```
1 import nltk
2 nltk.download
3 import nltk.corpus
4 nltk.download('punkt')
```

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

```
1 from nltk.tokenize import word_tokenize
```

```
1 snow_lepard = "The viral image shows the wildcat hiding somewhere in a rocky mountainc
```

```
1 snow_lepard_tokens = word_tokenize(snow_lepard)
2 snow_lepard_tokens
```

```
 'somewhere',
 'in',
 'a',
 'rocky',
 'mountainous',
 'region',
 'and',
 'snow',
 '.',
 'The',
 'image',
 ',',
 'originally',
 'captured',
 'by',
 'a',
 'Utah',
 'man',
 'Ryan',
 'Cragun',
 'recently',
 ',',
 'has',
 'left',
 'people',
 'on',
 'Twitter',
 'confused',
 ',',
 'with',
 'many',
 'failing',
 'to',
 'spot',
 'the',
 'animal',
 'and',
 'calling',
 'it',
```

```
 'fake',
 '.',
 'Snow',
 'leopard',
 'doing',
 'a',
 'descent',
 ',',
 'Cragun',
 'wrote',
 'while',
 'sharing',
 'the',
 'image',
 'tagging',
 'a',
 'nature',
 'enthusiast',
 '.']
```

```
# Type and number of tokens
type(snow_lepard_tokens), len(snow_lepard_tokens)
```

```
(list, 65)
```

```
# Frequency of tokens
from nltk.probability import FreqDist
fdist=FreqDist()
```

```
for i in snow_lepard_tokens:
    fdist[i]=fdist[i]+1
fdist
```

```
FreqDist({',': 4,
          '.': 3,
          'Cragun': 2,
          'Ryan': 1,
          'Snow': 1,
          'The': 2,
          'Twitter': 1,
          'Utah': 1,
          'a': 4,
          'and': 2,
          'animal': 1,
          'by': 1,
          'calling': 1,
          'captured': 1,
          'confused': 1,
          'descent': 1,
          'doing': 1,
          'enthusiast': 1,
          'failing': 1,
          'fake': 1,
          'has': 1,
          'hiding': 1,
          'image': 3,
          'in': 1,
```

```
 'it': 1,
 'left': 1,
 'leopard': 1,
 'man': 1,
 'many': 1,
 'mountainous': 1,
 'nature': 1,
 'on': 1,
 'originally': 1,
 'people': 1,
 'recently': 1,
 'region': 1,
 'rocky': 1,
 'sharing': 1,
 'shows': 1,
 'snow': 1,
 'somewhere': 1,
 'spot': 1,
 'tagging': 1,
 'the': 3,
 'to': 1,
 'viral': 1,
 'while': 1,
 'wildcat': 1,
 'with': 1,
 'wrote': 1})
```

```
top_10 =fdist.most_common(10)
top_10
```

```
[('a', 4),
 (',', 4),
 ('image', 3),
 ('the', 3),
 ('.', 3),
 ('The', 2),
 ('and', 2),
 ('Cragun', 2),
 ('viral', 1),
 ('shows', 1)]
```

```
# Unigram Bigrams trigrams and ngrams
Afghanistan = 'Calling Taliban the current reality in Afghanistan, Russia said Wednesc
```

```
Afghanistan_tokens = word_tokenize(Afghanistan)
Afghanistan_tokens
```

```
 'Afghanistan',
 ',',
 'Russia',
 'said',
 'Wednesday',
 'its',
 'for',
 'India',
 'to',
 'decide',
 'to'
```

```
 'to',
'what',
'extent',
'it',
'wants',
'to',
'be',
'involved',
'in',
'the',
'country',
'.',
'Responding',
'to',
'queries',
'on',
'the',
'current',
'security',
'situation',
'in',
'Afghanistan',
',',
'the',
'Russian',
'embassy',
'here',
'said',
'that',
'after',
'the',
'US',
'withdrawal',
'from',
'the',
'war-torn',
'country',
'there',
'can',
'be',
'no',
'military',
'intervention',
'by',
'any',
'foreign',
'power',
'.']
```

```
1  list(nltk.bigrams(Afghanistan_tokens))
```

```
