assignment4

October 9, 2019

1 Assignment 4

John Flanigan

October 7, 2019

[1]:
```
import nltk
```

1.1 Exercise 5.18

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

a. What proportion of word types are always assigned the same part-of-speech tag?

[2]:
```
tag_dict = {}

for word, tag in nltk.corpus.brown.tagged_words(tagset='universal'):
    if word in tag_dict:
        tag_dict[word].add(tag)
    else:
        tag_dict[word] = set(tag)

same_part_of_speech_count = len([word for word, tags in tag_dict.items() if␣
 ↪len(tags) == 1])
same_part_of_speech_count / len(tag_dict)
```

[2]:
```
0.012451611752323528
```

b. How many words are ambiguous, in the sense that they appear with at least two tags?

[3]:
```
ambiguous_words = []
for word, tags in tag_dict.items():
    if len(tags) > 1:
        ambiguous_words.append(word)

len(ambiguous_words) / len(tag_dict)
```

[3]:
```
0.9875483882476764
```

c. What percentage of word tokens in the Brown Corpus involve these ambiguous words?

[4]:
```
count = 0
for word in nltk.corpus.brown.words():
    if word in ambiguous_words:
        count += 1
count / len(nltk.corpus.brown.words())
```

[4]:
```
0.9350675857222578
```

1.2 Exercise 5.19

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:

a. 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?

Answer: It must first retrieve the words from the correctly tagged text and then re-tag these words.

b. 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?

Answer: Evaluate would likely count the number of correct tags (true positives), the number of tags that were incorrect (false positives), and the number of tags that it missed (false negatives). It would then use these three metrics to calculate an accuracy score.

c. 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 (be sure to use the api.py file and not the compiled api.pyc binary file).

[5]:
```
nltk.tag.api.__file__
```

[5]:
```
'//anaconda3/lib/python3.7/site-packages/nltk/tag/api.py'
```

[6]:
```
with open('//anaconda3/lib/python3.7/site-packages/nltk/tag/api.py', 'r') as f:
    print(f.read())
```

```
# Natural Language Toolkit: Tagger Interface
#
# Copyright (C) 2001-2019 NLTK Project
# Author: Edward Loper <edloper@gmail.com>
#         Steven Bird <stevenbird1@gmail.com> (minor additions)
# URL: <http://nltk.org/>
# For license information, see LICENSE.TXT

"""
```

```
Interface for tagging each token in a sentence with supplementary
information, such as its part of speech.
"""
from abc import ABCMeta, abstractmethod
from itertools import chain

from six import add_metaclass

from nltk.internals import overridden
from nltk.metrics import accuracy
from nltk.tag.util import untag


@add_metaclass(ABCMeta)
class TaggerI(object):
    """
    A processing interface for assigning a tag to each token in a list.
    Tags are case sensitive strings that identify some property of each
    token, such as its part of speech or its sense.

    Some taggers require specific types for their tokens.  This is
    generally indicated by the use of a sub-interface to ``TaggerI``.
    For example, featureset taggers, which are subclassed from
    ``FeaturesetTagger``, require that each token be a ``featureset``.

    Subclasses must define:
      - either ``tag()`` or ``tag_sents()`` (or both)
    """

    @abstractmethod
    def tag(self, tokens):
        """
        Determine the most appropriate tag sequence for the given
        token sequence, and return a corresponding list of tagged
        tokens.  A tagged token is encoded as a tuple ``(token, tag)``.

        :rtype: list(tuple(str, str))
        """
        if overridden(self.tag_sents):
            return self.tag_sents([tokens])[0]

    def tag_sents(self, sentences):
        """
        Apply ``self.tag()`` to each element of *sentences*.  I.e.:

            return [self.tag(sent) for sent in sentences]
        """
        return [self.tag(sent) for sent in sentences]
```

```
    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)

    def _check_params(self, train, model):
        if (train and model) or (not train and not model):
            raise ValueError('Must specify either training data or trained
model.')


class FeaturesetTaggerI(TaggerI):
    """
    A tagger that requires tokens to be ``featuresets``.  A featureset
    is a dictionary that maps from feature names to feature
    values.  See ``nltk.classify`` for more information about features
    and featuresets.
    """
```

1.3 Exercise 5.20

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

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

[7]:
```
md_words = set([word.lower() for word, tag in nltk.corpus.brown.tagged_words()␣
 ↪if tag == 'MD'])
sorted(md_words)
```

