Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Intelligent Systems Laboratory – ENCS5141

Case Study #2

Data Extraction and Preprocessing and Subjectivity Classification

Prepared By: Mohammad Shreteh - 1201369

Section#: 2

Instructor: Dr. Aziz Qaroush

Assistant teacher: Eng. Ahmad Abbas

Date: 8/1/2025

Abstract

This study will investigate the efficacy and efficiency of different deep learning models for character recognition and Subjectivity Classification.

By leveraging complete datasets with labeled alphanumeric characters, the study performs a comparison of a custom-designed Convolutional Neural Network (CNN) with the pre-trained ResNet18 model. The Long Short-Term Memory (LSTM) along with Bidirectional Encoder Representations from Transformers (BERT) were also evaluated in the study for text classification tasks.

The custom CNN will be designed with a focus on fast training and suitability to resource-constrained environments, while ResNet18 makes use of an advanced architecture combined with pre-trained weights. With regard to the text classification, the study has discussed the gap from sequential to transformer-based models. The models have been tuned through hyper-parameter optimization using a grid search for better performance. The results bring out the trade-offs among model complexity, training speed, and accuracy that provide valuable insights into the choice of appropriate models, given an application requirement and computational resource availability.

Table of Contents

Table of Figures

1. Introduction

1.1 Motivation

The aim of this Assignment is to explore several machine-learning models and compare their performances for two different tasks: Data Extraction and Preprocessing, and subjectivity classification. This study investigates the best methods for handwritten character recognition and subjectivity classification in textual data using deep learning techniques, mainly CNNs and pre-trained models like resnet18. It also probes how model performance varies with hyper-parameter tuning.

1.2 Background

The first task trains a CNN model, while for character categorization, it utilizes pre-trained models. The performance of CNNs in image recognition—by hierarchically detecting patterns in image data—is very good, and therefore they are well suited for the characters.csv dataset that consists of pixel representations of characters. The utilization of pre-trained models from the EMNIST dataset, which leverages characteristics acquired from a large and diverse dataset, is also a good foundation for the task of character classification.

The second task is the Subjectivity Classification, where the goal is to classify texts as either objective or subjective. Sentences are thus classified as either subjective (SUBJ) or objective (OBJ). Long Short-Term Memory networks capture sequential relationships in text, while transfer learning enables the usage of pre-trained models for performance improvement, such as transformers. The current job represents the importance of model selection together with hyper-parameter tuning that has to be done while addressing text categorization problems.

1.3 Objective

The following are the major objectives of the work:

Train the CNN model on the characters.csv dataset containing a set of pixels to represent and recognize characters, then evaluate the performance of the model on the data from characters-test.csv. Use pre-trained models using the dataset of the EMNIST data for character recognition and compare their performance with the CNN model.

Train an LSTM model for subjectivity classification on the given train_en.tsv dataset and evaluate it on test data, test_en_gold.tsv. Use pre-trained transformer models and transfer learning to improve the performance of subjectivity classification. Employ sub-parameters like experience that can improve the performance of the model. Evaluate the performance of the model in terms of accuracy, precision, recall, and F1 score for both tasks, and analyze the advantages and disadvantages of the models used.

