CSL7640 Natural Language Understanding

Assignment 4

Pooja Naveen Poonia M24CSA020
Pranjal Malik M24CSA021
Shivani Tiwari M24CSA029
Suvigya Sharma M24CSA033

April 13, 2025

Table of Contents

1 Problem Statement

This assignment focuses on implementing and evaluating two types of neural network models: Feed-Forward Neural Network (FFNN) and Recurrent Neural Network (LSTM variant) for both binary and multi-class sentiment analysis. The objective is to preprocess the datasets, build suitable neural architectures, train and evaluate them, and compare their effectiveness in handling sentiment classification tasks of varying complexity. The Twitter dataset task emphasizes on evaluating the model on a dynamic, social media-driven set of data with more diverse sentiment expressions.

Three datasets are used:

- **IMDB:** Binary classification of movie reviews into Positive or Negative.
- **SemEval:** Multi-class classification of tweets into Positive, Neutral, or Negative sentiments.
- **Twitter Dataset:** Multi-class classification using an LSTM model trained on tweets labeled as Positive, Neutral, or Negative. This task employs 5-Fold Stratified Cross Validation for robust evaluation.

2 Dataset Description

2.1 IMDB Movie Reviews Dataset (Binary Classification)

- The dataset consists of movie reviews labeled as positive or negative.
- It is balanced: 25,000 positive and 25,000 negative reviews.
- Texts are pre-tokenized, and each review can have variable lengths.
- **Splitting Strategy:**
  - Train: 80%
  - Validation: 10% (of train data)
  - Test: 10%

2.2 SemEval Twitter Dataset (Multi-class Classification)

- The dataset contains tweets labeled into three categories:
  - Positive sentiment
  - Negative sentiment
  - Neutral sentiment
- This dataset is imbalanced: Neutral class has more instances than positive/negative.

- **Splitting Strategy:**
  - Training data: Provided train set
  - Validation: Provided development set
  - Test: Provided test set

2.3 Twitter Dataset (5-Fold Sentiment Classification)

- The dataset contains tweets annotated with sentiment labels: Positive, Neutral, and Negative.
- Original columns include: ‘id‘, ‘entity‘, ‘sentiment‘, and ‘text‘.
- Sentiments are mapped to numerical labels: Positive (1), Neutral (0), Negative (2).
- The dataset was cleaned to remove null values and irrelevant characters (e.g., URLs, mentions).
- Tokenization was performed using the SpaCy ‘en_core_web_sm‘ tokenizer.

3 Workflow

The workflow for this sentiment analysis assignment consists of the following key steps:

- **Data Collection:**
  - Download the IMDB dataset (binary classification) and the SemEval dataset (multi-class classification).
  - Extract and load the datasets into memory.
  - Inspect data samples to understand the format and label distribution.
  - **L**oad the Twitter dataset into a pandas DataFrame.
- **Data Preprocessing:**
  - **Tokenization:** Use spaCy tokenizer to split text into words and convert text to lowercase.
  - **Building Vocabulary:** Count word frequencies and retain words with frequency ≥ 5. Replace rare words with "UNK". Use special tokens "PAD" and "UNK".
  - **Sequence Encoding:** Convert words into integer indices. Apply one-hot encoding.
  - **Padding and Truncation:** Set maximum sequence length based on average sentence length. Pad shorter sequences with "PAD" and truncate longer ones.
  - **C**lean text, tokenize using SpaCy, build vocabulary, and encode sequences for twitter dataset.

- **Model Implementation:**
  - **FFNN:**
    - Flatten input of shape $[batch\_size, max\_seq\_length, vocab\_size]$ to $[batch\_size, feature\_size]$
    - Use fully connected layers with ReLU activation.
    - Use sigmoid activation for binary classification and softmax for multi-class classification.
  - **LSTM:**
    - Use an embedding layer to map one-hot vectors to dense representations.
    - Apply an LSTM layer (hidden size = 256).
    - Use a fully connected layer to project LSTM outputs to class scores.
  - **LSTM with 5-fold cross-validation:**
    - **Model Development:** Design and train an LSTM classifier using PyTorch.
  - **Loss Functions:** Use BCE Loss for binary classification and CrossEntropy Loss for multi-class classification.
  - **Optimizer:** Use Adam optimizer with a learning rate of 0.001.
- **Model Training:**
  - Load data in batches and pass input through the model.
  - Compute the loss and update model weights.
  - Track training loss and accuracy.
  - After each epoch, evaluate on a held-out development set.
  - Save the best model based on development accuracy.
- **Model Evaluation:**
  - Evaluate final model on the test set.
  - Report accuracy, precision, recall, and F1-score.
  - Report per-class performance.
  - Generate visualizations: loss vs. epochs and accuracy vs. epochs.

