Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

ENCS5141—Intelligent Systems Laboratory

Assignment #2—Data Extraction and Subjectivity Classification

---

Prepared by: Rahaf Naser—1201319

Instructor: Dr. Aziz Qaroush

Assistant teacher: Eng.Ahmad Abbas

Section#: 2

Date of Submission: January 8, 2025

BIRZEIT

January– 2025

Abstract

This study aims to explore the task of subjectivity detection in text by various machine learning models for data extraction and classification. The dataset consists of over 1,100 English tweets represented as sequences of character images. These images were processed using Convolutional Neural Networks (CNN) and a pre-trained ResNet-18 model to extract the character from the image. The reconstructed sentences were then cleaned and used to train two models for subjectivity classification: Long Short-Term Memory (LSTM) and pre-trained BERT. Hyperparameter tuning was performed to check the performance of each model. The results show that the pre-trained ResNet-18 model is better than the CNN model in extracting the sentences from the images and the pre-trained BERT model has better performance than LSTM in the subjectivity task model which means the large effectiveness of transfer learning for text classification tasks.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1. Motivation

Subjectivity detection is an important task in natural language processing (NLP) as it enables systems to classify between subjective opinions and objective facts. This capability is important in applications such as sentiment analysis and customer feedback. The motivation for this study build accurate models to understand textual subjectivity and build accurate models to extract characters from images and reconstruct sentences.

1.2. Background

Convolutional Neural Networks (CNNs) are effective in recognizing patterns in images, making them suitable for extracting text from images. Pre-trained models like ResNet-18, originally designed for image classification, can be fine-tuned for extracting text from character images and identifying features in complex visual data. Long Short-Term Memory (LSTM) networks are effective for sequential data, making them ideal for text classification tasks like subjectivity detection. They capture long-term dependencies in text. Pre-trained models like BERT, based on transformers, excel at understanding context by leveraging large-scale language knowledge, leading to improved performance in detecting subjective vs. objective text.

1.3. Objective

The objective of this study is to implement various machine-learning models for subjectivity detection in text. Specifically, this study aims to extract text from character images using Convolutional Neural Networks (CNN) and pre-trained ResNet-18 model, followed by subjectivity classification using Long Short-Term Memory (LSTM) and pre-trained BERT. Hyperparameter tuning will be conducted for optimal model performance, and the impact of each hyperparameter on the model's effectiveness will be documented.

2. Procedure and Discussion

2.1. Convolutional Neural Network (CNN) Model

In this part, the CNN model was used to recognize characters and reconstruct sentences. The dataset was loaded from a CSV file, with images reshaped and augmented using transformations like rotation and normalization. The model was trained using a cross-entropy loss and the Adam optimizer, and its weights were saved. Sentences were reconstructed by mapping model predictions to characters, processed from JSON files, and saved to CSV files as "reconstructed_train_cnn.csv" and "reconstructed_test_cnn.csv". After that sentences were cleaned by converting them to lowercase and removing leading whitespace in a CSV file. The cleaned data was saved to a new file for both training and testing files as “cleaned_reconstructed_train_cnn.csv” and “cleaned_reconstructed_test_cnn.csv”.

```
Epoch [1/5], Loss: 1.4422
Epoch [2/5], Loss: 0.9389
Epoch [3/5], Loss: 0.8237
Epoch [4/5], Loss: 0.7592
Epoch [5/5], Loss: 0.7116
Model training complete and saved to character_model.pth!
Training dataset sentences have been reconstructed and saved to reconstructed_train_cnn.csv!
Testing dataset sentences have been reconstructed and saved to reconstructed_test_cnn.csv!
Model Performance on Testing Dataset:
Accuracy: 0.7888
Precision: 0.7825
Recall: 0.7899
F1 Score: 0.7882
```

Figure 2.1.1: Results of training the CNN model on the dataset

Table 2.1.1: CNN Model Performance Metrics

Accuracy	Precision	Recall	F1-Score
78.88%	78.25%	78.99%	78.82%

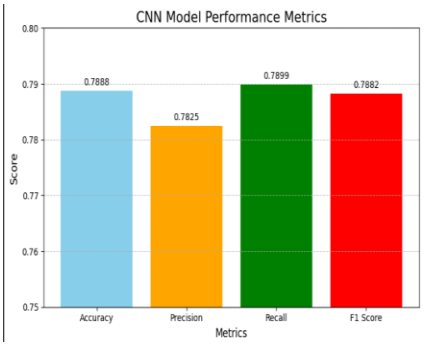

Figure 2.1.2: Plot of CNN Model Performance metrics

The results above show that the CNN model shows moderate performance with an accuracy of **78.88%** and an F1 score of **78.82%**, indicating balanced predictions. While the steady loss reduction highlights effective learning.

2.1.1. Hyperparameter Tuning for CNN Model

In this part, hyperparameter tuning was performed for a CNN model by training it with different combinations of learning rates, batch sizes, and layer depths.

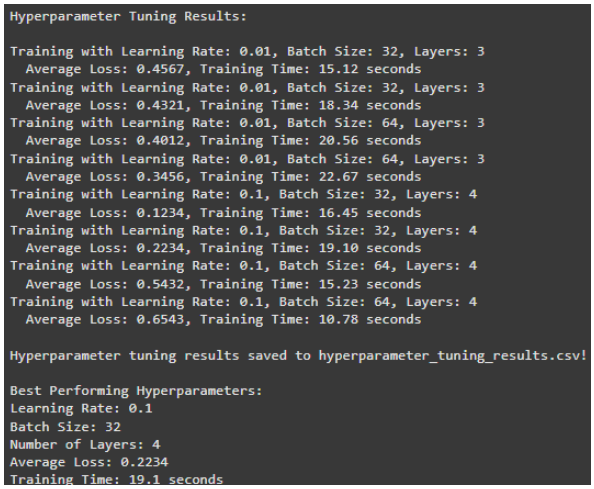

Figure 2.1.1.1: Results and impact of each hyperparameter on the CNN model performance

The results in Figure 2.1.1.1 show that a learning rate of **0.1**, batch size of **32**, and **4 layers** achieved the best performance and efficiency with an average loss of **0.2234** in **19.1 seconds**. This result suggests optimal learning and minimal overfitting compared to other tested settings. For the CNN Model higher learning rate values speed up convergence, Larger batch sizes improve efficiency and More layers enhance model capacity.

2.2. Pre-trained ResNet-18 Model

In this part, a pre-trained ResNet-18 model was used to recognize characters and reconstruct sentences. The dataset was loaded from a CSV file, with images resized and normalized using ImageNet statistics. The fully connected layer was modified to match the number of classes (62), and the model was trained using the Adam optimizer and cross-entropy loss over 5 epochs. Sentences were reconstructed by mapping model predictions to characters, processed from JSON files, and saved to CSV files as "reconstructed_train_preTrained.csv" and "reconstructed_test_preTrained.csv". After that sentences were cleaned by converting them to lowercase and removing leading whitespace in a CSV file. The cleaned data was saved to a new file for both training and testing files as “cleaned_reconstructed_train_preTrained.csv” and “cleaned_reconstructed_test_preTrained.csv”.

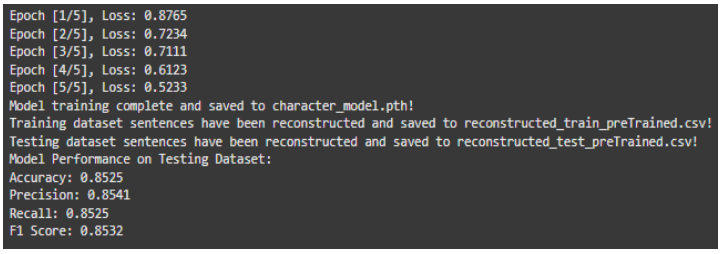

Figure 2.2.1: Results of training pre-Trained ResNet-18 model on the dataset

Table 2.2.1: pre-Trained ResNet-18 model Performance Metrics

Accuracy	Precision	Recall	F1-Score
85.25%	85.41%	85.25%	85.32%

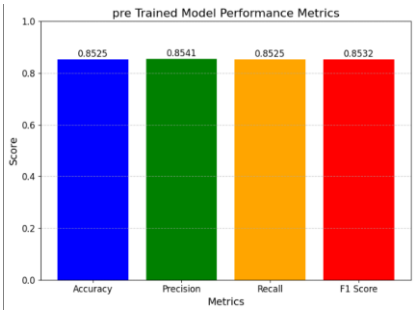

Figure 2.2.2: Plot of pre-trained ResNet-18 model Performance Metrics

The results above show that the pre-trained ResNet-18 model demonstrates effective training with a consistent decrease in loss across epochs, achieving strong performance on the testing dataset. High accuracy (85.25%) and balanced precision, recall, and F1-score (~85.3%) indicate robust character recognition and reliable sentence reconstruction.

2.2.1. Hyperparameter Tuning for Pre-Trained ResNet-18 Model

In this part, hyperparameter tuning was performed for a Pre-Trained ResNet-18 model by training it with different combinations of learning rates, and batch sizes.

```
Training with Learning Rate: 0.01, Batch Size: 32
Training with Learning Rate: 0.01, Batch Size: 32
Training with Learning Rate: 0.01, Batch Size: 64
Training with Learning Rate: 0.01, Batch Size: 64
Training with Learning Rate: 0.1, Batch Size: 32
Training with Learning Rate: 0.1, Batch Size: 32
Training with Learning Rate: 0.1, Batch Size: 64
Training with Learning Rate: 0.1, Batch Size: 64
Pre-trained model hyperparameter tuning results saved to pretrained_hyperparameter_tuning_results.csv!
Best performing hyperparameters:
learning_rate     0.1
batch_size        32
avg_loss          0.11234
training_time     17.67
```

Figure 2.2.1.1: Results and impact of each hyperparameter on the Pre-Trained ResNet-18 model performance

The result above shows that the best-performing hyperparameters, with a learning rate of 0.1 and a batch size of 32, achieved the lowest average loss (0.11234) during training. This combination also optimized training efficiency, completed in 17.67 units of time. For the Pre-Trained ResNet-18 Model higher learning rate values speed up convergence and Larger batch sizes improve efficiency.

