COMP 4332 - Project 1
Sentiment Analysis

Group Number: 14
Group Members: CHEN, Jiawei (20763842), Leung King Suen (20770625), Kwong Ka Lok (20772439)

Introduction

1. LSTM

LSTM stands for Long Short-Term Memory, and it is a type of recurrent neural network (RNN) architecture that is designed to overcome the limitations of traditional RNNs in capturing and remembering long-term dependencies in sequential data. The key feature of LSTM networks is their ability to maintain and update a cell state, which acts as a memory unit. This cell state allows the network to retain information over long sequences and selectively forget or remember information based on the relevance to the current task. This allows the network to adapt to the specific patterns and dependencies in the input data.

2. BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a powerful natural language processing (NLP) model developed by Google. It leverages the power of transformers, a type of attention-based neural network architecture. Unlike traditional models that read text in a sequential manner (e.g., LSTM or RNN), BERT utilizes a bidirectional approach which reads the input text both from left to right and from right to left, allowing it to capture the context and dependencies of each word based on the entire surrounding context. This bidirectional nature enables BERT to have a deeper understanding of the meaning and nuances of words within a sentence. BERT has been pre-trained on a massive amount of unlabeled text from the internet, allowing it to learn general and contextual language representations and has the ability to generate high-quality word embeddings. Due to its versatility and effectiveness, BERT has become a popular choice for many NLP applications.

Attempted Models

1. Freeze BERT + Linear Layer

Model Architecture Description

Our model uses the pre-trained BERT model with its tokenizer to convert the input texts to word embeddings. We then take the pooler output from the bert model, which has a feature dimension of 768, as input to a linear layer and convert it to a tensor which has a feature space of 5, representing our 5 predicted labels. Before passing the input word embeddings to the linear layer, we add a dropout layer to randomly drop some nodes while model training to prevent the model from overfitting too fast. Finally, we use the cross entropy loss as the loss function since it is more suitable for multi-class classification problems. As the pytorch implementation for this cross entropy loss function has already included a softmax operation inside, we can directly take the output logits from our linear layer and pass to the loss function for calculating the loss. For this model, we will freeze the Bert model and only train the single linear layer.

Analysis on Initial Training Results

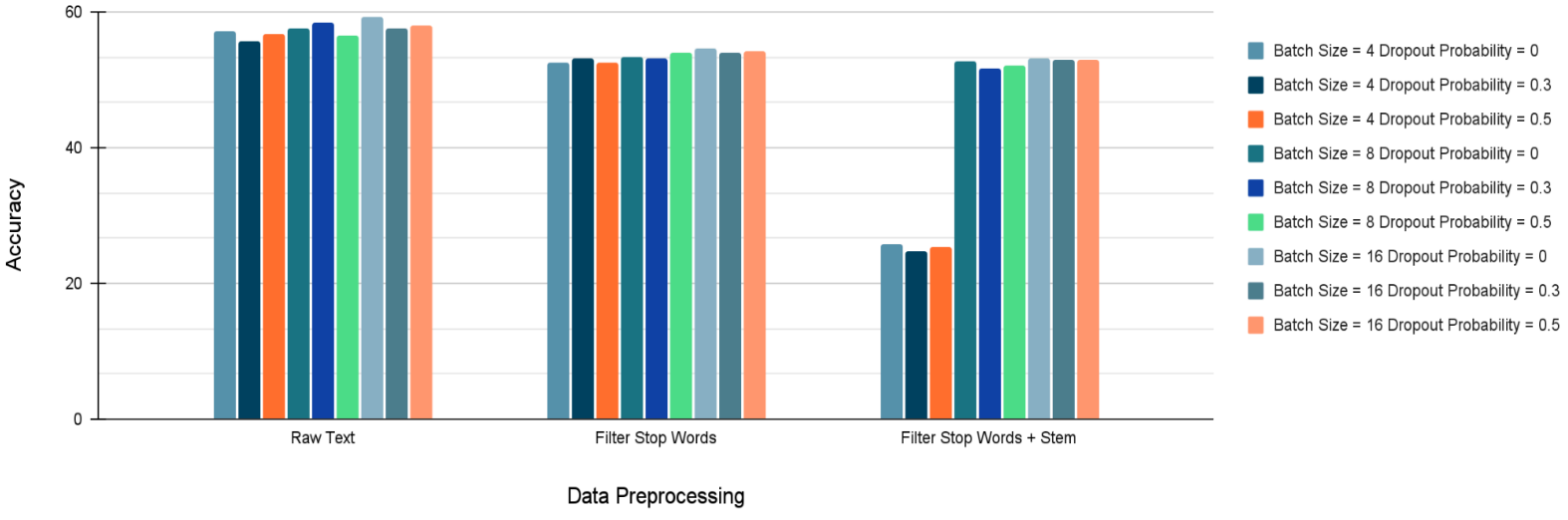
The model is underfitting and cannot learn all the features well with limited trainable parameters during our initial trials, the highest validation accuracy is 41%. So we decided to increase the complexity of our model with more trainable parameters.

2. Finetune BERT + Linear Layer

Model Architecture Description

As mentioned above, we need to add more trainable parameters to enhance the model's learning capacity. Instead of adding more linear layers, we try to just unfreeze the BERT model and finetune it with our dataset. Although the architecture of this design is still simple, we yield a significant performance increase. We have tried to train this model with different hyperparameters and data preprocessing settings. Below shows some of the results with large performance increases.

Experiment Results (For table format see Appendix)



Analysis on Results

Using preprocessing techniques on input texts like filtering out stopwords or using stemming on top of stopwords filtering does not help improve the accuracy of the model but instead decreases the model's performance. While the batch size and dropout rate have less impact on the model's performance, we found a batch size of 16 and a dropout rate of 0 turns out to be the best combination for model training among all tested settings. The model reaches the highest accuracy (59.20%) on validation set when using raw texts as data inputs with a batch size of 16 and dropout rate of 0 during training.

3. Freeze BERT + LSTM

Model Architecture Description

Fine-tuning BERT can be resource intensive (Used all 10GB VRAM on my RTX3080), therefore in this part, parameters in BERT are frozen using `with torch.no_grad():`. To compensate for the performance loss, an RNN built with LSTM as hidden units is added before the linear layer as we think RNN will be good at processing sentence input.

This new architecture has an additional 2-layer RNN with 256/512 hidden LSTM units after the BERT 768 dimension pooler output. At the end it's a {hidden units * 5} linear layer.

Experiment Results

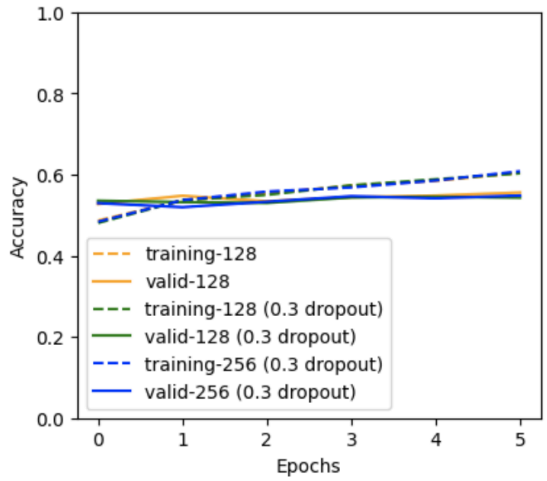
Thanks to more VRAM headroom, we can train at a batch size of 256. To promote faster convergence, we used Adam optimizer and learning rate scheduler to reduce the learning rate during the training.

Hyperparameters are configured as following:

optimizer: Adam, initial learning rate: 1e-3, weight_decay: 5e-4

learning rate scheduler: CosineAnnealingLR(T_max=7)

Epochs: 6, Training time: 9 min 45 seconds



In the training I noticed more epochs does not help much with the validation accuracy. The maximum increase in validation accuracy is at most 2%. Therefore I will present all 1-epoch training experiments in the following table.

Model	Training Acc. (%)	Validation Acc. (%)
128 hidden units, 0.3 dropout	48.8%	53.0%
256 hidden units, 0.3 dropout	48.8%	53.0%
512 hidden units, 0.3 dropout	49.1%	51.8%
128 hidden units	48.6%	53.0%
256 hidden units	49.0%	52.7%
512 hidden units	48.9%	51.9%

Analysis on Results

We've found out the RNN model we've tested in this configuration will overfit after one epoch on the training set. Moreover, the number of hidden units and dropout setting does not affect the validation scores. However, without fine-tuning the BERT layer to generate more correct embeddings for this movie dataset, it is hard for this frozen BERT experiment to 'catch' on the fine-tuned BERT results.

Final Model

Finetune BERT + Linear Layer

Data Preprocessing: None (Raw Texts)

Batch Size: 16

Dropout Probability for Linear Layer: 0

Accuracy on Validation Set: 59.20%

Appendix

1. Finetune BERT + Linear Layer Experiment Results Table

Data Preprocessing	Batch Size	Dropout Probability	Validation Accuracy
Raw Text	4	0	57.10
		0.3	55.65
		0.5	56.70
	8	0	57.60
		0.3	58.45
		0.5	56.50
	16	0	**59.20**
		0.3	57.70
		0.5	58.00
Filter Stop Words	4	0	52.65
		0.3	53.10
		0.5	52.50
	8	0	53.40
		0.3	53.25
		0.5	54.00
	16	0	54.60
		0.3	54.05
		0.5	54.15
Filter Stop Words + Stem	4	0	25.70
		0.3	24.80
		0.5	25.40
	8	0	52.80
		0.3	51.70
		0.5	52.05

	16	0	53.10
		0.3	53.05
		0.5	53.00