In [14]:

```
import string
string.punctuation
```

Out[14]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [17]:

```
text="I a,,m.. TeacHing, /n NLP (NATURAL LanguaGe ProceSSinG) ??"
txt=[c for c in text if c not in string.punctuation]
txt=''.join(txt)
txt
```

Out[17]:

```
'I am TeacHing n NLP NATURAL LanguaGe ProceSSinG '
```

In [18]:

```
txt.lower()
```

Out[18]:

```
'i am teaching n nlp natural language processing '
```

In [5]:

```
# Tokenization of paragraphs/sentences
import nltk
paragraph = """I have three visions for India. In 3000 years of our history, people from all over the world have come and
               invaded us, captured our lands, conquered our minds.my first vision is that of freedom. I believe that India got its first vision of
               this in 1857, when we started the War of Independence."""

# Tokenizing sentences
sentences = nltk.sent_tokenize(paragraph)

print(sentences)
```

```
['I have three visions for India.', 'In 3000 years of our history, people from all over the world have come and \n               invaded us, captured our lands, conquered our minds.my first vision is that of freedom.', 'I believe that India got its first vision of \n               this in 1857, when we started the War of Independence.']
```

In [6]:

```
# Tokenizing words
words = nltk.word_tokenize(paragraph)
print(words)
```

```
['I', 'have', 'three', 'visions', 'for', 'India', '.', 'In', '3000', 'years', 'of', 'our', 'history', ',', 'people', 'from', 'all', 'over', 'the', 'world', 'have', 'come', 'and', 'invaded', 'us', ',', 'captured', 'our', 'lands', ',', 'conquered', 'our', 'minds.my', 'first', 'vision', 'is', 'that', 'of', 'freedom', '.', 'I', 'believe', 'that', 'India', 'got', 'its', 'first', 'vision', 'of', 'this', 'in', '1857', ',', 'when', 'we', 'started', 'the', 'War', 'of', 'Independence', '.']
```

In [8]:

```
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords

paragraph = """I have three visions for India. In 3000 years of our history, people from all over
               the world have come and invaded us, captured our lands, conquered our mind
```

```
s.
                From Alexander onwards, the Greeks, the Turks, the Moguls, the Portuguese,
the British,
                the French, the Dutch, all of them came and looted us, took over what was
ours.
                Yet we have not done this to any other nation. We have not conquered anyon
e. """


sentences = nltk.sent_tokenize(paragraph)
stemmer = PorterStemmer()

# Stemming
for i in range(len(sentences)):
    words = nltk.word_tokenize(sentences[i])
    words = [stemmer.stem(word) for word in words if word not in set(stopwords.words('en
glish'))]
    sentences[i] = ' '.join(words)
print(sentences)
```

```
['I three vision india .', 'In 3000 year histori , peopl world come invad us , captur lan
d , conquer mind .', 'from alexand onward , greek , turk , mogul , portugues , british ,
french , dutch , came loot us , took .', 'yet done nation .', 'We conquer anyon .']
```

In [10]:

```
import nltk
from nltk.corpus import stopwords

paragraph = """I have three visions for India. In 3000 years of our history, people from
all over
                the world have come and invaded us, captured our lands, conquered our mind
s.
                From Alexander onwards, the Greeks, the Turks, the Moguls, the Portuguese,
the British,
                the French, the Dutch, all of them came and looted us, took over what was
ours.
                Yet we have not done this to any other nation. We have not conquered anyon
e. """


sentences = nltk.sent_tokenize(paragraph)

# Stop word removal
for i in range(len(sentences)):
    words = nltk.word_tokenize(sentences[i])
    words = [word for word in words if word not in set(stopwords.words('english'))]
    sentences[i] = ' '.join(words)
print(sentences)
```

```
['I three visions India .', 'In 3000 years history , people world come invaded us , captu
red lands , conquered minds .', 'From Alexander onwards , Greeks , Turks , Moguls , Portu
guese , British , French , Dutch , came looted us , took .', 'Yet done nation .', 'We con
quered anyone .']
```

