NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

AI6127 - Deep Neural Networks For Natural Language Processing, Assignment - I

March 11, 2024

Name: **Aradhya Dhruv (G2303518F)**

---

1 Introduction

In this report, a comprehensive exploration of sentiment analysis models is presented, encompassing experiments with various optimizers, epochs, and pre-trained embeddings. The report also explores the intricacies of different neural network architectures, including feedforward networks, convolutional neural networks (CNN), long short-term memory (LSTM), and bidirectional LSTM (BiLSTM). Through meticulous analysis of accuracy, precision, recall, and F1-score, this report provides valuable insights into the performance nuances of each model, ultimately identifying the most effective approach for sentiment analysis tasks.

Question 2: Conduct experiments with different optimizers: SGD, Adam, Adagrad and record the experimental results

SGD, ADAM and ADAGRAD each have certain benefits and tradeoffs over the other. SGD uses a single random training example (or a small batch) to calculate the gradient and update the model parameters. This random selection introduces randomness into the optimization process, hence the term "stochastic" in stochastic Gradient Descent. This also makes it a bit slower compared to other algorithms at the cost computational efficiency.

The given figure[1] shows the loss curve results observed on the test set when we trained the RNN model using either SGD, ADAM and ADAGRAD as our optimizer:

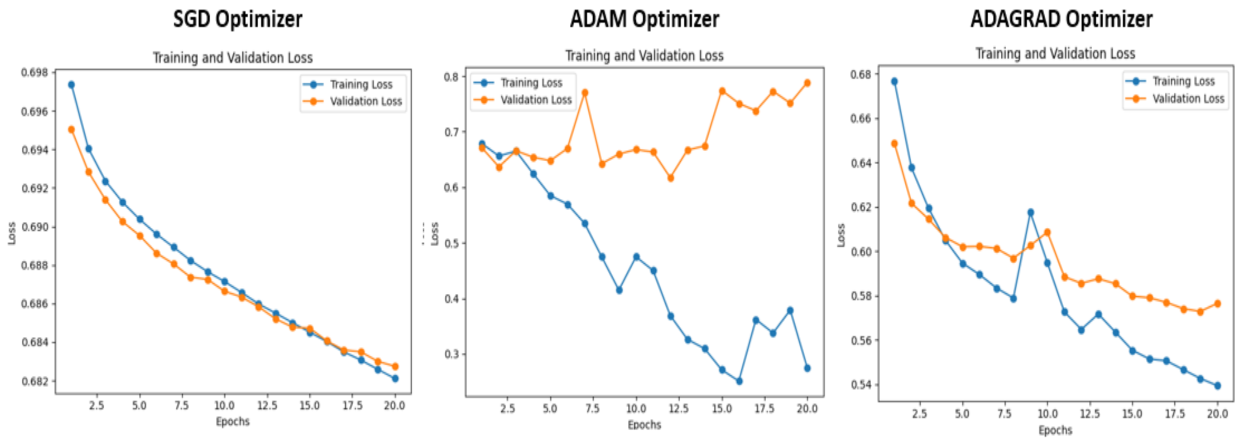

Figure 1: Training and Validation loss with different optimizers for 20 Epochs

Graph Inference

- The graph of SGD optimizer suggests that training was stopped prematurely, the training and validation loss was decreasing until 20 epochs and could have reduced further. Moreover, SGD is highly sensitive to learning rate, we can increase the learning rate to get a faster convergence with SGD
- The learning rate is a critical hyper parameter in ADAM optimizer. If the learning rate is too high, it can make significant adjustments to the model's weights during each training iteration. These large steps can cause the model to quickly latch onto specific patterns in the training data, potentially leading towards overfitting. As evident in the case of ADAM optimizer, the validation loss is increasing whereas, there is a gap between validation accuracy and training accuracy
- In case of ADAGRAD optimizer, validation loss goes down until a turning point is found, and there it starts going up again. That point represents the beginning of overfitting.