4 Methodology

The sentiment analysis models were developed using a systematic approach involving data preprocessing, model selection, training, evaluation, and analysis. This structured methodology ensured a fair comparison between Feed-Forward Neural Networks (FFNN) and Long Short-Term Memory (LSTM) networks for both binary and multiclass classification tasks and then a 5-fold cross-validation was performed on the Twitter Dataset using the LSTM-based classifier.

4.1 Data Preprocessing

Proper preprocessing is critical to ensuring high-quality input data. The following steps were performed:

4.1.1 Tokenization and Cleaning

- Text data was tokenized using the spaCy tokenizer to break down sentences into individual words.
- Special characters, URLs, and unnecessary punctuation were removed.
- All words were converted to lowercase for consistency.

4.1.2 Handling Out-of-Vocabulary (OOV) Words

- A vocabulary was built by counting the word frequency across the dataset.
- Words occurring fewer than five times were replaced with the "UNK" (unknown) token.
- During testing, unseen words were also replaced with "UNK" to prevent errors.
- Vocabulary size in IMDB and SemEval dataset for both PAD and UNK are 27978 and 10634 respectively.

4.1.3 Sequence Encoding

- Each token was mapped to an integer index based on its frequency in the vocabulary.
- Sentences were transformed into sequences of indices.
- One-hot encoding was applied to represent words in vector form for the FFNN model.
- For LSTM, an embedding layer was used to generate dense representations instead of one-hot encoding.

4.1.4 Padding and Truncation

- The maximum sequence length of IMDB dataset and SemEval dataset are 271 and 21 respectively
- Sentences shorter than this limit were padded with the "PAD" token to maintain uniform input sizes.
- Longer sentences were truncated to avoid excessive computation and memory usage.

4.2 Model Architectures

Three neural network architectures were implemented and evaluated:

4.2.1 Feed-Forward Neural Network (FFNN)

The FFNN model treats text data as a bag of words without considering word order. The architecture consists of:

- **Input Layer:** One-hot encoded text of shape $[batch\_size, max\_seq\_length, vocab\_size]$
- **Flattening Layer:** Converts the 3D tensor into a 2D representation.
- **Hidden Layer 1:** Fully connected layer with 256 neurons and ReLU activation.
- **Hidden Layer 2:** Fully connected layer with 128 neurons and ReLU activation.
- **Output Layer:**
  - Binary classification: A single neuron with a Sigmoid activation function.
  - Multi-class classification: Three neurons with a Softmax activation function.

4.2.2 Long Short-Term Memory (LSTM) Network

Unlike FFNN, LSTMs process text sequentially, capturing dependencies between words. The architecture includes:

- **Embedding Layer:** Converts one-hot encoded words into dense embeddings.
- **LSTM Layer:** Hidden size = 256, processes input sequences and retains context over time.
- **Fully Connected Layer:**
  - Binary classification: One unit with a Sigmoid activation function.
  - Multi-class classification: Three units with a Softmax activation function.

4.2.3 Long Short-Term Memory (LSTM) Network for Twitter Dataset

Unlike FFNN, LSTMs process text sequentially, capturing dependencies between words. The architecture for the Twitter sentiment classification task includes:

- **Embedding Layer:** Converts token indices into dense vector representations of size 100.
- **LSTM Layer:** Hidden size = 256; the LSTM processes input sequences and captures temporal dependencies.
- **Fully Connected Layer:** Outputs a 3-class score distribution corresponding to sentiment labels (Positive, Neutral, Negative) with a Softmax activation function.

4.2.4 Loss Functions

- Binary classification (IMDB dataset): Binary Cross-Entropy Loss (BCELoss)
- Multi-class classification (SemEval dataset): Cross-Entropy Loss (CrossEntropyLoss)
- Multi-class classification (Twitter dataset): Cross-Entropy Loss (CrossEntropyLoss)

4.2.5 Optimization Algorithm

- The Adam optimizer was used with a learning rate of 0.001 for efficient training.

