',  
'forces', 'functions', 'gains', 'guides', 'harbors',  
'helps', 'hits', 'holds', 'honors', 'hopes',  
'influences', 'issues', 'lands', 'levels', 'limits',  
'lines', 'lists', 'lives', 'marches', 'markets',  
'means', 'meets', 'misses', 'moves', 'needs',  
'notes', 'objects', 'offers', 'orders', 'pictures',  
'places', 'plans', 'powers', 'practices',  
'projects', 'purchases', 'questions', 'rates',  
'reasons', 'rebels', 'records', 'regrets',  
'remains', 'remarks', 'replies', 'reports',  
'results', 'returns', 'rises', 'rolls', 'rules',  
'sanctions', 'services', 'sets', 'shares', 'shows',  
'snows', 'speeds', 'sports', 'springs', 'stakes',  
'stands', 'states', 'stems', 'steps', 'stops',  
'studies', 'subjects', 'supplies', 'supports',  
'switches', 'tastes', 'tests', 'times', 'tops',  
'toys', 'transfers', 'travels', 'trusts', 'turns',  
'uses', 'values', 'views', 'visits', 'winds',  
'works']
```

3. Identify three-word prepositional phrases of the form IN + DET + NN (eg. in the lab).

Code:

```
def find_phrase():  
    for tagged_sent in brown.tagged_sents():  
        for (w1,t1), (w2,t2), (w3,t3) in  
            nltk.trigrams(tagged_sent):  
                if (t1=='IN' and t2 == 'DT' and t3=='NN'):  
                    print(w1, w2, w3)
```

Output:

```
$python find_phrase.py
[]at that time
over this couple
of this nation
of this collection
for that matter
of this idea
by this news
of this sort
of this letter
with this subject
of that picture
in that position
at this juncture
of this nut
on this side
in that outfit
In this manner
of this veteran
In this country
In this work
in that milieu
In that case
for this autumn
...]
```

4. What is the ratio of masculine to feminine pronouns?

Problem 26

4.1 plotted a curve showing change in the performance of a lookup tagger as the model size was increased. Plot the performance curve for a unigram tagger, as the amount of training data is varied.