SOC: NLP & Sentiment Analysis Assignment-1

Name: *Parth Maheshwari*
Dataset: Twitter US Airline Sentiment
Tools Used: NLTK, scikit-learn, matplotlib, seaborn, TextBlob/VADER, gensim
Date: *8 July 2025*

Q1. Pre-processing & Token Analysis

Sample Text used:

```
sample_texts = [
    "Orphan is flawed 2009 psychological horror film that will keep you hooked via its characters and Isabelle Fuhrman's portrayal of Esther. Despite the flaws in storytelling, it still has it moments of fantastic horror.",

    "I enjoyed much of this but there were a few silly moments and I feel if the film had been cut down to 90 minutes we could have got rid of those embarrasing bits and made the whole ting tighter. The acting is fine, especially the two girls and that of Isabelle Fuhrman, who at times seems to be carrying the film, is remarkable. Inevitably, perhaps there are unfortunately those times, as in 'gas lighting' films where we sit frustrated urging those on screen to notice the obvious to no avail and then on we go as everything gets more and more embroiled.",

    "The movie is overall good but the husband in this movie is stupid",

    "You either love it or you hate it. That's really the effect of any great twist. I was one of those who absolutely loved it and found this creepy, disturbing movie even more terrifying with the reveal. Vera Farmiga (Conjuring) stars as a mother whose just gone through a miscarriage and with her husband played by Peter Sarsgaard (Skeleton Key) decides to adopt a mysterious young girl played by a very creepy Isabel Fuhrman (Hunger Games). At first Esther seems a little off but it's mostly acceptable until people begin having horrific accidents and some go missing. There is a palpable sense of dread that builds throughout the film and so many setpieces are expertly executed. The scene in the playground is an especially effective one. The film culminates in a surprising and edge of your seat fashion and all though there's a small loop hole here and there. Its completely successful at being a stylish, unique, intense piece of genre.",

    "Loved the movie, it's just so disturbing and thrilling and suspenseful, the opening scene is very perturbing as the baby covered in blood and the husband taping her as the poor woman miscarriages is disgusting and chills me. Now I really loved Esther and I really clapped in the theater when she dumped the bratty girl down the slide. The fluorescent paint is a good touch.
```

Now people should realize the movie has a whole sexual climax and the movie's theme is about Esther trying to find her sexuality. We find that out on the twist when we realize she was a 35 year old girl with dwarfism and she wanted to find her sexual manners around men who think she is a girl. And I was really disturbed when the men found the fluorescent paintings in the walls with she and him having sex and all of them are dead except for them. The only reason I gave it an 8 was because of the 2 hours with 3 minutes length and had an unnecessary long character development which almost had me sleeping in the movie theater but then I got woken up by the intensity that followed the the boringness."
]

Word	**Token**	**Stopword Removed**	**Stemmed**	**Lammatized**	**POS**
Orphan	True	True	orphan	Orphan	NNP
flawed	True	True	flaw	flawed	VBD
psychological	True	True	psycholog	psychological	JJ
horror	True	True	horror	horror	NN
characters	True	True	charact	character	NNS

Used NLTK tools for tokenization, stop-word removal, stemming (Porter), lemmatization (WordNet), and POS tagging.

Q2. Vectorization Comparison

Used 20 tweets from the Twitter US Airline Sentiment Dataset.

Code:

```
# Load the dataset
df = pd.read_csv('Tweets.csv')

# Show basic info
print(df.shape)
df[['airline_sentiment', 'text']].head()

# Take the first 20 tweets
sample_texts = df['text'].head(20).tolist()
```

Bag of Words (BoW)

```
from sklearn.feature_extraction.text import CountVectorizer

bow_vectorizer = CountVectorizer()
bow_features = bow_vectorizer.fit_transform(sample_texts)

print("Bag of Words Shape:", bow_features.shape)
```

TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer()
tfidf_features = tfidf_vectorizer.fit_transform(sample_texts)

print("TF-IDF Shape:", tfidf_features.shape)
```

Word Embeddings

```
import nltk
nltk.download('punkt')

from nltk.tokenize import word_tokenize
from gensim.models import Word2Vec

# Preprocess: lowercase and tokenize
tokenized_texts = [word_tokenize(text.lower()) for text in sample_texts]

# Train Word2Vec model
w2v_model = Word2Vec(sentences=tokenized_texts, vector_size=100, window=5,
min_count=1, workers=4)

# Example embedding
print("Vector for 'flight':", w2v_model.wv['flight'][:10])
```

Matrix Shapes:

Vectorization	Matrix Shape
BoW	(20, 178)
TF-IDF	(20, 178)
Word2Vec	vocab size: 205

Comparison:

Among the three, **Word Embeddings** clearly capture semantics best, as they encode contextual similarity between words into dense vectors, unlike BoW and TF-IDF which rely on surface-level frequency statistics. Between BoW and TF-IDF, TF-IDF is preferable as it emphasizes informative words and reduces the weight of common ones, making it more effective for traditional text classification.

Q3. Text Classification: Logistic Regression vs Naive Bayes

Used 50 labeled tweets with sentiment labels (positive, neutral, negative).
Preprocessed using TF-IDF.

- **Logistic Regression:**
  Accuracy: 0.3
  F1 Score: 0.18333333333333332
  Confusion Matrix:

```
 [[0 3 3]
 [0 1 0]
 [0 1 2]]
```

	Precision	**Recall**	**fi-score**	**Support**
negative	0.00	0.00	0.00	6
neutral	0.20	1.00	0.33	1
positive	0.40	0.67	0.50	3

accuracy			0.30	10
Macro avg	0.20	0.56	0.28	10
Weighted avg	0.14	0.30	0.18	10

- **Naive Bayes:**
  Accuracy: 0.3
  F1 Score: 0.17333333333333334
  Confusion Matrix:

```
 [[0 2 4]
 [0 1 0]
 [0 1 2]]
```

	Precision	Recall	fi-score	Support
negative	0.00	0.00	0.00	6
neutral	0.25	1.00	0.40	1
positive	0.33	0.67	0.44	3
accuracy			0.30	10
Macro avg	0.19	0.56	0.28	10
Weighted avg	0.12	0.30	0.17	10

Logistic Regression performed better in this setup, likely due to its ability to handle sparse TF-IDF vectors more robustly than Naive Bayes.

Q4. Emotional Trajectory in a Passage

Text Used:
text = """
The castle stood silent under a heavy veil of fog, its towers piercing the grey sky like spears. Harry walked quietly through the empty corridor, the distant echoes of footsteps making his heart pound faster. He wasn't supposed to be out this late, but something deep inside urged him forward. The shadows danced along the walls, and every flicker of torchlight seemed to whisper secrets he wasn’t meant to hear.

As he reached the base of the Astronomy Tower, he paused. The night was colder than he expected, and his hands trembled not from the chill, but from fear. What if it was a trap? What if Malfoy really had something terrible planned? He took a deep breath, clutching his wand tighter, and began to climb.

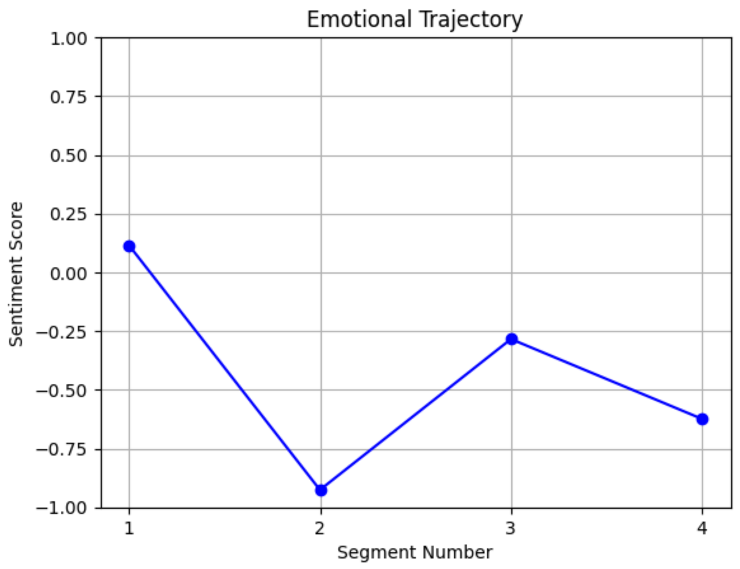
At the top, the wind howled and the moon painted silver lines across the floor. But there he was — Malfoy — not with hatred, but tears in his eyes. “I don’t want this anymore,” Malfoy whispered. Harry froze. The boy he once considered an enemy looked broken, fragile. For a moment, the walls between them faded. They stood in silence, sharing the weight of expectations too big for either of them.

Then, from the shadows, a voice rang out — cold, clear, final. “Expelliarmus!” Harry turned just in time to see Snape step forward. The world collapsed into chaos. Wands clashed, sparks flew, and everything Harry thought he understood came undone. As the smoke cleared and silence returned, Harry stood alone, the storm outside now echoing the one within him.
"""

Segment Scores:

Segment	Score
1	0.1154
2	-0.9261
3	-0.2842
4	-0.6239

Plot:



The emotional tone begins slightly positive, suggesting curiosity or anticipation. It then drops sharply into negativity, indicating fear or uncertainty. This is followed by a partial emotional recovery — perhaps a moment of connection — before dipping again into sadness or despair, reflecting a dramatic and unresolved ending.

Q5 – Conceptual Reflection

1. **Why is lemmatization often preferred over stemming?**
   Lemmatization is often preferred over stemming in natural language processing (NLP) because it produces valid words (lemmas) that are actually in the language's dictionary, while stemming might generate non-dictionary words. Lemmatization also considers the context and part of speech of a word, leading to more accurate results.

2. **How does TF-IDF down-weight common words?**
   TF-IDF down-weights common words by assigning them lower importance scores compared to less frequent, more specific words. This is achieved through the IDF (Inverse Document Frequency) component of the TF-IDF calculation. IDF essentially measures how much information a word provides; rare words get high IDF scores, while common words get low scores.

3. **Describe the curse of dimensionality in text data.**
   The curse of dimensionality in text data refers to the challenges that arise when dealing with high-dimensional text representations, such as those created using techniques like one-hot encoding or word embeddings. As the number of features (dimensions) increases, data can become sparse, and algorithms struggle to find meaningful patterns, leading to decreased performance and increased computational complexity.

4. **When should you use word embeddings instead of BoW/TF-IDF?**
   Word embeddings like Word2Vec or GloVe are generally preferred over Bag-of-Words (BoW) or TF-IDF when dealing with complex NLP tasks that require understanding semantic relationships between words. BoW and TF-IDF treat words as independent entities, while embeddings capture contextual and semantic information, leading to better performance in tasks like sentiment analysis, machine translation, and text classification.

5. **How can POS tagging enhance NLP pipelines?**
   POS (Part-of-Speech) tagging significantly enhances NLP pipelines by providing crucial grammatical information about words in a sentence, which aids in various downstream tasks like sentiment analysis, information retrieval, and machine translation. It essentially helps machines understand the structure and meaning of text by assigning grammatical roles to each word.