Name : Asaha Belcilda

Roll:225229104

NLP Lab 9 : Building Bigram Tagger

EXERCISE - 1

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

In [2]:
```
from nltk.tokenize import sent_tokenize,word_tokenize
```

In [5]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\1MSCDSA13\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[5]:
```
True
```

In [7]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\1MSCDSA13\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[7]:
```
True
```

In [8]:
```
text = word_tokenize("And now for something completely different")
nltk.pos_tag(text)
```

Out[8]:
```
[('And', 'CC'),
 ('now', 'RB'),
 ('for', 'IN'),
 ('something', 'NN'),
 ('completely', 'RB'),
 ('different', 'JJ')]
```

EXERCISE - 2

In [9]:
```
from nltk.corpus import brown
```

In [10]:
```
nltk.download('brown')
```

```
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\1MSCDSA13\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\brown.zip.
```

Out[10]:
```
True
```

In [11]:
```
tagsen = brown.tagged_sents()
tagsen
```

Out[11]:
```
[[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-TL'),
('Jury', 'NN-TL'), ('said', 'VBD'), ('Friday', 'NR'), ('an', 'AT'), ('investiga
tion', 'NN'), ('of', 'IN'), ("Atlanta's", 'NP$'), ('recent', 'JJ'), ('primary',
'NN'), ('election', 'NN'), ('produced', 'VBD'), ('``', '``'), ('no', 'AT'), ('e
vidence', 'NN'), ("''", "''"), ('that', 'CS'), ('any', 'DTI'), ('irregularitie
s', 'NNS'), ('took', 'VBD'), ('place', 'NN'), ('.', '.')], [('The', 'AT'), ('ju
ry', 'NN'), ('further', 'RBR'), ('said', 'VBD'), ('in', 'IN'), ('term-end', 'N
N'), ('presentments', 'NNS'), ('that', 'CS'), ('the', 'AT'), ('City', 'NN-TL'),
('Executive', 'JJ-TL'), ('Committee', 'NN-TL'), (',', ','), ('which', 'WDT'),
('had', 'HVD'), ('over-all', 'JJ'), ('charge', 'NN'), ('of', 'IN'), ('the', 'A
T'), ('election', 'NN'), (',', ','), ('``', '``'), ('deserves', 'VBZ'), ('the',
'AT'), ('praise', 'NN'), ('and', 'CC'), ('thanks', 'NNS'), ('of', 'IN'), ('th
e', 'AT'), ('City', 'NN-TL'), ('of', 'IN-TL'), ('Atlanta', 'NP-TL'), ("''",
"''"), ('for', 'IN'), ('the', 'AT'), ('manner', 'NN'), ('in', 'IN'), ('which',
'WDT'), ('the', 'AT'), ('election', 'NN'), ('was', 'BEDZ'), ('conducted', 'VB
N'), ('.', '.')], ...]
```

Prepare data sets

In [12]:
```
len(tagsen)
```

Out[12]:
```
57340
```

In [13]:
```
br_train = tagsen[0:50000]
br_test = tagsen[50000:]
br_test[0]
```

Out[13]:
```
[('I', 'PPSS'),
 ('was', 'BEDZ'),
 ('loaded', 'VBN'),
 ('with', 'IN'),
 ('suds', 'NNS'),
 ('when', 'WRB'),
 ('I', 'PPSS'),
 ('ran', 'VBD'),
 ('away', 'RB'),
 (',', ','),
 ('and', 'CC'),
 ('I', 'PPSS'),
 ("haven't", 'HV*'),
 ('had', 'HVN'),
 ('a', 'AT'),
 ('chance', 'NN'),
 ('to', 'TO'),
 ('wash', 'VB'),
 ('it', 'PPO'),
 ('off', 'RP'),
 ('.', '.')]
```

Build a bigram tagger

In [14]:
```
t0 = nltk.DefaultTagger('NN')
t1 = nltk.UnigramTagger(br_train, backoff=t0)
t2 = nltk.BigramTagger(br_train, backoff=t1)
```

Evaluate

In [15]:
```
t2.evaluate(br_test)
```

Out[15]:
```
0.9111006662708622
```

Explore

In [16]:
```
# 1.
total_train = [len(l) for l in br_train]
sum(total_train)
```

Out[16]:
```
1039920
```

In [17]:
```
total_test = [len(l) for l in br_test]
sum(total_test)
```
Out[17]:
```
121272
```

In [18]:
```
# 2.
t1.evaluate(br_test)
```
Out[18]:
```
0.8897849462365591
```

In [19]:
```
t2.evaluate(br_test)
```
Out[19]:
```
0.9111006662708622
```

In [20]:
```
# 3.
br_train[0]
```
Out[20]:
```
[('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'),
 ('.', '.')]
```

In [21]:
```
br_train
[1277]
```
Out[21]:
```
[1277]
```

In [22]:
```
br_train[1277] [11]
```
Out[22]:
```
('cold', 'JJ')
```

In [23]:
```
# 4.
br_train_flat = [(word, tag) for sent in br_train for (word, tag) in sent]
```

