In [4]:

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

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Shrushti\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[4]:

```
True
```

Tokenization

In [5]:

```
from nltk import word_tokenize, sent_tokenize
```

In [6]:

```
sent = "I will walk 500 miles and I would walk 500 more, just to be the man who walks a tho
```

In [10]:

```
print(word_tokenize(sent))
```

```
['I', 'will', 'walk', '500', 'miles', 'and', 'I', 'would', 'walk', '500', 'm
ore', ',', 'just', 'to', 'be', 'the', 'man', 'who', 'walks', 'a', 'thousan
d', 'miles', 'to', 'fall', 'down', 'at', 'your', 'door']
```

In [11]:

```
print(sent_tokenize(sent))
```

```
['I will walk 500 miles and I would walk 500 more, just to be the man who wa
lks a thousand miles to fall down at your  door']
```

Stop-Words

In [12]:

```
from nltk.corpus import stopwords
```

In [14]:

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Shrushti\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[14]:

```
True
```

In [17]:

```
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're
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', 'tho
se', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',
'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'bu
t', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for',
'with', 'about', 'against', 'between', 'into', 'through', 'during', 'befor
e', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'o
n', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'the
re', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'mo
re', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'sa
me', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "d
on't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y',
'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
oesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 's
han', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "were
n't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:

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

In [21]:

```
stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(sent)

token = word_tokenize(sent)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
print("This is the unclean version:", token)
print("This is the cleaned version:", cleaned_token)
```

```
This is the unclean version: ['I', 'will', 'walk', '500', 'miles', 'and',
'I', 'would', 'walk', '500', 'more', ',', 'just', 'to', 'be', 'the', 'man',
'who', 'walks', 'a', 'thousand', 'miles', 'to', 'fall', 'down', 'at', 'you
r', 'door']
This is the cleaned version: ['I', 'walk', '500', 'miles', 'I', 'would', 'wa
lk', '500', ',', 'man', 'walks', 'thousand', 'miles', 'fall', 'door']
```

Stemming

In [22]:

```
from nltk.stem import PorterStemmer
```

In [23]:

```
stemmer = PorterStemmer()
```

In [24]:

```
words = ['play', 'playing', 'plays', 'played', 'playfullness', 'playful']
stemmed = [stemmer.stem(word) for word in words]
print(stemmed)
```

```
['play', 'play', 'play', 'play', 'playful', 'play']
```

In [25]:

```
sent2 = "I played the play playfully as the players were playing inthe play with playfullne
token = word_tokenize(sent2)
stemmed = ""
for word in token:
    stemmed += stemmer.stem(word) + " "
print(stemmed)
```

```
i play the play play as the player were play inth play with playful
```

Tagging parts of speech

In [27]:

```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Shrushti\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[27]:

```
True
```

In [28]:

```
from nltk import pos_tag
token = word_tokenize(sent) + word_tokenize(sent2)
tagged = pos_tag(cleaned_token)
print(tagged)
```

```
[('I', 'PRP'), ('walk', 'VBP'), ('500', 'CD'), ('miles', 'NNS'), ('I', 'PR
P'), ('would', 'MD'), ('walk', 'VB'), ('500', 'CD'), (',', ','), ('man', 'N
N'), ('walks', 'NNS'), ('thousand', 'VBP'), ('miles', 'NNS'), ('fall', 'V
B'), ('door', 'NN')]
```

Lemmatization

In [29]:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

In [31]:

```
nltk.download('wordnet')
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Shrushti\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
```

```
rocks : rock
corpora : corpus
better : good
```

Part 2

In [35]:

```
import pandas as pd
import sklearn as sk
import math

first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"

#split so each word have their own string
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")
#join them to remove common duplicate words

total= set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'of', 'Data', 'machine', 'learning', 'for', 'the', '21st', 'key', 'sexies
t', 'century', 'data', 'science', 'job', 'is', 'Science'}
```

In [45]:

```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)

for word in first_sentence:
    wordDictA[word]+=1
for word in second_sentence:
    wordDictB[word]+=1
pd.DataFrame([wordDictA, wordDictB])
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)

    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
        return(tfDict)

#running our sentences through the tf function:

tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)

#Converting to dataframe for visualization

tf = pd.DataFrame([tfFirst, tfSecond])

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['Data', 'machine', 'learning', '21st', 'key', 'sexiest', 'century', 'data',
'science', 'job', 'Science']

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

In [ ]:

```

```