NLP Capstone Project

Pauline Ng

Objectives

- This project aims to conduct binary sentiment classification on IMDB movie reviews using traditional machine learning models, such as logistic regression and support vector machines, as well as the deep learning transformer-based model, DistilBERT.
- This project also aims to compare DistilBERT's performance on raw text without preprocessing, showcasing its robustness to handle real-world data effectively.

Dataset

- The IMDB dataset is sourced from Hugging Face's datasets library and originates from StanfordNLP.
- The dataset used is a benchmark dataset for binary sentiment analysis with 50,000 movie reviews labelled as positive or negative (1 for positive; 0 for negative).
- https://huggingface.co/datasets/stanfordnlp/imdb
- The 50,000 labelled reviews were preprocessed as a combined dataset before splitting into an 80-20 train-test set.

Data Preprocessing

These 7 steps are applied as a preprocessing pipeline:

Steps	Rationale	Code
1. Drop duplicate reviews	Done first to prevent redundant operations on identical data points	df = df.drop_duplicates(subset=["text"])
2. Remove hyperlinks	Remove irrelevant tokens	text = re.sub(r'http\S+', '', text)
3. Remove HTML tags	Remove noise that interfere with subsequent processing steps like contraction expansion and emoticon conversion	text = re.sub(r'<.*?>', '', text)
4. Expand Contractions	For better tokenization. Done before emoticon conversion ensures clear separation of words and emoticons.	text = contractions.fix(text)
5. Convert emoticons to text	Preserve sentiment-rich information as text	emoticons_dict = emot_obj.emoticons(text)
6. Remove non-ASCII characters	Special symbol irrelevant to sentiment analysis removed to reduce errors.	text = "".join(char for char in text if ord(char) < 128).strip()
7. Convert lowercase	Ensure consistency across tokens. Done later in this sequence to ensure no interference with emoticon conversion or contraction expansion which may be case-sensitive	text = text.lower()

Data Preprocessing

Steps	Rationale	Code
8. Lemmatization using spacy	To reduce words to their base forms	Load SpaCy model, apply lemmatize_texts() with nlp.pipe() on dataset, then save the resulting CSV.
9. Stopword removal	Further refining the text by removing uninformative words. Done after lemmatization as stopword list are in base form	Load SpaCy model, apply remove_stopwords() using nlp.pipe() on the lemmatized dataset, then save the filtered CSV.
10. Split dataset into 80% train and 20% test	Done before feature extraction to ensure consistent train-test data for evaluation, as fine-tuning DistilBERT requires raw data as input	train_data, test_data = train_test_split(dataset, test_size=0.2, random_state=42)
11. Feature extraction using DistilBert (For logistic regression and SVM) Selected as DistilBert can provide embeddings that capture the contextual meaning of words in a sentence.	Extract DistilBERT embeddings to produce vector representations	Tokenize texts with DistilBERT tokenizer, run them through the model in batches, save CLS embeddings as NumPy arrays for train and test sets.

Traditional ML (Logistic Regression)

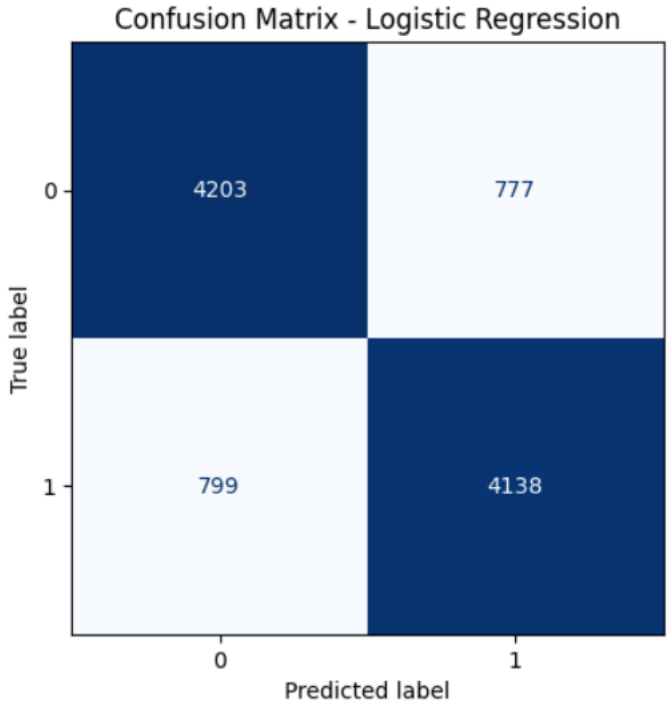
- Model selected as it can perform binary classification and is efficient.
- Trained with labels (positive/negative sentiment) as the target.
- model = LogisticRegression(max_iter=1000, random_state=42)

Traditional ML (Logistic Regression)

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.84	0.84	0.84	4980
1	0.84	0.84	0.84	4937
accuracy			0.84	9917
macro avg	0.84	0.84	0.84	9917
weighted avg	0.84	0.84	0.84	9917

Logistic Regression Test Report saved to: /content/drive



Test Results

- Balanced performance with an accuracy, precision, recall, and F1-score of 0.84.
- This indicates it can handle both positive and negative sentiment prediction well.

Traditional ML (SVM)

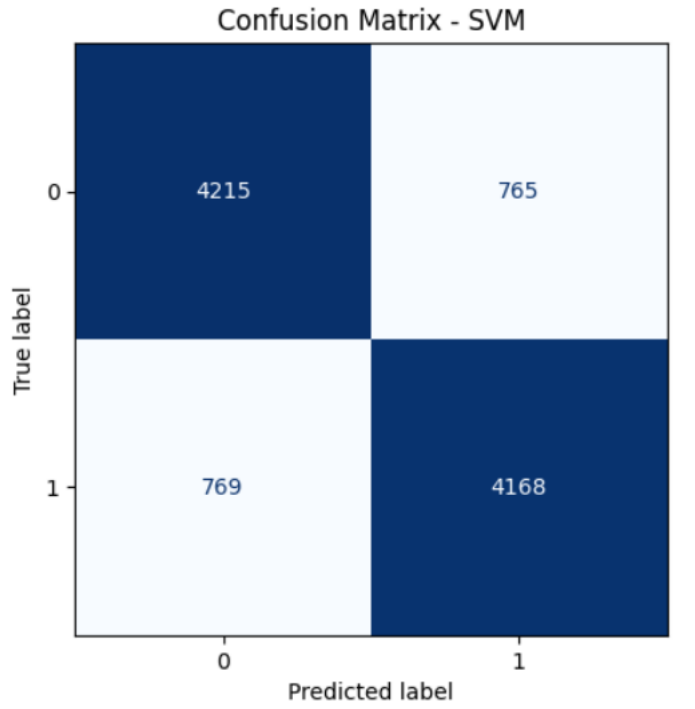
- Model selected as it can work with high dimensional features like DistilBert embeddings. Sentiments inherently complex/non-linear, and SVMs combined with the RBF kernel can effectively capture non-linear decision boundaries.
- SVM is sensitive to scale of features and is scaled using StandardScaler.
- Trained using default parameters (Kernel: 'rbf‘, C: 1.0, Gamma: 'scale‘)
- svm_model = SVC()

Traditional ML (SVM)

SVM Classification Report:

	precision	recall	f1-score	support
0	0.85	0.85	0.85	4980
1	0.84	0.84	0.84	4937
accuracy			0.85	9917
macro avg	0.85	0.85	0.85	9917
weighted avg	0.85	0.85	0.85	9917

SVM Test Report saved to: /content/drive/MyDrive/SPNLP/C



Test Results

- Balanced performance with an accuracy, precision, recall, and F1-score of around 0.85.
- Results are slightly better than logistic regression in classifying both sentiments.

Deep Learning with DistilBERT (1st Test)

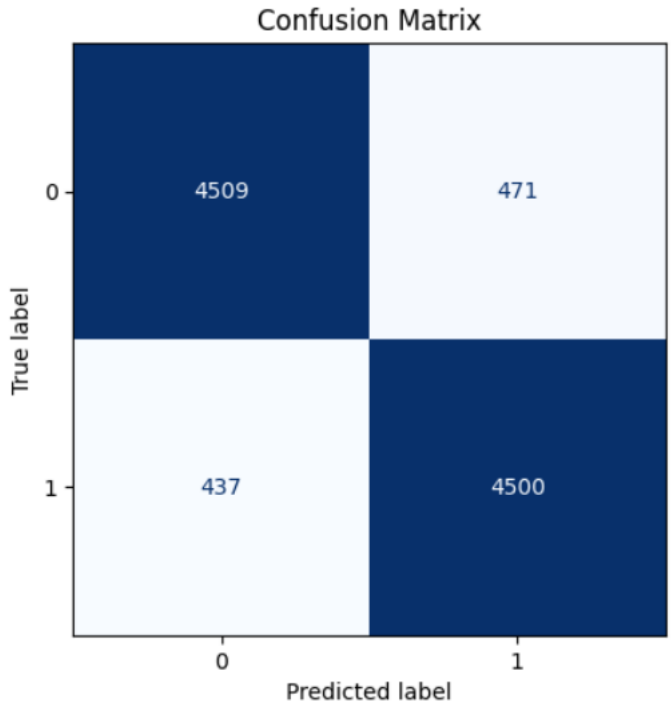
- DistilBERT training and evaluation uses preprocessed text.
- Use PyTorch DataLoader for batching (batch size of 16)
- model = DistilBertForSequenceClassification.from_pretrained(model_name, num_labels=2)
- Optimizer used is AdamW for fine-tuning
- Trained for 3 epochs with loop to tokenize each batch (convert text into token IDs with padding, truncation, and max length of 512), forward pass (compute loss of prediction) and backward pass (update model weights using backpropagation).
- Loss decreased from 0.293 to 0.082, showing the model's improved performance in binary sentiment classification.

Deep Learning with DistilBERT (1st Test)

```
Using GPU: Tesla T4
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      4980
           1       0.91      0.91      0.91      4937

    accuracy                           0.91      9917
   macro avg       0.91      0.91      0.91      9917
weighted avg       0.91      0.91      0.91      9917
```

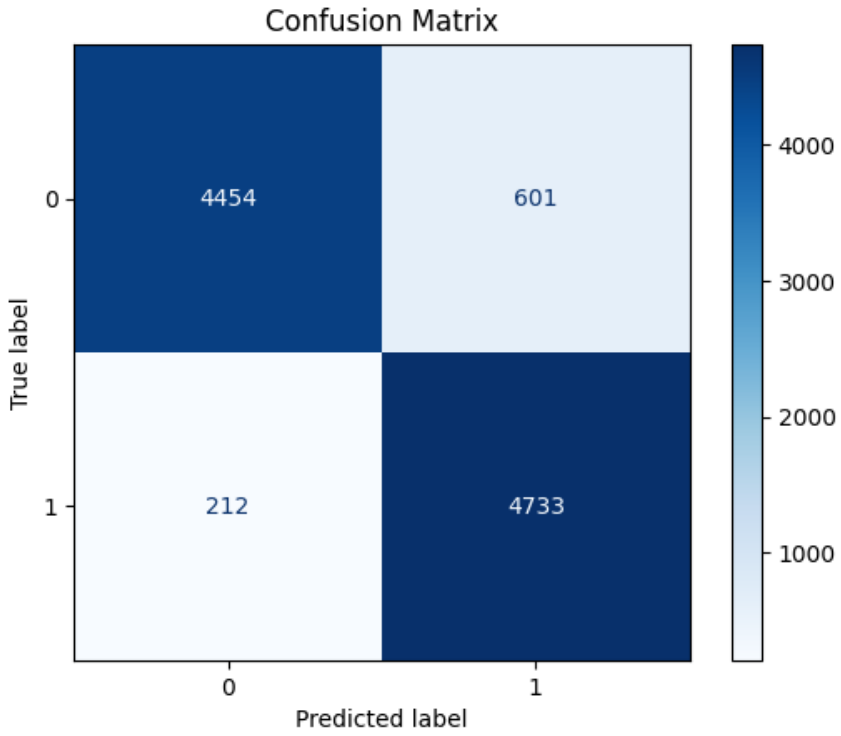


- Balanced performance with an accuracy, precision, recall, and F1-score of 0.91
- Performance is better than traditional models due to its ability to adapt contextual embeddings to sentiment analysis.

Deep Learning with DistilBERT (2nd Test)

Classification Report:

	precision	recall	f1-score	support
0	0.95	0.88	0.92	5055
1	0.89	0.96	0.92	4945
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000



- Tested on raw data without any preprocessing.
- Only split the dataset into 80-20 train-test before fine tuning.
- Balanced performance with an accuracy, precision, recall, and F1-score of 0.92.
- Best performance out of all models.

Evaluation of all models

Model	Accuracy	Precision	Recall	F1 Score
1. Logistic Regression Input Text → DistilBERT (frozen) → Static Embedding → LR (trainable)	0.84	0.84	0.84	0.84
2. Support Vector Machine Input Text → DistilBERT (frozen) → Static Embedding → SVM (trainable)	0.85	0.85	0.85	0.85
3. DistilBERT (with preprocessing) Input Text → Tokenizer → DistilBERT (trainable) → Classification Head (trainable)	0.91	0.91	0.91	0.91
4. DistilBERT (raw text without any preprocessing)	**0.92**	**0.92**	**0.92**	**0.92**

- For Model (1) and (2), performance is limited by static DistilBERT embeddings.
- For Model (3), performance improved due to its ability to adapt contextual embeddings.
- For Model (4), DistilBERT without any preprocessing outperforms the other models which highlights its robustness in handling raw text.