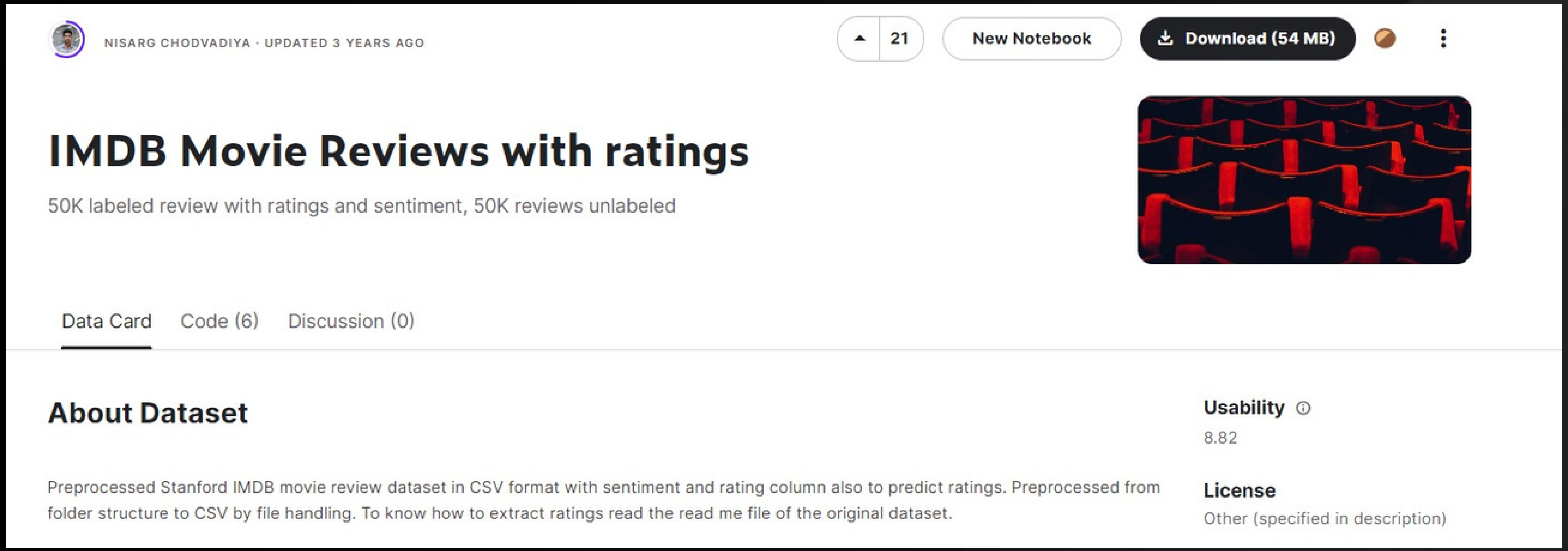
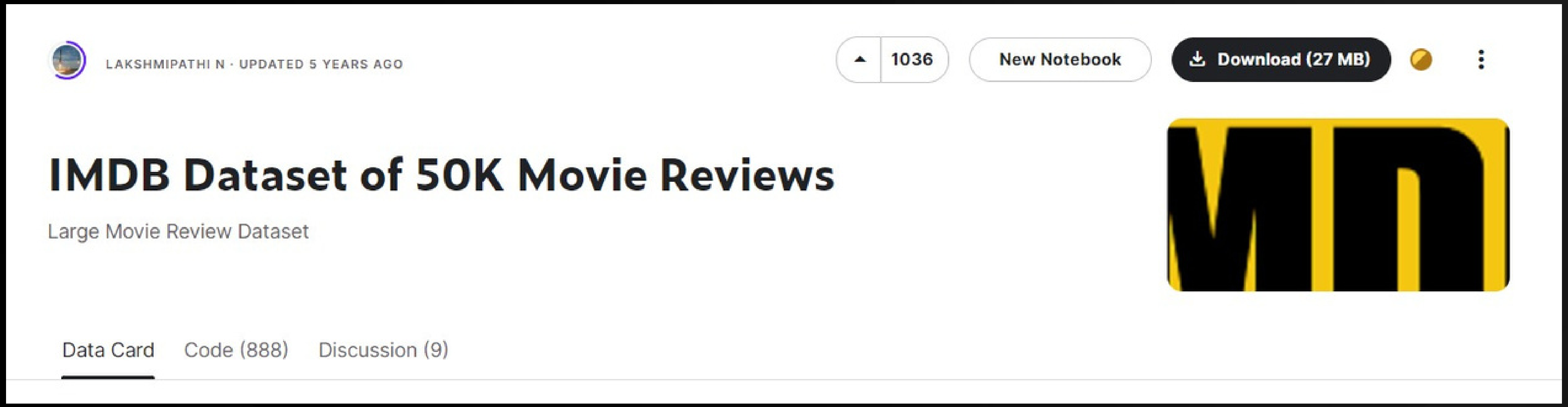
SENTIMENT ANALYSIS PLATFORM FOR MEDIA PRODUCTIONS

