```
data = """o
Sr. NO 123  - XYZ Data
XYZ Data  is a no-code and highly intuitive data pipeline platform. was founded by Mr. Som

The built-in CDC for database sources allows you to get up-to-date data from any database
The current CEO is Mr. Pankaj Gupta
"""
```

```
import nltk
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import spacy
from spacy import displacy
nltk.download('maxent_ne_chunker')
from nltk import word_tokenize
nltk.download('punkt')
from nltk.probability import FreqDist
from nltk.util import bigrams, trigrams, ngrams
nltk.download('wordnet')
from nltk.stem import wordnet
nltk.download('words')
from nltk.corpus import stopwords
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

Statistics

```
data_tokens = word_tokenize(data)                        #Converting the dataset into sing
data_tokens
```

```
 'Services',
 '.',
 'You',
 'can',
         'also',
         'customized',
         'your',
         'data',
         'connections',
         'using',
         'Rest',
         'API',
         'and',
         'Webhook',
         'integration',
         'to',
         'interact',
         'with',
         'all',
         'kinds',
         'of',
         'applications',
         '.',
         'The',
         'built-in',
         'CDC',
         'for',
         'database',
         'sources',
         'allows',
         'you',
         'to',
         'get',
         'up-to-date',
         'data',
         'from',
         'any',
         'database',
         'to',
         'your',
         'destination',
         'in',
         'real-time',
         'without',
         'putting',
         'additional',
         'load',
         'on',
         'your',
         'database',
         '.',
         'The',
         'current',
         'CEO',
         'is',
         'Mr.',
         'Pankaj',
         'Gupta']
```

```
fdist = FreqDist()
```

```
for word in data_tokens:
  fdist[word.lower()]+=1                   #FreqDist is a dict with tokens as keys and the
fdist
```

```
             'customized': 1,
             'data': 6,
             'database': 3,
             'databases': 1,
             'destination': 1,
             'for': 1,
             'founded': 1,
             'from': 2,
             'get': 1,
             'gupta': 1,
             'highly': 1,
             'in': 2,
             'integration': 1,
             'integrations': 1,
             'interact': 1,
             'intuitive': 1,
             'is': 2,
             'it': 1,
             'kinds': 1,
             'load': 2,
             'minutes': 1,
             'mr.': 2,
             'no': 1,
             'no-code': 1,
             'o': 1,
             'of': 1,
             'on': 1,
             'pankaj': 1,
             'pipeline': 1,
             'platform': 1,
             'provides': 1,
             'putting': 1,
             'ready-to-use': 1,
             'real-time': 1,
             'reliably': 1,
             'rest': 1,
             'saas': 1,
             'sdks': 1,
             'services': 1,
             'sharma': 1,
             'somnath': 1,
             'sources': 2,
             'sr.': 1,
             'storage': 1,
             'streaming': 1,
             'the': 3,
             'to': 4,
             'up-to-date': 1,
             'using': 1,
              'warehouse': 1,
              'was': 1,
              'webhook': 1,
              'with': 1,
              'without': 1,
              'xyz': 2,
              'you': 3,
              'your': 4})
```

```
len(fdist)
```

```
80
```

This implies we have 80 distinct features in the given data.

```
fdist['mr.']
```

```
2
```

```
fdist_top20 = fdist.most_common(20)                      #To extract the top 20 occuring f
fdist_top20
```

```
[('data', 6),
 ('.', 6),
 ('your', 4),
 ('to', 4),
 (',', 4),
 ('and', 3),
 ('you', 3),
 ('the', 3),
 ('database', 3),
 ('xyz', 2),
 ('is', 2),
 ('mr.', 2),
 ('can', 2),
 ('load', 2),
 ('from', 2),
 ('all', 2),
 ('sources', 2),
 ('in', 2),
 ('applications', 2),
 ('o', 1)]
```

```
#Plot of features
fig, ax = plt.subplots(figsize=(10,10))
all_fdist = pd.Series(dict(fdist_top20))
pplot = sns.barplot(x=all_fdist.index, y=all_fdist.values, ax=ax)
plt.xticks(rotation=30);
```

n-gram is a contiguous sequence of n items generated from a given sample of text where the items can be characters or words and n can be any numbers like 1,2,3, etc.