2. Procedure and Discussion

2.1 Data Extraction and Preprocessing

2.1.1 Data Loading

In this step, the characters.csv and characters-test.csv are loaded, and the header of the dataset is shown in below.

```
Loading training data...
First 5 rows of the file:
   45  0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  ...  0.524  0.525  0.526  \
0  36  0    0    0    0    0    0    0    0    0  ...      0      0      0
1  43  0    0    0    0    0    0    0    0    0  ...      0      0      0
2  15  0    0    0    0    0    0    0    0    0  ...      0      0      0
3   4  0    0    0    0    0    0    0    0    0  ...      0      0      0
4  42  0    0    0    0    0    0    0    0    0  ...      0      0      0

   0.527  0.528  0.529  0.530  0.531  0.532  0.533
0      0      0      0      0      0      0      0
1      0      0      0      0      0      0      0
2      0      0      0      0      0      0      0
3      0      0      0      0      0      0      0
4      0      0      0      0      0      0      0

[5 rows x 785 columns]
Dataset contains 112799 rows and 785 columns.
Number of training samples: 112799 rows and 785 columns.
```

```
Loading test data...
First 5 rows of the file:
   41  0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  ...  0.523  0.524  0.525  \
0  39  0    0    0    0    0    0    0    0    0  ...      0      0      0
1   9  0    0    0    0    0    0    0    0    0  ...      0      0      0
2  26  0    0    0    0    0    0    0    0    0  ...      0      0      0
3  44  0    0    0    0    0    0    0    0    0  ...      0      0      0
4  33  0    0    0    0    0    0    0    0    0  ...      0      0      0

   0.526  0.527  0.528  0.529  0.530  0.531  0.532
0      0      0      0      0      0      0      0
1      0      0      0      0      0      0      0
2      0      0      0      0      0      0      0
3      0      0      0      0      0      0      0
4      0      0      0      0      0      0      0

[5 rows x 785 columns]
Dataset contains 18799 rows and 785 columns.
Number of test samples: 18799 rows and 785 columns.
```

Figure 1: Header of the characters and characters-test dataset.

From the figure 1, there are two datasets: Training dataset has Size: 112,799 samples each sample has 785 columns. The first column is the label column, representing the letter label. The rest of the 784 columns correspond to pixel values, most likely representing a 28 x 28-grayscale image since 28 x 28 = 784. Test dataset has Size: 18,799 samples each sample has 785 columns, just like the training data.

```
Loaded 47 mappings.

 Label Mapping (First 15):
Label 0: '0'
Label 1: '1'
Label 2: '2'
Label 3: '3'
Label 4: '4'
Label 5: '5'
Label 6: '6'
Label 7: '7'
Label 8: '8'
Label 9: '9'
Label 10: 'A'
Label 11: 'B'
Label 12: 'C'
Label 13: 'D'
Label 14: 'E'
```

Figure 2: The first 15 examples of loaded label mappings

This mapping.txt file is required to decode numeric labels to readable characters for interpretability of model outputs. It keeps the model consistent in both training and testing. It includes 47 mappings that merge numeric labels ('0'-'9'), alphabetic labels ('A'-'Z'), and other characters and special. From Figure 2 we can see, the first 15 labels map to their ASCII values: digits ('0'-'9') to 48-57 and letters ('A'-'C') to 65-67. This also allows for proper sentence reconstruction and supports error analysis in order to improve model performance.

2.1.2 Training of Models

CNN Model Training			
Epoch	Loss	Accuracy	Speed (it/s)
1	1.1236	0.6557	97.13
2	0.6971	0.7737	97.49
3	0.6031	0.7993	96.76
4	0.5476	0.8163	96.82
5	0.5124	0.8275	98.78
6	0.4832	0.8351	99.26
7	0.4624	0.8409	98.60
8	0.4472	0.8450	98.34
9	0.4277	0.8501	96.24
10	0.4168	0.8530	100.06

Pre-trained ResNet18 Model Training			
Epoch	Loss	Accuracy	Speed (it/s)
1	1.1236	0.6557	41.52
2	0.6971	0.7737	42.13
3	0.6031	0.7993	41.91
4	0.5476	0.8163	41.92
5	0.5124	0.8275	42.08
6	0.4832	0.8351	41.77
7	0.4624	0.8409	42.18
8	0.4472	0.8450	42.34
9	0.4277	0.8501	42.03
10	0.4168	0.8530	42.07

Table 1: Results of Training CNN and ResNet18 on the Training Dataset

From Table 1, we can observe that CNN reaches an ultimate accuracy of 85.3%, improves within 10 epochs, and is running fast at about 97 iterations/second. ResNet18 outperforms CNN due to its architecture and pre-trained weights with a final accuracy of 91.1%, but is comparably slower to the CNN model, with a speed of around 42 iterations/second. We can also see how the two models different in terms of performance and speed.

