NLP Lab 9 : Building Bigram Tagger

DINESH KUMAR K

225229108

EXERCISE - 1

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

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

In [3]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\1mscdsa08\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[3]:
```
True
```

In [4]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\1mscdsa08\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

In [5]:
```
text = word_tokenize("And now for something completely different")
nltk.pos_tag(text)
```

Out[5]:
```
[('And', 'CC'),
 ('now', 'RB'),
 ('for', 'IN'),
 ('something', 'NN'),
 ('completely', 'RB'),
 ('different', 'JJ')]
```

EXERCISE - 2

In [6]:
```
from nltk.corpus import brown
```

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

```
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\1mscdsa08\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\brown.zip.
```

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

In [8]:
```
tagsen = brown.tagged_sents()
tagsen
```

Out[8]:
```
[[('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 [9]:
```
len(tagsen)
```

Out[9]:
```
57340
```

In [10]:
```
br_train = tagsen[0:50000]
br_test = tagsen[50000:]
br_test[0]
```

Out[10]:
```
[('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 [11]:
```
t0 = nltk.DefaultTagger('NN')
t1 = nltk.UnigramTagger(br_train, backoff=t0)
t2 = nltk.BigramTagger(br_train, backoff=t1)
```

In [12]:
```
t2.evaluate(br_test)
```

Out[12]:
```
0.9111006662708622
```

In [13]:
```
total_train = [len(l) for l in br_train]
sum(total_train)
```

Out[13]:
```
1039920
```

In [14]:
```
total_test = [len(l) for l in br_test]
sum(total_test)
```

Out[14]:
```
121272
```

In [15]:
```
t1.evaluate(br_test)
```

Out[15]:
```
0.8897849462365591
```

In [16]:
```
t2.evaluate(br_test)
```

Out[16]:
```
0.9111006662708622
```

In [19]:
```
br_train[0]
```

Out[19]:
```
[('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 [20]:
```
br_train[1277]
```

Out[20]:
```
[('``', '``'),
 ('I', 'PPSS'),
 ('told', 'VBD'),
 ('him', 'PPO'),
 ('who', 'WPS'),
 ('I', 'PPSS'),
 ('was', 'BEDZ'),
 ('and', 'CC'),
 ('he', 'PPS'),
 ('was', 'BEDZ'),
 ('quite', 'QL'),
 ('cold', 'JJ'),
 ('.', '.')]
```

In [21]:
```
br_train[1277] [11]
```

Out[21]:
```
('cold', 'JJ')
```

In [22]:
```
br_train_flat = [(word, tag) for sent in br_train for (word, tag) in sent]
```

In [23]:
```
br_train_flat[:40]
```

Out[23]:
```
[('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 [24]:
```
br_train_flat[13]
```

Out[24]:
```
('election', 'NN')
```

In [25]:
```
fd = nltk.FreqDist(br_train_flat)
cfd = nltk.ConditionalFreqDist(br_train_flat)
```

In [26]:
```
cfd['cold'].most_common()
```

Out[26]:
```
[('JJ', 110), ('NN', 8), ('RB', 2)]
```

In [27]:
```
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[27]:
```
['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 [28]:
```
br_pre = [(w2+"/"+t2, t1) for ((w1,t1),(w2,t2)) in br_train_2grams]
br_pre_cfd = nltk.ConditionalFreqDist(br_pre)
br_pre
```

Out[28]:
```
[('Fulton/NP-TL', 'AT'),
 ('County/NN-TL', 'NP-TL'),
 ('Grand/JJ-TL', 'NN-TL'),
 ('Jury/NN-TL', 'JJ-TL'),
 ('said/VBD', 'NN-TL'),
 ('Friday/NR', 'VBD'),
 ('an/AT', 'NR'),
 ('investigation/NN', 'AT'),
 ('of/IN', 'NN'),
 ("Atlanta's/NP$", 'IN'),
 ('recent/JJ', 'NP$'),
 ('primary/NN', 'JJ'),
 ('election/NN', 'NN'),
 ('produced/VBD', 'NN'),
 ('``/``', 'VBD'),
 ('no/AT', '``'),
 ('evidence/NN', 'AT'),
 ("''/''", 'NN'),
 ('that/CS', "''"),
```