Question 3: Use Adam optimizer, conduct experiments with different number of epochs: 5, 10, 20, and 50

Table 1: Classification Report - RNN with ADAM Optimizer trained on 5, 10, 20 and 50 Epochs

Epochs	Sentiment	Precision	Recall	F1-Score	Accuracy
5	Negative	0.69	0.64	0.66	67.44%
	Positive	0.66	0.71	0.69	
10	Negative	0.74	0.68	0.71	**71.74%**
	Positive	0.70	0.76	0.73	
20	Negative	0.67	0.68	0.67	67.13%
	Positive	0.68	0.66	0.67	
50	Negative	0.68	0.63	0.66	66.95%
	Positive	0.66	0.71	0.68	

This table presents the performance of an RNN model with the ADAM optimizer trained for various epochs (training iterations). We'll analyze the observations focusing on precision, recall, F1-score, and accuracy.

- Precision-Recall Trade-off: As expected, there's a trade-off between precision and recall across epochs. For instance, in epoch 10, the model achieves higher recall for positive reviews (0.76) but lower precision (0.70) compared to epoch 5 (0.66 precision, 0.71 recall) (Refer Table[1]). This suggests the model might be capturing more true positives at epoch 10 but also making more false positives.
- Accuracy Fluctuation: The overall accuracy shows some fluctuation between 66% and 72%. This could be due to the model overfitting slightly on the training data at some points. Overfitting occurs when the model learns specific patterns in the training data that don't generalize well to unseen data.
- Limited Improvement after 10 Epochs: While the metrics improve slightly from epoch 5 to 10, there seems to be limited improvement beyond that. This suggests the model might be overfitting after 10 epochs.

Potential Improvements

- Experimenting with hyperparameters like learning rate, batch size, or RNN architecture could improve performance. Tuning these settings can lead to a better fit between the model and the dataset.
- Applying techniques like dropout or L1/L2 regularization can help prevent overfitting and improve generalization. These techniques can help the model focus on more generalizable features in the data.

Question 4: Use Adam optimizer and 50 epochs, download and use pretrained Word2Vec embeddings as initialization of the models; compare the performance with the previous one.

The RNN model with pre-trained Word2Vec embeddings achieved significantly higher accuracy (75.56%) compared to random initialization (66.95%). However, the erratic training curve (Refer to figure [2]) suggests that some pre-processing technqiues and alternative optimizers might be better suited when we use pretrained Word2Vec embeddings.

Table 2: Classification Report - RNN with ADAM Optimizer trained on 5, 10, 20 and 50 Epochs

Model	Sentiment	Precision	Recall	F1-Score	Accuracy
Randomly initialized embedding	Negative	0.68	0.63	0.66	66.95%
	Positive	0.66	0.71	0.68	
With Word2Vec embeddings	Negative	0.82	0.65	0.73	**75.56%**
	Positive	0.71	0.86	0.78	

- Negative Reviews: The Word2Vec model has a higher precision (0.82) than recall(0.65) for negative reviews, meaning most reviews classified as negative are truly negative. However, it might miss some actual negative reviews (lower recall: 0.65).
- Conversely, the same Word2Vec model has a higher recall (0.86) for positive reviews, meaning it captures most actual positive reviews but might also misclassify some negative reviews as positive due to lower precision.