[7]:
```
["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']
```

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

[8]:
```
plural_nouns = set([word.lower() for word, tag in nltk.corpus.brown.
 ↪tagged_words() if tag == 'NNS'])
third_person_singular_verbs = set([word.lower() for word, tag in nltk.corpus.
 ↪brown.tagged_words() if tag == 'VBZ'])

combined = [word for word in plural_nouns if word in␣
 ↪third_person_singular_verbs]
sorted(combined)
```

[8]:
```
['accounts',
 'acts',
 'addresses',
 'advances',
 'affects',
 'aids',
 'aims',
 'amounts',
 'answers',
 'appeals',
 'approaches',
 'arches',
 'assaults',
 'associates',
 'attacks',
 'attempts',
```

```
'attributes',
'backs',
'bangs',
'banks',
'bargains',
'bars',
'bases',
'bats',
'beats',
'bellows',
'belts',
'bends',
'benefits',
'bites',
'blankets',
'blots',
'blows',
'blueprints',
'boards',
'bodies',
'boils',
'borders',
'bores',
'bottles',
'bows',
'breaks',
'bridges',
'bristles',
'bubbles',
'bugs',
'bulletins',
'bullies',
'burns',
'butts',
'calls',
'caps',
'captures',
'cares',
'casts',
'catches',
'causes',
'censors',
'centers',
'challenges',
'champions',
'changes',
'charges',
```

```
'checks',
'claims',
'claps',
'clicks',
'clouds',
'clucks',
'clutches',
'colors',
'commands',
'comments',
'compounds',
'compresses',
'concentrates',
'concerns',
'conducts',
'conflicts',
'contacts',
'contracts',
'contrasts',
'controls',
'coordinates',
'costs',
'counts',
'courts',
'covers',
'cracks',
'credits',
'cries',
'crops',
'crosses',
'cuts',
'cycles',
'damages',
'dances',
'dashes',
'dates',
'deals',
'declines',
'decreases',
'decrees',
'deeds',
'delays',
'delights',
'demands',
'deserts',
'designs',
'desires',
```

```
'dictates',
'dies',
'dishes',
'dislikes',
'displays',
'dogs',
'doubles',
'drains',
'dreams',
'drifts',
'drinks',
'drives',
'drops',
'dwarfs',
'embraces',
'encounters',
'ends',
'equals',
'escapes',
'estimates',
'excuses',
'exercises',
'exhibits',
'experiences',
'extracts',
'faces',
'factors',
'falls',
'fans',
'fashions',
'favors',
'fears',
'features',
'feeds',
'fields',
'fights',
'figures',
'files',
'finishes',
'fishes',
'fits',
'flags',
'flares',
'flies',
'flourishes',
'flows',
'forces',
```

```
'forms',
'fractures',
'functions',
'gains',
'gestures',
'glories',
'graduates',
'guarantees',
'guides',
'handles',
'harbors',
'hates',
'hauls',
'haunts',
'heads',
'helps',
'hides',
'hinges',
'hints',
'hits',
'holds',
'honors',
'hopes',
'houses',
'howls',
'hunts',
'hurts',
'imports',
'increases',
'influences',
'initiates',
'interests',
'issues',
'jokes',
'jumps',
'keeps',
'kicks',
'kids',
'kills',
'kisses',
'knocks',
'knuckles',
'labels',
'labors',
'lags',
'lands',
'laps',
```

```
'lapses',
'laughs',
'lays',
'leads',
'leaps',
'leases',
'leaves',
'levels',
'lies',
'lifts',
'lights',
'likes',
'limits',
'lines',
'lists',
'lives',
'looks',
'looms',
'loves',
'makes',
'marches',
'markets',
'marks',
'matches',
'matters',
'means',
'measures',
'meets',
'mentions',
'merits',
'mirrors',
'misses',
'mistakes',
'moderates',
'mounts',
'moves',
'names',
'needs',
'notes',
'numbers',
'objects',
'offers',
'orders',
'outlines',
'paints',
'parades',
'parallels',
```

```
'passes',
'pauses',
'peers',
'permits',
'petitions',
'phones',
'photographs',
'picks',
'pictures',
'piles',
'places',
'plans',
'plays',
'plots',
'plunges',
'points',
'powers',
'practices',
'presents',
'preserves',
'presses',
'proceeds',
'projects',
'promises',
'protests',
'pulls',
'purchases',
'purges',
'pushes',
'quarrels',
'questions',
'rains',
'raises',
'rallies',
'ranches',
'ranges',
'ranks',
'rates',
'reaches',
'reasons',
'rebels',
'records',
'regards',
'registers',
'regrets',
'releases',
'remains',
```

```
'remarks',
'replies',
'reports',
'requests',
'reserves',
'respects',
'rests',
'results',
'returns',
'reviews',
'rides',
'rings',
'rises',
'rocks',
'rolls',
'rules',
'runs',
'rushes',
'sanctions',
'says',
'scales',
'scans',
'seals',
'searches',
'senses',
'services',
'sets',
'shakes',
'shapes',
'shares',
'sheds',
'shifts',
'shocks',
'shouts',
'shows',
'signals',
'signs',
'sketches',
'skins',
'skirts',
'slips',
'smells',
'smiles',
'snatches',
'sneers',
'snowballs',
'snows',
```

```
'solos',
'sounds',
'spans',
'sparks',
'speeds',
'spies',
'splashes',
'splits',
'sponsors',
'sports',
'spreads',
'springs',
'stains',
'stakes',
'stands',
'starts',
'states',
'stays',
'stems',
'steps',
'sticks',
'stops',
'stresses',
'stretches',
'strikes',
'struggles',
'studies',
'subjects',
'suits',
'sums',
'supplies',
'supports',
'surveys',
'switches',
'swoops',
'talks',
'tastes',
'terms',
'tests',
'thrusts',
'ties',
'times',
'tires',
'tops',
'tortures',
'totals',
'touches',
```

```
 'towers',
 'toys',
 'traces',
 'trades',
 'trains',
 'transfers',
 'transports',
 'traps',
 'travels',
 'treats',
 'tries',
 'trusts',
 'turns',
 'upsets',
 'urges',
 'uses',
 'values',
 'views',
 'visits',
 'votes',
 'vows',
 'walks',
 'wants',
 'watches',
 'weights',
 'winds',
 'wins',
 'wishes',
 'wonders',
 'works',
 'worries',
 'wrenches',
 'yields']
```