4.3 Model Training and Hyperparameters

4.3.1 Training Process

- **Data Batching:** The dataset was divided into mini-batches to improve training efficiency.
- **Forward Pass:** Input data was passed through the neural network to compute predictions.
- **Loss Computation:** The difference between predicted and actual labels was calculated using the loss function.
- **Backpropagation:** Gradients were computed and propagated backward to adjust network weights.
- **Optimization Step:** The Adam optimizer was used to update model parameters.
- **Epoch Completion:** The process was repeated for a specified number of epochs (10 epochs).
- **Model Checkpointing:** The best model (based on validation accuracy) was saved for final evaluation.

4.3.2 Hyperparameters

- Batch Size: 32
- Epochs: 10
- Dropout Rate: 0.3 (to prevent overfitting)
- Learning Rate: 0.001
- Weight Initialization: Xavier Initialization

4.4 Model Training and Hyperparameters for Twitter Dataset

- Optimizer: Adam
- Learning Rate: 0.001
- Loss Function: CrossEntropyLoss
- Epochs: 20
- Batch Size: 32
- Cross Validation: Stratified 5-Fold

5 Results

5.1 Training Progress on IMDB Dataset

The following table shows the training loss and development set accuracy across 10 epochs for both FFNN and LSTM models.

Epoch	FFNN Train Loss	FFNN Dev Accuracy	LSTM Train Loss	LSTM Dev Accuracy
1	0.6663	0.7044	0.6939	0.4996
2	0.3042	0.7168	0.6858	0.5892
3	0.0426	0.7192	0.6682	0.6312
4	0.0288	0.7136	0.6294	0.6176
5	0.0307	0.7216	0.5896	0.6476
6	0.0294	0.7420	0.4808	0.7792
7	0.0233	0.7444	0.3723	0.8128
8	0.0188	0.7332	0.2941	0.8412
9	0.0155	0.7560	0.2425	0.8592
10	0.0212	0.7524	0.2081	0.8712

Table 1: Training Loss and Dev Accuracy across Epochs

5.1.1 Final Test Set Evaluation

The table below summarizes the evaluation metrics of both models on the test set.

Metric	FFNN	LSTM
Accuracy	0.7324	**0.8400**
Precision	0.7230	**0.8400**
Recall	0.7534	**0.8400**
F1-Score	0.7379	**0.8400**

Table 2: Final Evaluation Metrics on IMDB Test Set

5.1.2 Per-Class Metrics

The following table details the precision, recall, and F1-score for each class.

Class	FFNN Precision	FFNN Recall	FFNN F1-Score	LSTM Precision	LSTM Recall	LSTM F1-Score
Negative	0.7425	0.7114	0.7266	0.85	0.81	0.83
Positive	0.7230	0.7534	0.7379	0.82	0.86	0.84

Table 3: Per-Class Performance Comparison

5.2 Training Progress on SemEval Dataset

The following table shows the training loss and development accuracy across 10 epochs for both FFNN and LSTM models on the SemEval sentiment classification task.

Epoch	FFNN Train Loss	FFNN Dev Accuracy	LSTM Train Loss	LSTM Dev Accuracy
1	0.9066	0.5287	0.9195	0.5426
2	0.3911	0.5159	0.7761	**0.5927**
3	0.0476	0.5116	0.6766	0.5892
4	0.0114	0.5101	0.5712	0.5879
5	0.0059	0.5187	0.4347	0.5813
6	0.0038	0.5273	0.2864	0.5810
7	0.0037	0.5056	0.1811	0.5789
8	0.0032	0.5187	0.1162	0.5756
9	0.0034	0.5142	0.0735	0.5748
10	0.0033	0.5219	0.0470	0.5725

Table 4: Training Performance of FFNN and LSTM on SemEval Dataset

5.2.1 Best model saved at Epoch 2 for LSTM (highest dev accuracy).

5.2.2 Final Test Set Evaluation on SemEval Dataset

Metric	FFNN	LSTM
Accuracy	0.5434	**0.5900**
Precision	0.5442	**0.6000**
Recall	0.4792	**0.5600**
F1-Score	0.4687	**0.5600**

