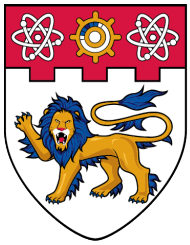
Acad. Year
2024-25

IS6751 Text & Web Mining: Take Home Project #2



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

LIANG HOU - G2401631D

[Words: 901]

Year 2024/25

IS6751 Text & Web Mining: Take Home Project #2

Task 1: Optimize various RNN models

SUBMITTED BY

LIANG HOU

(G2401631D)

Wee Kim Wee School of Communication and Information

Year 2024/25

1 Introduction

In this task, we aim to optimize various Recurrent Neural Network (RNN) architectures for sentiment classification using the provided dataset. The baseline implementation was based on `Sentiment-Classification-with-RNNs-v3-revised.ipynb`, and we extended it by testing multiple RNN variants and performing extensive hyperparameter tuning to identify the best-performing configuration.

2 Approach

2.1 Model Variants Tested

We first compared three different types of RNN architectures:

- Standard RNN
- Gated Recurrent Unit (GRU)
- Long Short-Term Memory (LSTM)

The performance is summarized in Table 7.

Table 1: Performance of Different RNN Models

Model	**Test Loss**	**Test Accuracy (%)**
RNN	0.63	65.89
GRU	0.60	68.52
LSTM	**0.58**	**69.65**

Given the superior performance of LSTM, we chose it as the basis for further optimization.

2.2 Output Aggregation Methods

We experimented with different ways to aggregate the LSTM output before the final classification layer:

- Last output vector
- Maximum over sequence (max pooling)
- Mean over sequence (mean pooling)

The results are summarized in Table 2.

Table 2: Performance of Different Output Aggregation Methods (LSTM)

Aggregation Method	**Test Loss**	**Test Accuracy (%)**
Last Output	0.58	69.65
Max Pooling	0.57	70.65
Mean Pooling	**0.56**	**72.03**

Mean pooling achieved the best results and was used in subsequent experiments.

2.3 Dropout Rate

We tested the effect of applying a dropout layer (with a rate of 0.4) after the LSTM outputs. The model without dropout yielded better accuracy:

Table 3: Effect of Dropout (Mean Pooling LSTM)

Setting	**Test Loss**	**Test Accuracy (%)**
Without Dropout	**0.56**	**72.03**
With Dropout (0.4)	0.58	69.91

We thus opted not to use dropout in the final model.

2.4 Word Embedding Size

We experimented with different word embedding sizes, and observed best performance at size 150:

Table 4: Effect of Word Embedding Size (Mean Pooling LSTM)

Embedding Size	**Test Loss**	**Test Accuracy (%)**
50	0.57	72.15
100	0.56	72.03
150	**0.53**	**73.04**
200	1.34	71.59

2.5 RNN Hidden Size

Finally, different LSTM hidden sizes were evaluated:

Table 5: Effect of Hidden Size (Embedding Size = 150)

Hidden Size	**Test Loss**	**Test Accuracy (%)**
50	**0.53**	**73.47**
100	0.53	73.04
150	0.52	72.78

The best result was achieved with a hidden size of 50.

3 Final Model Configuration

The optimized configuration for our LSTM model is:

Table 6: Final LSTM Model Configuration

Parameter	**Value**
Output aggregation	Mean pooling
Dropout	None
Word embedding size	150
LSTM hidden size	50

3.1 Final Performance

- **Test Loss: 0.53**
- **Test Accuracy: 73.47%**

4 Conclusion

Through systematic comparison and hyperparameter tuning, the LSTM model with mean pooling, embedding size of 150, and hidden size of 50 achieved the best performance, improving test accuracy from an initial 65.89% (RNN baseline) to 73.47%.

Task 2: Optimize various transformer-based models

SUBMITTED BY

LIANG HOU

(G2401631D)

Wee Kim Wee School of Communication and Information

Year 2024/25

5 Introduction

This section details the optimization process and results of applying various transformer-based models for sentiment classification, based on the implementation in `Sentiment-Classification-with-transformer-models-v1.ipynb`. The objective was to compare models such as BERT, RoBERTa, and DistilRoBERTa, and to optimize the best-performing model through hyperparameter tuning.

6 Approach

6.1 Transformer Models Tested

We first evaluated the performance of three transformer-based pre-trained models:

- `bert-base-cased`
- `roberta-base`
- `distilroberta-base`

The performance is summarized in Table 7.

Table 7: Performance of Different Transformer Models

Model	Test Loss	Test Accuracy
`bert-base-cased`	1.11	0.86
`roberta-base`	**0.76**	**0.89**
`distilroberta-base`	0.84	0.86

`roberta-base` showed the best performance and was selected for further optimization.

6.2 Hyperparameter Tuning

6.2.1 Number of Data Loader Workers (`num_workers`)

We tested two different settings for the `num_workers` parameter:

Table 8: Effect of num_workers on roberta-base

num_workers	**Test Loss**	**Test Accuracy**
0	**0.76**	**0.89**
4	0.91	0.89

Analysis: Increasing num_workers did not improve performance and slightly worsened test loss, possibly due to data loading overhead or environment limitations. Thus, we retained num_workers = 0.

6.2.2 Batch Size

Next, we tested different batch sizes while keeping other settings constant:

Table 9: Effect of Batch Size on roberta-base (num_workers = 0)

Batch Size	**Test Loss**	**Test Accuracy**
16	0.76	0.89
32	**0.37**	**0.89**

Analysis: Increasing batch size to 32 significantly reduced test loss from 0.76 to 0.37 while maintaining accuracy at 0.89. A larger batch size stabilizes gradient updates and leads to a smoother loss landscape, which may explain this improvement.

7 Final Model Configuration

The final optimized configuration for our transformer model is:

- Model: roberta-base
- num_workers: 0
- Batch size: 32

7.1 Final Performance

- **Test Loss: 0.37**
- **Test Accuracy: 0.89**

8 Conclusion

Through systematic model comparison and hyperparameter tuning, `roberta-base` consistently outperformed other transformer models in this task. The key finding was that increasing the batch size from 16 to 32 significantly reduced the test loss while preserving accuracy. The `num_workers` parameter did not yield positive effects in this context.