In [11]:

```
import nltk
from nltk.corpus import stopwords

stop = stopwords.words('english')
stop[0:10]
```

Out[11]:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

In [12]:

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
```

```
paragraph = """Thank you all so very much. Thank you to the Academy.
               Thank you to all of you in this room. I have to congratulate
               the other incredible nominees this year. The Revenant was
               the product of the tireless efforts of an unbelievable cast
               and crew.Thank you so very much."""


sentences = nltk.sent_tokenize(paragraph)
lemmatizer = WordNetLemmatizer()

# Lemmatization
for i in range(len(sentences)):
    words = nltk.word_tokenize(sentences[i])
    words = [lemmatizer.lemmatize(word) for word in words if word not in set(stopwords.w
ords('english'))]
    sentences[i] = ' '.join(words)
print(sentences)
```

```
['Thank much .', 'Thank Academy .', 'Thank room .', 'I congratulate incredible nominee ye
ar .', 'The Revenant product tireless effort unbelievable cast crew.Thank much .']
```

In [13]:

```
import string
string.punctuation
```

Out[13]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [17]:

```
text = "I am... teaching/n NLP??"
txt = [c for c in text if c not in string.punctuation]
txt= ''.join(txt)
txt
```

Out[17]:

```
'I am teachingn NLP'
```

In [22]:

```
import nltk

paragraph =  """I have three visions for India. In 3000 years of our history, people from
all over
               the world have come and invaded us, captured our lands, conquered our mind
s.
               From Alexander onwards, the Greeks, the Turks, the Moguls, the Portuguese,
the British,
               the French, the Dutch, all of them came and looted us, took over what was
ours.
               Yet we have not done this to any other nation. We have not conquered anyon
e. """


# Cleaning the texts
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

ps = PorterStemmer()
wordnet=WordNetLemmatizer()
sentences = nltk.sent_tokenize(paragraph)
corpus = []
for i in range(len(sentences)):
    review = re.sub('[^a-zA-Z]', ' ', sentences[i])
    review = review.lower()
    review = nltk.word_tokenize(review)
```

```
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('engli
sh'))]
    review = ' '.join(review)
    corpus.append(review)

# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 70)
X = cv.fit_transform(corpus).toarray()
print(X)
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0]
 [0 0 0 0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0]
 [1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

In [23]:

```
import nltk

paragraph =  """I have three visions for India. In 3000 years of our history, people from
all over
               the world have come and invaded us, captured our lands, conquered our mind
s.
               From Alexander onwards, the Greeks, the Turks, the Moguls, the Portuguese,
the British,
               the French, the Dutch, all of them came and looted us, took over what was
ours.
               Yet we have not done this to any other nation. We have not conquered anyon
e. """


# Cleaning the texts
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

ps = PorterStemmer()
wordnet=WordNetLemmatizer()
sentences = nltk.sent_tokenize(paragraph)
corpus = []
for i in range(len(sentences)):
    review = re.sub('[^a-zA-Z]', ' ', sentences[i])
    review = review.lower()
    review = nltk.word_tokenize(review)
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('engli
sh'))]
    review = ' '.join(review)
    corpus.append(review)

# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 70)
X = cv.fit_transform(corpus).toarray()
print(X)
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0]
 [0 0 0 0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0]
 [1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

In [27]:

```
import pandas as pd
corpus =  ['this is sentence is',
           'this is another sentence this ',
           'Third document is here']

# Creating the Bag of Words model
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X = cv.fit_transform(corpus)
print(X.shape)
print(X)
print(X.toarray())
```

```
(3, 7)
  (0, 6) 1
  (0, 3) 2
  (0, 4) 1
  (1, 6) 2
  (1, 3) 1
  (1, 4) 1
  (1, 0) 1
  (2, 3) 1
  (2, 5) 1
  (2, 1) 1
  (2, 2) 1
[[0 0 0 2 1 0 1]
 [1 0 0 1 1 0 2]
 [0 1 1 1 0 1 0]]
```

In [28]:

```
df= pd.DataFrame(X.toarray(), columns= cv.get_feature_names())
df
```

Out[28]:

	another	document	here	is	sentence	third	this
0	0	0	0	2	1	0	1
1	1	0	0	1	1	0	2
2	0	1	1	1	0	1	0

In [29]:

```
from nltk.util import bigrams,trigrams,ngrams
string="""Thank you all so very much. Thank you to the Academy.
                Thank you to all of you in this room. I have to congratulate
                the other incredible nominees this year. The Revenant was
                the product of the tireless efforts of an unbelievable cast
                and crew.Thank you so very much."""
tokens = nltk.word_tokenize(string)
tokens_bigrams=list(nltk.bigrams(tokens))
tokens_bigrams
```

Out[29]:

```
[('Thank', 'you'),
 ('you', 'all'),
 ('all', 'so'),
 ('so', 'very'),
 ('very', 'much'),
 ('much', '.'),
 ('.', 'Thank'),
 ('Thank', 'you'),
 ('you', 'to'),
 ('to', 'the'),
 ('the', 'Academy'),
 ('Academy', '.'),
 ('.', 'Thank'),
 ('Thank', 'you'),
 ('you', 'to'),
 ('to', 'all'),
 ('all', 'of'),
 ('of', 'you'),
 ('you', 'in'),
 ('in', 'this'),
 ('this', 'room'),
```

```
 ('room', '.'),
 ('.', 'I'),
 ('I', 'have'),
 ('have', 'to'),
 ('to', 'congratulate'),
 ('congratulate', 'the'),
 ('the', 'other'),
 ('other', 'incredible'),
 ('incredible', 'nominees'),
 ('nominees', 'this'),
 ('this', 'year'),
 ('year', '.'),
 ('.', 'The'),
 ('The', 'Revenant'),
 ('Revenant', 'was'),
 ('was', 'the'),
 ('the', 'product'),
 ('product', 'of'),
 ('of', 'the'),
 ('the', 'tireless'),
 ('tireless', 'efforts'),
 ('efforts', 'of'),
 ('of', 'an'),
 ('an', 'unbelievable'),
 ('unbelievable', 'cast'),
 ('cast', 'and'),
 ('and', 'crew.Thank'),
 ('crew.Thank', 'you'),
 ('you', 'so'),
 ('so', 'very'),
 ('very', 'much'),
 ('much', '.')]
```

In [30]:

```
tokens_trigrams=list(nltk.trigrams(tokens))
tokens_trigrams
```

Out[30]:

```
[('Thank', 'you', 'all'),
 ('you', 'all', 'so'),
 ('all', 'so', 'very'),
 ('so', 'very', 'much'),
 ('very', 'much', '.'),
 ('much', '.', 'Thank'),
 ('.', 'Thank', 'you'),
 ('Thank', 'you', 'to'),
 ('you', 'to', 'the'),
 ('to', 'the', 'Academy'),
 ('the', 'Academy', '.'),
 ('Academy', '.', 'Thank'),
 ('.', 'Thank', 'you'),
 ('Thank', 'you', 'to'),
 ('you', 'to', 'all'),
 ('to', 'all', 'of'),
 ('all', 'of', 'you'),
 ('of', 'you', 'in'),
 ('you', 'in', 'this'),
 ('in', 'this', 'room'),
 ('this', 'room', '.'),
 ('room', '.', 'I'),
 ('.', 'I', 'have'),
 ('I', 'have', 'to'),
 ('have', 'to', 'congratulate'),
 ('to', 'congratulate', 'the'),
 ('congratulate', 'the', 'other'),
 ('the', 'other', 'incredible'),
 ('other', 'incredible', 'nominees'),
 ('incredible', 'nominees', 'this'),
 ('nominees', 'this', 'year'),
 ('this', 'year', '.'),
```