```
data_bigrams = list(nltk.bigrams(data_tokens))
data_bigrams
```

```
    ('Streaming', 'Services'),
    ('Services', '.'),
    ('.', 'You'),
    ('You', 'can'),
    ('can', 'also'),
    ('also', 'customized'),
    ('customized', 'your'),
    ('your', 'data'),
    ('data', 'connections'),
    ('connections', 'using'),
    ('using', 'Rest'),
    ('Rest', 'API'),
    ('API', 'and'),
    ('and', 'Webhook'),
 ('Webhook', 'integration'),
 ('integration', 'to'),
 ('to', 'interact'),
 ('interact', 'with'),
 ('with', 'all'),
 ('all', 'kinds'),
 ('kinds', 'of'),
 ('of', 'applications'),
 ('applications', '.'),
 ('.', 'The'),
 ('The', 'built-in'),
 ('built-in', 'CDC'),
 ('CDC', 'for'),
 ('for', 'database'),
 ('database', 'sources'),
 ('sources', 'allows'),
 ('allows', 'you'),
 ('you', 'to'),
 ('to', 'get'),
 ('get', 'up-to-date'),
 ('up-to-date', 'data'),
 ('data', 'from'),
 ('from', 'any'),
 ('any', 'database'),
 ('database', 'to'),
 ('to', 'your'),
 ('your', 'destination'),
 ('destination', 'in'),
 ('in', 'real-time'),
 ('real-time', 'without'),
 ('without', 'putting'),
 ('putting', 'additional'),
 ('additional', 'load'),
 ('load', 'on'),
 ('on', 'your'),
 ('your', 'database'),
 ('database', '.'),
 ('.', 'The'),
 ('The', 'current'),
 ('current', 'CEO'),
 ('CEO', 'is'),
 ('is', 'Mr.'),
 ('Mr.', 'Pankaj'),
 ('Pankaj', 'Gupta')]
```

```
data_trigrams = list(nltk.trigrams(data_tokens ))
data_trigrams
```

```
 ('and', 'Streaming', 'Services'),
 ('Streaming', 'Services', '.'),
 ('Services', '.', 'You'),
 ('.', 'You', 'can'),
 ('You', 'can', 'also'),
 ('can', 'also', 'customized'),
 ('also', 'customized', 'your'),
 ('customized', 'your', 'data'),
  ('your', 'data', 'connections'),
  ('data', 'connections', 'using'),
  ('connections', 'using', 'Rest'),
  ('using', 'Rest', 'API'),
  ('Rest', 'API', 'and'),
  ('API', 'and', 'Webhook'),
  ('and', 'Webhook', 'integration'),
  ('Webhook', 'integration', 'to'),
  ('integration', 'to', 'interact'),
  ('to', 'interact', 'with'),
  ('interact', 'with', 'all'),
  ('with', 'all', 'kinds'),
  ('all', 'kinds', 'of'),
  ('kinds', 'of', 'applications'),
  ('of', 'applications', '.'),
  ('applications', '.', 'The'),
  ('.', 'The', 'built-in'),
  ('The', 'built-in', 'CDC'),
  ('built-in', 'CDC', 'for'),
  ('CDC', 'for', 'database'),
  ('for', 'database', 'sources'),
  ('database', 'sources', 'allows'),
  ('sources', 'allows', 'you'),
  ('allows', 'you', 'to'),
  ('you', 'to', 'get'),
  ('to', 'get', 'up-to-date'),
  ('get', 'up-to-date', 'data'),
  ('up-to-date', 'data', 'from'),
  ('data', 'from', 'any'),
  ('from', 'any', 'database'),
  ('any', 'database', 'to'),
  ('database', 'to', 'your'),
  ('to', 'your', 'destination'),
  ('your', 'destination', 'in'),
  ('destination', 'in', 'real-time'),
  ('in', 'real-time', 'without'),
  ('real-time', 'without', 'putting'),
  ('without', 'putting', 'additional'),
  ('putting', 'additional', 'load'),
  ('additional', 'load', 'on'),
  ('load', 'on', 'your'),
  ('on', 'your', 'database'),
  ('your', 'database', '.'),
  ('database', '.', 'The'),
  ('.', 'The', 'current'),
  ('The', 'current', 'CEO'),
  ('current', 'CEO', 'is'),
  ('CEO', 'is', 'Mr.'),
  ('is', 'Mr.', 'Pankaj'),
  ('Mr.', 'Pankaj', 'Gupta')]
```

```
data_ngrams = list(nltk.ngrams(data_tokens, 5 ))
data_ngrams
```

```
  ('SDKs', ',', 'and', 'Streaming', 'Services'),
  (',', 'and', 'Streaming', 'Services', '.'),
('and', 'Streaming', 'Services', '.', 'You'),
('Streaming', 'Services', '.', 'You', 'can'),
('Services', '.', 'You', 'can', 'also'),
('.', 'You', 'can', 'also', 'customized'),
('You', 'can', 'also', 'customized', 'your'),
('can', 'also', 'customized', 'your', 'data'),
('also', 'customized', 'your', 'data', 'connections'),
('customized', 'your', 'data', 'connections', 'using'),
('your', 'data', 'connections', 'using', 'Rest'),
('data', 'connections', 'using', 'Rest', 'API'),
('connections', 'using', 'Rest', 'API', 'and'),
('using', 'Rest', 'API', 'and', 'Webhook'),
('Rest', 'API', 'and', 'Webhook', 'integration'),
('API', 'and', 'Webhook', 'integration', 'to'),
('and', 'Webhook', 'integration', 'to', 'interact'),
('Webhook', 'integration', 'to', 'interact', 'with'),
('integration', 'to', 'interact', 'with', 'all'),
('to', 'interact', 'with', 'all', 'kinds'),
('interact', 'with', 'all', 'kinds', 'of'),
('with', 'all', 'kinds', 'of', 'applications'),
('all', 'kinds', 'of', 'applications', '.'),
('kinds', 'of', 'applications', '.', 'The'),
('of', 'applications', '.', 'The', 'built-in'),
('applications', '.', 'The', 'built-in', 'CDC'),
('.', 'The', 'built-in', 'CDC', 'for'),
('The', 'built-in', 'CDC', 'for', 'database'),
('built-in', 'CDC', 'for', 'database', 'sources'),
('CDC', 'for', 'database', 'sources', 'allows'),
('for', 'database', 'sources', 'allows', 'you'),
('database', 'sources', 'allows', 'you', 'to'),
('sources', 'allows', 'you', 'to', 'get'),
('allows', 'you', 'to', 'get', 'up-to-date'),
('you', 'to', 'get', 'up-to-date', 'data'),
('to', 'get', 'up-to-date', 'data', 'from'),
('get', 'up-to-date', 'data', 'from', 'any'),
('up-to-date', 'data', 'from', 'any', 'database'),
('data', 'from', 'any', 'database', 'to'),
('from', 'any', 'database', 'to', 'your'),
('any', 'database', 'to', 'your', 'destination'),
('database', 'to', 'your', 'destination', 'in'),
('to', 'your', 'destination', 'in', 'real-time'),
('your', 'destination', 'in', 'real-time', 'without'),
('destination', 'in', 'real-time', 'without', 'putting'),
('in', 'real-time', 'without', 'putting', 'additional'),
('real-time', 'without', 'putting', 'additional', 'load'),
('without', 'putting', 'additional', 'load', 'on'),
('putting', 'additional', 'load', 'on', 'your'),
('additional', 'load', 'on', 'your', 'database'),
('load', 'on', 'your', 'database', '.'),
('on', 'your', 'database', '.', 'The'),
('your', 'database', '.', 'The', 'current'),
('database', '.', 'The', 'current', 'CEO'),
('.', 'The', 'current', 'CEO', 'is'),
('The', 'current', 'CEO', 'is', 'Mr.'),
('current', 'CEO', 'is', 'Mr.', 'Pankaj'),
('CEO', 'is', 'Mr.', 'Pankaj', 'Gupta')]
```

Chunking is used to identify parts of speech and short phrases present in a given sentence. The different parts of speech present in the given data are as follows:

```
from nltk import ne_chunk
data_tokens = word_tokenize(data)
data_tags = nltk.pos_tag(data_tokens)
data_chunks = ne_chunk(data_tags)
print(data_chunks)
```

```
(ORGANIZATION SDKs/NNP)
,/,
and/CC
(ORGANIZATION Streaming/NNP Services/NNPS)
./.
You/PRP
can/MD
also/RB
customized/VB
your/PRP$
data/NNS
connections/NNS
using/VBG
(PERSON Rest/NNP API/NNP)
and/CC
(GPE Webhook/NNP)
integration/NN
to/TO
interact/VB
with/IN
all/DT
kinds/NNS
of/IN
applications/NNS
./.
The/DT
built-in/JJ
(ORGANIZATION CDC/NNP)
for/IN
database/NN
sources/NNS
allows/VBZ
you/PRP
to/TO
get/VB
up-to-date/JJ
data/NNS
from/IN
any/DT
database/NN
to/TO
your/PRP$
destination/NN
in/IN
real-time/NN
```

```
  real-time/NN
  without/IN
  putting/VBG
  additional/JJ
  load/NN
  on/IN
  your/PRP$
  database/NN
  ./.
  The/DT
  current/JJ
  (ORGANIZATION CEO/NNP)
  is/VBZ
  (PERSON Mr./NNP Pankaj/NNP Gupta/NNP))
```

Method 1: From Scratch

```python
import regex as re
```

```python
opt = re.sub(r'[.|,]','', data)                    #Removing Punctuations
opt = re.sub(r'[-|\n]', '', opt)                   #Removing new line characters
opt = re.split(' ', opt)                           #Splitting the words
print(opt)
```

```
['oSr', 'NO', '123', '', '', 'XYZ', 'DataXYZ', 'Data', '', 'is', 'a', 'nocode', 'and',
```

```python
serial_nos = []
for word in opt:
    if (word.isdigit()) :                          #To check if the any word is numeric or no
        serial_nos.append(word)
print(f"The serial number is: {serial_nos[0]}")
```

```
The serial number is: 123
```

```python
names = []
i = 0
for word in opt:
    if (word == 'Mr'):
        names.append(opt[i+1] +' ' + opt[i+2])
    i = i+1
print(names)
```

```
['Somnath Sharma', 'Pankaj Gupta']
```

So, we can extract the names and the serial number from scratch but not the job designation.

Approach 1 : Using NLTK

```
data_tokens = nltk.word_tokenize(data)                       #Spillting the data into tokens
data_tags = nltk.pos_tag(data_tokens)                        #Part of Speech Tagging
data_chunks = nltk.ne_chunk(data_tags, binary = True)        #whether NE or not NE
```

```
for chunk in data_chunks:
  print(chunk)
```

```
('o', 'JJ')
('Sr.', 'NNP')
('NO', 'NNP')
('123', 'CD')
('-', ':')
(NE XYZ/NN Data/NNP)
('XYZ', 'NNP')
('Data', 'NNP')
('is', 'VBZ')
('a', 'DT')
('no-code', 'JJ')
('and', 'CC')
('highly', 'RB')
('intuitive', 'JJ')
('data', 'NNS')
('pipeline', 'NN')
('platform', 'NN')
('.', '.')
('was', 'VBD')
('founded', 'VBN')
('by', 'IN')
(NE Mr./NNP Somnath/NNP Sharma/NNP)
('.', '.')
('You', 'PRP')
('can', 'MD')
('reliably', 'VB')
('load', 'NN')
('data', 'NNS')
('from', 'IN')
('all', 'DT')
('your', 'PRP$')
('sources', 'NNS')
('to', 'TO')
('the', 'DT')
('warehouse', 'NN')
('in', 'IN')
('minutes', 'NNS')
('.', '.')
('It', 'PRP')
('provides', 'VBZ')
('100+', 'CD')
('ready-to-use', 'NN')
```

```
('integrations', 'NNS')
('across', 'IN')
(NE Databases/NNP)
(',', ',')
(NE SaaS/NNP Applications/NNP)
(',', ',')
(NE Cloud/NNP Storage/NNP)
(',', ',')
(NE SDKs/NNP)
(',', ',')
('and', 'CC')
('Streaming', 'NNP')
('Services', 'NNPS')
('.', '.')
('You', 'PRP')
('can', 'MD')
```

Created a dataframe for only the named entities recognized by nltk

```
entities = []
labels = []
for chunk in data_chunks:
  if hasattr(chunk,'label'):                          #to check if the has the correct giv
    entities.append(' '.join(c[0] for c in chunk))
    labels.append(chunk.label())

entities_labels = list(set(zip(entities, labels)))
entities_df = pd.DataFrame(entities_labels)
entities_df.columns = ['Entities', 'labels']
entities_df
```

Next is another dataframe with entities being organization, person and geopolitical entity. We observe that it has plenty of errors such as 'databases' considered under person and so on.

```
data_chunks = nltk.ne_chunk(data_tags, binary = False)      #either NE or not NE
entities = []
labels = []
for chunk in data_chunks:
  if hasattr(chunk,'label'):
    entities.append(' '.join(c[0] for c in chunk))
    labels.append(chunk.label())

entities_labels = list(set(zip(entities, labels)))
entities_df = pd.DataFrame(entities_labels)
entities_df.columns = ['Entities', 'labels']
entities_df
```

```
entities = []
labels = []

sentence = nltk.sent_tokenize(data)
for sent in sentence:
  for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent)), binary = False):
    if hasattr(chunk, 'label'):
      entities.append(' '.join(c[0] for c in chunk))
      labels.append(chunk.label())

```

```
entities_labels = list(set(zip(entities, labels)))
entities_df = pd.DataFrame(entities_labels)
entities_df.columns = ['Entities','Labels']
entities_df
```

In our last dataframe, we observe the people being classified rightly with their job designations being under 'organizatio' entity. This is moderately good as it still has errors which we will try to remove with spacy.

Approach 2: Using Spacy

```
spacy.__version__
```

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

```
data = nlp(data)
entities = []
labels = []
pos_start = []
pos_end = []
```

```

for ent in data.ents:
  entities.append(ent)
  labels.append(ent.label_)
  pos_start.append(ent.start_char)
  pos_end.append(ent.end_char)

df = pd.DataFrame({'Entities':entities, 'Labels':labels, 'Position Start':pos_start
                   , 'Position End': pos_end})
df
```

In spacy, we keep a track of the entities' startig and ending positions aswell. We get both the cardinals as preferred and the people correctly classified aswell along with its designation. XYZ Data beig classified under Organization whu=ich was not the case for nltk.

```
df[df["Labels"]=="PERSON"]
```

For visualizing the entities in data:

```
displacy.serve(data,style = 'ent')
```

```
Shutting down server on port 5000.
```

```
spacy.displacy.serve(data, style = 'ent')
```

```
Using the 'ent' visualizer
Serving on http://0.0.0.0:5000 ...

Shutting down server on port 5000.
```

Stanford NER

```
pip install stanza
```

```
import stanza
```

```
data = """o
Sr. NO 123  - XYZ Data
XYZ Data  is a no-code and highly intuitive data pipeline platform. was founded by Mr. Som

The built-in CDC for database sources allows you to get up-to-date data from any database
The current CEO is Mr. Pankaj Gupta
"""
```

```
stanza.download('en')
nlp = stanza.Pipeline('en')
```

```
Downloading                                                           142k/?
https://raw.githubusercontent.com/stanfordnlp/stanza-                 [00:00<00:00,
resources/main/resources_1.3.0.json:                                  1.15MB/s]
2022-04-04 12:28:28 INFO: Downloading default packages for language: en (English)...
```

```
doc = nlp(data)
```

```
for ent in doc.entities:
  print(f"{ent.text} : {ent.type}")
```

```
123 : CARDINAL
Somnath Sharma : PERSON
minutes : TIME
100 : CARDINAL
Databases : ORG
SaaS Applications : PRODUCT
Cloud Storage : PRODUCT
SDKs : ORG
Streaming Services : ORG
Rest API : PRODUCT
Webhook : PRODUCT
CDC : ORG
Pankaj Gupta : PERSON
```

This a better estimation of the entities compared to the rest as we've gotten our cardinals, person and Organization(along with job designation) right on point.

✓ 0s completed at 8:01 PM