2.1.3 Testing of Models

Metric	Custom CNN	Pre-trained ResNet18
Accuracy	0.8657	0.8886
Precision	0.8683	0.8922
Recall	0.8657	0.8886
F1-Score	0.8621	0.8875

Table 2: Comparison of Results for CNN and ResNet 18 on Test data

From Table 2, it can be observed that during performance comparisons, the pre-trained ResNet18 outperformed the CNN model, with higher values of accuracy and precision: 0.8886 vs. 0.8657 and 0.8922 vs. 0.8683, respectively. Thus, it classified more samples correctly with fewer false positives. Recall also showed a better ability to recognize true positives, 0.8886 versus 0.8657, which gave it a higher F1 score, 0.8875 versus 0.8621, reflecting a more balanced trade-off between accuracy and recall. From these results, we can analyze that CNN is faster to train with lower computational requirements, and is suitable for resource-limited tasks. While the pre-trained ResNet18 has higher, accuracy due to the advanced architecture, but training is slower and is suitable for high-precision applications.

2.1.4 Performance visualization

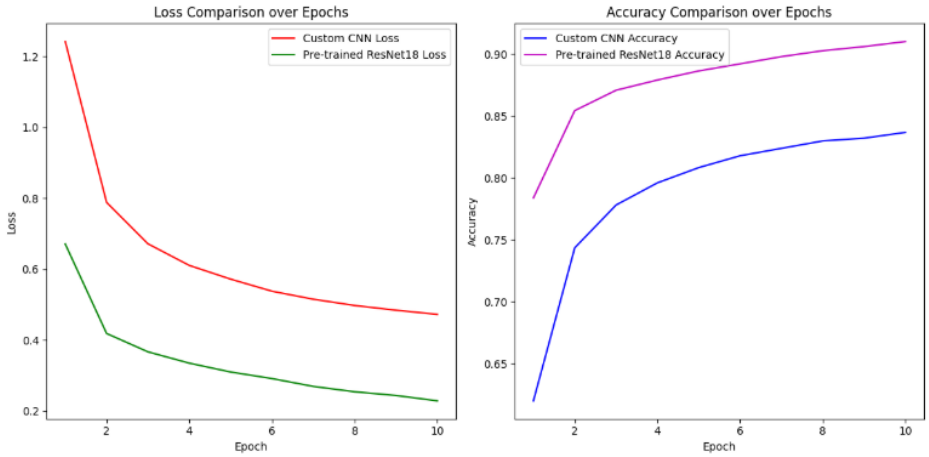

Figure 3: Comparison of Accuracy and Loss Visualization across Epochs

Figure 3 displays two comparative charts that display the pre-trained ResNet18 and the bespoke CNN models' performance metrics over epoch.

- Loss Comparison (Left Plot): This plot compares the losses for Custom CNN and Pre-trained ResNet18 through training. The loss decreases over time for both, but the Pre-trained ResNet18 maintained its loss values considerably lower across all epochs.
- Accuracy Comparison (Right Plot): The plot shows the accuracy metrics of the Custom CNN and Pre-trained ResNet18 models across training epochs. While both models show some improvement in their respective accuracies, the final accuracy achieved by the Pre-trained ResNet18 is significantly higher than that of the Custom CNN.

