In [1]:
```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to /home/TE/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /home/TE/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/TE/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to /home/TE/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [18]:
```
text='Real madrid is set to win the UCL for the season . Benzema might
text2 = 'I felt happy because I saw that others were happy . And becau
```

In [3]:
```
tokens_sents = nltk.sent_tokenize(text)
print(tokens_sents)
```

```
['Real madrid is set to win the UCL for the season .', 'Benzema might
win Balon dor .', 'Salah might be the runner up']
```

In [4]:
```
tokens_words = nltk.word_tokenize(text)
print(tokens_words)
```

```
['Real', 'madrid', 'is', 'set', 'to', 'win', 'the', 'UCL', 'for', 'th
e', 'season', '.', 'Benzema', 'might', 'win', 'Balon', 'dor', '.', 'S
alah', 'might', 'be', 'the', 'runner', 'up']
```

In [5]:
```
from nltk.stem import PorterStemmer
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import LancasterStemmer
```

In [6]:
```
stem=[]
for i in tokens_words:
  ps = PorterStemmer()
  stem_word= ps.stem(i)
  stem.append(stem_word)
print(stem)
```

```
['real', 'madrid', 'is', 'set', 'to', 'win', 'the', 'ucl', 'for', 'th
e', 'season', '.', 'benzema', 'might', 'win', 'balon', 'dor', '.', 's
alah', 'might', 'be', 'the', 'runner', 'up']
```

Lemmatization

In [7]:
```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [8]:
```
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in stem])
print(lemmatized_output)
```

```
real madrid is set to win the ucl for the season . benzema might win
balon dor . salah might be the runner up
```

In [9]:
```
leme=[]
for i in stem:
  lemetized_word=lemmatizer.lemmatize(i)
  leme.append(lemetized_word)
print(leme)
```

```
['real', 'madrid', 'is', 'set', 'to', 'win', 'the', 'ucl', 'for', 'th
e', 'season', '.', 'benzema', 'might', 'win', 'balon', 'dor', '.', 's
alah', 'might', 'be', 'the', 'runner', 'up']
```

In [10]:
```
print("Parts of Speech: ",nltk.pos_tag(leme))
```

```
Parts of Speech:  [('real', 'JJ'), ('madrid', 'NN'), ('is', 'VBZ'),
('set', 'VBN'), ('to', 'TO'), ('win', 'VB'), ('the', 'DT'), ('ucl',
'NN'), ('for', 'IN'), ('the', 'DT'), ('season', 'NN'), ('.', '.'),
('benzema', 'NN'), ('might', 'MD'), ('win', 'VB'), ('balon', 'NN'),
('dor', 'NN'), ('.', '.'), ('salah', 'NN'), ('might', 'MD'), ('be',
'VB'), ('the', 'DT'), ('runner', 'NN'), ('up', 'RP')]
```

In [11]:
```
sw_nltk = stopwords.words('english')
print(sw_nltk)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'th
eir', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'wer
e', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'doe
s', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'beca
use', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'abou
t', 'against', 'between', 'into', 'through', 'during', 'before', 'aft
er', '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',
'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not',
'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can',
'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'l
l', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "co
uldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'has
n', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mi
ghtn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 's
houldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "w
on't", 'wouldn', "wouldn't"]
```

In [12]:
```
words = [word for word in text.split() if word.lower() not in sw_nltk]
new_text = " ".join(words)
print(new_text)
```

```
Real madrid set win UCL season . Benzema might win Balon dor . Salah
might runner
```

In [13]:
```
from nltk.probability import FreqDist
freq_of_words = FreqDist(tokens_words)
```

In [14]:
```
freq_of_words
```

Out[14]:
```
FreqDist({'the': 3, 'win': 2, '.': 2, 'might': 2, 'Real': 1, 'madri
d': 1, 'is': 1, 'set': 1, 'to': 1, 'UCL': 1, ...})
```

In [19]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
string = [text2]
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)
print('\nIDF Values: ')
for ele1,ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ':', ele2)

print('\nWord Indexes:')
print(tfidf.vocabulary_)


print('\nTf-Idf Values:')
print(result.toarray())
```

```
IDF Values:
and : 1.0
because : 1.0
but : 1.0
feel : 1.0
felt : 1.0
happy : 1.0
knew : 1.0
others : 1.0
really : 1.0
saw : 1.0
should : 1.0
that : 1.0
wasn : 1.0
were : 1.0

Word Indexes:
{'felt': 4, 'happy': 5, 'because': 1, 'saw': 9, 'that': 11, 'others':
7, 'were': 13, 'and': 0, 'knew': 6, 'should': 10, 'feel': 3, 'but':
2, 'wasn': 12, 'really': 8}

Tf-Idf Values:
[[0.1767767  0.35355339 0.1767767  0.1767767  0.1767767  0.70710678
  0.1767767  0.1767767  0.1767767  0.1767767  0.1767767  0.1767767
  0.1767767  0.1767767 ]]
```

In [ ]:
```

```