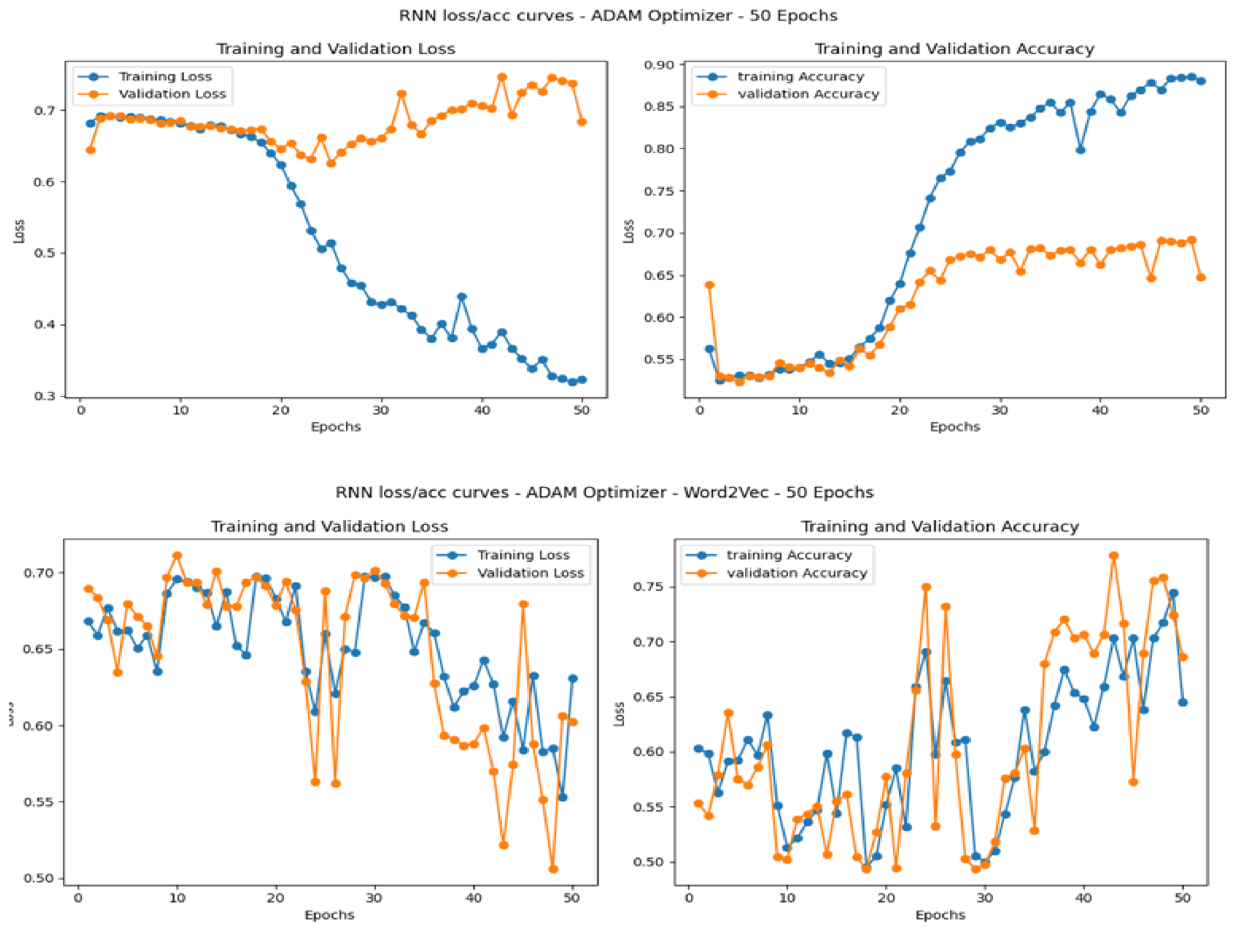

Figure 2: Top row - Loss curve **without** pretrained embedding, bottom row - Loss curve with Word2Vec embedding

Potential Improvements

- The fluctuations observed in training and validation loss while training Word2Vec model suggests that there may be noise in our training dataset which may be limiting the ability of the RNN model to find meaningful patterns in the data. We should consider pre-processing techniques like stopword removal, stemming etc.