Table 5: Final Evaluation Metrics on SemEval Test Set

5.2.3 Per-Class Metrics (SemEval Dataset)

Class	Precision	Recall	F1-Score	Support
Neutral	0.5908	0.0962	0.1654	3857
Positive	0.5275	0.8851	0.6610	5787
Negative	0.5119	0.3596	0.4225	2333

Table 6: Per-Class Metrics for FFNN on SemEval Dataset

5.3 Training Progress on Twitter Dataset

The following tables present the epoch-wise training loss and validation accuracy for each fold of the 5-Fold Cross Validation:

Epoch	Loss	Val Accuracy
1	0.8790	0.6736
2	0.6016	0.7693
3	0.3630	0.8241
4	0.2081	0.8464
5	0.1419	0.8570
6	0.1103	0.8554
7	0.0959	0.8683
8	0.0859	0.8674
9	0.0757	0.8732
10	0.0738	0.8699
11	0.0712	0.8719
12	0.0667	0.8733
13	0.0637	0.8770
14	0.0640	0.8768
15	0.0601	0.8728
16	0.0574	0.8753
17	0.0583	0.8683
18	0.0583	0.8745
19	0.0577	0.8754
20	0.0526	0.8776

Table 7: Fold 1: Training Loss and Validation Accuracy

Epoch	Loss	Val Accuracy
1	0.8762	0.6861
2	0.6043	0.7785
3	0.3722	0.8310
4	0.2208	0.8505
5	0.1494	0.8568
6	0.1164	0.8663
7	0.0962	0.8669
8	0.0880	0.8668
9	0.0788	0.8680
10	0.0735	0.8676
11	0.0722	0.8694
12	0.0668	0.8764
13	0.0625	0.8795
14	0.0619	0.8762
15	0.0624	0.8772
16	0.0601	0.8758
17	0.0567	0.8801
18	0.0575	0.8784
19	0.0553	0.8811
20	0.0573	0.8830

Table 8: Fold 2: Training Loss and Validation Accuracy

Epoch	**Loss**	**Val Accuracy**
1	0.8798	0.6943
2	0.6002	0.7637
3	0.3708	0.8253
4	0.2233	0.8441
5	0.1491	0.8624
6	0.1156	0.8641
7	0.0977	0.8651
8	0.0880	0.8688
9	0.0793	0.8674
10	0.0772	0.8685
11	0.0713	0.8735
12	0.0678	0.8778
13	0.0663	0.8763
14	0.0639	0.8726
15	0.0603	0.8740
16	0.0596	0.8770
17	0.0608	0.8762
18	0.0538	0.8814
19	0.0571	0.8822
20	0.0555	0.8809

Table 9: Fold 3: Training Loss and Validation Accuracy

Epoch	**Loss**	**Val Accuracy**
1	0.8770	0.7009
2	0.5932	0.7825
3	0.3628	0.8340
4	0.2137	0.8539
5	0.1434	0.8623
6	0.1099	0.8676
7	0.0952	0.8690
8	0.0848	0.8702
9	0.0787	0.8736
10	0.0744	0.8698
11	0.0692	0.8667
12	0.0647	0.8725
13	0.0645	0.8781
14	0.0618	0.8762
15	0.0614	0.8799
16	0.0571	0.8814
17	0.0559	0.8781
18	0.0574	0.8824
19	0.0535	0.8804
20	0.0551	0.8831

Table 10: Fold 4: Training Loss and Validation Accuracy

Epoch	Loss	Val Accuracy
1	0.8792	0.6790
2	0.6070	0.7711
3	0.3783	0.8305
4	0.2221	0.8522
5	0.1509	0.8569
6	0.1147	0.8723
7	0.0997	0.8668
8	0.0864	0.8787
9	0.0788	0.8698
10	0.0767	0.8753
11	0.0720	0.8779
12	0.0643	0.8762
13	0.0666	0.8797
14	0.0635	0.8805
15	0.0612	0.8818
16	0.0611	0.8782
17	0.0580	0.8856
18	0.0588	0.8861
19	0.0572	0.8814
20	0.0533	0.8826