Sentiment Analysis and Rating Prediction of Movie Reviews

Presented by: Lawrence(Zhiwei) Qiu and Gurnoor Singh



https://www.kaggle.com/datasets/nisargchodavadiya/imdb-movie-reviews-with-ratings-50k



These two 3D scatter plots show the frequency distribution of positive and negative words respectively under different movie ratings.

Development Environment & Tools

- *Python*
- *Logistic Regression & Naive Bayes*
- *FastText*
- *Linear Regression*
- *spaCy*
- *TfidfVectorizer*
- *Jupyter Notebook*

Part - 1

Data Processing and Feature Extraction

```python
# Load the dataset
imdb_data = pd.read_csv("review_sent.csv")
print("Dataset Loaded")
print("Dataset shape:", imdb_data.shape)
print("First few rows of the dataset:\n", imdb_data.head())
```

```
Dataset Loaded
Dataset shape: (50000, 2)
First few rows of the dataset:
                                               review sentiment
0  One of the other reviewers has mentioned that ...  positive
1  A wonderful little production. <br /><br />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
```

```python
# Clean the review text
def clean_text(text):
    text = re.sub(r'<.*?>', ' ', text)
    text = re.sub(r'[^a-zA-Z0-9\s]', ' ', text)
    return text
imdb_data['clean_review'] = imdb_data['review'].apply(clean_text)
print("\nSample review before cleaning:", imdb_data['review'][0])
print("Sample review after cleaning:", imdb_data['clean_review'][0])
```

```
Sample review before cleaning: One of the other reviewers has mentioned that after wa
right, as this is exactly what happened with me.<br /><br />The first thing that stru
enes of violence, which set in right from the word GO. Trust me, this is not a show f
unches with regards to drugs, sex or violence. Its is hardcore, in the classic use of
e nickname given to the Oswald Maximum Security State Penitentary. It focuses mainly 
son where all the cells have glass fronts and face inwards, so privacy is not high on
ms, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death sta
far away.<br /><br />I would say the main appeal of the show is due to the fact that 
tty pictures painted for mainstream audiences, forget charm, forget romance...OZ does
k me as so nasty it was surreal. I couldn't say I was ready for it, but as I watched 
d to the high levels of graphic violence. Not just violence, but injustice (crooked g
o'll kill on order and get away with it, well mannered, middle class inmates being tu
t skills or prison experience) Watching Oz, you may become comfortable with what is u
ch with your darker side.
Sample review after cleaning: One of the other reviewers has mentioned that after wat
```

```python
# TF-IDF vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english', ngram_range=(1, 2))
features = tfidf_vectorizer.fit_transform(imdb_data['clean_review'])
print("\nTF-IDF Vectorization complete")
print("Shape of the TF-IDF matrix:", features.shape)
```

```
TF-IDF Vectorization complete
Shape of the TF-IDF matrix: (50000, 5000)
```

```python
# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(features, imdb_data['sentiment'], test_size=0.2, random_state=42)
print("\nTrain-Test Split complete")
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
Train-Test Split complete
X_train shape: (40000, 5000)
X_test shape: (10000, 5000)
y_train shape: (40000,)
y_test shape: (10000,)
```

Model Training & Evaluation

```python
# Logistic Regression Model
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

```python
# Naive Bayes Model
naive_bayes_model = MultinomialNB()
naive_bayes_model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```python
# Evaluation
evaluation = {
    'Model': ['Logistic Regression', 'Naive Bayes'],
    'Accuracy': [accuracy_score(y_test, logistic_predictions), accuracy
    'Precision': [precision_score(y_test, logistic_predictions, pos_lab
    'Recall': [recall_score(y_test, logistic_predictions, pos_label='po
    'F1-Score': [f1_score(y_test, logistic_predictions, pos_label='posi
}

evaluation_df = pd.DataFrame(evaluation)
print(evaluation_df)
```

	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.8900	0.879261	0.906132	0.892494
1	Naive Bayes	0.8541	0.845426	0.869419	0.857255

Results of Sentiment Analysis Model