- SGD optimizer might be better suited when we use Word2Vec embeddings since the convergence is slow and not aggressive
- Techniques like hyperparameter tuning (learning rate, optimizer) and early stopping can help the model converge more smoothly.
- We could normalize the Word2Vec embedding before utilizing it in RNN, this may remove the erratic training behavior observed in figure[2]
- Instead of directlly using the pretrained embeddings we can consider fine-tuning the pretrained Word2Vec embeddings during training. This allows the model to slightly adjust them for better alignment with training data

---

Question 5: Use Adam optimizer, 50 epochs and randomly initialized embeddings, run the experiments with the following models:

- One-layer feed forward neural network, hidden dimension is 500.
- Two-layer feed forward neural network, hidden dimensions are 500 and 300.
- Three-layer feed forward neural network, hidden dimensions are 500, 300, and 200
- CNN model (using three feature maps with the sizes are 1, 2, and 3)
- LSTM model

Model	No. of Trainable Parameters	Accuracy - Test Set
FFN (500 dimensions)	7,651,601	85.96%
FFN (500,200 dimensions)	7,801,701	86.60%
FFN (500,200,300 dimensions)	7,861,801	85.77%
CNN (Kernel size - 1,2,3)	8,403,601	**88.81**%
LSTM	9,105,101	86.14%
BiLSTM	10,709,601	85.22%

Table 3: Accuracy of different models long with their respective trainable parameter size

Answer: The table[3] provided above summarizes the accuracies observed for different networks along with their trainable parameter size. It also indicates that that the CNN model achieved the highest accuracy among all the models tested. Several factors could contribute to this:

- CNNs are less sensitive to the position of features in the input sequence. In sentiment analysis, the location of sentiment-bearing words or phrases may vary, and the CNN's position-invariant property allows it to capture such information more effectively.
- The hierarchical structure of CNNs allows for automatic feature learning at different levels of abstraction. This is advantageous in sentiment analysis, where sentiments can be expressed through various linguistic nuances and combinations of words.
- CNN has a relatively lower number of trainable parameters compared to the LSTM and BiLSTM, which might contribute to better generalization.

Model	Sentiment	Precision	Recall	F1-Score
FFN (500 dimensions)	Negative	0.83	0.90	0.87
	Positive	0.89	0.82	0.85
FFN (500,300 dimensions)	Negative	0.87	0.87	0.87
	Positive	0.87	0.87	0.87
FFN (500,300,200 dimensions)	Negative	0.87	0.85	0.86
	Positive	0.85	0.87	0.86
CNN (Kernel size - 1,2,3)	Negative	0.90	0.87	0.89
	Positive	0.88	0.90	0.89
LSTM	Negative	0.83	0.91	0.87
	Positive	0.90	0.81	0.85
BiLSTM	Negative	0.88	0.81	0.85
	Positive	0.83	0.89	0.86

Table 4: Caption

Individual Model Explanations

1. One-layer feed forward neural networks (Dimensions - [500], [500,300] and [500,300,200]) Precision and recall variations for feed forward neural networks indicate challenges in capturing nuanced sentiment patterns, possibly due to overfitting.

- **FFN (500 dimensions):**
  - Competitive accuracy but struggles to find a balance between precision and recall for both positive and negative sentiments. (Refer Table[3] and [4])
  - The three-dimensional configuration may indicate challenges in capturing nuanced sentiment patterns, possibly due to overfitting.
  - Increasing dimensions does not consistently improve performance, suggesting potential limitations in the model's capacity to leverage additional layers effectively.
- **FFN (500,200 dimensions):**
  - Limited improvement in accuracy compared to the 500-dimensional configuration.
  - The addition of a second layer does not consistently enhance the model's ability to capture sentiment patterns.
  - Challenges in achieving a well-balanced performance across precision and recall for both sentiment classes. (Refer Table[4])
- **FFN (500,200,300 dimensions):**
  - Slight drop in accuracy compared to the 500-dimensional configuration. (Refer Table[3])
  - The three-dimensional configuration faces challenges in maintaining a balanced trade-off between precision and recall for positive and negative sentiments.
  - Limited improvement with the addition of the third layer, indicating potential limitations in the model's capacity to benefit from increased complexity.
- **CNN (Kernel size - 1,2,3):**
  - Outperforms other models with the highest accuracy, precision, and recall on the test set, showcasing effective sentiment pattern capture. (Refer Table[3] and [4]). However, the model is seems to be overfitting quite aggressively (Refer Image 3) due to a high

  learning rate. We can consider decreasing the learning rate or increasing the vocabulary size to introduce complexity in the training data.
- The choice of kernel sizes allows the model to capture information at different granular levels, contributing to its success.
- Achieves a well-balanced precision and recall for both sentiment classes, indicating robust performance.