```
 ('year', '.', 'The'),
 ('.', 'The', 'Revenant'),
 ('The', 'Revenant', 'was'),
 ('Revenant', 'was', 'the'),
 ('was', 'the', 'product'),
 ('the', 'product', 'of'),
 ('product', 'of', 'the'),
 ('of', 'the', 'tireless'),
 ('the', 'tireless', 'efforts'),
 ('tireless', 'efforts', 'of'),
 ('efforts', 'of', 'an'),
 ('of', 'an', 'unbelievable'),
 ('an', 'unbelievable', 'cast'),
 ('unbelievable', 'cast', 'and'),
 ('cast', 'and', 'crew.Thank'),
 ('and', 'crew.Thank', 'you'),
 ('crew.Thank', 'you', 'so'),
 ('you', 'so', 'very'),
 ('so', 'very', 'much'),
 ('very', 'much', '.')]
```

In [31]:

```
tokens_ngrams=list(nltk.ngrams(tokens,5))
tokens_ngrams
```

Out[31]:

```
[('Thank', 'you', 'all', 'so', 'very'),
 ('you', 'all', 'so', 'very', 'much'),
 ('all', 'so', 'very', 'much', '.'),
 ('so', 'very', 'much', '.', 'Thank'),
 ('very', 'much', '.', 'Thank', 'you'),
 ('much', '.', 'Thank', 'you', 'to'),
 ('.', 'Thank', 'you', 'to', 'the'),
 ('Thank', 'you', 'to', 'the', 'Academy'),
 ('you', 'to', 'the', 'Academy', '.'),
 ('to', 'the', 'Academy', '.', 'Thank'),
 ('the', 'Academy', '.', 'Thank', 'you'),
 ('Academy', '.', 'Thank', 'you', 'to'),
 ('.', 'Thank', 'you', 'to', 'all'),
 ('Thank', 'you', 'to', 'all', 'of'),
 ('you', 'to', 'all', 'of', 'you'),
 ('to', 'all', 'of', 'you', 'in'),
 ('all', 'of', 'you', 'in', 'this'),
 ('of', 'you', 'in', 'this', 'room'),
 ('you', 'in', 'this', 'room', '.'),
 ('in', 'this', 'room', '.', 'I'),
 ('this', 'room', '.', 'I', 'have'),
 ('room', '.', 'I', 'have', 'to'),
 ('.', 'I', 'have', 'to', 'congratulate'),
 ('I', 'have', 'to', 'congratulate', 'the'),
 ('have', 'to', 'congratulate', 'the', 'other'),
 ('to', 'congratulate', 'the', 'other', 'incredible'),
 ('congratulate', 'the', 'other', 'incredible', 'nominees'),
 ('the', 'other', 'incredible', 'nominees', 'this'),
 ('other', 'incredible', 'nominees', 'this', 'year'),
 ('incredible', 'nominees', 'this', 'year', '.'),
 ('nominees', 'this', 'year', '.', 'The'),
 ('this', 'year', '.', 'The', 'Revenant'),
 ('year', '.', 'The', 'Revenant', 'was'),
 ('.', 'The', 'Revenant', 'was', 'the'),
 ('The', 'Revenant', 'was', 'the', 'product'),
 ('Revenant', 'was', 'the', 'product', 'of'),
 ('was', 'the', 'product', 'of', 'the'),
 ('the', 'product', 'of', 'the', 'tireless'),
 ('product', 'of', 'the', 'tireless', 'efforts'),
 ('of', 'the', 'tireless', 'efforts', 'of'),
 ('the', 'tireless', 'efforts', 'of', 'an'),
 ('tireless', 'efforts', 'of', 'an', 'unbelievable'),
 ('efforts', 'of', 'an', 'unbelievable', 'cast'),
 ('of', 'an', 'unbelievable', 'cast', 'and'),
 ('an', 'unbelievable', 'cast', 'and', 'crew.Thank')
```

```
('all', 'unbelievable', 'cast', 'and', 'crew.Thank'),
('unbelievable', 'cast', 'and', 'crew.Thank', 'you'),
('cast', 'and', 'crew.Thank', 'you', 'so'),
('and', 'crew.Thank', 'you', 'so', 'very'),
('crew.Thank', 'you', 'so', 'very', 'much'),
('you', 'so', 'very', 'much', '.')]
```

In [38]:

```
string="""Thank you all so very much. I welcome you to this Academy. """
tokens = nltk.word_tokenize(string)
for token in tokens:
    print(nltk.pos_tag([token]))
```

```
[('Thank', 'NN')]
[('you', 'PRP')]
[('all', 'DT')]
[('so', 'RB')]
[('very', 'RB')]
[('much', 'JJ')]
[('.', '.')]
[('I', 'PRP')]
[('welcome', 'NN')]
[('you', 'PRP')]
[('to', 'TO')]
[('this', 'DT')]
[('Academy', 'NN')]
[('.', '.')]
```

In [44]:

```
quote="Steve, the CEO of Apple Inc. is living in San Francisco"
tokens= nltk.word_tokenize(quote)
chunks= nltk.ne_chunk(nltk.pos_tag(tokens))
for chunk in chunks:
    print(chunk)
```

```
(PERSON Steve/NNP)
(',', ',')
('the', 'DT')
(ORGANIZATION CEO/NNP)
('of', 'IN')
(ORGANIZATION Apple/NNP Inc./NNP)
('is', 'VBZ')
('living', 'VBG')
('in', 'IN')
(GPE San/NNP)
('Francisco', 'NNP')
```

In [56]:

```
import nltk

paragraph =  """I have visions for India. In 3000 years of our history, people have come
and invaded us"""


# Creating the TF-IDF model
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()
sentences = nltk.sent_tokenize(paragraph)
X = cv.fit_transform(sentences)
df= pd.DataFrame(X.toarray(), columns= cv.get_feature_names())
df
```

Out[56]:

	3000	and	come	for	have	history	in	india	invaded	of	our	people	
0	0.000000	0.000000	0.000000	0.534046	0.379978	0.000000	0.000000	0.534046	0.000000	0.000000	0.000000	0.000000	0.
1	0.294804	0.294804	0.294804	0.000000	0.209755	0.294804	0.294804	0.000000	0.294804	0.294804	0.294804	0.294804	0.

In [70]:

```
!python -m pip install -U gensim
```

```
Collecting gensim
  Downloading https://files.pythonhosted.org/packages/09/ed/b59a2edde05b7f5755ea68648487c
150c7c742361e9c8733c6d4ca005020/gensim-3.8.1-cp37-cp37m-win_amd64.whl (24.2MB)
Requirement already satisfied, skipping upgrade: numpy>=1.11.3 in c:\users\vinti\appdata\
roaming\python\python37\site-packages (from gensim) (1.17.4)
Collecting smart-open>=1.8.1 (from gensim)
  Using cached https://files.pythonhosted.org/packages/0c/09/735f2786dfac9bbf39d244ce75c0
313d27d4962e71e0774750dc809f2395/smart_open-1.9.0.tar.gz
Requirement already satisfied, skipping upgrade: six>=1.5.0 in c:\users\vinti\appdata\roa
ming\python\python37\site-packages (from gensim) (1.13.0)
Requirement already satisfied, skipping upgrade: scipy>=0.18.1 in c:\users\vinti\anaconda
3\lib\site-packages (from gensim) (1.3.1)
Requirement already satisfied, skipping upgrade: boto>=2.32 in c:\users\vinti\anaconda3\l
ib\site-packages (from smart-open>=1.8.1->gensim) (2.49.0)
Requirement already satisfied, skipping upgrade: requests in c:\users\vinti\appdata\roami
ng\python\python37\site-packages (from smart-open>=1.8.1->gensim) (2.22.0)
Collecting boto3 (from smart-open>=1.8.1->gensim)
  Downloading https://files.pythonhosted.org/packages/a5/bc/ffa0cef8de2bfadb3af1c2073eb96
32c0f0c766f24850b7291aefe69bcf7/boto3-1.12.6-py2.py3-none-any.whl (128kB)
Requirement already satisfied, skipping upgrade: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in c:\users\vinti\appdata\roaming\python\python37\site-packages (from requests->smart-ope
n>=1.8.1->gensim) (1.25.7)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in c:\users\vinti\app
data\roaming\python\python37\site-packages (from requests->smart-open>=1.8.1->gensim) (20
19.9.11)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in c:\users\vinti\
appdata\roaming\python\python37\site-packages (from requests->smart-open>=1.8.1->gensim)
(3.0.4)
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in c:\users\vinti\appdata
\roaming\python\python37\site-packages (from requests->smart-open>=1.8.1->gensim) (2.8)
Collecting jmespath<1.0.0,>=0.7.1 (from boto3->smart-open>=1.8.1->gensim)
  Downloading https://files.pythonhosted.org/packages/a3/43/1e939e1fcd87b827fe192d0c9fc25
b48c5b3368902bfb913de7754b0dc03/jmespath-0.9.5-py2.py3-none-any.whl
Collecting s3transfer<0.4.0,>=0.3.0 (from boto3->smart-open>=1.8.1->gensim)
  Using cached https://files.pythonhosted.org/packages/69/79/e6afb3d8b0b4e96cefbdc690f741
d7dd24547ff1f94240c997a26fa908d3/s3transfer-0.3.3-py2.py3-none-any.whl
Collecting botocore<1.16.0,>=1.15.6 (from boto3->smart-open>=1.8.1->gensim)
  Downloading https://files.pythonhosted.org/packages/d8/37/651779db53f36693992376f2d0659
933c8780fca58f616d0fff4a9728552/botocore-1.15.6-py2.py3-none-any.whl (5.9MB)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.1 in c:\users\
vinti\anaconda3\lib\site-packages (from botocore<1.16.0,>=1.15.6->boto3->smart-open>=1.8.
1->gensim) (2.7.5)
Requirement already satisfied, skipping upgrade: docutils<0.16,>=0.10 in c:\users\vinti\a
naconda3\lib\site-packages (from botocore<1.16.0,>=1.15.6->boto3->smart-open>=1.8.1->gens
im) (0.15.2)
Building wheels for collected packages: smart-open
  Building wheel for smart-open (setup.py): started
  Building wheel for smart-open (setup.py): finished with status 'done'
  Created wheel for smart-open: filename=smart_open-1.9.0-cp37-none-any.whl size=73092 sh
a256=4de88d3561745f708b84ddb10b474dfb4f447d99ef9d2d9190db8c1a017d044c
  Stored in directory: C:\Users\vinti\AppData\Local\pip\Cache\wheels\ab\10\93\5cff86f5b72
1d77edaecc29959b1c60d894be1f66d91407d28
Successfully built smart-open
Installing collected packages: jmespath, botocore, s3transfer, boto3, smart-open, gensim
Successfully installed boto3-1.12.6 botocore-1.15.6 gensim-3.8.1 jmespath-0.9.5 s3transfe
r-0.3.3 smart-open-1.9.0
```

In [1]:

```
import nltk
from gensim.models import Word2Vec
from nltk.corpus import stopwords
import re
paragraph = """I have three visions for India. In 3000 years of our history, people from
all over
             the world have come and invaded us, captured our lands, conquered our mind
```

