NLP 202 Assignment 1

Ishika Kulkarni
University of California, Santa Cruz
Natural Language Processing
ikulkar1@ucscs.edu

Abstract

This assignment report presents the implementation of a logistic regression model and an LSTM model for sentiment analysis on the IMDB dataset. We look into the impact of mini batching by evaluating the models on different batch sizes. We also do hyperparameter tuning to get the best results.

1 Introduction

In Natural Language Processing, sentiment analysis helps machines classify textual data based on users' sentiments. In this assignment, we analyze the feelings on the IMDB dataset for positive and negative reviews.

Mini batching allows the models to process multiple samples for analysis by reducing the computation time and making the model more efficient. We thus explore how these batches impact the accuracy and time, and we perform hyperparameter tuning to optimize the models better.

2 Dataset

As mentioned before, we use the IMDB dataset for this assignment. Below is the classification of the positive and negative labels we can use for sentiment analysis.

Label	Count
pos	12500
neg	12500

Table 1: Class Distribution of Reviews

As we can see, the dataset is balanced. Thus, the results would not be skewed.

Column	Non-Null Count
review	25000 non-null
label	25000 non-null

Table 2: Column Data Types and Non-Null Counts

We have two columns in this dataset: review and label. Below are some examples of positive and negative reviews.

ID	Review	Label
0	Before Sunrise has many remarkable things going...	pos
1	Lars Von Trier's Europa is an extremely good fi...	pos
2	This movie took my breath away at some points,...	pos
3	"Emma Woodhouse" Gwyneth Paltrow (Shakespeare ...	pos
4	It is a well-known fact that when Gene Roddenb...	pos
12500	I am so confused. What in the world was this m...	neg
12501	Although it's an R rated movie, I really doubt...	neg
12502	Timberlake's performance almost made attack th...	neg
12503	This was disappointing. It started well enough...	neg
12504	An executive, very successful in his profession...	neg

Table 3: Sample Reviews and Their Sentiment Labels (Positive and Negative)

3 Pre processing

- **Text Tokenization**
  - We utilized the spaCy tokenizer to split each review into individual words.
- **Vocabulary Construction**
  - We built a vocabulary of the most frequent 25,000 words from the training set.
  - Two special tokens were included:
    * <pad> for sequence padding
    * <unk> for out-of-vocabulary words
- **Numericalization**
  - Each review was converted into a sequence of integers based on the vocabulary.
  - Words not found in the vocabulary were replaced with <unk>.
- **Padding**
  - Since reviews have variable lengths, we applied padding to ensure uniform input sizes.
  - Shorter sequences were padded with <pad> to match the longest sequence in a batch.
- **Minibatching and Collation**
  - We implemented mini batching to enable efficient training.
  - The collate_fn function ensured that batches of different sequence lengths were handled properly by padding sequences dynamically.

4 Part 1: Logistic Regression

For the logistic regression model, we have 2 layers: the embedding layer and a linear output layer. The embedding layer converts the input text to dense vectors, and the linear layer generates the classification labels for us.

Below is the model architecture:

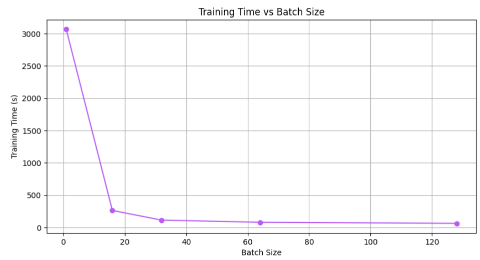


Figure 1: LR: Training time vs Batch Size

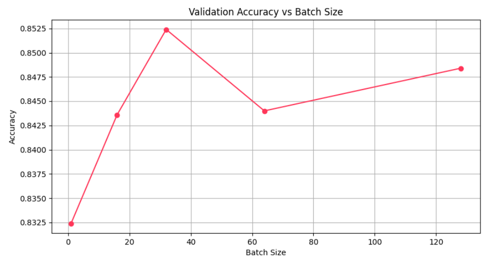


Figure 2: LR: Validation accuracy vs Batch size

Logistic Regression Architecture

+ embedding: nn.Embedding

+ dropout: nn.Dropout

+ fc: nn.Linear

+ forward(x): Tensor

5 Logistic Regression Results

For training with different batch sizes, below are the results:

Batch Size	Training Time (s)	Validation Accuracy
1	3967.03	0.8324
16	327.09	0.8436
32	127.11	0.8524
64	82.07	0.8440
128	70.59	0.8484

Table 4: Training Time and Validation Accuracy for Different Batch Sizes

When tested on the test set with the same parameters, the results were as follows:

Test Loss: 0.3914, Test Accuracy: 0.8408, Precision: 0.7947, Recall: 0.9190, F1: 0.8523

For hyperparameter tuning:

Batch Size	Epochs	Learning Rate	Dropout	Validation Accuracy
32	10	0.001	0.5	0.8286
32	10	0.0001	0.3	0.8284
32	30	0.001	0.3	0.8634
32	30	0.0001	0.3	0.6528
64	10	0.001	0.3	0.7942
64	30	0.001	0.5	0.8466
64	30	0.0001	0.5	0.6030
128	10	0.001	0.3	0.7230
128	30	0.001	0.3	0.8420
128	30	0.0001	0.5	0.8364

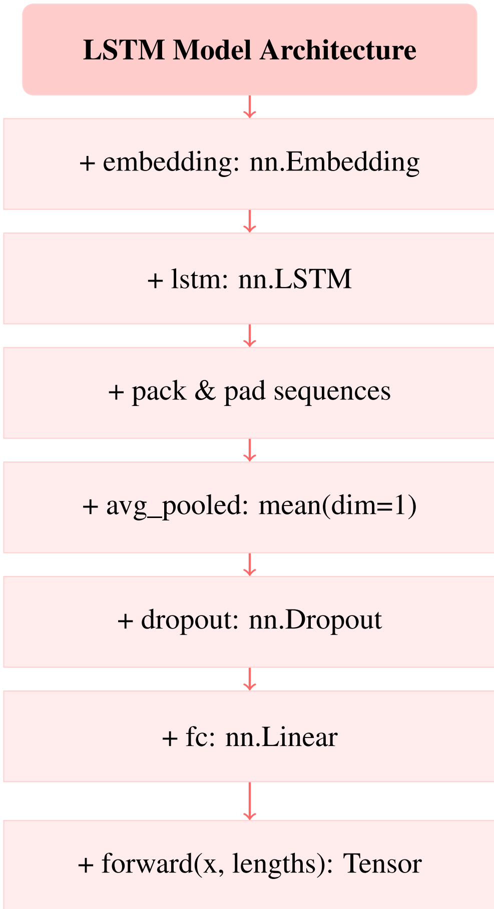
Table 5: Validation Accuracy for Different Batch Sizes, Epochs, Learning Rates, and Dropout Rates

As we can see in Figure 1, the initial training time when we used batch size = 1, the training time was the most, and then it just decreased as the batch size increased.

For validation accuracy, as in Figure 2, the model peaked when the batch size was 32, and it was the lowest when the batch size was 1.

6 Part 2: LSTM

LSTM or Long Short Term Memory helps the model capture long-term dependencies in the text sequences and thus is helpful in sentiment analysis tasks. Similar to Logistic Regression, the LSTM model also has an embedding layer, then we have the LSTM layer followed by a pooling and classification layer.



6.1 LSTM Experiments

Hyperparameter	Values
Batch Sizes	1, 32, 64, 128
Learning Rates	0.001, 0.0001
Dropout Rates	0.3, 0.5
Number of Epochs	10
LSTM Options	True, False

Table 6: Hyperparameter Settings for LSTM Model

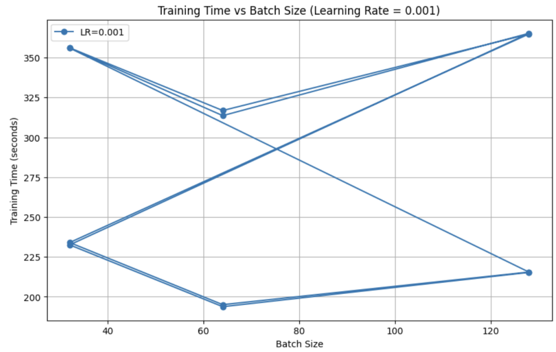


Figure 3: Training time vs batch size, LR = 0.001

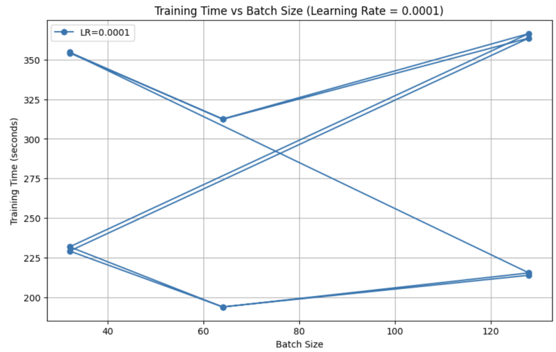


Figure 4: Training time vs batch size, LR = 0.0001

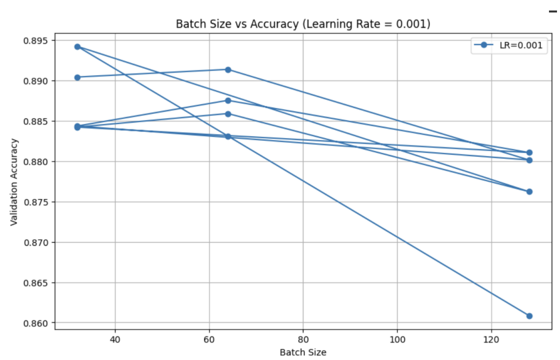


Figure 5: Batch Size vs Accuracy, LR = 0.001

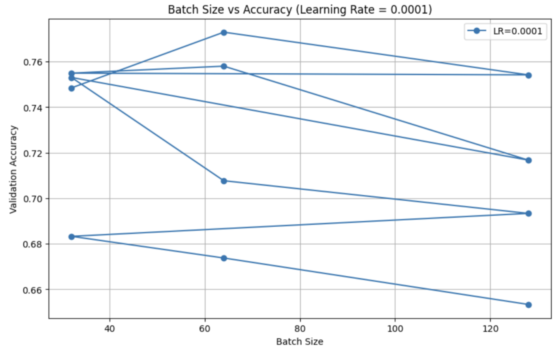


Figure 6: Batch size vs Accuracy; LR = 0.0001

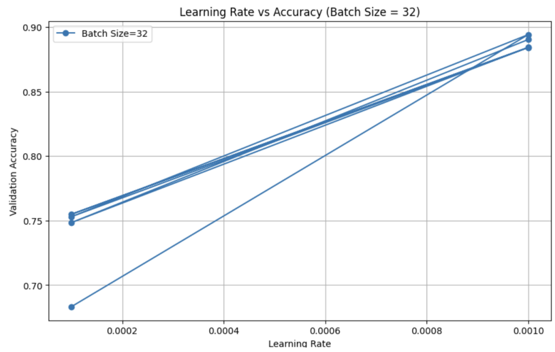


Figure 7: LR vs Accuracy 1

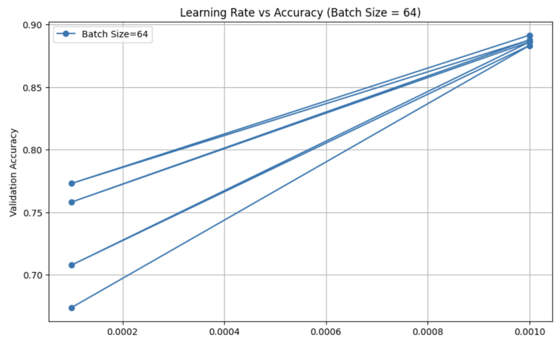


Figure 8: LR vs Accuracy 2

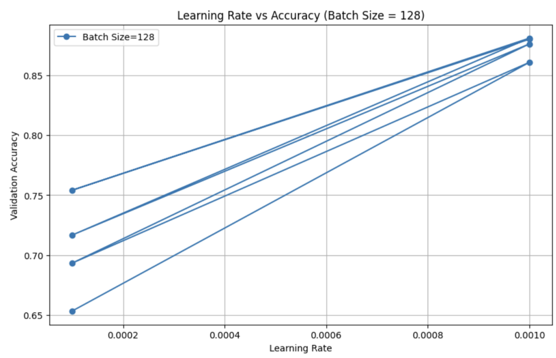


Figure 9: LR vs Accuracy 3

Table 7: Results for BiLSTM = True

Dropout	Batch Size	Learning Rate	Validation Accuracy	Training Time (s)
0.3	32	0.001	0.8869	416.59
0.3	32	0.0001	0.8044	408.59
0.3	64	0.001	0.8825	372.80
0.3	64	0.0001	0.7718	371.87
0.3	128	0.001	0.8837	440.78
0.3	128	0.0001	0.7200	440.68
0.5	32	0.001	0.8966	410.42
0.5	32	0.0001	0.7223	413.09
0.5	64	0.001	0.8928	369.51
0.5	64	0.0001	0.6729	372.62
0.5	128	0.001	0.8844	441.27
0.5	128	0.0001	0.6863	441.07

Table 8: Results for BiLSTM = False

Dropout	Batch Size	Learning Rate	Validation Accuracy	Training Time (s)
0.3	32	0.001	0.8868	276.64
0.3	32	0.0001	0.7859	276.48
0.3	64	0.001	0.8839	236.59
0.3	64	0.0001	0.7405	237.56
0.3	128	0.001	0.8719	263.82
0.3	128	0.0001	0.7281	263.48
0.5	32	0.001	0.8890	275.68
0.5	32	0.0001	0.6930	272.54
0.5	64	0.001	0.8934	235.58
0.5	64	0.0001	0.6766	236.66
0.5	128	0.001	0.8708	265.05
0.5	128	0.0001	0.6686	263.97

6.2 LSTM Results

- **Effect of BiLSTM:**
  - Increases training time significantly (up to ~441s vs. 276s).
  - Slightly improves accuracy for learning rate $= 0.001$.
  - Best for accuracy, but disabling BiLSTM improves efficiency.
- **Effect of Dropout:**
  - **Dropout = 0.5** improves accuracy when learning rate is **0.001**.
  - **Dropout = 0.3** is better for **0.0001 learning rate**.
- **Effect of Batch Size:**
  - Best validation accuracy with **batch size = 32** (BiLSTM) or **64** (Non-BiLSTM).
  - **Batch size = 128** increases training time without improving accuracy.
- **Effect of Learning Rate:**
  - **0.001** performs significantly better than **0.0001** across all settings.
  - Learning rate has no major impact on training time.
- **Best Accuracy:**
  - **BiLSTM = True**, Dropout = **0.5**, Batch Size = **32**, LR = **0.001**
  - **Validation Accuracy:** 0.8966
- **Best Efficiency (Lower Training Time, High Accuracy):**
  - **BiLSTM = False**, Dropout = **0.3**, Batch Size = **64**, LR = **0.001**
  - **Validation Accuracy:** 0.8934, **Training Time:** 235s

7 Conclusion

For Logistic Regression, the results indicate that a batch size of 32 provided the best balance between training efficiency and validation accuracy, achieving the highest performance.

Using LSTM improves accuracy slightly but significantly increases training time. A learning rate of 0.001 consistently outperforms 0.0001, with Dropout = 0.5 yielding the best results. For optimal performance, use LSTM (Dropout = 0.5, Batch Size = 32), while for efficiency, disable LSTM with Dropout = 0.3, Batch Size = 64.