NN DL Project

Group Members: Liyao Cai, Ying Liu, Modi Li

Introduction

- Primary goal:
  - Classify the sentiment of IMDB movie reviews into positive or negative categories.
  - Explore and compare the performance of models in textual sentiment analysis
- Models:
  - RNN
  - LSTM
  - Transformer
- Dataset: IMDB dataset provided by the Keras library
  - 50,000 labeled movie reviews with positive and negative sentiments
- Why?
  - It provides a hands-on opportunity to understand and compare the capabilities of traditional and modern NLP models in handling sequential data.

Outline

- Data preprocessing
- Model discussion
    - RNN
    - LSTM
    - Transformer
- Model results
- Comparative analysis

Data Preprocessing

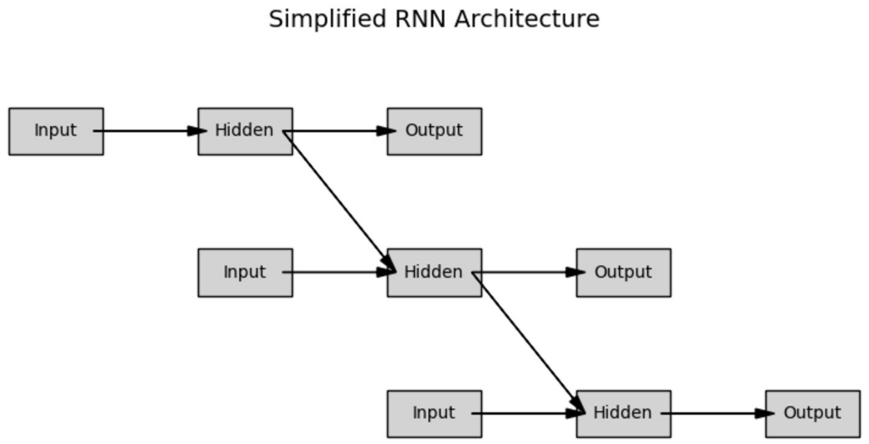
1. Define the max features and sequence length

2. Load and encode

3. Combine and split

4. Pad or truncate the merged data

```
max_features = 10000
max_len = 500  # Maximum length of each comment

# Load the data and convert the reviews into integer-encoded sequences
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

# Combine all data
x_full_train = np.concatenate([x_train, x_test])
y_full_train = np.concatenate([y_train, y_test])

# The first 40,000 records are used as the new training set, and the remaining 10,000 records are used as the new test set.
x_train_new = x_full_train[:40000]
y_train_new = y_full_train[:40000]

x_test_new = x_full_train[40000:]
y_test_new = y_full_train[40000:]

# Pad or truncate the merged data
x_train_new = sequence.pad_sequences(x_train_new, maxlen=max_len)
x_test_new = sequence.pad_sequences(x_test_new, maxlen=max_len)
```

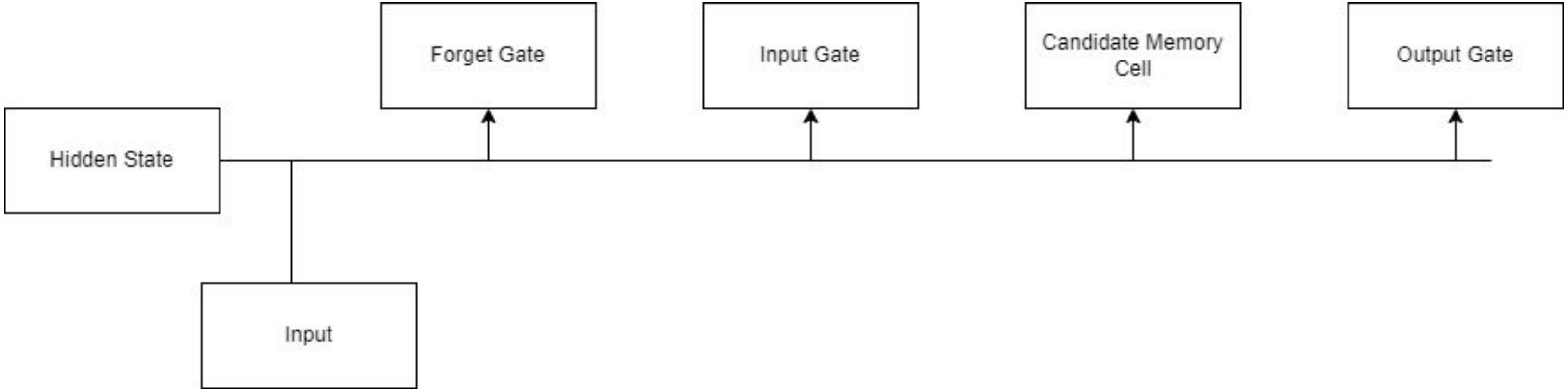


Model: RNN

- **Recurrent Neural Networks (RNNs)** are a type of neural network designed for processing sequential data. RNNs have a feedback loop that allows information to persist across time steps. This makes them ideal for tasks involving sequences, such as:
    - Text data
    - Time-Series Data
    - Speech and Audio