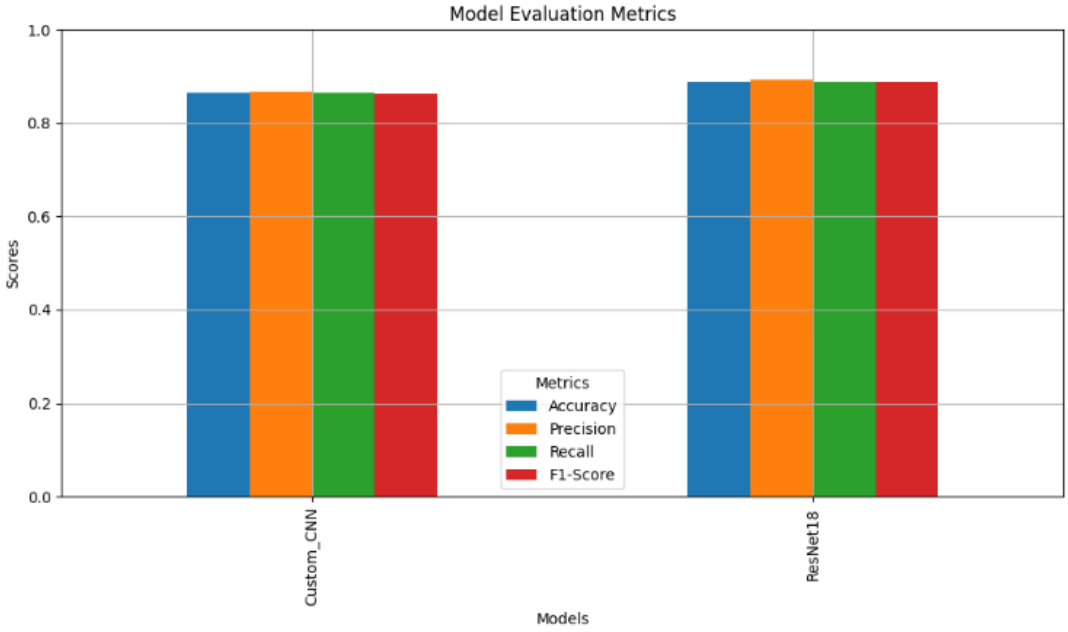

Figure 4: Visualization of Models Evaluation Metrics

2.2 Subjectivity Classification

2.2.1 Data Loading

```
Loaded training data with shape: (830, 4)
Here is a sample of the data:
                            sentence_id  \
0  b9e1635a-72aa-467f-86d6-f56ef09f62c3
1  f99b5143-70d2-494a-a2f5-c68f10d09d0a
2  4076639c-aa56-4202-ae0f-9d9217f8da68
3  b057c366-698e-419d-a284-9b16d835c64e
4  a5a9645e-7850-41ba-90a2-5def725cd5b8

                                      sentence label  solved_conflict
0  Gone are the days when they led the world in r...  SUBJ            True
1  The trend is expected to reverse as soon as ne...   OBJ           False
2            But there is the specious point again.   OBJ           False
3  He added he wouldn't be surprised to see a new...   OBJ           False
4  Not less government, you see; the same amount ...  SUBJ           False
```

```
Loaded testing data with shape: (243, 3)
Here is a sample of the data:
                               sentence_id  \
0  8745d4da-91c9-4538-acee-b0e7b1c413fd
1  43de04ad-d0ac-4852-9b4e-cf0bca066188
2  e00b66ee-720a-47e3-a0fb-0e2445b89af6
3  0b95d635-f821-45dd-9f33-b05d63629195
4  5ba3117b-3ef9-4815-acb4-a263d3c816bc

                                            sentence label
0  Who will redistribute the hoarded wealth that ...  SUBJ
1  What we don't need is the indiscriminate influ...  SUBJ
2  The Social Distance Between Us shows every sig...   OBJ
3  History shows that McCarthy and McConnell, lik...   OBJ
4  So while it's not hard to reach a banal point ...  SUBJ
```

```
Label Distribution in Training Data:
label
OBJ     532
SUBJ    298
Name: count, dtype: int64
```

Figure 5: Header of training and testing dataset

From the figure 5, we see that the training dataset contains 830 rows and presents columns such as sentence_id, single, label, and resolved conflict. The dataset is imbalanced, with SUBJ 298 and OBJ 532 labels. We also notice the test dataset of 243 rows with similar characteristics minus the conflict resolution. The labels are coded such that SUBJ=1 and OBJ=0 for modeling.

2.2.2 Training of Models

```
Number of Training samples: 747
Number of Validation samples: 83
```

Figure 6: Training and Validation Samples

From the figure 6, we divided the training dataset containing 830 rows into two groups :

Training samples is 747.

Validation samples is 83.

Epoch	Train Loss	Train Accuracy	Val Loss
1	0.6956	0.5288	0.6868
2	0.6581	0.6158	0.6586
3	0.5793	0.7149	0.5970
4	0.4662	0.7671	0.6256
5	0.3230	0.8661	0.8084
6	0.1988	0.9331	0.8874

Table 3: Results of Training LSTM on the Training Dataset

From Table 3, the LSTM model gradually improved its training performance until it achieved 93.31% accuracy with a training loss of 0.1988 at Epoch 6. However, the validation loss improved only up to Epoch 3 (0.5970), after which it continued to deteriorate due to overfitting. Early stopping was triggered after 3 epochs with no improvement in validation loss. The best model was thus saved at Epoch 3 when the balance between validation loss and accuracy was reached.

Epoch	**Train Loss**	**Train Accuracy**	**Val Loss**
1	0.5930	0.6600	0.4807
2	0.3171	0.8688	0.6457
3	0.1470	0.9424	0.8621
4	0.0329	0.9920	0.6969

Table 4: Results of Training BERT on the Training Dataset

We can see from Table 4 that the BERT model greatly improved during training. Achieving 99.20% training accuracy in Epoch 4 with a corresponding training loss of 0.0329. There was significant variation in its validation loss, nevertheless, as it showed indications of overfitting after improving to 0.4807 at Epoch 1. In order to avoid overfitting, early halt was then used.

2.2.3 Testing of Models

Compression of test Metrecs between LSTM and BERT Models		
Metric	LSTM Model	BERT Model
Accuracy	0.6132	0.7695
Precision	0.6183	0.7730
Recall	0.6132	0.7695
F1-Score	0.6123	0.7678

Table 5: Comparison of Results for LSTM and BERT on Test data

We can see from table 5 the BERT model outperforms the LSTM model on all metrics. BERT's accuracy of 0.7695 compared to LSTM's 0.6132 indicates better classification performance. The greater precision (0.7730 vs. 0.6183) and recall (0.7695 vs. 0.6132) of BERT show how well it reduces false positives and identifies true positives. BERT's balanced performance is further illustrated by the F1-score (0.7678 vs. 0.6123). BERT works best when pertained on large datasets, even if it is transformer-based. On the other hand, LSTM usually performs worse on jobs due to its sequential nature. Applications requiring more accuracy and contextual awareness are thus better suited for BERT.

