Natural Language Processing

```
:
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk import FreqDist
```

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk import FreqDist
```

```
ted by David Attenborough. The story would have probably been more interesting than talking reali
```

```
#Tokenize words and sentences from review

word_token = word_tokenize(review.lower())

print(word_token)
```

```
#Tokenize words and sentences from review

word_token = word_tokenize(review.lower())

print(word_token)
```

```
['the', 'animators', 'deserve', 'five', 'stars', 'as', 'the', 'scenery', 'and', 'anim
o', 'keep', 'reminding', 'myself', 'that', 'this', 'was', 'animated', 'and', 'not',
'the', 'very', 'first', 'opening', 'scene', 'was', 'filmed', '.', 'otherwise', ',',
k', 'of', 'animators', '.', 'amazing', '.', 'however', ',', 'as', 'a', 'story', 'it',
s', 'very', 'disappointed', '.', 'several', 'songs', 'are', 'changed', ',', 'and', 'm
e', 'scaled', 'way', 'back', 'and', 'lose', 'their', 'charm', 'completely', '.', 'can
t', ',', 'during', 'the', 'day', '?', '?', '?', 'late', 'afternoon', 'at', 'best', '.
s', ',', 'they', 'were', 'all', 'disappointed', 'except', 'the', 'two', 'year', 'old'
```

```python
# filter out stopwords

mystopwords = set(stopwords.words("english"))

filtered_words = []
for word in word_token:
    if word not in mystopwords and word.isalpha():
        filtered_words.append(word)

print(filtered_words)
```

```
['animators', 'deserve', 'five', 'stars', 'scenery', 'animals', 'lovely', 'keep', 'reminding'
ilm', 'first', 'opening', 'scene', 'filmed', 'otherwise', 'entire', 'movie', 'work', 'animato
y', 'really', 'falls', 'flat', 'disappointed', 'several', 'songs', 'changed', 'many', 'iconic
k', 'lose', 'charm', 'completely', 'feel', 'love', 'tonight', 'day', 'late', 'afternoon', 'be
```

```
# stemming - eliminating affixes (suffixed, prefixes, infixes, circumfixes) to obtain a word stem

ps = PorterStemmer()

stemmed_words=[]
for word in filtered_words:
    stemmed_words.append(ps.stem(word))

print("stemmed words: ",stemmed_words)
```

```
stemmed words:  ['anim', 'deserv', 'five', 'star', 'sceneri', 'anim', 'love', 'keep', 'remind', 'a
m', 'first', 'open', 'scene', 'film', 'otherwis', 'entir', 'movi', 'work', 'anim', 'amaz', 'howev'
'flat', 'disappoint', 'sever', 'song', 'chang', 'mani', 'icon', 'scene', 'scale', 'way', 'back', '
eel', 'love', 'tonight', 'day', 'late', 'afternoon', 'best', 'weird', 'four', 'kid', 'disappoint',
d', 'thought', 'pretti', 'great', 'point', 'thought', 'would', 'build', 'good', 'pun', 'excit', 's
```

```
# Calculating frequency of words
freq = FreqDist(stemmed_words)

for word, frequency in freq.most_common(5):
      print("{}:{}".format(word, frequency))

#plot frequency for top 10
freq.plot(10,title="Top 10 from Review", linewidth=10, color="g")
```

```
anim:7
would:4
love:3
scene:3
movi:3
```

NLP Lemmatizer

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

Word = "skies"

print(lemmatizer.lemmatize(Word))
```

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

lemmatizer = WordNetLemmatizer()

Word = "skies"

print(lemmatizer.lemmatize(Word))
```

```
sky
```

```
print(lemmatizer.lemmatize(Word, pos = "v"))
```

```
write
```

In [3]:
```
Word = "better"
print(lemmatizer.lemmatize(Word, pos ="a"))
```

```
good
```

In [4]:
```
Word = "feet"
print(lemmatizer.lemmatize(Word))
```

```
foot
```

In [5]:
```
Word = "exhausted"
print(lemmatizer.lemmatize(Word, pos = "v"))
```

```
exhaust
```

VIDEOS

Lion King Review

```
# Read Text File and read lines
myfile = open("lionkingreviews.txt", "r")
lines = myfile.readlines()
myfile.close()
```

```
lines
```

```
mystr=""
for line in lines:

    mystr= mystr+line

print(mystr)
```

Lion King Review

Sentiment

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer as stm
```

```
stmscore = stm().polarity_scores(mystr)
print(stmscore)
```

```
print(stmscore['compound'])
if stmscore['compound'] > 0.9:
    print('reviewers are really happy with this movie')
elif stmscore['compound'] > 0.7:
    print('reviewers are somewhat happy with this movie')
else:
    print('They don\'t like it!')
```

In [ ]:
```
# Read Text File and read lines
myfile = open("lionkingreviews.txt", "r")
lines = myfile.readlines()
myfile.close()
```

In [ ]:
```
lines
```

In [ ]:
```
mystr=""
for line in lines:

    mystr= mystr+line

print(mystr)
```

In [ ]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer as stm
```

In [ ]:
```
stmscore = stm().polarity_scores(mystr)
print(stmscore)
```

In [ ]:
```
print(stmscore['compound'])
if stmscore['compound'] > 0.9:
    print('reviewers are really happy with this movie')
elif stmscore['compound'] > 0.7:
    print('reviewers are somewhat happy with this movie')
```