Arrow Connections
- Input to Hidden State
- Hidden state to Hidden state
- Hidden state to Output

Model: LSTM



LSTM (Long Short-Term Memory) is a special type of RNN designed specifically to solve the long-term dependency problem of RNN.

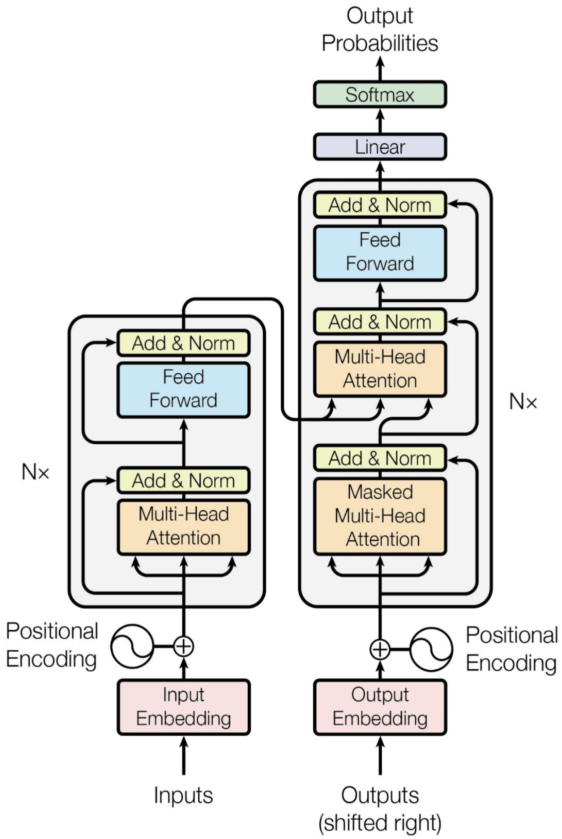
Important parts

1. Memory Cell

2. Gating Mechanism
   a. Forget Gate
   b. Input Gate
   c. Candidate Memory Cell
   d. Output Gate

Model: Transformer

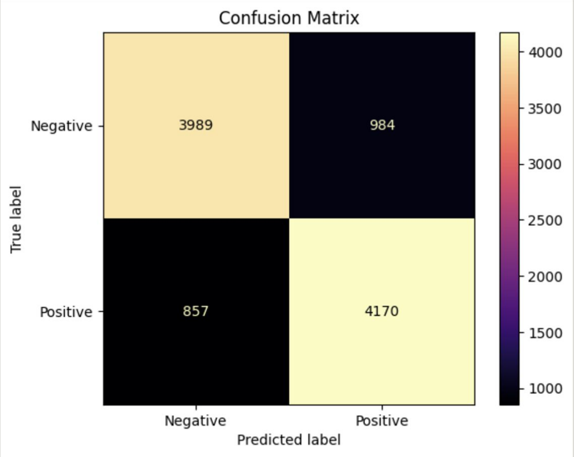
Transformer is a deep learning model architecture based on the Attention mechanism. It solves the shortcomings of traditional RNN/CNN in long sequence modeling and is widely used in Natural Language Processing, image processing and time series tasks.



Important parts

1. Positional encoding: Using sine and cosine positional encoding helps the model capture sequence order information.
2. Multi-Head Attention: Capturing the correlation between positions in the input sequence.

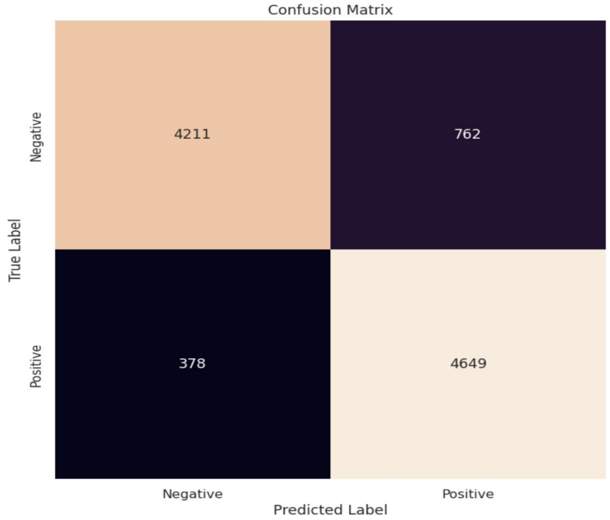
Results



Classification Report:

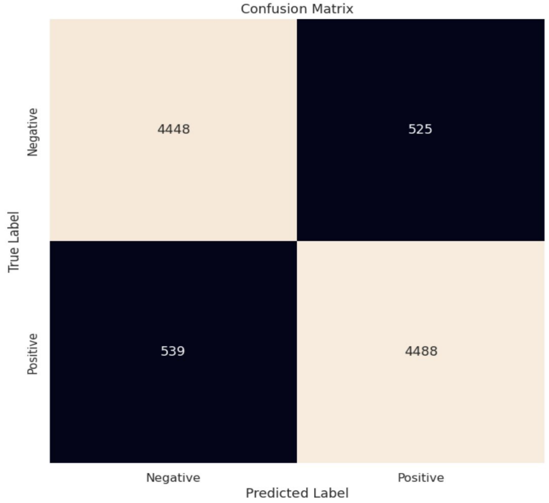
	precision	recall	f1-score	support
Negative	0.82	0.80	0.81	4973
Positive	0.81	0.83	0.82	5027
accuracy			0.82	10000
macro avg	0.82	0.82	0.82	10000
weighted avg	0.82	0.82	0.82	10000

RNN



	precision	recall	f1-score	support
Negative	0.92	0.85	0.88	4973
Positive	0.86	0.92	0.89	5027
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

LSTM



	precision	recall	f1-score	support
Negative	0.89	0.89	0.89	4973
Positive	0.90	0.89	0.89	5027
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

Transformer

Result Comparison

Model / Data Label	Precision	Recall	F1 Score	Accuracy	Running Time
RNN - Positive	0.82	0.80	0.81	0.82	6m
RNN - Negative	0.81	0.83	0.82		
LSTM - Positive	0.92	0.85	0.88	0.89	12m 44s
LSTM - Negative	0.86	0.92	0.89		
Transformer - Positive	0.89	0.89	0.89	0.89	2h 54m 26s
Transformer - Negative	0.90	0.89	0.89		

Result Analysis

- RNN
  - Has lowest metrics
  - Shortest running time
  - Reason: struggles with long-range dependencies due to the vanishing gradient problem
  - Basic model with limited ability to capture context in texts
- LSTM
  - Has higher metrics than RNN
  - Longer running time than RNN
  - Very high recall for negative texts, showing better recognition of negative text patterns
  - Reason: overcomes RNN’s limitations using the memory cells
  - Better at long-term dependencies
- Transformer
  - Has very high and most balanced performance among all models
  - Longest running time
  - Reason: uses the attention mechanism and captures long-range dependencies and the context more effectively while enabling parallel processing

DistilBERT

- A small, fast, cheap and light Transformer model trained by distilling BERT base

```
Epoch 1/4
WARNING:tensorflow:AutoGraph could not transform <function infer_framework at 0x7f9f2e17a5f0> and will run it as-i
s.
Cause: for/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function infer_framework at 0x7f9f2e17a5f0> and will run it as-is.
Cause: for/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
1250/1250 [==============================] - 2304s 2s/step - loss: 0.7252 - accuracy: 0.5000 - val_loss: 0.6932 - v
al_accuracy: 0.5027
Epoch 2/4
1250/1250 [==============================] - 2208s 2s/step - loss: 0.6935 - accuracy: 0.5038 - val_loss: 0.6938 - v
al_accuracy: 0.5027
Epoch 3/4
1250/1250 [==============================] - 2228s 2s/step - loss: 0.6936 - accuracy: 0.4999 - val_loss: 0.6937 - v
al_accuracy: 0.5027
Epoch 4/4
1250/1250 [==============================] - 2226s 2s/step - loss: 0.6935 - accuracy: 0.5034 - val_loss: 0.6932 - v
al_accuracy: 0.4973
```

```
313/313 [==============================] - 182s 582ms/step - loss: 0.6932 - accuracy: 0.4973
```

```
Test Loss: 0.693188488483429
Test Accuracy: 0.49729999899864197
```

- Reasons
  - No find-tuning on the large dataset
  - Maybe the pre-trained model was not trained on sentiment specific data

Thank You