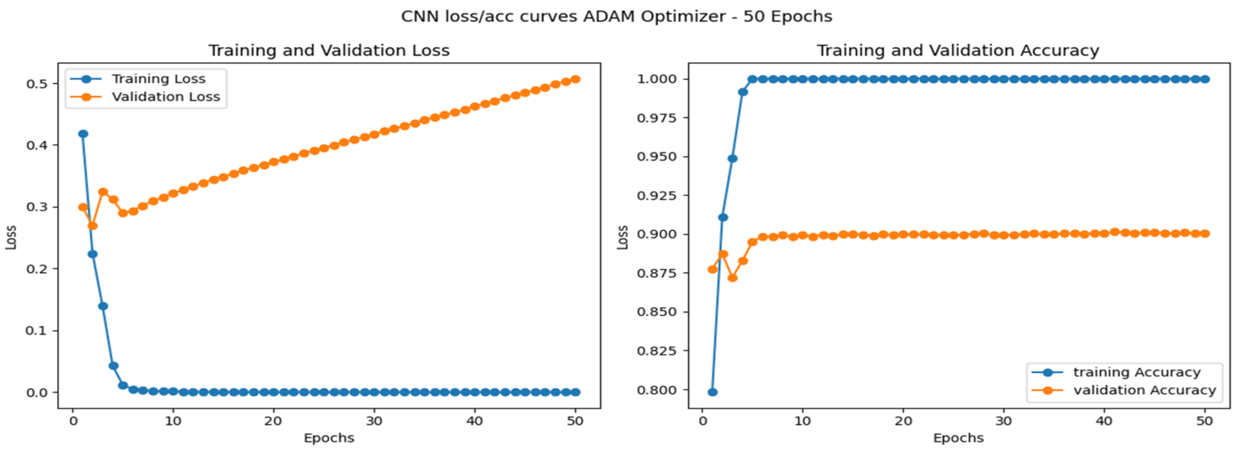

Figure 3: Training and Validation loss with CNN Model - ADAM Optimizer and 50 Epochs

- **Long Short-Term Memory (LSTM):**
  - Competitive accuracy, but slightly lower than the CNN.
  - Challenges in achieving a balanced precision and recall, especially for positive sentiments. (Refer Table[4])
  - Struggles with capturing long-range dependencies, impacting its ability to discern positive sentiment patterns effectively.
- **Bidirectional Long Short-Term Memory (BiLSTM):**
  - Lowest accuracy among the models, indicating difficulties in capturing sentiment patterns.
  - Precision is higher for negative sentiments, suggesting a better ability to identify negative expressions. (Refer Table[4])
  - Faces challenges in achieving a balanced trade-off between precision and recall for both sentiment classes, similar to the LSTM.

Conclusion

In summary, our experiments revealed insights on optimizers, epochs, and pre-trained embeddings. Adam optimizer proved effective but required careful hyper-parameter tuning, and epochs beyond 10 showed diminishing returns. Pre-trained Word2Vec embeddings significantly boosted accuracy but also explored ways to mitigate the erratic training behavior. Among all the models, CNN excelled due to position invariance and hierarchical feature learning. Whereas, feedforward networks faced challenges in precision-recall balance. Likewise, LSTM and BiLSTM showed competitive yet less balanced performance. Overall, CNN demonstrated superior accuracy and generalization, making it the preferred choice for sentiment analysis tasks.