In [29]:
```
br_pre_cfd['cold/NN'].most_common()
```

Out[29]:
```
[('AT', 4), ('JJ', 2), (',', 1), ('DT', 1)]
```

In [30]:
```
br_pre_cfd['cold/JJ'].most_common()
```

Out[30]:
```
[('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 [31]:
```
bigram_tagger = nltk.BigramTagger(br_train)
```

In [32]:
```
text1 = word_tokenize('I was very cold.')
bigram_tagger.tag(text1)
```

Out[32]:
```
[('I', 'PPSS'), ('was', 'BEDZ'), ('very', 'QL'), ('cold', 'JJ'), ('.', '.')]
```

In [33]:
```
text2 = word_tokenize('I had a cold.')
bigram_tagger.tag(text2)
```

Out[33]:
```
[('I', 'PPSS'), ('had', 'HVD'), ('a', 'AT'), ('cold', 'JJ'), ('.', '.')]
```

In [34]:
```
text3 = word_tokenize('I had a severe cold.')
bigram_tagger.tag(text3)
```

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

In [35]:
```
text4 = word_tokenize('January was a cold month.')
bigram_tagger.tag(text4)
```

Out[35]:
```
[('January', None),
 ('was', None),
 ('a', None),
 ('cold', None),
 ('month', None),
 ('.', None)]
```

In [36]:
```
text5 = word_tokenize('I failed to do so.')
bigram_tagger.tag(text5)
```

Out[36]:
```
[('I', 'PPSS'),
 ('failed', 'VBD'),
 ('to', 'TO'),
 ('do', 'DO'),
 ('so', 'RB'),
 ('.', '.')]
```

In [37]:
```
text6 = word_tokenize('I was happy,but so was my enemy.')
bigram_tagger.tag(text6)
```

Out[37]:
```
[('I', 'PPSS'),
 ('was', 'BEDZ'),
 ('happy', 'JJ'),
 (',', ','),
 ('but', 'CC'),
 ('so', 'RB'),
 ('was', 'BEDZ'),
 ('my', 'PP$'),
 ('enemy', 'NN'),
 ('.', '.')]
```

In [38]:
```
text7 = word_tokenize('So, how was the exam?')
bigram_tagger.tag(text7)
```

Out[38]:
```
[('So', 'RB'),
 (',', ','),
 ('how', 'WRB'),
 ('was', 'BEDZ'),
 ('the', 'AT'),
 ('exam', None),
 ('?', None)]
```

In [39]:
```
text8 = word_tokenize('The students came in early so they can get good seats.')
bigram_tagger.tag(text8)
```

Out[39]:
```
[('The', 'AT'),
 ('students', 'NNS'),
 ('came', 'VBD'),
 ('in', 'IN'),
 ('early', 'JJ'),
 ('so', 'CS'),
 ('they', 'PPSS'),
 ('can', 'MD'),
 ('get', 'VB'),
 ('good', 'JJ'),
 ('seats', 'NNS'),
 ('.', '.')]
```

In [40]:
```
text9 = word_tokenize('She failed the exam, so she must take it again.')
bigram_tagger.tag(text9)
```

Out[40]:
```
[('She', 'PPS'),
 ('failed', 'VBD'),
 ('the', 'AT'),
 ('exam', None),
 (',', None),
 ('so', None),
 ('she', None),
 ('must', None),
 ('take', None),
 ('it', None),
 ('again', None),
 ('.', None)]
```

In [41]:
```
text10 = word_tokenize('That was so incredible.')
bigram_tagger.tag(text10)
```

Out[41]:
```
[('That', 'DT'),
 ('was', 'BEDZ'),
 ('so', 'QL'),
 ('incredible', 'JJ'),
 ('.', '.')]
```

In [42]:
```
text11 = word_tokenize('Wow, so incredible.')
bigram_tagger.tag(text11)
```

Out[42]:
```
[('Wow', None), (',', None), ('so', None), ('incredible', None), ('.', None)]
```

In [ ]:
```

```