

Assignment 4.2

```

import nltk
import spacy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer # Import TfidfVectorizer
nltk.download('wordnet')
nltk.download("stopwords")
nltk.download('punkt_tab')

[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True

```

```
a=pd.read_csv('/content/IMDB Dataset.csv')
```

```
a.head()
```

	review	sentiment	
0	One of the other reviewers has mentioned that ...	positive	
1	A wonderful little production. The...	positive	
2	I thought this was a wonderful way to spend ti...	positive	
3	Basically there's a family where a little boy ...	negative	
4	Petter Mattei's "Love in the Time of Money" is...	positive	

Next steps: [Generate code with a](#) [New interactive sheet](#)

Sentence Tokenization

```

nltk_sentences = sent_tokenize(a['review'].iloc[0])
print("NLTK Sentences:")
nltk_sentences

```

```

NLTK Sentences:
["One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked.", 
 'They are right, as this is exactly what happened with me.<br /><br />The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO.', 
 'Trust me, this is not a show for the faint hearted or timid.', 
 'This show pulls no punches with regards to drugs, sex or violence.', 
 'Its is hardcore, in the classic use of the word.<br /><br />It is called OZ as that is the

```

nickname given to the Oswald Maximum Security State Penitentiary.',
'It focuses mainly on Emerald City, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda.',
"Em City is home to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away.

I would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare.",
"Forget pretty pictures painted for mainstream audiences, forget charm, forget romance...OZ doesn't mess around.",
"The first episode I ever saw struck me as so nasty it was surreal, I couldn't say I was ready for it, but as I watched more, I developed a taste for Oz, and got accustomed to the high levels of graphic violence.",
"Not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) Watching Oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side."]

Word Tokenization

```
nltk_words = word_tokenize(a['review'].iloc[0])
print("NLTK Words:")
nltk_words
```

```
uncomfortable ,  
'viewing' ,  
'....' ,  
'thats' ,  
'if' ,  
'you' ,  
'can' ,  
'get' ,  
'in' ,  
'touch' ,  
'with' ,  
'your' ,  
'darker' ,  
'side' ,  
'.' ]
```

▼ Stemming

```
stemmer = PorterStemmer()  
words = nltk_words  
stemmed_words = [stemmer.stem(word) for word in words]  
stemmed_words
```

```
....',
'that',
'if',
'you',
'can',
'get',
'in',
'touch',
'with',
'your',
'darker',
'side',
' ']
```

▼ Lemmatization

```
lemmatizer = WordNetLemmatizer()
nltk_lemmas = [lemmatizer.lemmatize(token) for token in nltk_words]
print("NLTK Lemmatization Output:")
nltk_lemmas
```

'IT',
'you',
'can',
'get',
'in',
'touch',
'with',
'your',
'darker',
'side',
'']

TT B I <> ☰ 🖼 “ ≡ ≡ – ψ 😊 📁 Close

Stop words

```
stop_words = set(stopwords.words("english"))
filtered_list = []
for word in words: # Changed 'words_in_quote' to 'words'
    if word.casfold() not in stop_words:
        filtered_list.append(word)
print(filtered_list)
```

```
watching ,  
'Oz',  
'',  
'may',  
'become',  
'comfortable',  
'uncomfortable',  
'viewing',  
'....',  
'thats',  
'get',  
'touch',  
'darker',  
'side',  
'..']
```

```
positive_reviews = a[a['sentiment'] == 'positive']['review']  
negative_reviews = a[a['sentiment'] == 'negative']['review']
```

```
positive_reviews
```

	review
0	One of the other reviewers has mentioned that ...
1	A wonderful little production. The...
2	I thought this was a wonderful way to spend ti...
4	Petter Mattei's "Love in the Time of Money" is...
5	Probably my all-time favorite movie, a story o...
...	...
49983	I loved it, having been a fan of the original ...
49985	Imaginary Heroes is clearly the best film of t...
49989	I got this one a few weeks ago and love it! It...
49992	John Garfield plays a Marine who is blinded by...
49995	I thought this movie did a down right good job...

25000 rows × 1 columns

dtype: object

```
negative_reviews
```

review

3 Basically there's a family where a little boy ...
 7 This show was an amazing, fresh & innovative i...
 8 Encouraged by the positive comments about this...
 10 Phil the Alien is one of those quirky films wh...
 11 I saw this movie when I was about 12 when it c...
 ...
 49994 This is your typical junk comedy.

T...
 49996 Bad plot, bad dialogue, bad acting, idiotic di...
 49997 I am a Catholic taught in parochial elementary...
 49998 I'm going to have to disagree with the previou...
 49999 No one expects the Star Trek movies to be high...

25000 rows × 1 columns

dtype: object

```
tfidf_pos = TfidfVectorizer(stop_words=list(stop_words), max_features=5000)
tfidf_neg = TfidfVectorizer(stop_words=list(stop_words), max_features=5000)
```

tfidf_pos

```
▼ TfidfVectorizer
TfidfVectorizer(max_features=5000,
stop_words={'a', 'about', 'above', 'after', 'again', 'against',
'ain', 'all', 'am', 'an', 'and', 'any', 'are',
'aren', "aren't", 'as', 'at', 'be', 'because',
'veen', 'before', 'being', 'below', 'between',
'both', 'but', 'by', 'can', 'couldn', "couldn't", ...})
```

tfidf_neg

```
▼ TfidfVectorizer
TfidfVectorizer(max_features=5000,
stop_words={'a', 'about', 'above', 'after', 'again', 'against',
'ain', 'all', 'am', 'an', 'and', 'any', 'are',
'aren', "aren't", 'as', 'at', 'be', 'because",
'veen', 'before', 'being', 'below', 'between',
'both', 'but', 'by', 'can', 'couldn', "couldn't", ...})
```

```
pos_tfidf_matrix = tfidf_pos.fit_transform(positive_reviews)
neg_tfidf_matrix = tfidf_neg.fit_transform(negative_reviews)

pos_scores = np.mean(pos_tfidf_matrix.toarray(), axis=0)
neg_scores = np.mean(neg_tfidf_matrix.toarray(), axis=0)
```

pos_scores

```
array([0.00044712, 0.00092389, 0.01145975, ..., 0.00117221, 0.0007349 ,
0.00063394])
```

```
neg_scores
```

```
array([0.00056652, 0.00120859, 0.01046598, ..., 0.00292957, 0.00233863,  
0.00064824])
```

```
pos_terms = tfidf_pos.get_feature_names_out()  
neg_terms = tfidf_neg.get_feature_names_out()
```

```
pos_terms
```

```
array(['00', '000', '10', ..., 'zombie', 'zombies', 'zone'], dtype=object)
```

```
neg_terms
```

```
array(['00', '000', '10', ..., 'zombie', 'zombies', 'zone'], dtype=object)
```

```
pos_df = pd.DataFrame({'term': pos_terms, 'score': pos_scores})  
neg_df = pd.DataFrame({'term': neg_terms, 'score': neg_scores})
```

```
pos_df
```

	term	score	grid icon
0	00	0.000447	edit icon
1	000	0.000924	
2	10	0.011460	
3	100	0.001673	
4	11	0.001471	
...	
4995	youth	0.001322	
4996	zero	0.000708	
4997	zombie	0.001172	
4998	zombies	0.000735	
4999	zone	0.000634	

5000 rows × 2 columns

Next steps: [Generate code with pos_df](#) [New interactive sheet](#)

```
neg_df
```

	term	score	
0	00	0.000567	
1	000	0.001209	
2	10	0.010466	
3	100	0.001896	
4	101	0.000426	
...	
4995	zero	0.002001	
4996	zoey	0.000380	
4997	zombie	0.002930	
4998	zombies	0.002339	
4999	zone	0.000648	

5000 rows × 2 columns

Next steps: [Generate code with neg_df](#) [New interactive sheet](#)

```
top_pos = pos_df.sort_values(by='score', ascending=False).head(15)
top_neg = neg_df.sort_values(by='score', ascending=False).head(15)
```

top_pos

	term	score	
560	br	0.104987	
2950	movie	0.052894	
1745	film	0.049761	
3135	one	0.032037	
2622	like	0.025213	
1969	good	0.024688	
1999	great	0.023290	
4256	story	0.021466	
3911	see	0.020986	
4873	well	0.020381	
4531	time	0.019942	
3593	really	0.019655	
4968	would	0.017881	
2688	love	0.017417	
210	also	0.017007	

Next steps: [Generate code with top_pos](#) [New interactive sheet](#)

top_neg

	term	score	
551	br	0.113804	
2937	movie	0.061162	
1723	film	0.047146	
3125	one	0.032345	
2614	like	0.029371	
379	bad	0.026157	
1946	good	0.023867	
1539	even	0.023061	
4960	would	0.022204	
3572	really	0.021262	
4509	time	0.020389	
4238	story	0.018907	
3878	see	0.018872	
2938	movies	0.017632	
2945	much	0.017500	

Next steps: [Generate code with top_neg](#) [New interactive sheet](#)

Side-by-side bar charts comparing both sets

```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
axes[0].barh(top_pos['term'], top_pos['score'])
axes[0].set_title("Top 15 TF-IDF Terms (Positive Reviews)")
axes[0].invert_yaxis()
axes[1].barh(top_neg['term'], top_neg['score'])
axes[1].set_title("Top 15 TF-IDF Terms (Negative Reviews)")
axes[1].invert_yaxis()
plt.tight_layout()
plt.show()
```