 ('in', 'Afghanistan'),
 ('Afghanistan', ','),
 (',', 'Russia'),

 ('Russia', 'said'),
 ('said', 'Wednesday'),
 ('Wednesday', 'its'),
 ('its', 'for'),
 ('for', 'India'),
 ('India', 'to'),
```

```
 ('India', 'to'),
 ('to', 'decide'),
 ('decide', 'to'),
 ('to', 'what'),
 ('what', 'extent'),
 ('extent', 'it'),
 ('it', 'wants'),
 ('wants', 'to'),
 ('to', 'be'),
 ('be', 'involved'),
 ('involved', 'in'),
 ('in', 'the'),
 ('the', 'country'),
 ('country', '.'),
 ('.', 'Responding'),
 ('Responding', 'to'),
 ('to', 'queries'),
 ('queries', 'on'),
 ('on', 'the'),
 ('the', 'current'),
 ('current', 'security'),
 ('security', 'situation'),
 ('situation', 'in'),
 ('in', 'Afghanistan'),
 ('Afghanistan', ','),
 (',', 'the'),
 ('the', 'Russian'),
 ('Russian', 'embassy'),
 ('embassy', 'here'),
 ('here', 'said'),
 ('said', 'that'),
 ('that', 'after'),
 ('after', 'the'),
 ('the', 'US'),
 ('US', 'withdrawal'),
 ('withdrawal', 'from'),
 ('from', 'the'),
 ('the', 'war-torn'),
 ('war-torn', 'country'),
 ('country', 'there'),
 ('there', 'can'),
 ('can', 'be'),
 ('be', 'no'),
 ('no', 'military'),
 ('military', 'intervention'),
 ('intervention', 'by'),
 ('by', 'any'),
 ('any', 'foreign'),
 ('foreign', 'power'),
 ('power', '.')]
```

```
1 list(nltk.trigrams(Afghanistan_tokens))
```

```
 ('reality', 'in', 'Afghanistan'),
 ('in', 'Afghanistan', ','),
 ('Afghanistan', ',', 'Russia'),
 (',', 'Russia', 'said'),
 ('Russia', 'said', 'Wednesday'),
 ('said', 'Wednesday', 'its'),
 ('Wednesday', 'its', 'for'),
 ('its', 'for', 'India')
```

```
 ('its', 'for', 'India'),
 ('for', 'India', 'to'),
 ('India', 'to', 'decide'),
 ('to', 'decide', 'to'),
 ('decide', 'to', 'what'),
 ('to', 'what', 'extent'),
 ('what', 'extent', 'it'),
 ('extent', 'it', 'wants'),
 ('it', 'wants', 'to'),
 ('wants', 'to', 'be'),
 ('to', 'be', 'involved'),
 ('be', 'involved', 'in'),
 ('involved', 'in', 'the'),
 ('in', 'the', 'country'),
 ('the', 'country', '.'),
 ('country', '.', 'Responding'),
 ('.', 'Responding', 'to'),
 ('Responding', 'to', 'queries'),
 ('to', 'queries', 'on'),
 ('queries', 'on', 'the'),
 ('on', 'the', 'current'),
 ('the', 'current', 'security'),
 ('current', 'security', 'situation'),
 ('security', 'situation', 'in'),
 ('situation', 'in', 'Afghanistan'),
 ('in', 'Afghanistan', ','),
 ('Afghanistan', ',', 'the'),
 (',', 'the', 'Russian'),
 ('the', 'Russian', 'embassy'),
 ('Russian', 'embassy', 'here'),
 ('embassy', 'here', 'said'),
 ('here', 'said', 'that'),
 ('said', 'that', 'after'),
 ('that', 'after', 'the'),
 ('after', 'the', 'US'),
 ('the', 'US', 'withdrawal'),
 ('US', 'withdrawal', 'from'),
 ('withdrawal', 'from', 'the'),
 ('from', 'the', 'war-torn'),
 ('the', 'war-torn', 'country'),
 ('war-torn', 'country', 'there'),
 ('country', 'there', 'can'),
 ('there', 'can', 'be'),
 ('can', 'be', 'no'),
 ('be', 'no', 'military'),
 ('no', 'military', 'intervention'),
 ('military', 'intervention', 'by'),
 ('intervention', 'by', 'any'),
 ('by', 'any', 'foreign'),
 ('any', 'foreign', 'power'),
 ('foreign', 'power', '.')]
```

```
1 list(nltk.ngrams(Afghanistan_tokens,5))
```

```