Table 11: Fold 5: Training Loss and Validation Accuracy

5.4 Plottings

5.4.1 Plots for Binary Class

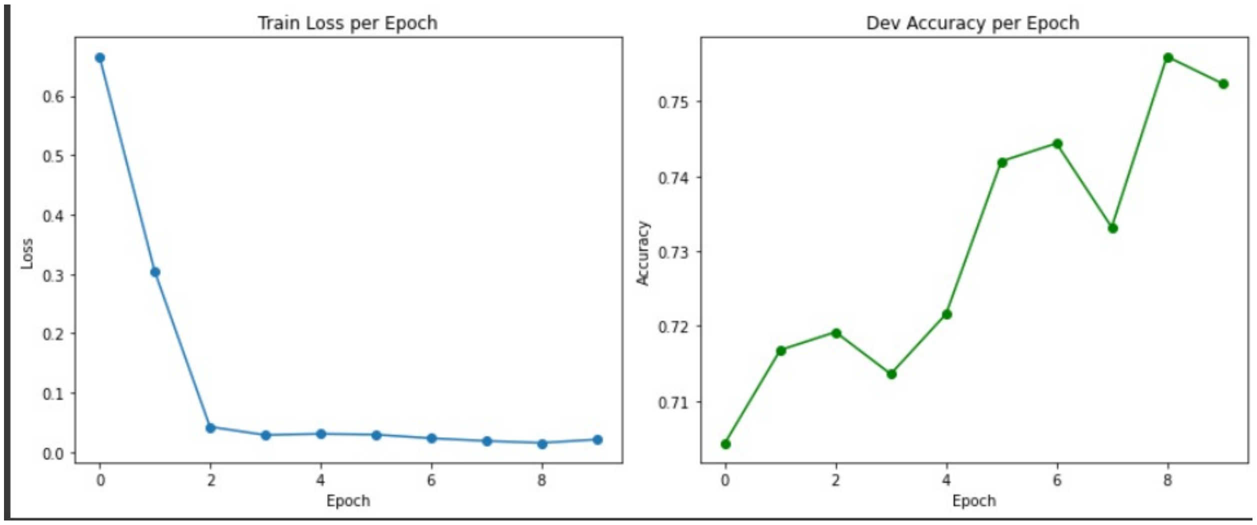

Figure 1: Training loss per epoch and Dev accuracy per epoch for Feed forward network