In [24]:
```
br_train_flat[:40]
```

Out[24]:
```
[('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'),
 ('.', '.'),
 ('The', 'AT'),
 ('jury', 'NN'),
 ('further', 'RBR'),
 ('said', 'VBD'),
 ('in', 'IN'),
 ('term-end', 'NN'),
 ('presentments', 'NNS'),
 ('that', 'CS'),
 ('the', 'AT'),
 ('City', 'NN-TL'),
 ('Executive', 'JJ-TL'),
 ('Committee', 'NN-TL'),
 (',', ','),
 ('which', 'WDT'),
 ('had', 'HVD')]
```

In [25]:
```
br_train_flat[13]
```

Out[25]:
```
('election', 'NN')
```

In [26]:
```
# 5. a)
fd = nltk.FreqDist(br_train_flat)
cfd = nltk.ConditionalFreqDist(br_train_flat)
```

In [27]:
```
cfd['cold'].most_common()
```

Out[27]:
```
[('JJ', 110), ('NN', 8), ('RB', 2)]
```

In [28]:
```
# 5. b)
br_train_2grams = list(nltk.ngrams(br_train_flat, 2))
br_train_cold = [a[1] for (a,b) in br_train_2grams if b[0] == 'cold']
fdist = nltk.FreqDist(br_train_cold)
[tag for (tag, _) in fdist.most_common()]
```

Out[28]:
```
['AT',
 'IN',
 'CC',
 'QL',
 'BEDZ',
 'JJ',
 ',',
 'DT',
 'PP$',
 'RP',
 '``',
 'NN',
 'VBN',
 'VBD',
 'CS',
 'BEZ',
 'DOZ',
 'RB',
 'PPSS',
 'BE',
 'VB',
 'VBZ',
 'NP$',
 'BEDZ*',
 '--',
 'DTI',
 'WRB',
 'BED']
```

In [29]:
```
# 5. c)
br_pre = [(w2+"/"+t2, t1) for ((w1,t1),(w2,t2)) in br_train_2grams]
br_pre_cfd = nltk.ConditionalFreqDist(br_pre)
br_pre
```

```
 ('had/HVD', 'WDT'),
 ('over-all/JJ', 'HVD'),
 ('charge/NN', 'JJ'),
 ('of/IN', 'NN'),
 ('the/AT', 'IN'),
 ('election/NN', 'AT'),
 (',/,', 'NN'),
 ('``/``', ','),
 ('deserves/VBZ', '``'),
 ('the/AT', 'VBZ'),
 ('praise/NN', 'AT'),
 ('and/CC', 'NN'),
 ('thanks/NNS', 'CC'),
 ('of/IN', 'NNS'),
 ('the/AT', 'IN'),
 ('City/NN-TL', 'AT'),
 ('of/IN-TL', 'NN-TL'),
 ('Atlanta/NP-TL', 'IN-TL'),
 ("''/''", 'NP-TL'),
 ('for/IN', "''"),
```

In [30]:
```
# 5. d)
br_pre_cfd['cold/NN'].most_common()
```

Out[30]:
```
[('AT', 4), ('JJ', 2), (',', 1), ('DT', 1)]
```

In [31]:
```
br_pre_cfd['cold/JJ'].most_common()
```

Out[31]:
```
[('AT', 38),
 ('IN', 14),
 ('CC', 8),
 ('QL', 7),
 ('BEDZ', 7),
 ('JJ', 4),
 ('DT', 3),
 (',', 3),
 ('PP$', 3),
 ('``', 2),
 ('NN', 2),
 ('VBN', 2),
 ('VBD', 2),
 ('CS', 1),
 ('BEZ', 1),
 ('DOZ', 1),
 ('RB', 1),
 ('PPSS', 1),
 ('BE', 1),
 ('VB', 1),
 ('VBZ', 1),
 ('NP$', 1),
 ('BEDZ*', 1),
 ('--', 1),
 ('RP', 1),
 ('DTI', 1),
 ('WRB', 1),
 ('BED', 1)]
```

In [32]:
```
# 6.
bigram_tagger = nltk.BigramTagger(br_train)
```

In [33]:
```
# 6. a)
text1 = word_tokenize('I was very cold.')
bigram_tagger.tag(text1)
```

Out[33]:
```
[('I', 'PPSS'), ('was', 'BEDZ'), ('very', 'QL'), ('cold', 'JJ'), ('.', '.')]
```

In [34]:
```
# 6. b)
text2 = word_tokenize('I had a cold.')
bigram_tagger.tag(text2)
```

Out[34]:
```
[('I', 'PPSS'), ('had', 'HVD'), ('a', 'AT'), ('cold', 'JJ'), ('.', '.')]
```

In [35]:
```
# 6. c)
text3 = word_tokenize('I had a severe cold.')
bigram_tagger.tag(text3)
```

Out[35]:
```
[('I', 'PPSS'),
 ('had', 'HVD'),
 ('a', 'AT'),
 ('severe', 'JJ'),
 ('cold', 'JJ'),
 ('.', '.')]
```

In [36]:
```
# 6. d)
text4 = word_tokenize('January was a cold month.')
bigram_tagger.tag(text4)
```

Out[36]:
```
[('January', None),
 ('was', None),
 ('a', None),
 ('cold', None),
 ('month', None),
 ('.', None)]
```

In [44]:
```
#7
sentences = [
    "I was very cold",
    "I had a cold",
    "I had a severe cold",
    "January was a cold month"
]

for sentence in sentences:
    words = nltk.word_tokenize(sentence)
    pos_tags = nltk.pos_tag(words)
    print(pos_tags)
```

```
[('I', 'PRP'), ('was', 'VBD'), ('very', 'RB'), ('cold', 'JJ')]
[('I', 'PRP'), ('had', 'VBD'), ('a', 'DT'), ('cold', 'JJ')]
[('I', 'PRP'), ('had', 'VBD'), ('a', 'DT'), ('severe', 'JJ'), ('cold', 'NN')]
[('January', 'NNP'), ('was', 'VBD'), ('a', 'DT'), ('cold', 'JJ'), ('month', 'N
N')]
```

In [37]:
```
# 8. a)
text5 = word_tokenize('I failed to do so.')
bigram_tagger.tag(text5)
```

Out[37]:
```
[('I', 'PPSS'),
 ('failed', 'VBD'),
 ('to', 'TO'),
 ('do', 'DO'),
 ('so', 'RB'),
 ('.', '.')]
```

In [38]:
```
# 8. b)
text6 = word_tokenize('I was happy,but so was my enemy.')
bigram_tagger.tag(text6)
```

Out[38]:
```
[('I', 'PPSS'),
 ('was', 'BEDZ'),
 ('happy', 'JJ'),
 (',', ','),
 ('but', 'CC'),
 ('so', 'RB'),
 ('was', 'BEDZ'),
 ('my', 'PP$'),
 ('enemy', 'NN'),
 ('.', '.')]
```

In [39]:
```
# 8. c)
text7 = word_tokenize('So, how was the exam?')
bigram_tagger.tag(text7)
```

Out[39]:
```
[('So', 'RB'),
 (',', ','),
 ('how', 'WRB'),
 ('was', 'BEDZ'),
 ('the', 'AT'),
 ('exam', None),
 ('?', None)]
```

In [40]:
```
# 8. d)
text8 = word_tokenize('The students came in early so they can get good seats.')
bigram_tagger.tag(text8)
```

Out[40]:
```
[('The', 'AT'),
 ('students', 'NNS'),
 ('came', 'VBD'),
 ('in', 'IN'),
 ('early', 'JJ'),
 ('so', 'CS'),
 ('they', 'PPSS'),
 ('can', 'MD'),
 ('get', 'VB'),
 ('good', 'JJ'),
 ('seats', 'NNS'),
 ('.', '.')]
```

In [41]:
```
# 8. e)
text9 = word_tokenize('She failed the exam, so she must take it again.')
bigram_tagger.tag(text9)
```

Out[41]:
```
[('She', 'PPS'),
 ('failed', 'VBD'),
 ('the', 'AT'),
 ('exam', None),
 (',', None),
 ('so', None),
 ('she', None),
 ('must', None),
 ('take', None),
 ('it', None),
 ('again', None),
 ('.', None)]
```

In [42]:
```
# 8. f)
text10 = word_tokenize('That was so incredible.')
bigram_tagger.tag(text10)
```

Out[42]:
```
[('That', 'DT'),
 ('was', 'BEDZ'),
 ('so', 'QL'),
 ('incredible', 'JJ'),
 ('.', '.')]
```

In [43]:
```
# 8. g)
text11 = word_tokenize('Wow, so incredible.')
bigram_tagger.tag(text11)
```

Out[43]:
```
[('Wow', None), (',', None), ('so', None), ('incredible', None), ('.', None)]
```

In [45]:
```
# 9.
text11 = word_tokenize('so')
bigram_tagger.tag(text11)
```

Out[45]:
```
[('so', 'RB')]
```

#10
The bigram tagger is a part-of-speech (POS) tagger that assigns a POS tag to each word in a text based on the preceding word. It is a simple and efficient algorithm that uses a bigram language model to predict the most likely POS tag for each word based on the probability of observing that tag given the previous tag.

Strengths:

The bigram tagger is fast and computationally efficient compared to other more complex POS taggers. It can process large amounts of text quickly and accurately.
It can handle unknown words by assigning them the most common POS tag in the training corpus.
The bigram model can capture some context and syntactic dependencies between adjacent words, leading to improved accuracy in some cases.

Limitations:

The bigram tagger relies heavily on the previous word to assign POS tags, so it may miss more complex dependencies or contextual cues that are further away.
It can't capture long-range dependencies or structural information that spans more than two adjacent words.
It can't handle words that have multiple POS tags or ambiguous meanings, leading to incorrect tagging.
The accuracy of the bigram tagger heavily depends on the quality and size of the training corpus. A small or biased corpus can lead to poor tagging performance.

In conclusion, the bigram tagger is a useful and efficient POS tagging tool for certain applications, especially in cases where a large training corpus is available and where simple contextual cues are sufficient for accurate tagging. However, it has clear limitations in handling more complex language structures, ambiguous words, and long-range dependencies.

In [ ]: