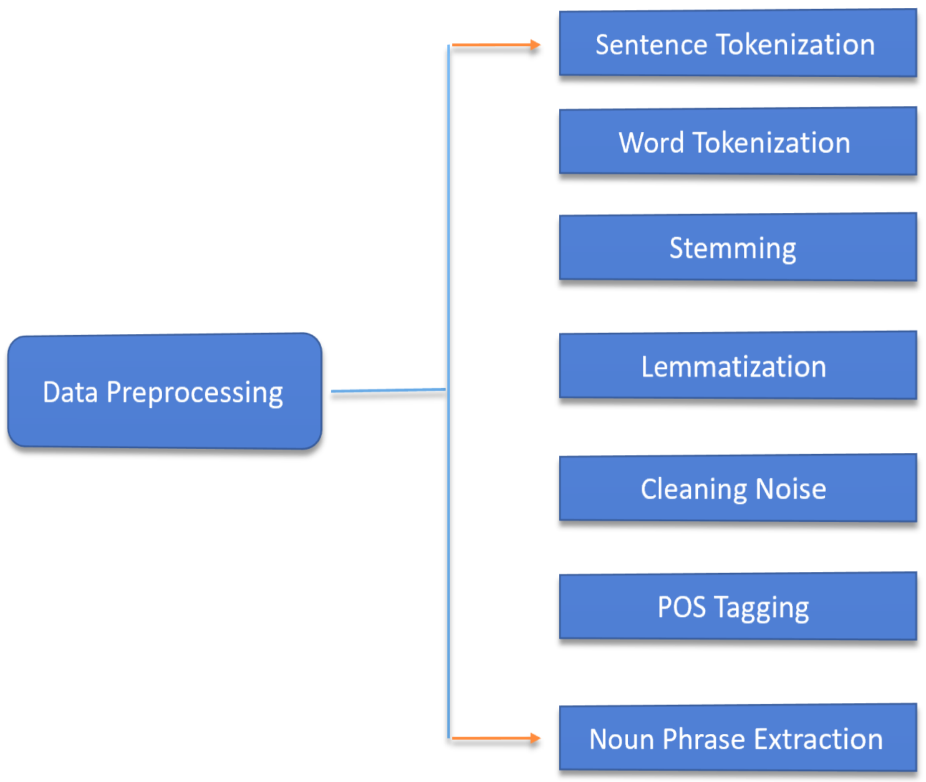
In this article we will explore most commonly researched tasks in natural language processing (NLP). Some of these tasks have direct real-world applications, while others more commonly serve as subtasks that are used to aid in solving larger tasks.



Every natural human readable data such as text data must go through data preprocessing steps as showed in the image,

All these task can be achieved in python using either of below tools,

- **The Natural Language Toolkit, or more commonly NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.
- **Spacy** an open-source software library for advanced natural language processing, written in the programming languages Python and Cython.

We are going to explore both tools for the mentioned tasks. So lets go ahead and install NLTK and Spacy first .

Command to install NLTK : **conda install -c anaconda nltk** OR **pip install nltk**

Command to install Spacy : **conda install -c conda-forge spacy** OR **pip install spacy**

Text Data for our usecases

> *Ramkumar Ramamoorthy was recently appointed as Chairman and Managing Director of tech major Cognizant’s India operations.Cognizant is US $61.65 comapny. With new CEO Brian Humphries taking charge at the global level, Cognizant has brought about changes to its strategy and operations. After leading industry growth over a little more than a decade, during which time it overtook Wipro and Infosys in revenue, Cognizant went through a period of low revenue growth. Now aiming to get its mojo back, the company is betting on talent in its India software delivery centres with their 2 lakh-plus headcount that services clients in the U.S., Europe and West Asia. Ramamoorthy also has views on the education system and what the country needs, to make the best of its demographic dividend.*

Text Data Preprocessing

A. Sentence Tokenization

Most common form of data in natural language processing is textual data. And when we look at a textual data in real time it is a group of sentences.
Senetence Tokenization is the process of splitting sentences from input text data. Lets see how we can achieve sentence tokenization using both NLTK and Spacy.

NLTK

sent_tokenize() splits text into sentences.Output is list of sentences.

In [113]:
```
from nltk.tokenize import sent_tokenize
data ='Ramkumar Ramamoorthy was recently appointed as Chairman and Managing
sentencesN_ = sent_tokenize(data)
print(sentencesN_)
```

```
['Ramkumar Ramamoorthy was recently appointed as Chairman and Managing Dire
ctor of tech major Cognizant’s India operations.Cognizant is US $61.65 coma
pny.', 'With new CEO Brian Humphries taking charge at the global level, Cog
nizant has brought about changes to its strategy and operations.', 'After l
eading industry growth over a little more than a decade, during which time
it overtook Wipro and Infosys in revenue, Cognizant went through a period o
f low revenue growth.', 'Now aiming to get its mojo back, the company is be
tting on talent in its India software delivery centres with their 2 lakh-pl
us headcount that services clients in the U.S., Europe and West Asia.', 'Ra
mamoorthy also has views on the education system and what the country need
s, to make the best of its demographic dividend.']
```

Spacy

In [115]:
```
import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp(data)
```

In [116]:
```
sentencesS_ = [sent for sent in doc.sents]
print(sentencesS_)
```

```
[Ramkumar Ramamoorthy was recently appointed as Chairman and Managing Direc
tor of tech major Cognizant’s India operations., Cognizant is US $61.65 com
apny., With new CEO Brian Humphries taking charge at the global level, Cogn
izant has brought about changes to its strategy and operations., After lead
ing industry growth over a little more than a decade, during which time it 
overtook Wipro and Infosys in revenue, Cognizant went through a period of l
ow revenue growth., Now aiming to get its mojo back, the company is betting
on talent in its India software delivery centres with their 2 lakh-plus hea
dcount that services clients in the U.S., Europe and West Asia., Ramamoorth
y also has views on the education system and what the country needs, to mak
e the best of its demographic dividend.]
```

B. Word Tokenization

Word tokenization is the process of splitting sentences/text into list of words.

NLTK

In [35]:
```
from nltk.tokenize import word_tokenize

print('Tokenizing actual text data....\n')
print(word_tokenize(data))

print('\n')
print('Tokenizing Each sentences from the tokenized sentence list....\n')
wordsN_ = [word_tokenize(sent) for sent in sentencesN_]
print(words_)
```

```
Tokenizing actual text data....

['Ramkumar', 'Ramamoorthy', 'was', 'recently', 'appointed', 'as', 'Cha
irman', 'and', 'Managing', 'Director', 'of', 'tech', 'major', 'Cogniza
nt', '’', 's', 'India', 'operations', '.', 'With', 'new', 'CEO', 'Bria
n', 'Humphries', 'taking', 'charge', 'at', 'the', 'global', 'level',
',', 'Cognizant', 'has', 'brought', 'about', 'changes', 'to', 'its',
'strategy', 'and', 'operations', '.', 'After', 'leading', 'industry',
'growth', 'over', 'a', 'little', 'more', 'than', 'a', 'decade', ',',
'during', 'which', 'time', 'it', 'overtook', 'Wipro', 'and', 'Infosy
s', 'in', 'revenue', ',', 'Cognizant', 'went', 'through', 'a', 'perio
d', 'of', 'low', 'revenue', 'growth', '.', 'Now', 'aiming', 'to', 'ge
t', 'its', 'mojo', 'back', ',', 'the', 'company', 'is', 'betting', 'o
n', 'talent', 'in', 'its', 'India', 'software', 'delivery', 'centres',
'with', 'their', '2', 'lakh-plus', 'headcount', 'that', 'services', 'c
lients', 'in', 'the', 'U.S.', ',', 'Europe', 'and', 'West', 'Asia.Rama
moorthy', 'also', 'has', 'views', 'on', 'the', 'education', 'system',
'and', 'what', 'the', 'country', 'needs', ',', 'to', 'make', 'the', 'b
est', 'of', 'its', 'demographic', 'dividend', '.']
```

Spacy

In [34]:
```
wordsS_ = [token.text for token in doc]
print(wordsS_)
```

```
['Ramkumar', 'Ramamoorthy', 'was', 'recently', 'appointed', 'as', 'Chairma
n', 'and', 'Managing', 'Director', 'of', 'tech', 'major', 'Cognizant',
'’s', 'India', 'operations', '.', 'With', 'new', 'CEO', 'Brian', 'Humphrie
s', 'taking', 'charge', 'at', 'the', 'global', 'level', ',', 'Cognizant',
'has', 'brought', 'about', 'changes', 'to', 'its', 'strategy', 'and', 'oper
ations', '.', 'After', 'leading', 'industry', 'growth', 'over', 'a', 'littl
e', 'more', 'than', 'a', 'decade', ',', 'during', 'which', 'time', 'it', 'o
vertook', 'Wipro', 'and', 'Infosys', 'in', 'revenue', ',', 'Cognizant', 'we
nt', 'through', 'a', 'period', 'of', 'low', 'revenue', 'growth', '.', 'No
w', 'aiming', 'to', 'get', 'its', 'mojo', 'back', ',', 'the', 'company', 'i
s', 'betting', 'on', 'talent', 'in', 'its', 'India', 'software', 'deliver
y', 'centres', 'with', 'their', '2', 'lakh', '-', 'plus', 'headcount', 'tha
t', 'services', 'clients', 'in', 'the', 'U.S.', ',', 'Europe', 'and', 'Wes
t', 'Asia', '.', 'Ramamoorthy', 'also', 'has', 'views', 'on', 'the', 'educa
tion', 'system', 'and', 'what', 'the', 'country', 'needs', ',', 'to', 'mak
e', 'the', 'best', 'of', 'its', 'demographic', 'dividend', '.']
```

C. Stop Word Removal

Stop words are generally the most common words in a language and these words needs to be filtered out before processing natural language data.

NLTK

In [44]:
```
from nltk.corpus import stopwords
print('NLTK Defined Stopwords are .... \n')
stopWordsN_ = stopwords.words("english")
print(stopWordsN_)
print('\n There are {} stopwords defined in NLTK '.format(len(stopWordsN_))
```

```
NLTK Defined Stopwords are ....

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
e", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'i
t', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselve
s', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'th
ose', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
s', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and',
'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'fo
r', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'be
fore', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
w', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'o
wn', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just',
'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 'v
e', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't",
'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "n
eedn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'were
n', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

 There are 179 stopwords defined in NLTK
```

In [60]:
```
#Removing Stop words
'''Step1: Sentence Tokenize
Step2: Word Tokenize
Step3: Drop Stopwords'''
wordsN_ = [word for sent in sent_tokenize(data) for word in word_tokenize(s
print('After removing stopwords ..... \n\n', wordsN_)
```

```
After removing stopwords .....

 ['Ramkumar', 'Ramamoorthy', 'recently', 'appointed', 'Chairman', 'Managin
g', 'Director', 'tech', 'major', 'Cognizant', '’', 'India', 'operations',
'.', 'With', 'new', 'CEO', 'Brian', 'Humphries', 'taking', 'charge', 'globa
l', 'level', ',', 'Cognizant', 'brought', 'changes', 'strategy', 'operation
s', '.', 'After', 'leading', 'industry', 'growth', 'little', 'decade', ',',
'time', 'overtook', 'Wipro', 'Infosys', 'revenue', ',', 'Cognizant', 'wen
t', 'period', 'low', 'revenue', 'growth', '.', 'Now', 'aiming', 'get', 'moj
o', 'back', ',', 'company', 'betting', 'talent', 'India', 'software', 'deli
very', 'centres', '2', 'lakh-plus', 'headcount', 'services', 'clients', 'U.
S.', ',', 'Europe', 'West', 'Asia.Ramamoorthy', 'also', 'views', 'educatio
n', 'system', 'country', 'needs', ',', 'make', 'best', 'demographic', 'divi
dend', '.']
```

Spacy

In [53]:
```python
stopWordsS_ = spacy.lang.en.stop_words.STOP_WORDS
print('Spacy Defined Stopwords are .... \n')
print(stopWordsS_)
print('\n There are {} stopwords defined in NLTK '.format(len(stopWordsS_)))
```

```
Spacy Defined Stopwords are ....

{'re', 'somewhere', 'least', 'an', 'further', 'for', 'either', 'would', 'in
to', 'too', 'nine', 'until', 'of', 'about', 'around', 'has', 'go', 'thru',
'eight', 'hereafter', 'whose', 'hers', 'against', 'wherever', 'side', 'quit
e', 'when', 'what', 'that', 'our', 'call', 'she', '’s', 'nor', 'all', 'wer
e', 'such', 'anyway', 'something', '‘s', 'yourself', 'at', 'herself', 'no
w', 'on', 'elsewhere', 'before', 'again', "'s", 'from', 'where', 'whereas',
'one', 'down', 'yours', '‘d', 'to', 'seemed', 'only', 'after', 'take', 'exc
ept', 'cannot', 'a', 'also', 'onto', 'among', 'together', 'her', 'seeming',
'within', 'been', '‘ve', 'almost', 'in', '’re', 'most', 'never', "'ll", 'an
ywhere', '’ve', 'no', 'as', 'used', 'everywhere', 'both', 'name', 'former',
'meanwhile', 'regarding', 'he', 'several', 'get', 'due', 'anyhow', 'betwee
n', 'always', 'under', 'move', 'whither', 'twenty', 'ourselves', 'which',
'however', 'each', 'their', 'enough', 'whom', 'any', 'every', 'give', 'othe
rs', 'unless', 'they', 'these', 'therein', 'much', 'became', 'the', 'whoeve
r', 'own', 'see', '‘m', 'alone', 'hereupon', 'few', 'why', 'sometime', 'whi
le', 'less', 'doing', 'whatever', 'some', 'empty', '’d', 'ever', 'his', 'hi
mself', 'throughout', 'latterly', 'next', 'there', 'hence', 'out', 'say',
'noone', 'therefore', 'nowhere', 'made', 'very', 'other', 'using', 'otherwi
se', 'but', 'seems', 'whole', 'with', 'more', 'beyond', 'often', 'is', 'wel
l', 'someone', "'re", 'being', 'behind', 'may', 'two', 'whereupon', 'mysel
f', 'not', 'bottom', 'over', 'perhaps', 'than', 'below', '‘ll', 'even', 'pe
r', 'can', 'those', 'sometimes', 'becomes', 'fifteen', 'if', 'become', 'its
elf', 'had', 'and', 'via', 'last', 'ours', 'him', 'rather', 'indeed', 'latt
er', 'same', 'toward', 'fifty', '’m', 'ca', 'thus', 'part', 'was', 'n‘t',
'towards', 'wherein', 'seem', 'above', 'your', 'because', 'whether', 'mak
e', 'top', 'amount', 'am', 'hereby', 'thence', '‘re', 'off', 'through', 'i
t', 'just', 'them', "'ve", 'up', 'n’t', 'does', 'still', 'yourselves', 'm
y', "'d", 'should', 'might', 'hundred', 'along', 'how', 'beside', 'everythi
ng', 'none', 'three', 'do', 'nobody', 'mine', 'already', 'yet', 'thereby',
'nevertheless', 'amongst', "n't", 'done', 'whenever', 'sixty', 'themselve
s', 'whereafter', 'who', 'everyone', 'thereupon', 'we', 'could', 'althoug
h', 'besides', 'front', 'please', 'us', 'back', 'various', 'show', 'someho
w', 'else', 'since', 'anything', 'put', 'i', 'forty', 'upon', 'namely', 'te
n', 'must', 'be', 'me', 'nothing', '’ll', 'this', 'third', 'or', 'formerl
y', 'its', 'six', 'neither', 'by', 'have', 'though', 'twelve', 'without',
'really', 'another', 'are', 'herein', 'mostly', 'thereafter', 'once', 'kee
p', 'did', 'whereby', 'here', 'then', 'becoming', 'moreover', 'whence', 'fu
ll', 'four', 'five', 'beforehand', "'m", 'first', 'serious', 'across', 'man
y', 'you', 'anyone', 'afterwards', 'eleven', 'so', 'during', 'will'}

 There are 326 stopwords defined in NLTK
```

In [61]:
```
#Removing Stopwords
# token.is_stop to check is the word is a stopword.
wordsS_ = [token for token in doc if token.is_stop == False]
print('After removing stopwords ..... \n\n', wordsS_)
```

```
After removing stopwords .....

 [Ramkumar, Ramamoorthy, recently, appointed, Chairman, Managing, Director,
tech, major, Cognizant, India, operations, ., new, CEO, Brian, Humphries, t
aking, charge, global, level, ,, Cognizant, brought, changes, strategy, ope
rations, ., leading, industry, growth, little, decade, ,, time, overtook, W
ipro, Infosys, revenue, ,, Cognizant, went, period, low, revenue, growth,
., aiming, mojo, ,, company, betting, talent, India, software, delivery, ce
ntres, 2, lakh, -, plus, headcount, services, clients, U.S., ,, Europe, Wes
t, Asia, ., Ramamoorthy, views, education, system, country, needs, ,, best,
demographic, dividend, .]
```

D. Stemming

Stemming is the process of reducing inflected words to their word stem, base or root form—generally a written word form. For e.g. a stemming algorithm reduce the words fishing, fished, and fisher to the stem word fish.

There are several types of stemming algorithms which differ in respect to performance and accuracy and rules to stemm. One of the popular stemming algorithm is **'PorterStemmer'** which is a **Suffix-stripping algorithm**. This algorithm works based on smaller list of rules. Some of examples of rules defined in this stemming algorithm are

- if the word ends in 'ed', remove the 'ed'
- if the word ends in 'ing', remove the 'ing'
- if the word ends in 'ly', remove the 'ly'

NLTK

In [65]:

```python
from nltk.stem import PorterStemmer

print('Before Stemming Words are .... \n\n', wordsN_)
stem_ = PorterStemmer()
stemmed_wordN_ =[stem_.stem(word) for word in wordsN_]
print('\nAfter Stemming Words are .... \n\n', stemmed_wordN_)
```

```
Before Stemming Words are ....

 ['Ramkumar', 'Ramamoorthy', 'recently', 'appointed', 'Chairman', 'Managin
g', 'Director', 'tech', 'major', 'Cognizant', '’', 'India', 'operations',
'.', 'With', 'new', 'CEO', 'Brian', 'Humphries', 'taking', 'charge', 'globa
l', 'level', ',', 'Cognizant', 'brought', 'changes', 'strategy', 'operation
s', '.', 'After', 'leading', 'industry', 'growth', 'little', 'decade', ',',
'time', 'overtook', 'Wipro', 'Infosys', 'revenue', ',', 'Cognizant', 'wen
t', 'period', 'low', 'revenue', 'growth', '.', 'Now', 'aiming', 'get', 'moj
o', 'back', ',', 'company', 'betting', 'talent', 'India', 'software', 'deli
very', 'centres', '2', 'lakh-plus', 'headcount', 'services', 'clients', 'U.
S.', ',', 'Europe', 'West', 'Asia.Ramamoorthy', 'also', 'views', 'educatio
n', 'system', 'country', 'needs', ',', 'make', 'best', 'demographic', 'divi
dend', '.']

After Stemming Words are ....

 ['ramkumar', 'ramamoorthi', 'recent', 'appoint', 'chairman', 'manag', 'dir
ector', 'tech', 'major', 'cogniz', '’', 'india', 'oper', '.', 'with', 'ne
w', 'ceo', 'brian', 'humphri', 'take', 'charg', 'global', 'level', ',', 'co
gniz', 'brought', 'chang', 'strategi', 'oper', '.', 'after', 'lead', 'indus
tri', 'growth', 'littl', 'decad', ',', 'time', 'overtook', 'wipro', 'infos
i', 'revenu', ',', 'cogniz', 'went', 'period', 'low', 'revenu', 'growth',
'.', 'now', 'aim', 'get', 'mojo', 'back', ',', 'compani', 'bet', 'talent',
'india', 'softwar', 'deliveri', 'centr', '2', 'lakh-plu', 'headcount', 'ser
vic', 'client', 'u.s.', ',', 'europ', 'west', 'asia.ramamoorthi', 'also',
'view', 'educ', 'system', 'countri', 'need', ',', 'make', 'best', 'demograp
h', 'dividend', '.']
```

Spacy

Spacy does not have implementation for PorterStemer. Instead it uses **Lemmatization** to convert words into its root form.

E. Lemmatization

Lemmatization is also used to convert a word to its root word. Difference between stemming and lemmatization is that stemming is rule based such as removing suffixes or prefixes from a word and after stem the word may not be exact word. But lemmatization actually looks at words and their roots (called lemma) as described in the dictionary—is more precise.

NLTK

In [87]:
```
from nltk.stem import WordNetLemmatizer
print('Before Lemmatizing Words are .... \n\n', wordsN_)
lemmatizer = WordNetLemmatizer()
lemmaN_ = [lemmatizer.lemmatize(word) for word in wordsN_]
print('\nAfter Lemmatizing Words are .... \n\n', lemmaN_)
```

```
Before Lemmatizing Words are ....

 ['Ramkumar', 'Ramamoorthy', 'recently', 'appointed', 'Chairman', 'Managin
g', 'Director', 'tech', 'major', 'Cognizant', '’', 'India', 'operations',
'.', 'With', 'new', 'CEO', 'Brian', 'Humphries', 'taking', 'charge', 'globa
l', 'level', ',', 'Cognizant', 'brought', 'changes', 'strategy', 'operation
s', '.', 'After', 'leading', 'industry', 'growth', 'little', 'decade', ',',
'time', 'overtook', 'Wipro', 'Infosys', 'revenue', ',', 'Cognizant', 'wen
t', 'period', 'low', 'revenue', 'growth', '.', 'Now', 'aiming', 'get', 'moj
o', 'back', ',', 'company', 'betting', 'talent', 'India', 'software', 'deli
very', 'centres', '2', 'lakh-plus', 'headcount', 'services', 'clients', 'U.
S.', ',', 'Europe', 'West', 'Asia.Ramamoorthy', 'also', 'views', 'educatio
n', 'system', 'country', 'needs', ',', 'make', 'best', 'demographic', 'divi
dend', '.']

After Lemmatizing Words are ....

 ['Ramkumar', 'Ramamoorthy', 'recently', 'appointed', 'Chairman', 'Managin
g', 'Director', 'tech', 'major', 'Cognizant', '’', 'India', 'operation',
'.', 'With', 'new', 'CEO', 'Brian', 'Humphries', 'taking', 'charge', 'globa
l', 'level', ',', 'Cognizant', 'brought', 'change', 'strategy', 'operatio
n', '.', 'After', 'leading', 'industry', 'growth', 'little', 'decade', ',',
'time', 'overtook', 'Wipro', 'Infosys', 'revenue', ',', 'Cognizant', 'wen
t', 'period', 'low', 'revenue', 'growth', '.', 'Now', 'aiming', 'get', 'moj
o', 'back', ',', 'company', 'betting', 'talent', 'India', 'software', 'deli
very', 'centre', '2', 'lakh-plus', 'headcount', 'service', 'client', 'U.
S.', ',', 'Europe', 'West', 'Asia.Ramamoorthy', 'also', 'view', 'educatio
n', 'system', 'country', 'need', ',', 'make', 'best', 'demographic', 'divid
end', '.']
```

Major difference between stemming and lemmatizing is, later uses parts of speech of the word to change it to root word. Default pos tag is 'noun'. Look at the example below, when we lemmatize word 'appointed' sepcifying pos tag 'v' (Verb) it change the word to its verb form 'appoint'.

In [84]:
```
lemmatizer.lemmatize('appointed', pos='v')
```

Out[84]:
```
'appoint'
```

Spacy

In [89]:
```
lemmaS_ = [token.lemma_ for token in doc]
print('Lammatized words are .... \n\n', lemmaS_)
```

```
Lammatized words are ....

 ['Ramkumar', 'Ramamoorthy', 'be', 'recently', 'appoint', 'as', 'Chairman',
'and', 'Managing', 'Director', 'of', 'tech', 'major', 'Cognizant', ''s', 'I
ndia', 'operation', '.', 'with', 'new', 'CEO', 'Brian', 'Humphries', 'tak
e', 'charge', 'at', 'the', 'global', 'level', ',', 'Cognizant', 'have', 'br
ing', 'about', 'change', 'to', '-PRON-', 'strategy', 'and', 'operation',
'.', 'after', 'lead', 'industry', 'growth', 'over', 'a', 'little', 'more',
'than', 'a', 'decade', ',', 'during', 'which', 'time', '-PRON-', 'overtak
e', 'Wipro', 'and', 'Infosys', 'in', 'revenue', ',', 'Cognizant', 'go', 'th
rough', 'a', 'period', 'of', 'low', 'revenue', 'growth', '.', 'now', 'aim',
'to', 'get', '-PRON-', 'mojo', 'back', ',', 'the', 'company', 'be', 'bet',
'on', 'talent', 'in', '-PRON-', 'India', 'software', 'delivery', 'centre',
'with', '-PRON-', '2', 'lakh', '-', 'plus', 'headcount', 'that', 'service',
'client', 'in', 'the', 'U.S.', ',', 'Europe', 'and', 'West', 'Asia', '.',
'Ramamoorthy', 'also', 'have', 'view', 'on', 'the', 'education', 'system',
'and', 'what', 'the', 'country', 'need', ',', 'to', 'make', 'the', 'good',
'of', '-PRON-', 'demographic', 'dividend', '.']
```

F. Parts of Speech Tagging

The part of speech indicates how the word functions in meaning as well as grammatically within the sentence.In english language 8 common POS are noun, pronoun, verb, adjective, adverb, preposition, conjunction, and interjection.

In NLP , POS tag can bee asigned to each word in a text.

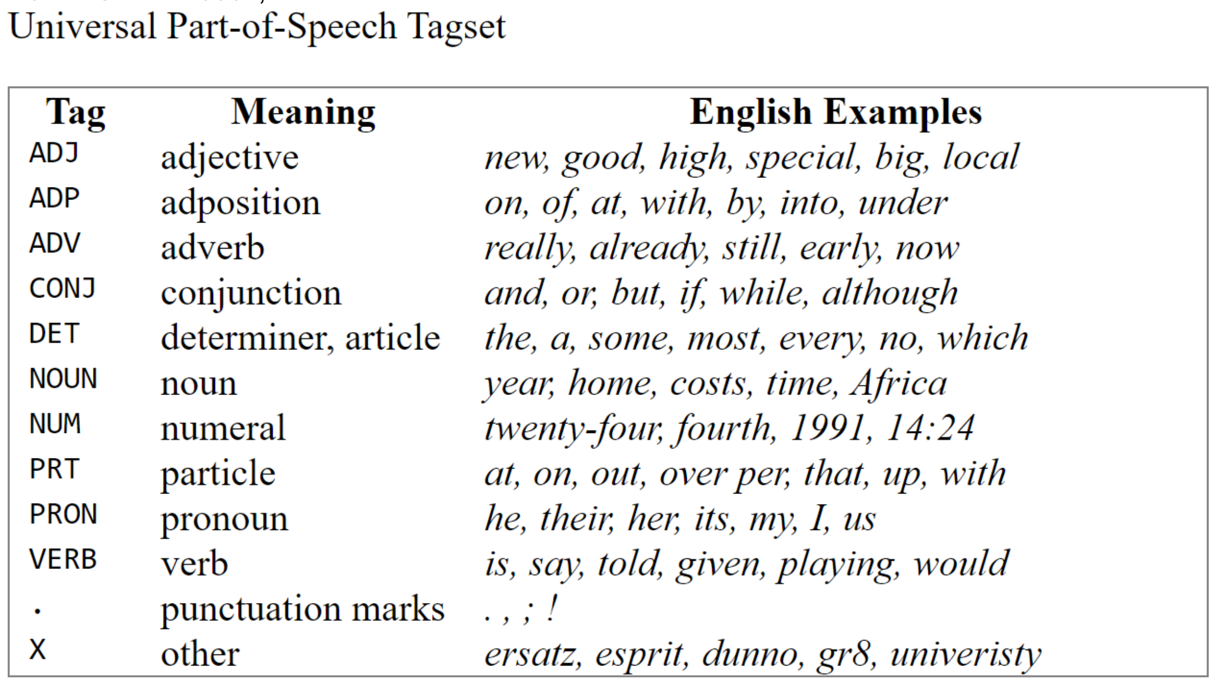
NLTK

```
In [99]:  pos_taggingN_ = nltk.pos_tag(wordsN_)
          print('POS Tag of each word are .... \n\n', pos_taggingN_)
```

```
POS Tag of each word are ....

 [('Ramkumar', 'NNP'), ('Ramamoorthy', 'NNP'), ('recently', 'RB'), ('appoin
ted', 'VBD'), ('Chairman', 'NNP'), ('Managing', 'NNP'), ('Director', 'NN
P'), ('tech', 'VBD'), ('major', 'JJ'), ('Cognizant', 'NNP'), ('’', 'NNP'),
('India', 'NNP'), ('operations', 'NNS'), ('.', '.'), ('With', 'IN'), ('ne
w', 'JJ'), ('CEO', 'NNP'), ('Brian', 'NNP'), ('Humphries', 'NNP'), ('takin
g', 'VBG'), ('charge', 'NN'), ('global', 'JJ'), ('level', 'NN'), (',',
','), ('Cognizant', 'NNP'), ('brought', 'VBD'), ('changes', 'NNS'), ('strat
egy', 'NN'), ('operations', 'NNS'), ('.', '.'), ('After', 'IN'), ('leadin
g', 'VBG'), ('industry', 'NN'), ('growth', 'NN'), ('little', 'JJ'), ('decad
e', 'NN'), (',', ','), ('time', 'NN'), ('overtook', 'JJ'), ('Wipro', 'NN
P'), ('Infosys', 'NNP'), ('revenue', 'NN'), (',', ','), ('Cognizant', 'NN
P'), ('went', 'VBD'), ('period', 'NN'), ('low', 'JJ'), ('revenue', 'NN'),
('growth', 'NN'), ('.', '.'), ('Now', 'RB'), ('aiming', 'VBG'), ('get', 'V
B'), ('mojo', 'JJ'), ('back', 'RB'), (',', ','), ('company', 'NN'), ('betti
ng', 'VBG'), ('talent', 'NN'), ('India', 'NNP'), ('software', 'NN'), ('deli
very', 'NN'), ('centres', 'VBZ'), ('2', 'CD'), ('lakh-plus', 'JJ'), ('headc
ount', 'NN'), ('services', 'NNS'), ('clients', 'NNS'), ('U.S.', 'NNP'),
(',', ','), ('Europe', 'NNP'), ('West', 'NNP'), ('Asia.Ramamoorthy', 'NN
P'), ('also', 'RB'), ('views', 'VBZ'), ('education', 'NN'), ('system', 'N
N'), ('country', 'NN'), ('needs', 'VBZ'), (',', ','), ('make', 'VB'), ('bes
t', 'JJS'), ('demographic', 'JJ'), ('dividend', 'NN'), ('.', '.')]
```

From the NLTK book,



Universal Part-of-Speech Tagset

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Spacy

In [101]:
```
pos_taggedS_ = [(token, token.pos_) for token in doc]
print('POS Tag of each word are .... \n\n', pos_taggedS_)
```

```
POS Tag of each word are ....

 [(Ramkumar, 'PROPN'), (Ramamoorthy, 'PROPN'), (was, 'AUX'), (recently, 'AD
V'), (appointed, 'VERB'), (as, 'SCONJ'), (Chairman, 'PROPN'), (and, 'CCON
J'), (Managing, 'PROPN'), (Director, 'PROPN'), (of, 'ADP'), (tech, 'NOUN'),
(major, 'ADJ'), (Cognizant, 'PROPN'), (’s, 'PROPN'), (India, 'PROPN'), (ope
rations, 'NOUN'), (., 'PUNCT'), (With, 'ADP'), (new, 'ADJ'), (CEO, 'PROP
N'), (Brian, 'PROPN'), (Humphries, 'PROPN'), (taking, 'VERB'), (charge, 'NO
UN'), (at, 'ADP'), (the, 'DET'), (global, 'ADJ'), (level, 'NOUN'), (,, 'PUN
CT'), (Cognizant, 'PROPN'), (has, 'AUX'), (brought, 'VERB'), (about, 'AD
P'), (changes, 'NOUN'), (to, 'ADP'), (its, 'PRON'), (strategy, 'NOUN'), (an
d, 'CCONJ'), (operations, 'NOUN'), (., 'PUNCT'), (After, 'ADP'), (leading,
'VERB'), (industry, 'NOUN'), (growth, 'NOUN'), (over, 'ADP'), (a, 'DET'),
(little, 'ADJ'), (more, 'ADJ'), (than, 'SCONJ'), (a, 'DET'), (decade, 'NOU
N'), (,, 'PUNCT'), (during, 'ADP'), (which, 'PRON'), (time, 'NOUN'), (it,
'PRON'), (overtook, 'VERB'), (Wipro, 'PROPN'), (and, 'CCONJ'), (Infosys, 'P
ROPN'), (in, 'ADP'), (revenue, 'NOUN'), (,, 'PUNCT'), (Cognizant, 'PROPN'),
(went, 'VERB'), (through, 'ADP'), (a, 'DET'), (period, 'NOUN'), (of, 'AD
P'), (low, 'ADJ'), (revenue, 'NOUN'), (growth, 'NOUN'), (., 'PUNCT'), (Now,
'ADV'), (aiming, 'VERB'), (to, 'PART'), (get, 'AUX'), (its, 'PRON'), (mojo,
'NOUN'), (back, 'ADV'), (,, 'PUNCT'), (the, 'DET'), (company, 'NOUN'), (is,
'AUX'), (betting, 'VERB'), (on, 'ADP'), (talent, 'NOUN'), (in, 'ADP'), (it
s, 'PRON'), (India, 'PROPN'), (software, 'NOUN'), (delivery, 'NOUN'), (cent
res, 'VERB'), (with, 'ADP'), (their, 'PRON'), (2, 'NUM'), (lakh, 'ADV'),
(-, 'PUNCT'), (plus, 'CCONJ'), (headcount, 'NOUN'), (that, 'PRON'), (servic
es, 'VERB'), (clients, 'NOUN'), (in, 'ADP'), (the, 'DET'), (U.S., 'PROPN'),
(,, 'PUNCT'), (Europe, 'PROPN'), (and, 'CCONJ'), (West, 'PROPN'), (Asia, 'P
ROPN'), (., 'PUNCT'), (Ramamoorthy, 'PROPN'), (also, 'ADV'), (has, 'AUX'),
(views, 'NOUN'), (on, 'ADP'), (the, 'DET'), (education, 'NOUN'), (system,
'NOUN'), (and, 'CCONJ'), (what, 'PRON'), (the, 'DET'), (country, 'NOUN'),
(needs, 'VERB'), (,, 'PUNCT'), (to, 'PART'), (make, 'VERB'), (the, 'DET'),
(best, 'ADJ'), (of, 'ADP'), (its, 'PRON'), (demographic, 'ADJ'), (dividend,
'NOUN'), (., 'PUNCT')]
```

To know more about POS tag list defined by spacy click on POS Tag List (https://spacy.io/api/annotation)

G. Named Entity Recognition (NER)

Entity detection, also called entity recognition, is a more advanced form of language processing that identifies important elements like places, people, organizations, and languages within an input string of text.

Spacy

In [121]:
```
named_entityS_ = [(entity,entity.label_) for entity in doc.ents]
print('Named Entities in our doc are .... \n\n', named_entityS_)
```

```
Named Entities in our doc are ....

 [(Ramkumar Ramamoorthy, 'PERSON'), (Cognizant’s India, 'ORG'), (US, 'GP
E'), (61.65, 'MONEY'), (Brian Humphries, 'PERSON'), (Cognizant, 'ORG'), (Wi
pro, 'PERSON'), (Infosys, 'GPE'), (Cognizant, 'ORG'), (India, 'GPE'), (2,
'CARDINAL'), (U.S., 'GPE'), (Europe, 'LOC'), (West Asia, 'LOC'), (Ramamoort
hy, 'PERSON')]
```

In [122]:
```
# We can visualize same using displaycy module in Spacy
from spacy import displacy
displacy.render(doc, style = "ent",jupyter = True)
```

Ramkumar Ramamoorthy **PERSON** was recently appointed as Chairman and Managing Director of tech major Cognizant’s India **ORG** operations.Cognizant is US **GPE** $ 61.65 **MONEY** comapny. With new CEO Brian Humphries **PERSON** taking charge at the global level, Cognizant **ORG** has brought about changes to its strategy and operations. After leading industry growth over a little more than a decade, during which time it overtook Wipro **PERSON** and Infosys **GPE** in revenue, Cognizant **ORG** went through a period of low revenue growth. Now aiming to get its mojo back, the company is betting on talent in its India **GPE** software delivery centres with their 2 **CARDINAL** lakh-plus headcount that services clients in the U.S. **GPE** , Europe **LOC** and West Asia **LOC** . Ramamoorthy **PERSON** also has views on the education system and what the country needs, to make the best of its demographic dividend.

H. Noun Phrase Extraction

It is the process of extracting noun phrases from a text.

Spacy

It has a function **noun_chunks** which breaks the input down into nouns and the words describing the nouns.

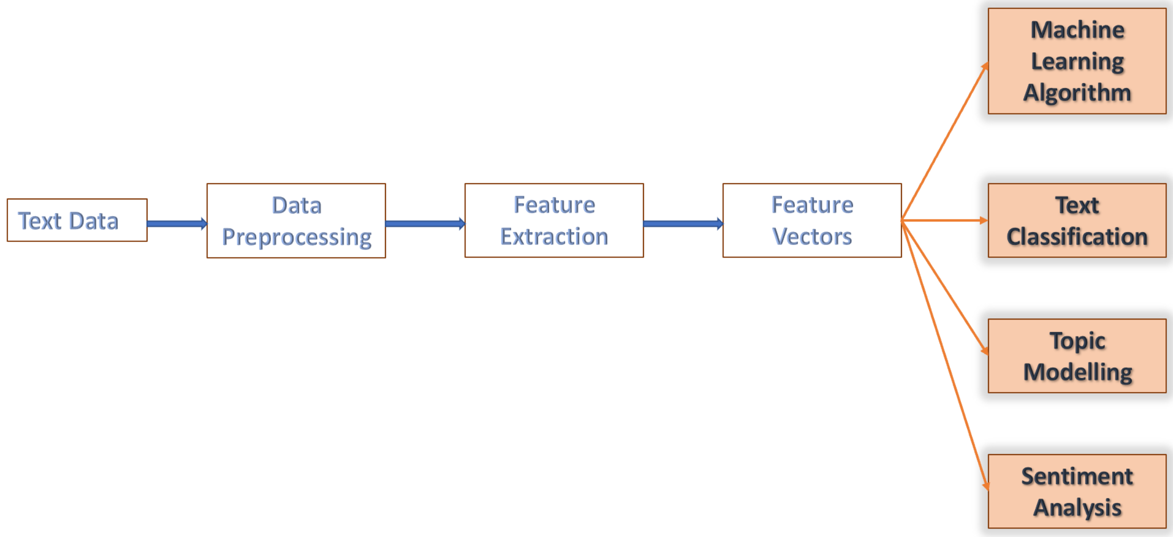
In [142]:
```
noun_phrasesS_ = [chunks for chunks in doc.noun_chunks]
print('Noun Phrases in our text are .... \n\n', noun_phrasesS_)
```

```
Noun Phrases in our text are ....

 [Ramkumar Ramamoorthy, Chairman, Managing Director, tech major Cognizant’s
India operations, Cognizant, US $61.65 comapny, new CEO Brian Humphries, ch
arge, the global level, Cognizant, changes, its strategy, operations, indus
try growth, than a decade, which time, it, Wipro, Infosys, revenue, Cogniza
nt, a period, low revenue growth, its mojo, the company, talent, headcount,
that, clients, the U.S., Europe, West Asia, Ramamoorthy, views, the educati
on system, what, the country, its demographic dividend]
```

Feature Extraction

Lets have a look at end to end steps involved in NLP, The imgae at right depicts the workflow.



So far we are familiar with data preprocessing tasks , now we will explore Feature Extraction.

For a machine learning algorithm , a dataset contains list of features and its values which are then passed to Machine learning algorithm. In case of text data we do not have such specific features. So we need to extract features for processing it through ML algorithms. **Feature Extraction** is the process of extracting/preparing features out of preprocessed text data.

We will explore different possible feature extraction techniques invented in NLP.

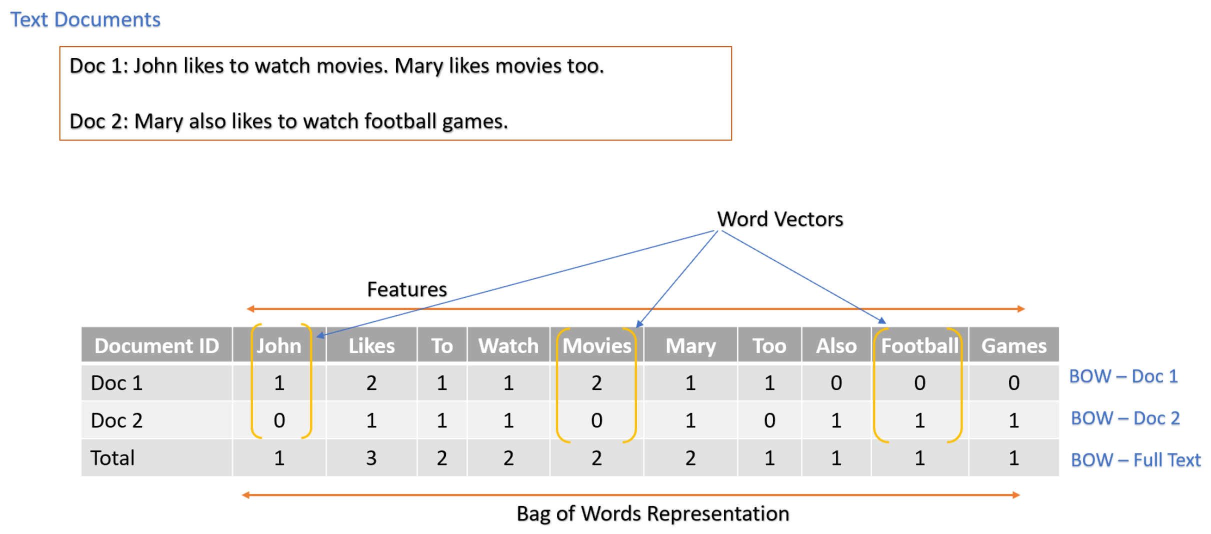
We know that a text data is a container of sentences and sentence is a container of words. In NLP a word in a text data is a feature for ML algorithm. Now the question is so can we train ML model on words of a text? answer is yes ML model would be trained on words of a text document. Word is a string data hence the next question arises is if the words can directly be feeded to model? answer is NO because ML algorithms are designed to support only numerical data. words should be converted into numerical format before sending it to any ML algorithm.

Words Embedding is the process of representing each word in a text into a numerical form. We human can easily understand connections between words by looking at the text. But computer can not understand it. For e.g. we can understand from the text data 'cars run using engines' that there is a connection between words cars and engines. But how can a computer can understand the connection? There are ways to represent words that captures mre these sorts of connections. **A word vector** is a numeric representation of a word that commuicates its relationship to other words.

There are several ways we can prepare word embediings (word vectors), we will dicuss those in following blogs.

Bag of Words:

A text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order .



Doc 1: John likes to watch movies. Mary likes movies too.

Doc 2: Mary also likes to watch football games.

Document ID	John	Likes	To	Watch	Movies	Mary	Too	Also	Football	Games
Doc 1	1	2	1	1	2	1	1	0	0	0
Doc 2	0	1	1	1	0	1	0	1	1	1
Total	1	3	2	2	2	2	1	1	1	1

Steps involved in this representations are

tokenizing strings for instance by using white-spaces and punctuation as token separators.

counting the occurrences of tokens in each document.

normalizing and weighting with diminishing importance tokens that occur in the majority of samples / documents.

In the example image we can see A corpus of documents can is represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.The words and its frequencies are features and the each word gets corresponding word vector from the matrix. for instance word vector of 'john' is [1, 0], word vector for 'movies' is [2,0]. In general term it is called **Vectorization**.

Lets see how can we vectorize in Python.

sklearn.feature_extraction.text module in python has different implementations for BOW vectorization. such implementations are **CountVectorizer and TfidfVectorizer**

CountVectorizer

The CountVectorizer implements Tokenizing and Counting steps in a BOW representation. Which means this class tokenize a input string and then counts word frequencies.

In [172]:
```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()

# We can transform list of text data OR list of sentences of a specific tex
# fit_transform to prepare the feature matrix (Matrix of features and its f
feature_matrix = vectorizer.fit_transform(sentencesN_)
```

feature_matrix is a Sparse matrix because many of its entries are 0 (did not occured in sentence). There are total 94 unique words in our text data. So feature matrix would have 94 features. And our text data has 5 sentences which we have vectorized , hence the feature_matrix would have 5 Rows. Lets visualise the feature_matrix through pandas dataframe.

In [173]:
```
import pandas as pd
pd.DataFrame(feature_matrix.toarray(),columns=vectorizer.get_feature_names(
```

Out[173]:

	61	65	about	after	aiming	also	and	appointed	as	asia	...	to	us	views	was	wer
0	1	1	0	0	0	0	1	1	1	0	...	0	1	0	1	
1	0	0	1	0	0	0	1	0	0	0	...	1	0	0	0	
2	0	0	0	1	0	0	1	0	0	0	...	0	0	0	0	
3	0	0	0	0	1	0	1	0	0	1	...	1	0	0	0	
4	0	0	0	0	0	1	1	0	0	0	...	1	0	1	0	

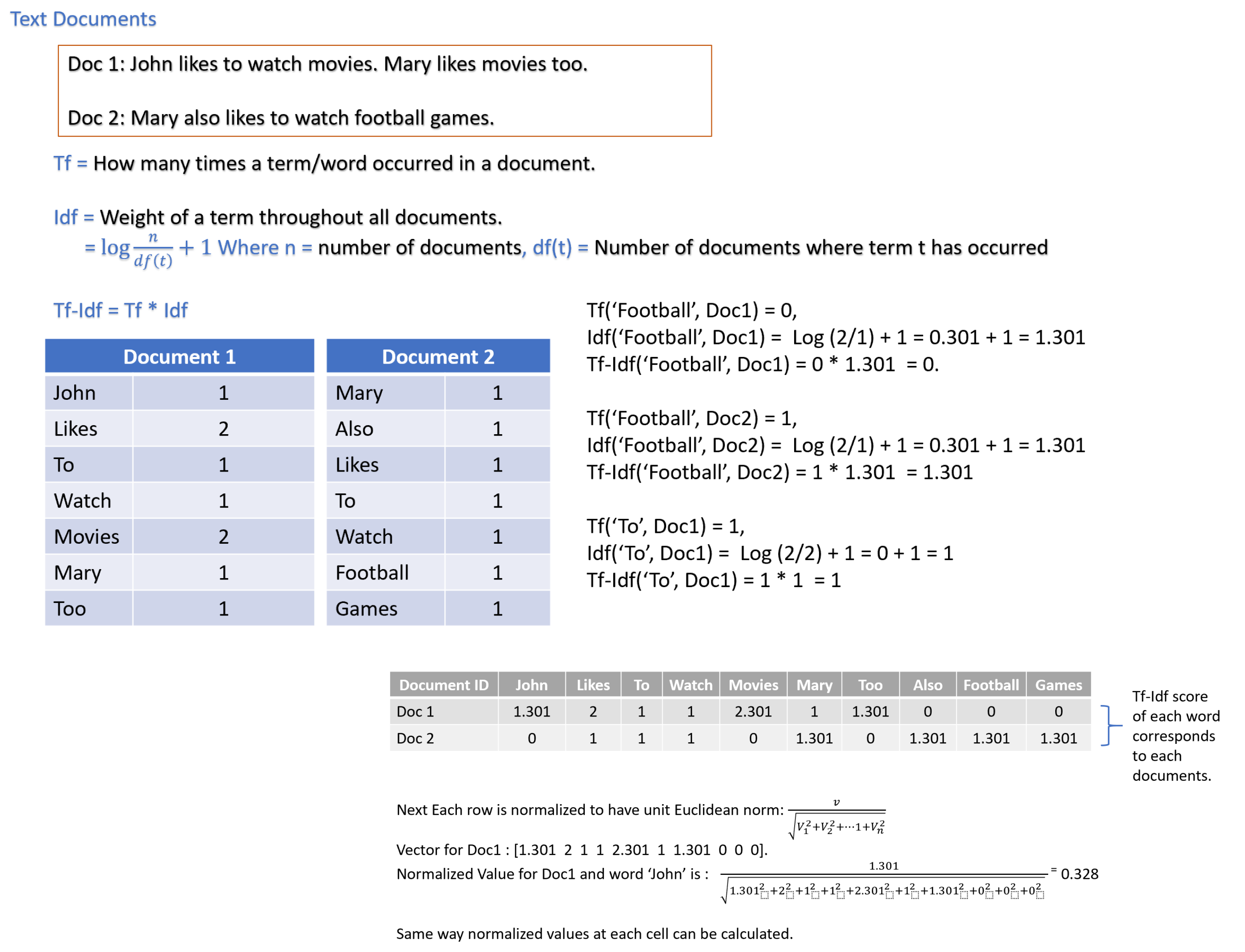
5 rows × 94 columns

The 0 indicates the word is not present in corresponding sentence. For instance word 'about' is not present in 1st sentence in our text data. also the word vector of word 'about' is [0 1 0 0 0]

TfIdfVectorizer

There are words (such as a, and, the, i, we, is , are, with, to and so on) which occurs many times in a text data or text corpus. So the word frequency in feature matrix for these words would be very high as compare to other words. which means these less importance words will kill the importance of other words. This is a drawback of using CountVectorizer. TfIdfVectorizer solves this importance problem of features by normalizing those features according to its weight in all documents.

Normalized TF-IDF score for each wrod corresponds to each document is depicted in the below images



Text Documents

Doc 1: John likes to watch movies. Mary likes movies too.

Doc 2: Mary also likes to watch football games.

Tf = How many times a term/word occurred in a document.

Idf = Weight of a term throughout all documents.

$= \log \frac{n}{df(t)} + 1$ Where n = number of documents, df(t) = Number of documents where term t has occurred

Tf-Idf = Tf * Idf

Document 1	
John	1
Likes	2
To	1
Watch	1
Movies	2
Mary	1
Too	1

Document 2	
Mary	1
Also	1
Likes	1
To	1
Watch	1
Football	1
Games	1

Tf('Football', Doc1) = 0,
Idf('Football', Doc1) = Log (2/1) + 1 = 0.301 + 1 = 1.301
Tf-Idf('Football', Doc1) = 0 * 1.301 = 0.

Tf('Football', Doc2) = 1,
Idf('Football', Doc2) = Log (2/1) + 1 = 0.301 + 1 = 1.301
Tf-Idf('Football', Doc2) = 1 * 1.301 = 1.301

Tf('To', Doc1) = 1,
Idf('To', Doc1) = Log (2/2) + 1 = 0 + 1 = 1
Tf-Idf('To', Doc1) = 1 * 1 = 1

Document ID	John	Likes	To	Watch	Movies	Mary	Too	Also	Football	Games
Doc 1	1.301	2	1	1	2.301	1	1.301	0	0	0
Doc 2	0	1	1	1	0	1.301	0	1.301	1.301	1.301

Tf-Idf score of each word corresponds to each documents.

Next Each row is normalized to have unit Euclidean norm: $\frac{v}{\sqrt{V_1^2+V_2^2+\cdots 1+V_n^2}}$

Vector for Doc1 : [1.301 2 1 1 2.301 1 1.301 0 0 0].

Normalized Value for Doc1 and word 'John' is : $\frac{1.301}{\sqrt{1.301^2+2^2+1^2+1^2+2.301^2+1^2+1.301^2+0^2+0^2+0^2}} = 0.328$

Same way normalized values at each cell can be calculated.

In [189]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

# We can transform list of text data OR list of sentences of a specific tex
# fit_transform to prepare the feature matrix (Matrix of features and its n
tf_idf_matrix = vectorizer.fit_transform(sentencesN_)
```

There are total 94 unique words in our text data. So tf_idf_matrix would have 94 features. And our text data has 5 sentences which we have vectorized , hence the tf_idf_matrix would have 5 Rows. Each entry in the matrix is normalized TF-Idf score . Lets visualise the tf_idf_matrix through pandas dataframe. import pandas as pd

In [191]:
```
pd.DataFrame(tf_idf_matrix.toarray(),columns=vectorizer.get_feature_names()
```

Out[191]:

```
         61        65     about    after    aiming      also       and  appointed        as
0  0.228975  0.228975  0.000000  0.00000  0.000000  0.000000  0.109108   0.228975  0.228975
1  0.000000  0.000000  0.242722  0.00000  0.000000  0.000000  0.115658   0.000000  0.000000
2  0.000000  0.000000  0.000000  0.18642  0.000000  0.000000  0.088830   0.000000  0.000000
3  0.000000  0.000000  0.000000  0.00000  0.181158  0.000000  0.086323   0.000000  0.000000
4  0.000000  0.000000  0.000000  0.00000  0.000000  0.232107  0.110600   0.000000  0.000000

5 rows × 94 columns
```

If a word is present in al documents that means the word has no significance. Hence Lesser the Tf-Idf score better the word in terms of importance.

In [ ]:
```

```