In [63]:
```
1 df
```

Out[63]:

	Phrases
0	Stay Hungry Stay Foolish
1	Faith Can Move Mountains
2	The way to get started is to quit talking and ...
3	If you set your goals ridiculously high and it...
4	They say dreams do come true but nightmare are...

In [64]:
```
1 df['tokenize_col'] = df.apply(lambda row: nltk.word_tokenize(row['Phrases']),axis=1)
```

In [65]:
```
1 df
```

Out[65]:

	Phrases	tokenize_col
0	Stay Hungry Stay Foolish	[Stay, Hungry, Stay, Foolish]
1	Faith Can Move Mountains	[Faith, Can, Move, Mountains]
2	The way to get started is to quit talking and ...	[The, way, to, get, started, is, to, quit, tal...
3	If you set your goals ridiculously high and it...	[If, you, set, your, goals, ridiculously, high...
4	They say dreams do come true but nightmare are...	[They, say, dreams, do, come, true, but, night...

In [ ]:
```
1
```

Stemming

In [ ]:
```
1
```

Stemming is the process of removing a part of word or reducing a word to its stem or root.

A stemming algorithm reduces the word

for eg : "chocolates" , "chocolatey", "choco"

when you apply the stemming concept the words are reduced to its root/stem (in the above eg: the word "choco" will be the root word)

"retrival", "retrives", "retrived" ==> retriv

In [ ]:
```
1
```

over stemming

```
1 over stemming is the process where a much larger part of word is chopped off (removed) than
2 what is required, which in turn leads to two or more words being reduced to the same root word or
3 stem incorrectly when they should have been reduced to two or more stem words.
```

```
1 universal university universe  ==> univers
```

In [ ]:
```
1
```

under stemming

under stemming is when two words that should be stemmed to the same root are not being done.

This is also known as false negative.

for eg:

alumnus alumni alumnae

In [ ]:
```
1
```

porter stemmer algorithm

In [66]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
```

In [67]:
```
ps = PorterStemmer()
word = ["program","programmer","programs","programming","programmers"]
for item in word:
    print(item," ",ps.stem(item))
```

```
program   program
programmer   programm
programs   program
programming   program
programmers   programm
```

In [68]:
```
sentence = "Provision Maximum multiple owned caring on go gone going was this"
```

In [69]:
```
tk_list = nltk.word_tokenize(sentence)
```

In [70]:
```
tk_list
```

Out[70]:
```
['Provision',
 'Maximum',
 'multiple',
 'owned',
 'caring',
 'on',
 'go',
 'gone',
 'going',
 'was',
 'this']
```

In [71]:
```
for word in tk_list:
    print(word," ",ps.stem(word))
```

```
Provision   provis
Maximum   maximum
multiple   multipl
owned   own
caring   care
on   on
go   go
gone   gone
going   go
was   wa
this   thi
```

In [73]:
```
words = ['generous','generate','generously','generation']
for word in words:
    print(word," ",ps.stem(word))
```

```
generous   gener
generate   gener
generously   gener
generation   gener
```

In [ ]:
```

```

Snowball Stemmer Algorithm

In [72]:
```
from nltk.stem import SnowballStemmer
snowball = SnowballStemmer(language='english')

words = ['generous','generate','generously','generation']

for word in words:
    print(word," ",snowball.stem(word))
```

```
generous   generous
generate   generat
generously   generous
generation   generat
```

In [ ]:
```

```

In [76]:
```
words = ['eating','eats','eaten','puts','putting']

for word in words:
    print(word," ",ps.stem(word))
```

```
eating   eat
eats   eat
eaten   eaten
puts   put
putting   put
```

In [ ]:
```

```

In [75]:
```
words = ['eating','eats','eaten','puts','putting']

for word in words:
    print(word," ",snowball.stem(word))
```

```
eating   eat
eats   eat
eaten   eaten
puts   put
putting   put
```

lancaster stemmer algorithm

Lancaster stemmer is simple but it tends to produce results with over stemming Over stemming causes the stem to be non meaningful

In [74]:
```
from nltk.stem import LancasterStemmer

lancaster = LancasterStemmer()

words = ['eating','eats','eaten','puts','putting']

for word in words:
    print(word," ",lancaster.stem(word))
```

```
eating   eat
eats   eat
eaten   eat
puts   put
putting   put
```

In [ ]:
```

```

In [ ]:
```

```

Lemmatization

Lemmatization is the process of converting a word to its base form.

The difference between stemming and lemmatization is lemmatization considers the context and converts the word into a meaningful base form

whereas stemming just removes the last few characters often leading to incorrect meaning and spelling errors

For eg

'Caring' => Lemmatization => 'care'

'Caring' => Stemming => 'Car'

word lemmatizer

spacy lemmatizer

textblob

clip pattern

stanford coreNLP

Genism Lemmatizer

TreeTagger

In [77]:
```
1 nltk.download('wordnet')
```

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

Out[77]: True

In [78]:
```
1 from nltk.stem import WordNetLemmatizer
```

In [79]:
```
1 lemmatizer = WordNetLemmatizer()
```

In [81]:
```
1 words = ['bats','are','feet','hands']
2
3 for item in words:
4     print(item," ",lemmatizer.lemmatize(item))
```

```
bats   bat
are   are
feet   foot
hands   hand
```

In [85]:
```
# words = ['bats','are','feet','hands']

# for item in words:
#     print(item," ",lancaster.stem(item))

# words = ['bats','are','feet','hands']

# for item in words:
#     print(item," ",snowball.stem(item))

# words = ['bats','are','feet','hands']

# for item in words:
#     print(item," ",ps.stem(item))
```

In [ ]:
```

```

Sentence Lemmatization

In [86]:
```
sentence = "The striped bats are hanging on their feet for best"
```

In [87]:
```
word_list = nltk.word_tokenize(sentence)
```

In [91]:
```
word_list
```

Out[91]:
```
['The',
 'striped',
 'bats',
 'are',
 'hanging',
 'on',
 'their',
 'feet',
 'for',
 'best']
```

In [89]:
```
1
2 lemmatized_output = ' '.join([lemmatizer.lemmatize(item) for item in word_list])
```

In [90]:
```
1 print(lemmatized_output)
```

```
The striped bat are hanging on their foot for best
```

In [ ]:
```
1
```

In [93]:
```
1 print(lemmatizer.lemmatize('stripes','v'))  # for converting the lemmatize word into verb (POS)
```

```
strip
```

In [95]:
```
1 print(lemmatizer.lemmatize('stripes','n')) # for converting the lemmatize word into noun (POS)
```

```
stripe
```

In [101]:
```
1 sentence = "My name is steve jobs"
2 word_list = nltk.word_tokenize(sentence)
3
4 lemmatized_output = ' '.join([lemmatizer.lemmatize(item) for item in word_list])
```

In [102]:
```
1 print(lemmatized_output)
```

```
My name is steve job
```

In [ ]:
```
1
```

In [105]:
```
print("rocks",lemmatizer.lemmatize("rocks"))

print("corpora",lemmatizer.lemmatize("corpora"))

print("better",lemmatizer.lemmatize("better","a"))  # 2nd argument is pos (part of speech (noun,verb,adje
```

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

In [119]:
```
print("can't",lemmatizer.lemmatize("can't"))
print("what's",lemmatizer.lemmatize("what's"))
print("couldn't",lemmatizer.lemmatize("couldn't"))
print("wasn't",lemmatizer.lemmatize("wasn't"))
```

```
can't can't
what's what's
couldn't couldn't
wasn't wasn't
```

In [115]:
```
print("can't",ps.stem("can't"))
```

```
can't can't
```

In [ ]:
```

```