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

[9]:
```
tagged_words = nltk.corpus.brown.tagged_words()
phrases = []

for (w1,t1), (w2,t2), (w3,t3) in nltk.trigrams(tagged_words):
    if t1 == 'IN' and t2 == 'DET' and t3 == 'NN':
        phrases.append([w1, w2, w3])

phrases
```

[9]:
```
[]
```

Note: I have double checked my work and all of the parts appear to be working as expected. I'm unsure why I'm not getting any results.

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

[10]:
```
[word.lower() for word, tag in nltk.corpus.brown.
 ↪tagged_words(tagset='universal') if tag == 'PRON']
```

[10]:
```
['it',
 'it',
 'them',
 'them',
 'it',
 'it',
 'it',
 'it',
 'they',
 'we',
 'it',
 'that',
 'it',
 'that',
 'they',
 'it',
 'he',
 'who',
 'he',
 'it',
 'who',
 'it',
 'it',
 'he',
 'who',
 'it',
 'they',
 'it',
 'itself',
 'it',
 'who',
 'who',
 'it',
 'he',
 'him',
 'it',
 'it',
 'i',
 'who',
 'i',
 'i',
 'we',
 'it',
```

```
'himself',
'he',
'he',
'he',
'himself',
'he',
'it',
'they',
'it',
'it',
'he',
'it',
'he',
'it',
'he',
'it',
'you',
'you',
'he',
'you',
'it',
'he',
'it',
'he',
'it',
'them',
'he',
'he',
'i',
'them',
'it',
'it',
'it',
'who',
'it',
'who',
'he',
'it',
'who',
'it',
'who',
'they',
'who',
'who',
'he',
'he',
'we',
```

```
'he',
'he',
'they',
'they',
'it',
'themselves',
'you',
'he',
'whom',
'he',
'i',
'who',
'he',
'he',
'he',
'them',
'it',
'he',
'who',
'who',
'he',
'them',
'who',
'it',
'it',
'who',
'who',
'he',
'who',
'who',
'he',
'it',
'they',
'he',
'they',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'them',
'who',
'who',
'them',
```

```
'they',
'that',
'who',
'them',
'who',
'that',
'he',
'himself',
'that',
'it',
'they',
'they',
'it',
'it',
'he',
'he',
'it',
'it',
'they',
'themselves',
'they',
'he',
'he',
'he',
'he',
'he',
'it',
'it',
'he',
'it',
'that',
'he',
'it',
'them',
'itself',
'it',
'it',
'whom',
'it',
'it',
'them',
'he',
'he',
'he',
'he',
'he',
'he',
```

```
'who',
'he',
'that',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'he',
'i',
'he',
'it',
'they',
'he',
'that',
'they',
'they',
'they',
'it',
'they',
'it',
'themselves',
'he',
'him',
'he',
'he',
'i',
'he',
'he',
'i',
'me',
'whom',
'i',
'me',
'he',
'me',
'he',
'he',
'he',
'who',
```

```
'us',
'we',
'us',
'it',
'he',
'he',
'he',
'he',
'who',
'he',
'them',
'they',
'he',
'he',
'he',
'he',
'he',
'it',
'i',
'we',
'it',
'he',
'it',
'i',
'he',
'he',
'he',
'them',
'he',
'it',
'it',
'who',
'he',
'them',
'it',
'it',
'it',
'it',
'they',
'he',
'we',
'we',
'him',
'he',
'he',
'he',
'he',
```

```
'he',
'i',
'he',
'he',
'i',
'he',
'he',
'he',
'you',
'it',
'he',
'i',
'he',
'i',
'who',
'who',
'who',
'i',
'it',
'it',
'it',
'who',
'he',
'we',
'we',
'who',
'them',
'he',
'he',
'he',
'me',
'who',
'he',
'he',
'he',
'he',
'he',
'i',
'we',
'he',
'we',
'he',
'you',
'i',
'he',
'he',
'we',
```

```
'they',
'that',
'he',
'it',
'he',
'he',
'he',
'it',
'he',
'he',
'i',
'you',
'who',
'they',
'who',
'she',
'she',
'that',
'himself',
'he',
'he',
'he',
'who',
'he',
'he',
'who',
'who',
'him',
'it',
'me',
'i',
'i',
'i',
'he',
'who',
'who',
'them',
'that',
'it',
'who',
'it',
'it',
'who',
'they',
'that',
'they',
'he',
```

```
'he',
'you',
'it',
'i',
'he',
'they',
'he',
'it',
'it',
'he',
'i',
'that',
'that',
'who',
'they',
'she',
'it',
'that',
'who',
'he',
'he',
'he',
'they',
'it',
'they',
'they',
'they',
'them',
'that',
'that',
'who',
'who',
'who',
'himself',
'it',
'it',
'who',
'they',
'who',
'it',
'he',
'it',
'they',
'it',
'it',
'him',
'it',
```

```
'who',
'it',
'who',
'it',
'it',
'it',
'he',
'him',
'he',
'they',
'it',
'they',
'he',
'they',
'he',
'them',
'it',
'it',
'it',
'it',
'it',
'it',
'it',
'that',
'they',
'it',
'who',
'it',
'he',
'themselves',
'that',
'they',
'i',
'we',
'we',
'we',
'you',
'we',
'who',
'we',
'who',
'it',
'it',
'who',
'who',
'she',
'he',
```

```
'he',
'he',
'it',
'who',
'he',
'he',
'that',
'it',
'it',
'that',
'he',
'he',
'he',
'it',
'he',
'i',
'he',
'him',
'it',
'it',
'it',
'who',
'he',
'he',
'them',
'who',
'he',
'we',
'them',
'ours',
'he',
'they',
'they',
'themselves',
'it',
'he',
'he',
'it',
'we',
'we',
'he',
'us',
'he',
'we',
'we',
'them',
'themselves',
```

```
'he',
'themselves',
'he',
'it',
'they',
'they',
'them',
'he',
'they',
'he',
'they',
'he',
'i',
'we',
'we',
'who',
'they',
'we',
'it',
'it',
'it',
'it',
'it',
'he',
'he',
'he',
'they',
'it',
'he',
'he',
'we',
'we',
'we',
'we',
'we',
'we',
'we',
'he',
'he',
'who',
'who',
'it',
'it',
'they',
'they',
'who',
'him',
'who',
```

```
'it',
'it',
'he',
'it',
'it',
'who',
'who',
'he',
'he',
'me',
'he',
'i',
'i',
'i',
'i',
'i',
'he',
'who',
'it',
'it',
'who',
'him',
'they',
'she',
'who',
'him',
'who',
'she',
'he',
'who',
'he',
'he',
'that',
'he',
'he',
'he',
'he',
'he',
'he',
'me',
'he',
'he',
'me',
'i',
'it',
'he',
'he',
```

```
'that',
'i',
'he',
'i',
'i',
'he',
'he',
'he',
'he',
'you',
'it',
'it',
'them',
'you',
'you',
'you',
'he',
'they',
'he',
'he',
'he',
'he',
'it',
'he',
'i',
'it',
'i',
'we',
'it',
'we',
'it',
'he',
'he',
'that',
'he',
'him',
'it',
'it',
'he',
'it',
'he',
'him',
'he',
'it',
'he',
'it',
'he',
```

```
'it',
'he',
'him',
'i',
'who',
'who',
'they',
'it',
'they',
'they',
'who',
'we',
'himself',
'he',
'he',
'we',
'we',
'they',
'they',
'they',
'you',
'you',
'them',
'it',
'he',
'themselves',
'him',
'it',
'us',
'i',
'it',
'it',
'they',
'he',
'they',
'it',
'who',
'he',
'it',
'he',
'it',
'he',
'i',
'he',
'she',
'she',
'i',
```

```
'him',
'i',
'you',
'him',
'him',
'i',