```
s.
                From Alexander onwards, the Greeks, the Turks, the Moguls, the Portuguese,
the British,
                the French, the Dutch, all of them came and looted us, took over what was
ours.
                Yet we have not done this to any other nation. We have not conquered anyon
e.
                We have not grabbed their land, their culture,
                their history and tried to enforce our way of life on them.
                Why? Because we respect the freedom of others.That is why my
                first vision is that of freedom. I believe that India got its first visio
n of
                this in 1857, when we started the War of Independence. It is this freedom
that
                we must protect and nurture and build on. If we are not free, no one will
respect us.
                My second vision for India's development. For fifty years we have been a
developing nation.
                It is time we see ourselves as a developed nation. We are among the top 5
nations of the world
                in terms of GDP. We have a 10 percent growth rate in most areas. Our pover
ty levels are falling.
                Our achievements are being globally recognised today. Yet we lack the self
-confidence to
                see ourselves as a developed nation, self-reliant and self-assured. Isn't
this incorrect?
                I have a third vision. India must stand up to the world. Because I believ
e that unless India
                stands up to the world, no one will respect us. Only strength respects str
ength. We must be
                strong not only as a military power but also as an economic power. Both mu
st go hand-in-hand.
                My good fortune was to have worked with three great minds. Dr. Vikram Sar
abhai of the Dept. of
                space, Professor Satish Dhawan, who succeeded him and Dr. Brahm Prakash,
father of nuclear material.
                I was lucky to have worked with all three of them closely and consider thi
s the great opportunity of my life.
                I see four milestones in my career"""
# Preprocessing the data
review = re.sub('[^a-zA-Z]', ' ', paragraph)
# Preparing the dataset
sentences = nltk.sent_tokenize(review)
sentences = [nltk.word_tokenize(sentence) for sentence in sentences]

for i in range(len(sentences)):
    sentences[i] = [word for word in sentences[i] if word not in stopwords.words('englis
h')]
# Training the Word2Vec model
model = Word2Vec(sentences, min_count=1)
words = model.wv.vocab

# Finding Word Vectors
vector = model.wv['War']
print(vector)
```

```
[ 3.3291515e-03 -1.9087734e-03 -1.4865078e-03  4.3227598e-03
 -2.0933771e-03  3.0054867e-03 -4.5966739e-03 -1.4548304e-03
 -4.6438802e-04  2.8921606e-03  1.9147887e-03 -1.5017448e-03
 -1.7856661e-03 -6.6398300e-04 -3.2864604e-03  2.2197445e-03
 -2.1672915e-03  1.4730210e-03 -3.6893750e-03  4.1312026e-03
  1.0475852e-03  1.6886365e-03 -1.9643459e-04 -3.8194784e-03
 -4.9211276e-03  4.4149095e-03 -2.6940438e-03  4.3267669e-04
  3.3932887e-03 -2.2869154e-03 -2.4002646e-03  1.6368385e-03
  3.8046195e-04 -2.4277857e-03  1.0705636e-03 -3.6923500e-04
 -1.9266544e-03  2.9768907e-03  4.6489141e-03  1.9019864e-03
  1.9591414e-04  2.8300409e-03 -4.6452968e-03 -2.5035783e-03
  3.5515221e-04 -2.1060600e-03 -4.7459225e-03 -3.6633173e-03
  8.6061373e-05  2.5354894e-03  1.9727182e-04  2.3900943e-04
 -4.7866930e-03  4.9839523e-03 -4.7437609e-03  4.0984415e-03
 -2.3877325e-03 -1.3153395e-03  2.1733081e-03 -2.9240672e-03
  2.3473229e-03 -2.0231518e-03  1.4679737e-03 -4.2611114e-03
```

```
 2.0707175e-03  3.6822788e-03  3.0437583e-04 -4.8873341e-03
-3.0707675e-03 -3.5392612e-04  1.5125524e-03  1.2713447e-04
 1.3945980e-03 -1.6426180e-03 -5.6841457e-04  3.3275848e-03
 2.3567670e-03 -7.1900350e-04 -1.0003500e-03  4.0262579e-03
 4.8180027e-03  3.9921193e-03 -6.6354044e-04  3.2855188e-03
 2.0033494e-03 -2.0155038e-03  3.4028639e-03 -2.6687763e-03
-1.9629521e-03 -4.2625633e-03  4.0843342e-03  1.2102015e-03
-3.7068473e-03  2.1465707e-03 -4.4148574e-03 -4.5918121e-05
 2.7090786e-03  4.2047054e-03  1.4919359e-03  2.6914137e-04]
```

In [9]:

```
# Most similar words
similar = model.wv.most_similar('closely')
similar[0]
```

Out[9]:

```
('levels', 0.22647951543331146)
```

In [12]:

```
model.wv.similarity(w1='history',w2='world')
```

Out[12]:

```
0.037967358
```