In [4]:
```
# -*- coding: utf-8 -*-
"""
Created July 2017

@author: arw
"""

from nltk.tag.sequential import ClassifierBasedPOSTagger
from nltk.classify import NaiveBayesClassifier, MaxentClassifier
from nltk.tag import TrigramTagger  # Considers previous 2 words
from nltk.tag import BigramTagger  # Considers previous word
from nltk.tag import UnigramTagger  # Context insentitive
from nltk.tag import RegexpTagger
from nltk.tag import DefaultTagger
from nltk.corpus import treebank
import nltk
sentence = "I saw the man with the telescope but he didn't see me"


# Using NLTK's built-in tagger based on PTB
tokens = nltk.word_tokenize(sentence)
tagged_sent = nltk.pos_tag(tokens, tagset='universal')
print(tagged_sent)


# # Using the pattern package (Python 2.x only) built-in tagger (optional)
# from pattern.en import tag
# tagged_sent = tag(sentence)
# print(tagged_sent)


# Building your own tagger
# - default tagger that tags all words the same!
# - regex tagger that doesn't care about context (most common tag per word)

# Fortunately the treebank corpus is bundled with NLTK for training a tagger
# We need to divide the data into training and test sets first
data = treebank.tagged_sents()
train_data = data[:3500]
test_data = data[3500:]
print(train_data[0])

# SAQ 1: How much data is there for training, testing?
# SAQ 2: What is the last training sentence; test sentence?

# Default 'naive' tagger - tags all words with a given tag!
# Can specify any default tag - NN gives best score - why?
dt = DefaultTagger('NN')

# Test score and example sentence tag output
print(dt.evaluate(test_data))
print(dt.tag(tokens))


# Regex tagger
# Define 'fixed' regex tag patterns
patterns = [
    (r'.*ing$', 'VBG'),               # gerunds
```

```
    (r'.*ed$', 'VBD'),                # simple past
    (r'.*es$', 'VBZ'),                # 3rd singular present
    (r'.*ould$', 'MD'),               # modals
    (r'.*\'s$', 'NN$'),               # possessive nouns
    (r'.*s$', 'NNS'),                 # plural nouns
    (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),  # cardinal numbers
    (r'.*', 'NN')                     # nouns (default) ...
]
rt = RegexpTagger(patterns)

# Test score and example sentence tag output
print(rt.evaluate(test_data))
print(rt.tag(tokens))


# Training your own tagger
# 1. using n-gram taggers and combining them with backoff
# 2. using naive bayes (statistical) model
# 3. using maximum entropy (classifier) model

# N gram taggers

# Traing the taggers
ut = UnigramTagger(train_data)
bt = BigramTagger(train_data)
tt = TrigramTagger(train_data)

# Test UnigramTagger score and example sentence tag output
print(ut.evaluate(test_data))
print(ut.tag(tokens))

# Test BigramTagger score and example sentence tag output
print(bt.evaluate(test_data))
print(bt.tag(tokens))

# Test TrigramTagger score and example sentence tag output
print(tt.evaluate(test_data))
print(tt.tag(tokens))

# Combining all 3 n-gram taggers with backoff (smoothing)


def combined_tagger(train_data, taggers, backoff=None):
    for tagger in taggers:
        backoff = tagger(train_data, backoff=backoff)
    return backoff


ct = combined_tagger(train_data=train_data,
                     taggers=[UnigramTagger, BigramTagger, TrigramTagger],
                     backoff=rt)

# Test Combined n-gram tagger score and example sentence tag output
print(ct.evaluate(test_data))
print(ct.tag(tokens))


# Treating POS tagging as a classification problem
# We use the ClassifierBasedPOSTagger class to build a classifier by specifying some
# classification algorithm - here the NaiveBayes abd Maxent algorithms which are passe
```

```
# to the class via the classifier_builder parameter

# First a Naive Bayes (statistical) classifier
nbt = ClassifierBasedPOSTagger(train=train_data,
                               classifier_builder=NaiveBayesClassifier.train)

# Test NBC tagger score and example sentence tag output
print(nbt.evaluate(test_data))
print(nbt.tag(tokens))


# Finally a Maximum entropy classifier (that would take sometime)
# met = ClassifierBasedPOSTagger(train=train_data,
#                                classifier_builder=MaxentClassifier.train)

met = ClassifierBasedPOSTagger(train=train_data,
                               classifier_builder=lambda train_feats: MaxentClassifier

# Test Maxent tagger score and example sentence tag output
print(met.evaluate(test_data))
print(met.tag(tokens))


# Final accuracies
print('Tagger accuracies:')
print()
print('Default tagger %.2f' % dt.evaluate(test_data))
print('Regex tagger %.2f' % rt.evaluate(test_data))
print('Unigram tagger %.2f' % ut.evaluate(test_data))
print('Bigram tagger %.2f' % bt.evaluate(test_data))
print('Trigram tagger %.2f' % tt.evaluate(test_data))
print('Combined tagger %.2f' % ct.evaluate(test_data))
print('Naive Bayes tagger %.2f' % nbt.evaluate(test_data))
print('Maxent tagger %.2f' % met.evaluate(test_data))
```

```
[('I', 'PRON'), ('saw', 'VERB'), ('the', 'DET'), ('man', 'NOUN'), ('with', 'ADP'),
('the', 'DET'), ('telescope', 'NOUN'), ('but', 'CONJ'), ('he', 'PRON'), ('did', 'VER
B'), ("n't", 'ADV'), ('see', 'VERB'), ('me', 'PRON')]
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'),
('old', 'JJ'), (',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board',
'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('director', 'NN'), ('No
v.', 'NNP'), ('29', 'CD'), ('.', '.')]