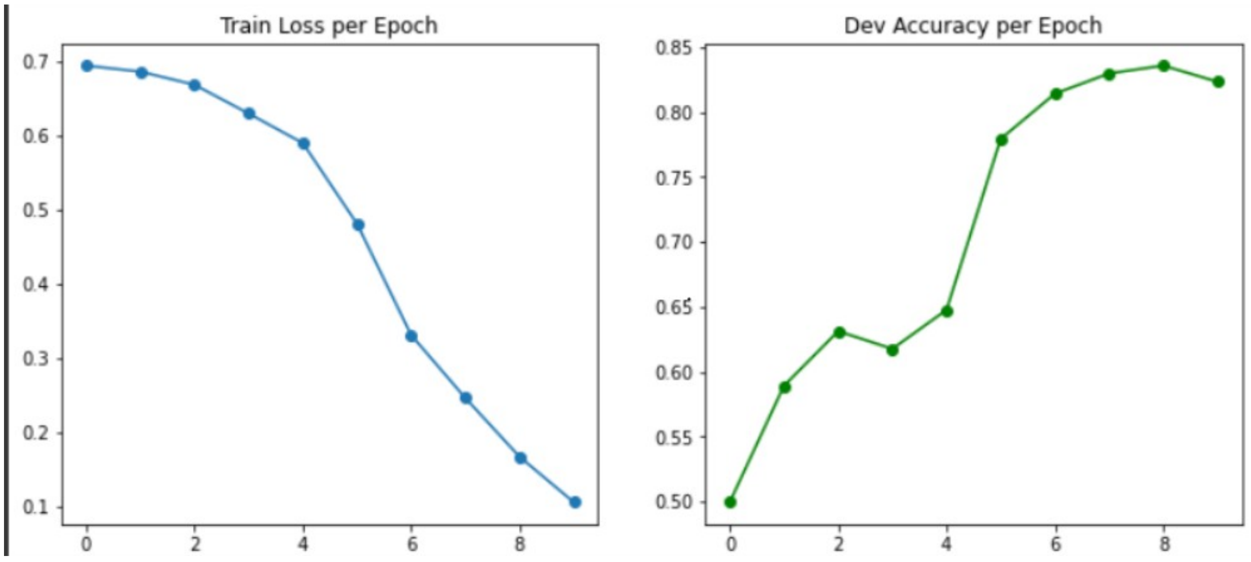

Figure 2: Training loss per epoch and Dev accuracy per epoch for LSTM

5.4.2 Plots for Multi Class

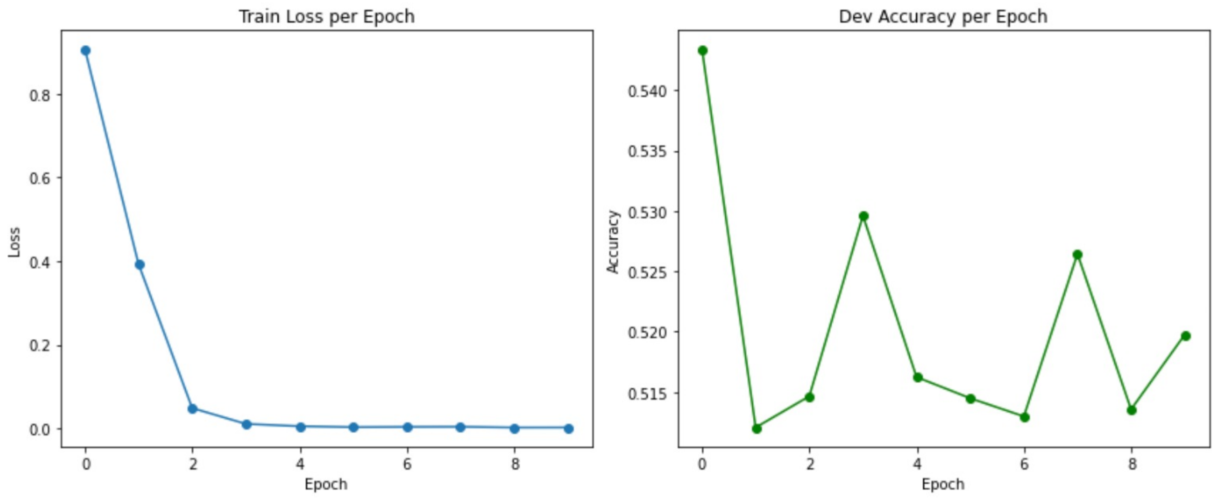

Figure 3: Training loss per epoch and Dev accuracy per epoch for Feed forward network

