PARTS OF SPEECH(POS)

The part of speech explains how a word is used in a sentence.

About Data

- The data used is indian corpus which is a tagged data.

TnT(Trigrams'n'Tags)

- It is a statistical tagger that works on second-order Markov models.
- It is a very efficient part-of-speech tagger that can be trained on different languages and on any tagset.
- Parameter to modify for TnT is N i.e. it controls the no. of possible solutions the tagger maintains.
- By defaults N = 100.
- Amount of memory will increase if increase the value of N, without any specific increase of accuracy.

Averaged Perceptron tagger

- Structured prediction:For POS tagging the simplest approach is break the problem down to per-token tagging and then tag left to right.
- The perceptron stores a sparse linear vector of (feature, weight) pairs. Observations are scored by taking the dot product of their feature vector with the perceptron. Training is done trying a prediction, and updating incorrect weights by reenforcing or penalising them depending on whether they correspond with the correct observation.
- To get the best weights the perceptron is trained for several iterations on reshuffled observations and the averaged weights across the whole training period are saved as the final weights.

```
import nltk
nltk.download('punkt')  #word_tokenize
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
import nltk
nltk.download("indian") #corpus
```

```
[nltk_data] Downloading package indian to /root/nltk_data...
[nltk_data]   Unzipping corpora/indian.zip.
True
```

- Importing libraries

```
from nltk import word_tokenize
from nltk.corpus import indian
from nltk.tag import tnt
from nltk.tag import perceptron
from nltk.tag import DefaultTagger
```

```
indian.fileids() #different pos
```

```
['bangla.pos', 'hindi.pos', 'marathi.pos', 'telugu.pos']
```

```
for i in indian.fileids():          #tagged words
  print(i)
  print(len(indian.words(i)))
```

```
bangla.pos
10281
hindi.pos
9408
marathi.pos
19066
telugu.pos
9999
```

```
tel_sent=indian.sents("telugu.pos")
print(tel_sent)
```

```
[['4', '.'], ['ఆడిట్', 'నిర్వహణ', 'ఆడిటర్', 'ఒక', 'కొత్త', 'ఆడిట్', 'చేపట్టే', 'ముందు', 'సక్రమ', 'పద్ధతి', 'లో', 'కార్య', 'ప్రణాళికను',
```

- Number of sentences

```
len(tel_sent)
```

```
994
```

```
a=indian.tagged_sents("telugu.pos")[1]
a
```

```
[('ఆడిట్', 'NN'),
 ('నిర్వహణ', 'NN'),
 ('ఆడిటర్', 'NN'),
 ('ఒక', 'QFNUM'),
 ('కొత్త', 'JJ'),
 ('ఆడిట్', 'NN'),
 ('చేపట్టే', 'VRB'),
 ('ముందు', 'PREP'),
 ('సక్రమ', 'JJ'),
 ('పద్ధతి', 'NN'),
 ('లో', 'PREP'),
 ('కార్య', 'JJ'),
 ('ప్రణాళికను', 'NN'),
 ('రూపొందించాలి', 'VFM'),
 ('.', 'SYM')]
```

training of tnt and accuracy of tnt tagger

```
 # initializing training and testing set
train_data = indian.tagged_sents("telugu.pos")[:900]
test_data = indian.tagged_sents("telugu.pos")[900:]
# initializing tagger
tnt_tagger = tnt.TnT(N=100)
# training
tnt_tagger.train(train_data)
# evaluating
a = tnt_tagger.evaluate(test_data)
print ("Accuracy of TnT Tagging : ", a)
```

```
Accuracy of TnT Tagging :  0.5234093637454982
```

```
ex="ఇది ఒక వాక్యం కాదు."
```

```
tokens=word_tokenize(ex)
tokens
```

```
['ఇది', 'ఒక', 'వాక్యం', 'కాదు', '.']
```

```
test=tnt_tagger.tag_sents([[i for i in tokens]])
test
```

```
[[('ఇది', 'PRP'),
  ('ఒక', 'JJ'),
  ('వాక్యం', 'Unk'),
  ('కాదు', 'NEG'),
  ('.', 'SYM')]]
```

training of perceptron tagger and accuracy of tagger

```
 # initializing tagger
pcp_tagger = perceptron.PerceptronTagger(load=False)
# training
pcp_tagger.train(train_data)
# evaluating
b = pcp_tagger.evaluate(test_data)
print ("Accuracy of perceptron Tagging : ", b)
```

```
Accuracy of perceptron Tagging :  0.7899159663865546
```

```
test1=pcp_tagger.tag_sents([[i for i in tokens]])
test1
```

```
[[('ఇది', 'PRP'),
  ('ఒక', 'JJ'),
  ('వాక్యం', 'NN'),
  ('కాదు', 'NEG'),
  ('.', 'SYM')]]
```

- conclusion

Average perceptron tagger is the best tagger(algorithm) among tnt tagger.