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:51: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(dt.evaluate(test_data))
0.1454158195372253
[('I', 'NN'), ('saw', 'NN'), ('the', 'NN'), ('man', 'NN'), ('with', 'NN'), ('the', 'N
N'), ('telescope', 'NN'), ('but', 'NN'), ('he', 'NN'), ('did', 'NN'), ("n't", 'NN'),
('see', 'NN'), ('me', 'NN')]
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:70: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(rt.evaluate(test_data))
0.24039113176493368
[('I', 'NN'), ('saw', 'NN'), ('the', 'NN'), ('man', 'NN'), ('with', 'NN'), ('the', 'N
N'), ('telescope', 'NN'), ('but', 'NN'), ('he', 'NN'), ('did', 'NN'), ("n't", 'NN'),
('see', 'NN'), ('me', 'NN')]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:87: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(ut.evaluate(test_data))
```

```
0.8607803272340013
[('I', 'PRP'), ('saw', 'VBD'), ('the', 'DT'), ('man', 'NN'), ('with', 'IN'), ('the',
'DT'), ('telescope', None), ('but', 'CC'), ('he', 'PRP'), ('did', 'VBD'), ("n't", 'R
B'), ('see', 'VB'), ('me', 'PRP')]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:91: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(bt.evaluate(test_data))
```

```
0.13466937748087907
[('I', 'PRP'), ('saw', None), ('the', None), ('man', None), ('with', None), ('the', N
one), ('telescope', None), ('but', None), ('he', None), ('did', None), ("n't", None),
('see', None), ('me', None)]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:95: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(tt.evaluate(test_data))
```

```
0.08064672281924679
[('I', 'PRP'), ('saw', None), ('the', None), ('man', None), ('with', None), ('the', N
one), ('telescope', None), ('but', None), ('he', None), ('did', None), ("n't", None),
('see', None), ('me', None)]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:112: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(ct.evaluate(test_data))
```

```
0.9094781682641108
[('I', 'PRP'), ('saw', 'VBD'), ('the', 'DT'), ('man', 'NN'), ('with', 'IN'), ('the',
'DT'), ('telescope', 'NN'), ('but', 'CC'), ('he', 'PRP'), ('did', 'VBD'), ("n't", 'R
B'), ('see', 'VB'), ('me', 'PRP')]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:126: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(nbt.evaluate(test_data))
```

```
0.9306806079969019
[('I', 'PRP'), ('saw', 'VBD'), ('the', 'DT'), ('man', 'NN'), ('with', 'IN'), ('the',
'DT'), ('telescope', 'NN'), ('but', 'CC'), ('he', 'PRP'), ('did', 'VBD'), ("n't", 'R
B'), ('see', 'VB'), ('me', 'PRP')]
  ==> Training (10 iterations)

      Iteration    Log Likelihood    Accuracy
      ---------------------------------------
             1          -3.82864        0.007
             2          -0.76176        0.957
         Final               nan        0.984
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:138: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print(met.evaluate(test_data))
```

```
0.9270984606447865
[('I', 'PRP'), ('saw', 'VBD'), ('the', 'DT'), ('man', 'NN'), ('with', 'IN'), ('the',
'DT'), ('telescope', 'NN'), ('but', 'CC'), ('he', 'PRP'), ('did', 'VBD'), ("n't", 'R
B'), ('see', 'VB'), ('me', 'PRP')]
Tagger accuracies:
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:145: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Default tagger %.2f' % dt.evaluate(test_data))
Default tagger 0.15
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:146: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Regex tagger %.2f' % rt.evaluate(test_data))
Regex tagger 0.24
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:147: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Unigram tagger %.2f' % ut.evaluate(test_data))
Unigram tagger 0.86
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:148: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Bigram tagger %.2f' % bt.evaluate(test_data))
Bigram tagger 0.13
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:149: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Trigram tagger %.2f' % tt.evaluate(test_data))
Trigram tagger 0.08
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:150: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Combined tagger %.2f' % ct.evaluate(test_data))
Combined tagger 0.91
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:151: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Naive Bayes tagger %.2f' % nbt.evaluate(test_data))
Naive Bayes tagger 0.93
C:\Users\User\AppData\Local\Temp\ipykernel_3036\844224969.py:152: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  print('Maxent tagger %.2f' % met.evaluate(test_data))
Maxent tagger 0.93
```

In [ ]: