

Name: Akhil.B

Roll Number: 2403a52344

Assignment Number: 04

```
# Analyse how TD-IDF differs between positive and negative reviews in python
```

```
import nltk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import movie_reviews, stopwords
from nltk.tokenize import word_tokenize
import string
```

```
plt.style.use('seaborn-v0_8-darkgrid')
```

```
# Downloading Required NLTK data
nltk.download('movie_reviews')
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
# Load Movie Reviews
postid = movie_reviews.fileids('pos')
negid = movie_reviews.fileids('neg')

print(f'Number of positive reviews: {len(postid)}')
print(f'Number of negative reviews: {len(negid)}')

Number of positive reviews: 1000
Number of negative reviews: 1000
```

```
# Defining Pre-Processing function
def preprocess_text(text):
    text = text.lower()

    tokens = word_tokenize(text)

    # Remove Punctuation and Stop Words
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens
              if token not in string.punctuation
              and token not in stop_words
              and len(token) > 3]
    return ' '.join(tokens)
```

```
nltk.download('punkt_tab')
# Create positive corpus
print("Processing positive reviews...")
positive_reviews = []
for fileid in postid:
    raw_text = movie_reviews.raw(fileid)
    preprocessed_text = preprocess_text(raw_text)
    positive_reviews.append(preprocessed_text)

[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
Processing positive reviews...
```

```
print("\nProcessing negative reviews...")
negative_reviews = []
for fileid in negid:
    raw_text = movie_reviews.raw(fileid)
    preprocessed_text = preprocess_text(raw_text)
    negative_reviews.append(preprocessed_text)
```

```
Processing negative reviews...
```

```
print(f"Processed {len(negative_reviews)} negative reviews")
print(f"Processed {len(positive_reviews)} positive reviews")
```

```
Processed 1000 negative reviews
Processed 1000 positive reviews
```

```
print("First 200 characters of positive reviews\n")
print(positive_reviews[0][:200] + "...\\n")
print("First 200 characters of negative reviews \\n")
print(negative_reviews[0][:200] + "...")
```

```
First 200 characters of positive reviews
```

```
films adapted comic books plenty success whether superheroes batman superman spawn geared toward kids casper arthouse crowd
```

```
First 200 characters of negative reviews
```

```
plot teen couples church party drink drive accident guys dies girlfriend continues life nightmares deal watch movie sorta fi
```

```
print("Computing TF-IDF for positive reviews...")
tfidf_vectorizer_pos = TfidfVectorizer(max_features=1000, min_df=2, max_df=0.8)
tfidf_matrix_pos = tfidf_vectorizer_pos.fit_transform(positive_reviews)
```

```
feature_names_pos = tfidf_vectorizer_pos.get_feature_names_out()
```

```
print(f"TF-IDF matrix shape: {tfidf_matrix_pos.shape}")
print(f"Number of unique terms: {len(feature_names_pos)}")
```

```
Computing TF-IDF for positive reviews...
```

```
TF-IDF matrix shape: (1000, 1000)
```

```
Number of unique terms: 1000
```

```
print("Computing TF-IDF for negative reviews...")
tfidf_vectorizer_neg = TfidfVectorizer(max_features=1000, min_df=2, max_df=0.8)
tfidf_matrix_neg = tfidf_vectorizer_neg.fit_transform(negative_reviews)
feature_names_neg = tfidf_vectorizer_neg.get_feature_names_out()
```

```
print(f"TF-IDF matrix shape: {tfidf_matrix_neg.shape}")
print(f"Number of unique terms: {len(feature_names_neg)}")
```

```
Computing TF-IDF for negative reviews...
```

```
TF-IDF matrix shape: (1000, 1000)
```

```
Number of unique terms: 1000
```

```
# Calculate mean TF-IDF scores for positive reviews
mean_tfidf_pos = np.array(tfidf_matrix_pos.mean(axis=0)).flatten()
top_indices_pos = mean_tfidf_pos.argsort()[-15:][::-1]
top_terms_pos = [(feature_names_pos[i], mean_tfidf_pos[i]) for i in top_indices_pos]
```

```
# Display top terms for positive reviews
print("Top 15 TF-IDF Terms in Positive Reviews: \\n")
for i, (term, score) in enumerate(top_terms_pos, 1):
    print(f"{i:2d}. {term:20s} - Score: {score:.6f}")
```

```
# Create DataFrame for positive terms
df_pos = pd.DataFrame(top_terms_pos, columns=['Term', 'TF-IDF Score'])
df_pos['Rank'] = range(1, 16)
print("\\n", df_pos)
```

```
Top 15 TF-IDF Terms in Positive Reviews:
```

1. movie	- Score: 0.065428
2. like	- Score: 0.043895
3. story	- Score: 0.036013
4. good	- Score: 0.035094
5. life	- Score: 0.034882
6. time	- Score: 0.033522
7. also	- Score: 0.033068
8. well	- Score: 0.032244
9. character	- Score: 0.031756
10. even	- Score: 0.031469
11. would	- Score: 0.030993
12. characters	- Score: 0.030914
13. much	- Score: 0.029628
14. first	- Score: 0.028960
15. films	- Score: 0.027729

	Term	TF-IDF Score	Rank
0	movie	0.065428	1
1	like	0.043895	2

```

2     story    0.036013   3
3     good     0.035094   4
4     life     0.034882   5
5     time     0.033522   6
6     also     0.033068   7
7     well     0.032244   8
8     character 0.031756   9
9     even     0.031469  10
10    would   0.030993  11
11    characters 0.030914  12
12    much     0.029628  13
13    first    0.028960  14
14    films    0.027729  15

```

```

# Calculate mean TF-IDF scores for negative reviews
mean_tfidf_neg = np.array(tfidf_matrix_neg.mean(axis=0)).flatten()
top_indices_neg = mean_tfidf_neg.argsort()[-15:][::-1]
top_terms_neg = [(feature_names_neg[i], mean_tfidf_neg[i]) for i in top_indices_neg]

# Display top terms for negative reviews
print("Top 15 TF-IDF Terms in Negative Reviews:")
print("*" * 50)
for i, (term, score) in enumerate(top_terms_neg, 1):
    print(f"{i:2d}. {term:20s} - Score: {score:.6f}")

# Create DataFrame for negative terms
df_neg = pd.DataFrame(top_terms_neg, columns=['Term', 'TF-IDF Score'])
df_neg['Rank'] = range(1, 16)
print("\n", df_neg)

```

Top 15 TF-IDF Terms in Negative Reviews:

```

=====
1. like      - Score: 0.048473
2. even      - Score: 0.038923
3. would    - Score: 0.036593
4. good      - Score: 0.036422
5. time      - Score: 0.035171
6. story     - Score: 0.033669
7. much      - Score: 0.032333
8. plot      - Score: 0.031488
9. character - Score: 0.031369
10. could   - Score: 0.030768
11. characters - Score: 0.030731
12. make     - Score: 0.028763
13. really   - Score: 0.028590
14. first    - Score: 0.028579
15. action   - Score: 0.028426

```

	Term	TF-IDF Score	Rank
0	like	0.048473	1
1	even	0.038923	2
2	would	0.036593	3
3	good	0.036422	4
4	time	0.035171	5
5	story	0.033669	6
6	much	0.032333	7
7	plot	0.031488	8
8	character	0.031369	9
9	could	0.030768	10
10	characters	0.030731	11
11	make	0.028763	12
12	really	0.028590	13
13	first	0.028579	14
14	action	0.028426	15

```

# Create side-by-side bar charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 8))

# Extract terms and scores
terms_pos = [term for term, _ in top_terms_pos]
scores_pos = [score for _, score in top_terms_pos]

terms_neg = [term for term, _ in top_terms_neg]
scores_neg = [score for _, score in top_terms_neg]

bars1 = ax1.bart(range(len(terms_pos)), scores_pos, color='#2ecc71', alpha=0.8, edgecolor='black')
ax1.set_yticks(range(len(terms_pos)))
ax1.set_yticklabels(terms_pos, fontsize=10)
ax1.invert_yaxis()
ax1.set_xlabel('Mean TF-IDF Score', fontsize=12, fontweight='bold')
ax1.set_title('Top 15 TF-IDF Terms in Positive Reviews', fontsize=14, fontweight='bold', pad=20)
ax1.grid(axis='x', alpha=0.3, linestyle='--')

# Add value labels on bars
for i, (bar, score) in enumerate(zip(bars1, scores_pos)):

```

```

    ax1.text(score + 0.0001, i, f'{score:.4f}', va='center', fontsize=9)

# Negative reviews bar chart
bars2 = ax2.barr(range(len(terms_neg)), scores_neg, color='#e74c3c', alpha=0.8, edgecolor='black')
ax2.set_yticks(range(len(terms_neg)))
ax2.set_yticklabels(terms_neg, fontsize=10)
ax2.invert_yaxis()
ax2.set_xlabel('Mean TF-IDF Score', fontsize=12, fontweight='bold')
ax2.set_title('Top 15 TF-IDF Terms in Negative Reviews', fontsize=14, fontweight='bold', pad=20)
ax2.grid(axis='x', alpha=0.3, linestyle='--')

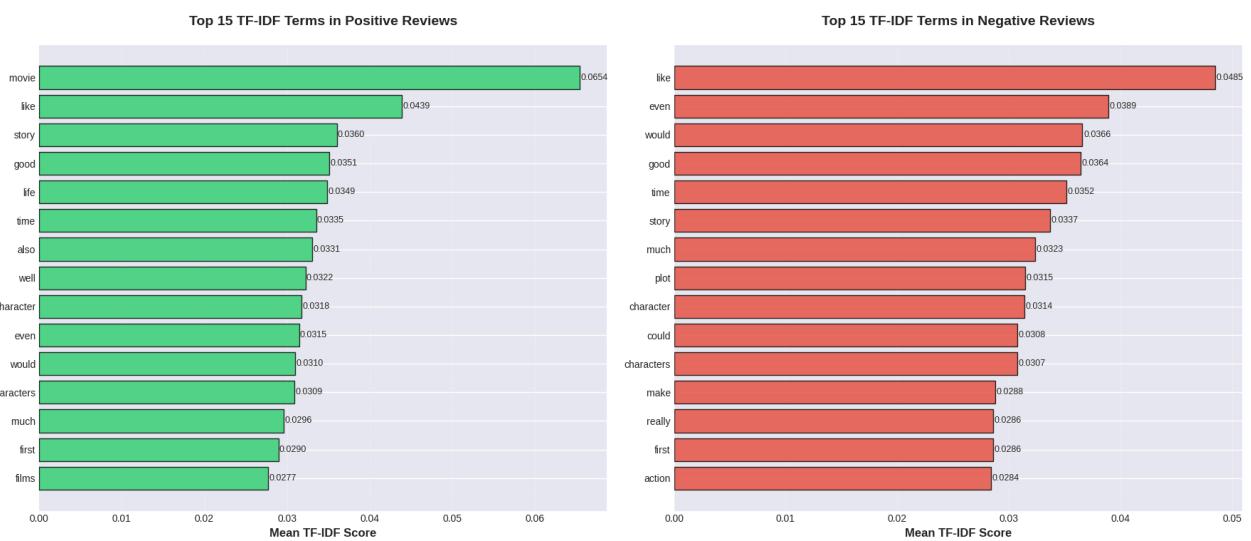
# Add value labels on bars
for i, (bar, score) in enumerate(zip(bars2, scores_neg)):
    ax2.text(score + 0.0001, i, f'{score:.4f}', va='center', fontsize=9)

plt.suptitle('TF-IDF Based Vocabulary Analysis: Positive vs Negative Movie Reviews',
             fontsize=16, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()

print("\n✓ Visualization complete!")

```

TF-IDF Based Vocabulary Analysis: Positive vs Negative Movie Reviews



✓ Visualization complete!

```

# Compare Common and unique terms
common_terms = set(feature_names_pos) & set(feature_names_neg)
unique_terms_pos = set(feature_names_pos) - common_terms
unique_terms_neg = set(feature_names_neg) - common_terms

print(f"Number of common terms: {len(common_terms)}")
print(f"Number of unique terms in positive reviews: {len(unique_terms_pos)}")
print(f"Number of unique terms in negative reviews: {len(unique_terms_neg)}")

```

Number of common terms: 797
Number of unique terms in positive reviews: 203
Number of unique terms in negative reviews: 203