'him',
'who',
'i',
'he',
'he',
'i',
'him',
'he',
'he',
'he',
'he',
'who',
'i',
'i',
'i',
'it',
'it',
'it',
'i',
'who',
'i',
'they',
'i',
'who',
'who',
'who',
'it',
'who',
'it',
'they',
'i',
'it',
'that',
'you',
'it',
'he',
'who',
'he',
'who',
'him',
'he',
```

```
'who',
'he',
'he',
'he',
'he',
'him',
'who',
'that',
'he',
'he',
'he',
'it',
'he',
'he',
'he',
'he',
'he',
'that',
'him',
'he',
'that',
'him',
'who',
'he',
'it',
'him',
'it',
'who',
'them',
'it',
'that',
'he',
'that',
'he',
'he',
'he',
'it',
'i',
'he',
'him',
'he',
'it',
'he',
'he',
'they',
'who',
'who',
```

```
'who',
'ye',
'he',
'him',
'he',
'who',
'who',
'that',
'it',
'who',
'it',
'i',
'it',
'i',
'it',
'who',
'it',
'he',
'he',
'he',
'it',
'that',
'i',
'i',
'you',
'it',
'it',
'who',
'you',
'he',
'he',
'who',
'he',
'he',
'they',
'they',
'they',
'they',
'who',
'them',
'that',
'who',
'he',
'i',
'i',
'he',
'he',
```

```
'it',
'who',
'who',
'he',
'him',
'them',
'they',
'it',
'that',
'that',
'who',
'they',
'who',
'themselves',
'i',
'who',
'you',
'they',
'they',
'it',
'i',
'they',
'i',
'me',
'it',
'who',
"'em",
'who',
'he',
'he',
'him',
'he',
'i',
'me',
'she',
'he',
'i',
'him',
'me',
'he',
'he',
'him',
'that',
'that',
'they',
'it',
'it',
```

```
'it',
'he',
'that',
'he',
'he',
'we',
'they',
'that',
'him',
'him',
'he',
'he',
'him',
'who',
'he',
'he',
'who',
'her',
'who',
'who',
'who',
'they',
'we',
'them',
'her',
'we',
'it',
'her',
'her',
'she',
'it',
'she',
'her',
'he',
'whom',
'who',
'who',
'he',
'he',
'you',
'we',
'me',
'who',
'you',
'you',
'she',
'you',
```

```
'who',
'it',
'it',
'me',
'me',
'he',
'him',
'he',
'he',
'he',
'me',
'he',
'who',
'who',
'who',
'it',
'who',
'her',
'that',
'he',
'he',
'himself',
'she',
'him',
'she',
'he',
'i',
'i',
'she',
'they',
'they',
'we',
'i',
'i',
'they',
'i',
'it',
'it',
'it',
'i',
'that',
'myself',
'who',
'we',
'we',
'who',
'he',
```

```
 'who',
 'she',
 'she',
 'they',
 'who',
 'they',
 'she',
 'she',
 'who',
 'her',
 'that',
 'we',
 'them',
 'we',
 'we',
 'it',
 'she',
 ...]
```

Note: Because I could not find a way to automatically categorize pronouns, I printed out all pronouns from the brown corpus and have attempted to manually categorize masculine and feminine pronouns.

[11]:
```
masculine_pronouns = ['he', 'him', 'himself', 'himselfe', 'his', 'hiself',␣
 ↪'hym', 'hymselfe']
feminine_pronouns = ['her', 'hers', 'herself', 'she']
```

[12]:
```
masculine_count = len([word for word, tag in nltk.corpus.brown.
 ↪tagged_words(tagset='universal') if word in masculine_pronouns])
feminine_count = len([word for word, tag in nltk.corpus.brown.
 ↪tagged_words(tagset='universal') if word in feminine_pronouns])
masculine_count / feminine_count
```

[12]:
```
3.258492462311558
```