```python
# Extracting Coefficients
feature_names = tfidf_vectorizer.get_feature_names_out()  # Updated method
coefficients = model.coef_.flatten()

# Combining coefficients with feature names
feature_importance = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

# Sorting by absolute coefficients to see the most influential words
top_influential_words = feature_importance.reindex(feature_importance.Coefficient.abs().sort_value

# Display top 20 words
print(top_influential_words.head(20))
```

	Feature	Coefficient
4942	worst	-9.417322
345	awful	-7.310270
515	boring	-6.866844
4824	waste	-6.865693
1509	excellent	6.759788
352	bad	-6.745389
1978	great	6.744836
3387	poor	-5.731439
4451	terrible	-5.642180
1351	dull	-5.274667
3388	poorly	-5.129762
4941	worse	-5.077302
432	best	5.075981
4915	wonderful	5.017101
207	amazing	4.962230
1230	disappointment	-4.935761
3288	perfect	4.903917
2181	horrible	-4.746227
553	brilliant	4.642281
1229	disappointing	-4.598668

Part - 2

Sentiment Analysis with FastText and Rating Predictions

145	10	positive	what if marylin mo
146	10	positive	kept my attention
147	7	positive	this long episode p

240	7	positive	mary pickford born
241	10	positive	clearly an hilarious r
242	7	positive	i was pleasantly surp

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Options:
1. Predict sentiment and rating
2. Compare two reviews
Type 'exit' to quit.
Please enter your choice: 1

Please enter a movie review: kept my attention from s
m mr pacino once again gives us another brilliant cha
This review seems positive to me!
Predicted rating: 9.49
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1. Predict sentiment and rating
2. Compare two reviews
Type 'exit' to quit.
Please enter your choice: 2
Enter first review: clearly an hilarious movie
nius please look at this for what it is a funny
Enter second review: kept my attention from sta
ino once again gives us another brilliant chara
Similarity between the two reviews: 0.75
```

Data Preprocessing and Feature Extraction for Rating Prediction

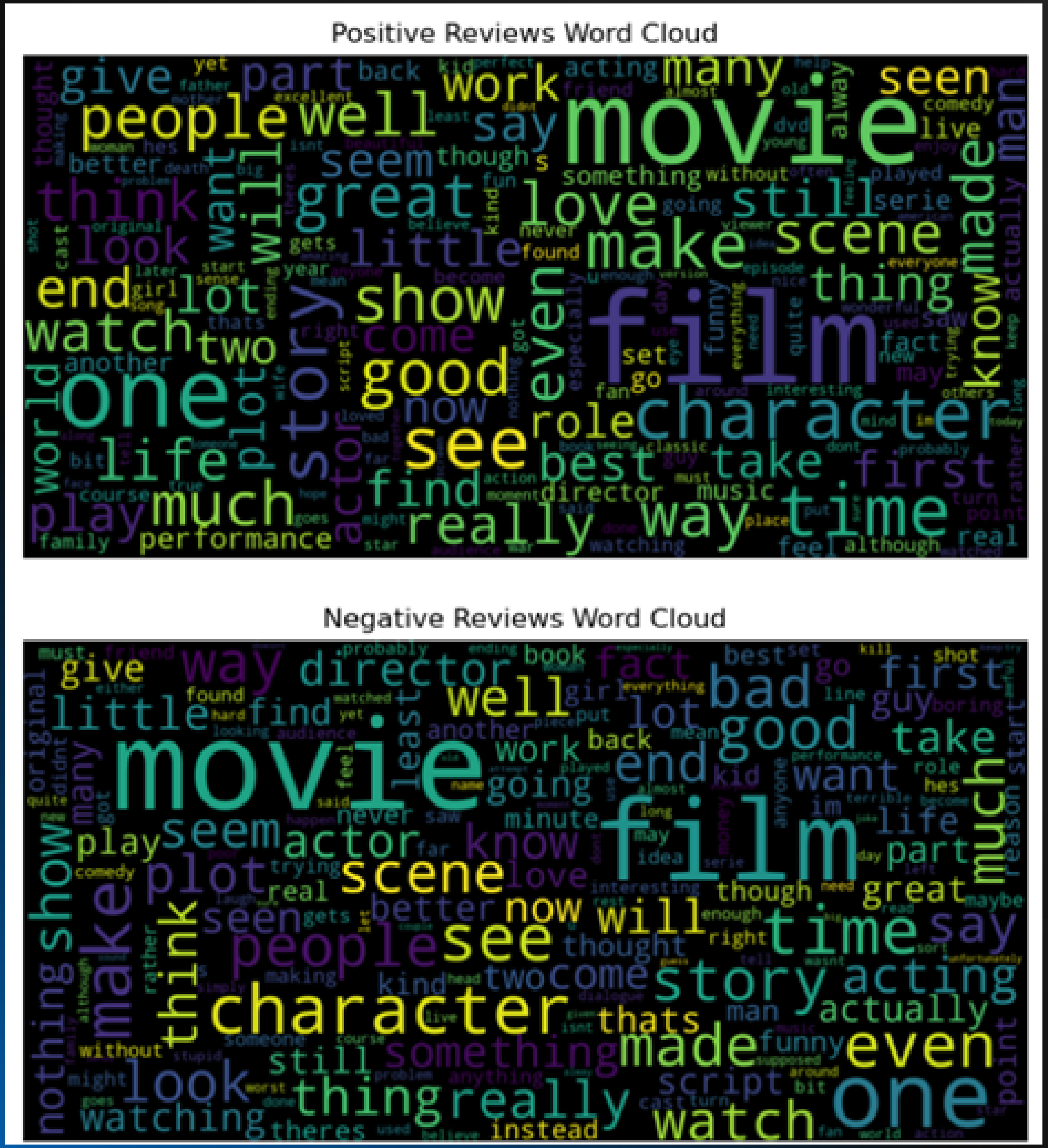
Before:

	Review	Rating	Sentiment
0	Kurt Russell's chameleon-like performance, cou...	10	1
1	It was extremely low budget(it some scenes it ...	8	1
2	James Cagney is best known for his tough chara...	8	1
3	Following the brilliant "Goyôkiba" (aka. "Hanz...	8	1
4	One of the last classics of the French New Wav...	10	1

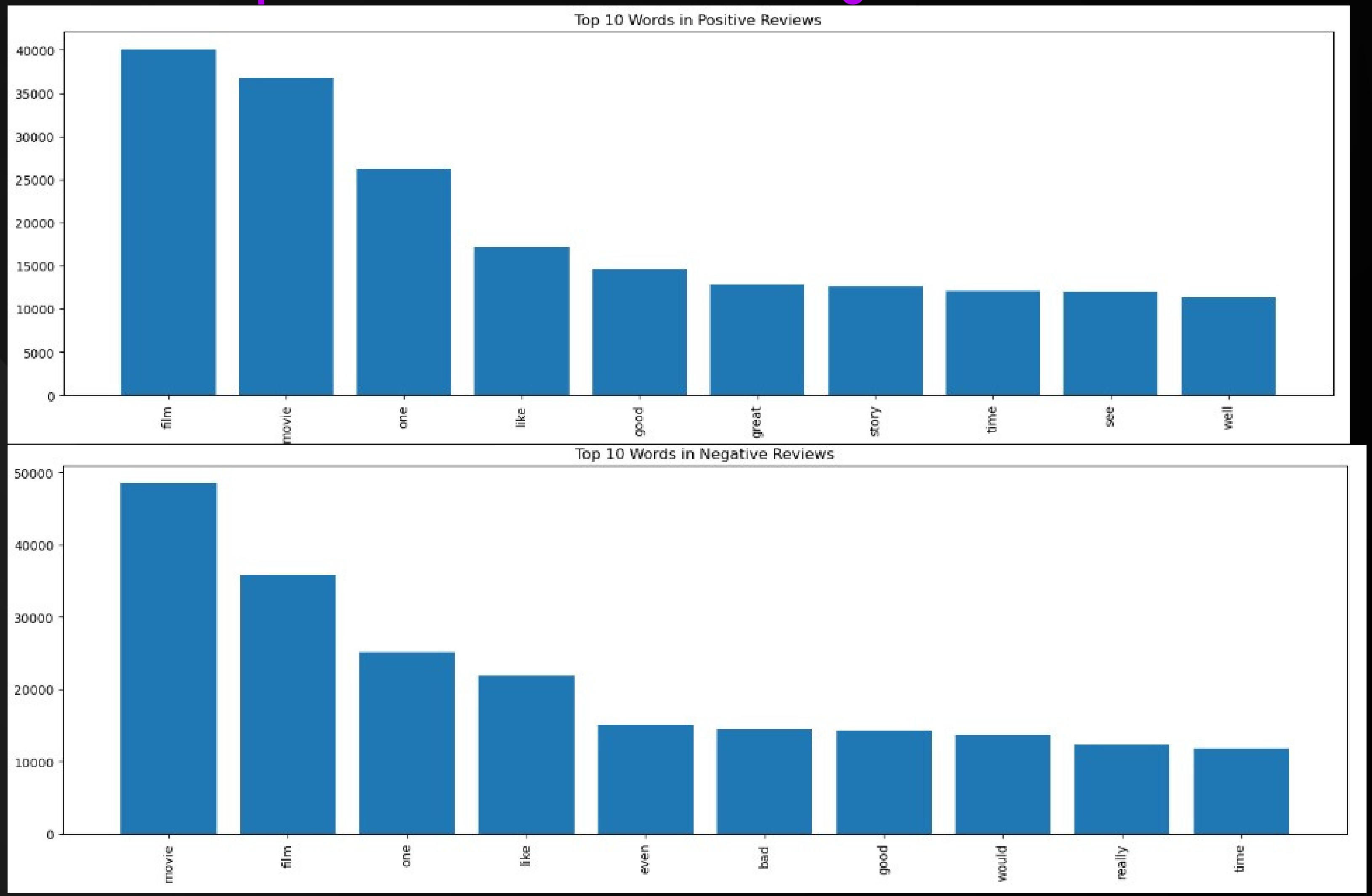
After:

	Rating	Sentiment	Clean_review
0	10	positive	kurt russells chameleonlike performance couple...
1	8	positive	it was extremely low budgetit some scenes it l...
2	8	positive	james cagney is best known for his tough chara...
3	8	positive	following the brilliant goykiba aka hanzo the ...
4	10	positive	one of the last classics of the french new wav...

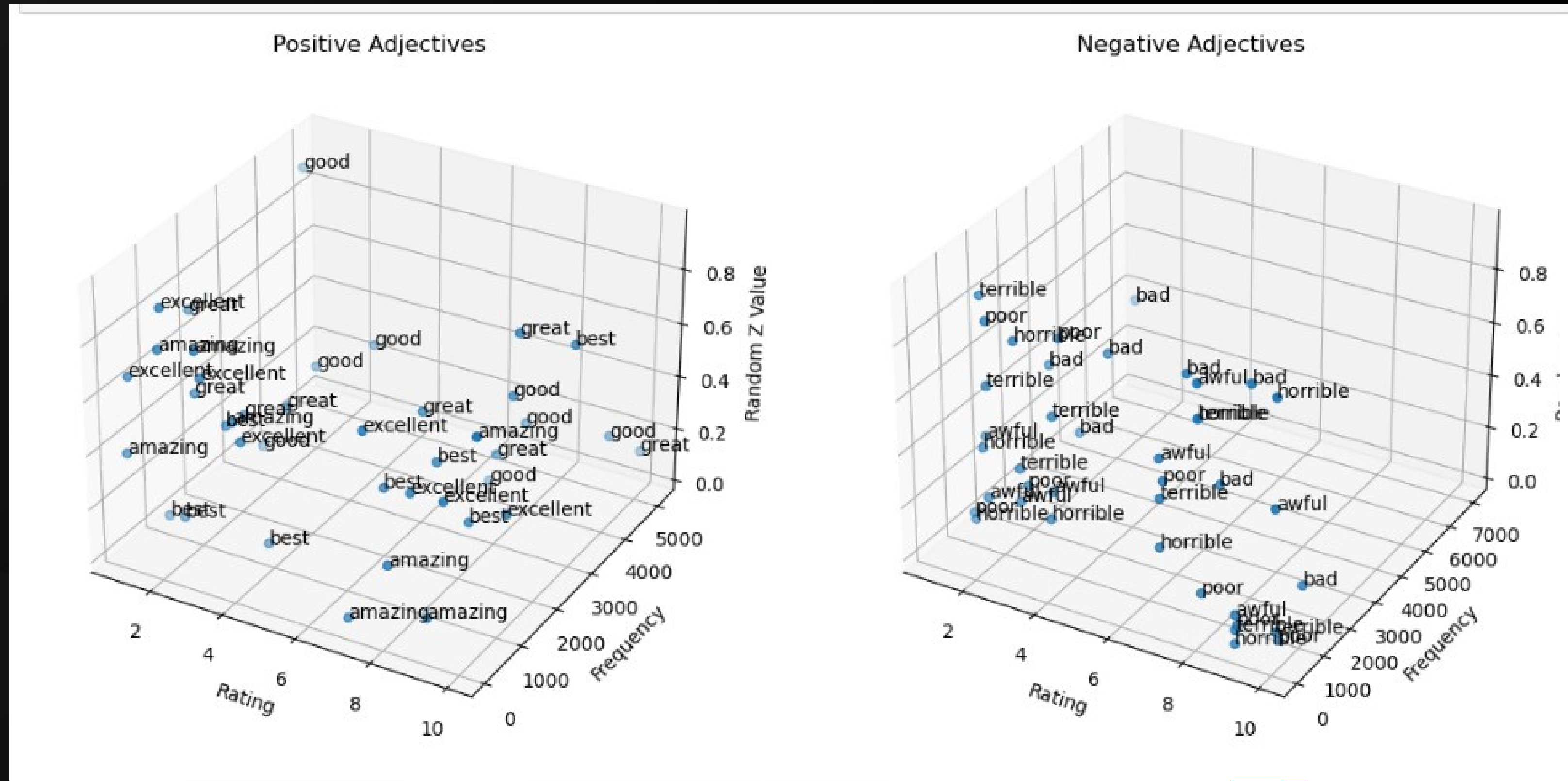
Visualization Techniques



Top 10 Words in Positive and Negative Reviews

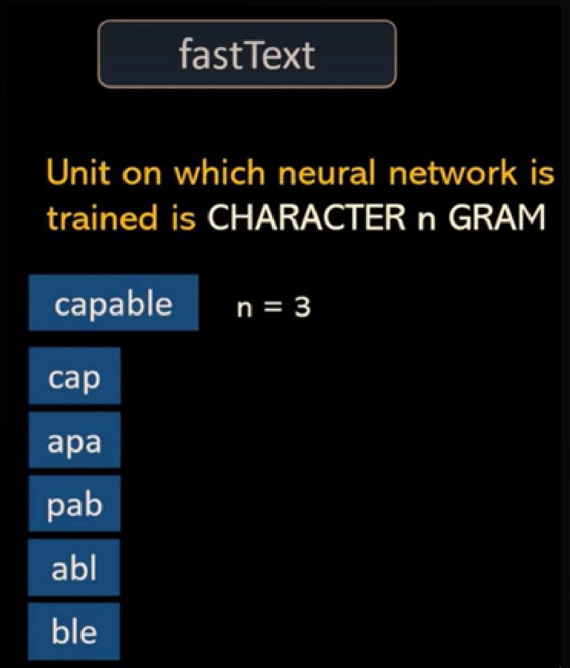


These two 3D scatter plots show the frequency distribution of positive and negative words respectively under different movie ratings.



Model Training and Evaluation

Classifying Movie Reviews with FastText



Problem

Before

```
# Make predictions on the test set and calculate evaluation metrics
test_df['predict_label'] = [classifier.predict(' ' + text)[0][0].replace('__label__', '') for text in test_df['Clean_review']]
true_labels = test_df['Sentiment']
predicted_labels = test_df['predict_label']
# Calculate and output performance metrics
accuracy = accuracy_score(true_labels, predicted_labels)
precision = precision_score(true_labels, predicted_labels, pos_label='positive')
recall = recall_score(true_labels, predicted_labels, pos_label='positive')
f1 = f1_score(true_labels, predicted_labels, pos_label='positive')
print(f'Accuracy: {accuracy}\nPrecision: {precision}\nRecall: {recall}\nF1 Score: {f1}')
```

```
Accuracy: 0.8815
Precision: 0.0
Recall: 0.0
F1 Score: 0.0
```

After

```
# Generate a detailed classification report
report = classification_report(true_labels, predicted_labels)
print("Classification Report:\n", report)
```

```
Model Accuracy: 0.9103
Model Precision: 0.9098901098901099
Model Recall: 0.9108
Model F1 Score: 0.9103448275862069
Classification Report:
              precision    recall  f1-score   support

    negative       0.91      0.91      0.91      5000
    positive       0.91      0.91      0.91      5000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

Average deviation of our predictions from the actual ratings

```
# Evaluate the model
# Calculate Mean Squared Error (MSE) - lower values are better
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Calculate R-squared score - values closer to 1 indicate better fit
r2 = r2_score(y_test, y_pred)
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 4.805162198134217
R-squared: 0.6007543201980878
```

CONCLUSION & FUTURE WORK