 ('the', 'current', 'reality', 'in', 'Afghanistan'),
 ('current', 'reality', 'in', 'Afghanistan', ','),
 ('reality', 'in', 'Afghanistan', ',', 'Russia'),
 ('in', 'Afghanistan', ',', 'Russia', 'said'),
 ('Afghanistan', ',', 'Russia', 'said', 'Wednesday'),
 (',', 'Russia', 'said', 'Wednesday', 'its'),
 ('Russia', 'said', 'Wednesday', 'its', 'for'),
```

```
 ('Russia', 'said', 'Wednesday', 'its', 'for'),
 ('said', 'Wednesday', 'its', 'for', 'India'),
 ('Wednesday', 'its', 'for', 'India', 'to'),
 ('its', 'for', 'India', 'to', 'decide'),
 ('for', 'India', 'to', 'decide', 'to'),
 ('India', 'to', 'decide', 'to', 'what'),
 ('to', 'decide', 'to', 'what', 'extent'),
 ('decide', 'to', 'what', 'extent', 'it'),
 ('to', 'what', 'extent', 'it', 'wants'),
 ('what', 'extent', 'it', 'wants', 'to'),
 ('extent', 'it', 'wants', 'to', 'be'),
 ('it', 'wants', 'to', 'be', 'involved'),
 ('wants', 'to', 'be', 'involved', 'in'),
 ('to', 'be', 'involved', 'in', 'the'),
 ('be', 'involved', 'in', 'the', 'country'),
 ('involved', 'in', 'the', 'country', '.'),
 ('in', 'the', 'country', '.', 'Responding'),
 ('the', 'country', '.', 'Responding', 'to'),
 ('country', '.', 'Responding', 'to', 'queries'),
 ('.', 'Responding', 'to', 'queries', 'on'),
 ('Responding', 'to', 'queries', 'on', 'the'),
 ('to', 'queries', 'on', 'the', 'current'),
 ('queries', 'on', 'the', 'current', 'security'),
 ('on', 'the', 'current', 'security', 'situation'),
 ('the', 'current', 'security', 'situation', 'in'),
 ('current', 'security', 'situation', 'in', 'Afghanistan'),
 ('security', 'situation', 'in', 'Afghanistan', ','),
 ('situation', 'in', 'Afghanistan', ',', 'the'),
 ('in', 'Afghanistan', ',', 'the', 'Russian'),
 ('Afghanistan', ',', 'the', 'Russian', 'embassy'),
 (',', 'the', 'Russian', 'embassy', 'here'),
 ('the', 'Russian', 'embassy', 'here', 'said'),
 ('Russian', 'embassy', 'here', 'said', 'that'),
 ('embassy', 'here', 'said', 'that', 'after'),
 ('here', 'said', 'that', 'after', 'the'),
 ('said', 'that', 'after', 'the', 'US'),
 ('that', 'after', 'the', 'US', 'withdrawal'),
 ('after', 'the', 'US', 'withdrawal', 'from'),
 ('the', 'US', 'withdrawal', 'from', 'the'),
 ('US', 'withdrawal', 'from', 'the', 'war-torn'),
 ('withdrawal', 'from', 'the', 'war-torn', 'country'),
 ('from', 'the', 'war-torn', 'country', 'there'),
 ('the', 'war-torn', 'country', 'there', 'can'),
 ('war-torn', 'country', 'there', 'can', 'be'),
 ('country', 'there', 'can', 'be', 'no'),
 ('there', 'can', 'be', 'no', 'military'),
 ('can', 'be', 'no', 'military', 'intervention'),
 ('be', 'no', 'military', 'intervention', 'by'),
 ('no', 'military', 'intervention', 'by', 'any'),
 ('military', 'intervention', 'by', 'any', 'foreign'),
 ('intervention', 'by', 'any', 'foreign', 'power'),
 ('by', 'any', 'foreign', 'power', '.')]
```

```
# Stemming
from nltk.stem import PorterStemmer
pst = PorterStemmer()
pst.stem('winning'), pst.stem('studies'), pst.stem('buying')
```

```
('win', 'studi', 'buy')
```

```
#Lemmatization
nltk.download('wordnet')
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

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

```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Unzipping corpora/omw-1.4.zip.
True
```

```
word_to_stem = ['studies', 'cats', 'cacti', 'geese', 'typeing']
for i in word_to_stem:
    print(i +':' + lemmatizer.lemmatize(i))
```

```
studies:study
cats:cat
cacti:cactus
geese:goose
typeing:typeing
```

```
# parts of speach (POS)
thuglife = 'At this point, Rahul is no longer interested in trying to keep people from

thuglife_tokens = word_tokenize(thuglife)
thuglife_tokens
```

```
['At',
 'this',
 'point',
 ',',
 'Rahul',
 'is',
 'no',
 'longer',
 'interested',
 'in',
 'trying',
 'to',
 'keep',
 'people',
 'from',
 'leaving',
 '.',
 'If',
 'you',
 'want',
 'to',
 'go',
 'then',
```

```
 'go',
 '.',
 'Now',
 'I',
 'only',
 'have',
 'space',
 'in',
 'my',
 'life',
 'for',
 'people',
 'who',
 'want',
 'to',
 'stay',
 'and',
 'for',
 'people',
 'who',
 'want',
 'to',
 'be',
 'with',
 'me']
```

```
nltk.download('averaged_perceptron_tagger')
for i in thuglife_tokens:
    print(nltk.pos_tag([i]))
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('At', 'IN')]
[('this', 'DT')]
[('point', 'NN')]
[(',', ',')]
[('Rahul', 'NN')]
[('is', 'VBZ')]
[('no', 'DT')]
[('longer', 'NN')]
[('interested', 'JJ')]
[('in', 'IN')]
[('trying', 'VBG')]
[('to', 'TO')]
[('keep', 'VB')]
[('people', 'NNS')]
[('from', 'IN')]
[('leaving', 'VBG')]
[('.', '.')]
[('If', 'IN')]
[('you', 'PRP')]
[('want', 'NN')]
[('to', 'TO')]
[('go', 'VB')]
[('then', 'RB')]
[('go', 'VB')]
[('.', '.')]
[('Now', 'RB')]
```

```
[('I', 'PRP')]
[('only', 'RB')]
[('have', 'VB')]
[('space', 'NN')]
[('in', 'IN')]
[('my', 'PRP$')]
[('life', 'NN')]
[('for', 'IN')]
[('people', 'NNS')]
[('who', 'WP')]
[('want', 'NN')]
[('to', 'TO')]
[('stay', 'NN')]
[('and', 'CC')]
[('for', 'IN')]
[('people', 'NNS')]
[('who', 'WP')]
[('want', 'NN')]
[('to', 'TO')]
[('be', 'VB')]
[('with', 'IN')]
[('me', 'PRP')]
```

```
# named entity recongnition
nltk.download('maxent_ne_chunker')
from nltk import ne_chunk
Rachana = 'HCL bought United Kingdom based Axon for 635 billion'
Rachana_tokens = word_tokenize(Rachana)
Rachana_tokens
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
['HCL',
 'bought',
 'United',
 'Kingdom',
 'based',
 'Axon',
 'for',
 '635',
 'billion']
```

```
Rachana_tags = nltk.pos_tag(Rachana_tokens)
Rachana_tags
```

```
[('HCL', 'NNP'),
 ('bought', 'VBD'),
 ('United', 'NNP'),
 ('Kingdom', 'NNP'),
 ('based', 'VBN'),
 ('Axon', 'NNP'),
 ('for', 'IN'),
 ('635', 'CD'),
 ('billion', 'CD')]
```

```
nltk.download('words')
Rachana_ner = ne_chunk(Rachana_tags)
print(Rachana_ner)
```

```
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
(S
  (ORGANIZATION HCL/NNP)
  bought/VBD
  (GPE United/NNP Kingdom/NNP)
  based/VBN
  (PERSON Axon/NNP)
  for/IN
  635/CD
  billion/CD)
```

```
import spacy
spacy.prefer_gpu()
nlp = spacy.load('en_core_web_sm')
```

```
# creating documents
doc = nlp("This is the first batch of Prime Intuit")
```

```
#tokensisation
for token in doc :
    print(token.text)
```

```
This
is
the
first
batch
of
Prime
Intuit
```

```
token = doc[2]
token
```

```
the
```

```
span = doc[2:5]
span
```

```
the first batch
```

```
doc = nlp('We are an excellent team')
```

```
for token in doc:
    print(token.text)
```

```
We
are
```

```
an
excellent
team
```

```
doc = nlp("Sidhath does not share icecream")
doc
```

```
Sidhath does not share icecream
```

```
for token in doc:
    print(token.text)
```

```
Sidhath
does
not
share
icecream
```

```
# Part of speach using spaCy

for token in doc:
    print (token.i, token.text, token.pos_)
```

```
0 Sidhath PROPN
1 does AUX
2 not PART
3 share VERB
4 icecream NOUN
```

```
doc = nlp(" HCL bought a UK based consulting firm Axon for 658 million")
```

```
for ent in doc.ents:
    print (ent.text, ent.label_)
```

```
HCL ORG
UK GPE
658 million CARDINAL
```

```
doc = nlp('Barack Obama the former president of United States will be vacating white h
```

```
for ent in doc.ents:
    print (ent.text, ent.label_)
```

```
United States GPE
today DATE
```

```
#Matcher function
from spacy.matcher import Matcher
```

```
pattern =[
```

```
    {
        'LEMMA':'vacate'
    },
    {
        'ORTH':'white'
    }
]
print(pattern)
```

```
[{'LEMMA': 'vacate'}, {'ORTH': 'white'}]
```

```
matcher = Matcher(nlp.vocab)
matcher.add('white_Pattern', [pattern])
matches = matcher(doc)
print(matches)
```

```
[(4191279314630736679, 10, 12)]
```

```
for match_id, start,end in matches:
    matched_span = doc[start:end]
    print(matched_span.text)
```

```
vacating white
```

```
doc = nlp("2020 European Championship cup : Italy won !!")
```

```
  pattern = [{'IS_DIGIT':True},{'LOWER':'european'},{'LOWER':'championship'},{'LOWER':'
  matcher2 = Matcher(nlp.vocab)
  matcher.add('euro_pattern',[pattern])
  matches = matcher(doc)
  print(matches)
```

```
[(2749811298878778488, 0, 4)]
```

```
  for match_id, start,end in matches:
      matched_span = doc[start:end]
      print(matched_span.text)
```

```
2020 European Championship cup
```