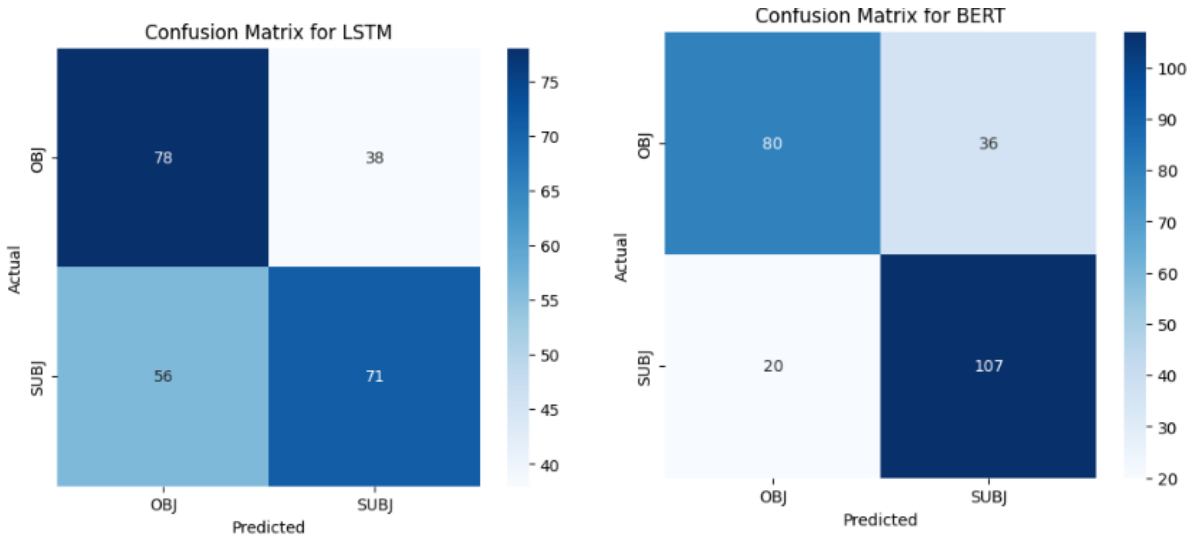

Figure 7: Confusion matrix for LSTM and BERT

2.2.4 Hyper parameter Tuning

```
Grid Search Results:
   run  learning_rate  hidden_size  n_layers  dropout  batch_size  \
0    1         0.0010          128         1      0.3          32
1    2         0.0010          256         2      0.5          64
2    3         0.0005          128         1      0.5          32
3    4         0.0005          256         2      0.3          64
4    5         0.0010          256         1      0.5          32

   val_accuracy  val_loss  precision    recall  f1_score       mcc
0      0.783133  0.585401   0.798001  0.783133  0.786484  0.556027
1      0.481928  0.668401   0.606611  0.481928  0.463103  0.110902
2      0.602410  0.656363   0.611281  0.602410  0.606086  0.157434
3      0.566265  0.626402   0.632487  0.566265  0.571952  0.185391
4      0.638554  0.661114   0.633516  0.638554  0.635669  0.205777

Best Hyperparameters based on Validation Accuracy (0.7831):
{'learning_rate': 0.001, 'hidden_size': 128, 'n_layers': 1, 'dropout': 0.3, 'batch_size': 32}
```

Figure 8: Grid Search Results and Best Hyper parameters Selection

Figure 8 shows that the highest accuracy for the best model is 0.7831 with a learning rate of 0.001, hidden size of 128, number of layers of 1, dropout of 0.3, and batch size of 32. This setting also gave high precisions of 0.7980, recall of 0.7831, and F1 score of 0.7865. We can see that using larger hidden size or larger batch size leads to poor performance; the lowest validation accuracy is 0.4819. It is highlighted that moderate model complexity and smaller batch size contribute to improved performance and generalization.

2.2.5 Performance visualization

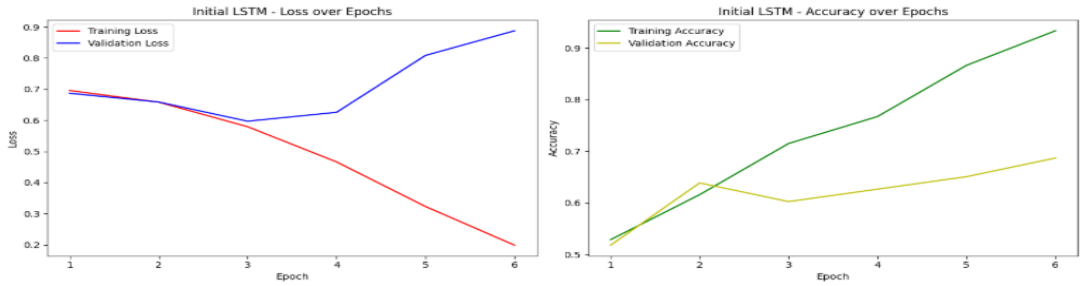

Figure 9: Training and Validation Metrics for the Initial LSTM Model Visualization