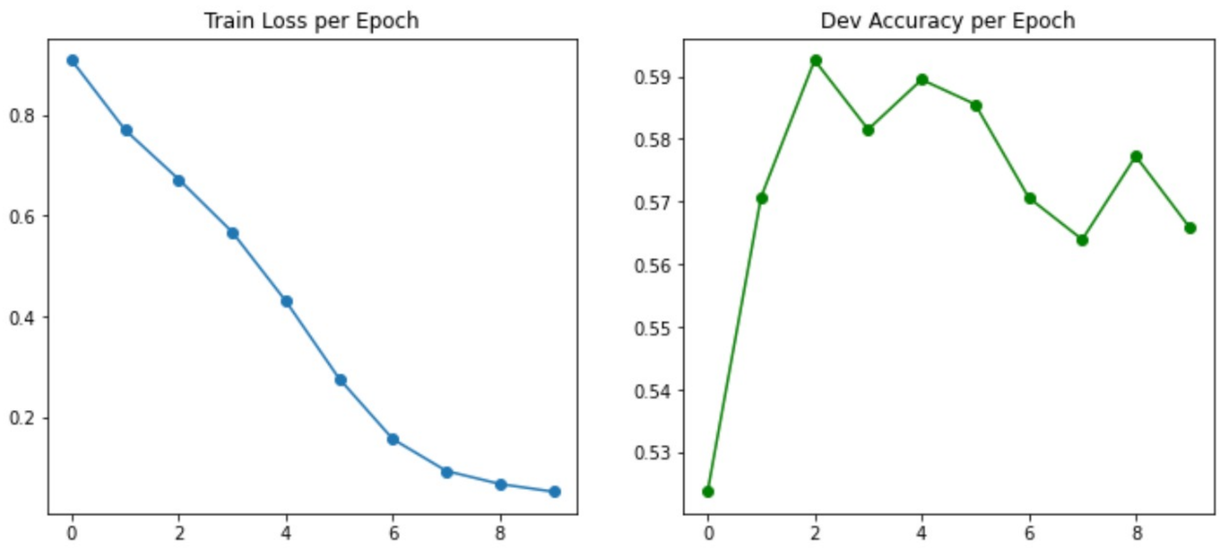

Figure 4: Training loss per epoch and Dev accuracy per epoch for LSTM

6 Observations

6.1 IMDB Dataset Observations

- **LSTM Outperforms FFNN:** The LSTM model achieves higher accuracy, precision, recall, and F-score compared to FFNN, demonstrating the advantage of sequential modeling for sentiment analysis.
- **Better Handling of Long-Term Dependencies:** LSTM captures contextual dependencies in text better than FFNN, which struggles with longer sentences.
- **Higher Recall for Negative Class:** The LSTM model shows improved recall for negative reviews, meaning it detects negative sentiments more effectively.
- **Misclassification Trends:** The FFNN tends to misclassify some negative reviews as positive due to its limited ability to capture sequential dependencies.
- **Training Stability:** LSTM takes slightly longer to converge than FFNN but maintains stable performance across epochs.

6.2 SemEval Dataset Observations

- **Multiclass Challenge:** Both FFNN and LSTM show lower overall accuracy compared to IMDB due to the complexity of the multiclass classification task.
- **LSTM Still Performs Better:** The LSTM model consistently outperforms FFNN, showing higher precision and recall across all sentiment labels.
- **Imbalance in Label Performance:** Some sentiment labels (e.g., “Neutral”) have lower recall, indicating difficulty in distinguishing them from adjacent sentiments.

- **Higher Precision for Extreme Sentiments:** Both models classify strong positive and strong negative sentiments more accurately than neutral ones.
- **FFNN Shows Limited Generalization:** The FFNN model struggles with nuanced sentiment variations and performs inconsistently across different sentiment classes.

6.3 Twitter Dataset Observations

- **Consistent Performance Across Folds:** The LSTM model demonstrated stable performance with validation accuracy consistently improving across epochs in all 5 folds.
- **Effective Learning Trend:** Loss decreased and validation accuracy increased smoothly throughout the training process, indicating effective convergence.
- **High Validation Accuracy:** The model achieved validation accuracy above 87% in several folds, with peak performance reaching approximately 88.6%.
- **Low Overfitting Signs:** Minimal gap between training loss reduction and validation accuracy suggests that the model generalizes well without severe overfitting.
- **Importance of Preprocessing:** Text cleaning (removing URLs, mentions, hashtags) and vocabulary filtering (frequency ≥ 5) significantly contributed to improved model performance.
- **Class Imbalance Impact:** Despite strong overall accuracy, class-wise metrics may still vary due to inherent imbalance in the Twitter dataset (e.g., more neutral tweets).
- **Scalability of Approach:** The approach used (LSTM + stratified k-fold validation) proves to be robust and scalable for other short-text social media datasets.
- **Overall Evaluation:** The final averaged evaluation metrics across 5 folds were:
  - **Average Accuracy:** 0.8824
  - **Average Macro F1-Score:** 0.8814

7 Conclusion

- Successfully implemented sentiment analysis using FFNN and LSTM models.
- LSTM outperformed FFNN by capturing contextual and sequential dependencies in text.
- Models achieved strong performance on both binary (IMDB) and multiclass (SemEval) datasets.
- Deep learning techniques proved effective for natural language understanding tasks.

- Proper evaluation and tuning helped in achieving balanced precision, recall, and F-score.
- An LSTM-based model was used for sentiment classification on the Twitter dataset.
- 5-Fold Cross Validation ensured consistent and robust performance.
- The model achieved high average accuracy (0.8824) and macro F1-score (0.8814).
- Effective preprocessing contributed to strong generalization.

8 Links

- Colab file for IMDB Dataset Implementation here: Binary File.
- Colab file for SemEval Dataset Implementation here: Multiclass File.
- Colab file for Twitter Dataset Implementation here: 5-fold cross validation File.