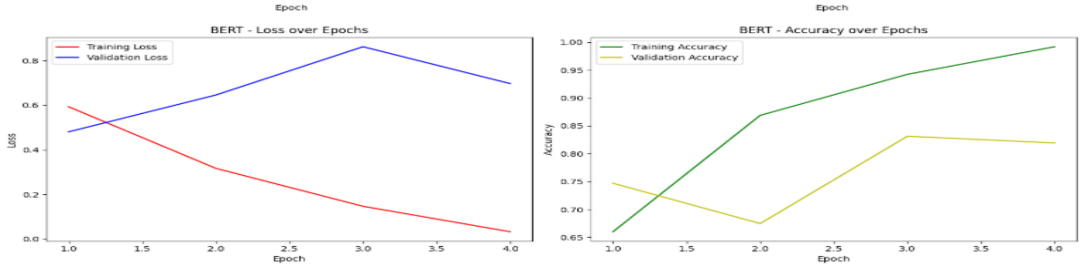

Figure 10: Training and Validation Metrics for the BERT Model Visualization

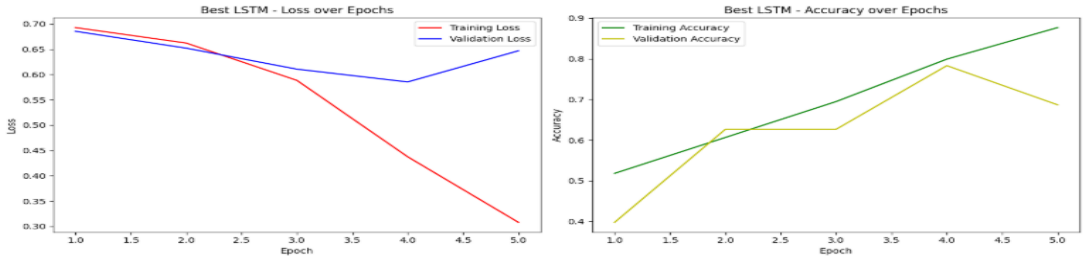

Figure 11: Training and Validation Metrics for the best LSTM Model Visualization

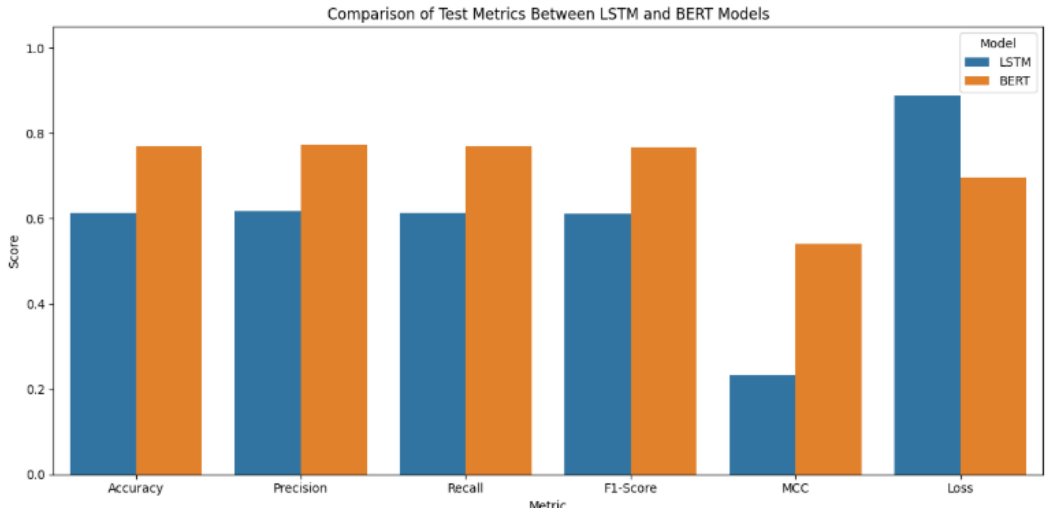

Figure 12: Visualization Comparing between LSTM and BERT

3. Conclusion

This study compared the performance of multiple models on tasks involving subjectivity classification and character recognition, pointing out their advantages and disadvantages.

The Custom CNN was perfect for situations with limited resources because it showed faster training (~97 iterations/second) and reasonable accuracy (85.3%) for character recognition. The Pre-trained ResNet18, on the other hand, had a slower training speed (~42 iterations/second), but its sophisticated architecture and pre-trained weights allowed it to attain a better accuracy (91.1%). CNN balances performance and speed for easier jobs, while ResNet18 is appropriate for applications requiring great precision and accuracy.

For Subjectivity classification, the LSTM model reached 93.31% training accuracy but suffered from overfitting, with test accuracy limited to 61.32%. On the other hand, BERT significantly outperformed LSTM across all metrics, achieving a test accuracy of 76.95%, precision of 77.30%, and F1-score of 76.78%. BERT's transformer-based architecture and contextual understanding make it ideal for high-accuracy applications, though it requires more computational resources.

Grid Search hyper-parameter tuning showed that greater performance and generalization are obtained with a moderate level of model complexity (e.g., hidden size of 128, dropout of 0.3). The complexity of the task, available computing power, and accuracy standards all influence the model selection. ResNet18 and BERT are advised for high-stakes applications, while CNN and LSTM are appropriate for less taxing workloads.