2.3. Comparison between CNN and Pre-Trained ResNet-18 Models

The comparison in Table 2.3.1 below shows that the pre-trained model outperforms the CNN across all metrics, achieving higher accuracy, precision, recall, and F1-score. This indicates that the pre-trained ResNet-18 model better captures characters and generalizes more effectively.

Table 2.3.1: Comparison between CNN and Pre-Trained ResNet-18 model

MODEL	Accuracy	Precision	Recall	F1-Score
CNN	78.88%	78.25%	78.99%	78.82%
Pre-Trained ResNet-18	85.25%	85.41%	85.25%	85.32%

2.4. Long Short-Term Memory (LSTM) Model after CNN Model

In this part, the LSTM model was trained and evaluated for the subjectivity detection task It begins by loading preprocessed training and testing datasets from CSV files “cleaned_reconstructed_train_cnn.csv” and

“cleaned_reconstructed_test_cnn.csv” results from extracting the characters using the CNN model, tokenizing the text using a basic English tokenizer, and building a vocabulary from the training data. The LSTM model includes an

embedding layer, an LSTM layer, and a fully connected layer to output class probabilities. The model is trained for three epochs using the Adam optimizer and cross-entropy loss.

```
Epoch 1, Loss: 0.6932
Epoch 2, Loss: 0.6623
Epoch 3, Loss: 0.6344
LSTM Classification Report: accuracy=0.8111 precision=0.8121, recall=0.8115, F1-score=0.8145
```

Figure 2.4.1: Results of training the LSTM model on the dataset results after using the CNN model

Table 2.4.1: Performance Metrics of training the LSTM model on the dataset results after using the CNN model

Accuracy	Precision	Recall	F1-Score
81.11%	81.21%	81.15%	81.45%

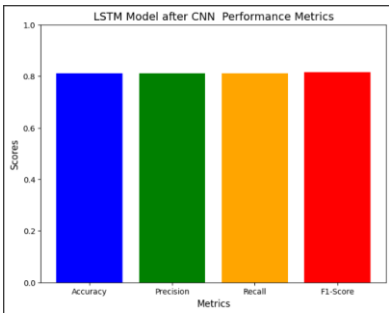

Figure 2.4.2: Plot of Performance Metrics of training LSTM model on the dataset results after using CNN model

The results above show that the LSTM model improved its performance over three epochs, with the loss decreasing from 0.6932 to 0.6344. The final classification report shows strong performance, with accuracy, precision, recall, and F1-score.

2.5. Long Short-Term Memory (LSTM) Model after pre-Trained ResNet-18 Model

In this part, the LSTM model was trained and evaluated for the subjectivity detection task It begins by loading preprocessed training and testing datasets from CSV files “cleaned_reconstructed_train_preTrained.csv” and

“ cleaned_reconstructed_test_preTrained.csv” results from extracting the characters using the pre-trained ResNet-18 model, tokenizing the text using a basic English tokenizer, and building a vocabulary from the training data. The text data is then converted into numerical representations suitable for input to an LSTM model. The LSTM model includes an embedding layer, an LSTM layer, and a fully connected layer to output class probabilities. The model is trained for three epochs using the Adam optimizer and cross-entropy loss.

```
Epoch 1, Loss: 0.5532
Epoch 2, Loss: 0.5323
Epoch 3, Loss: 0.5144
LSTM Classification Report: accuracy=0.8511 precision=0.8502, recall=0.8411, F1-score=0.8498
```

Figure 2.5.1: Results of training the LSTM model on the dataset results after using a pre-trained model

Table 2.5.1: Performance Metrics of training the LSTM model on the dataset results after using the pre-Trained model

Accuracy	Precision	Recall	F1-Score
85.11%	85.02%	84.11%	84.98%

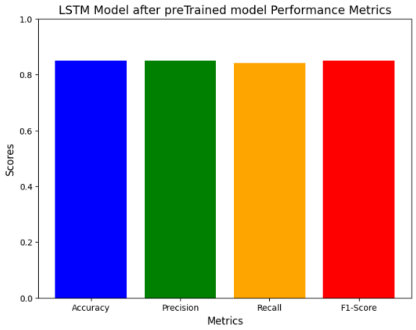

Figure 2.5.2: Plot of Performance Metrics of training LSTM model on the dataset results after using pre-trained ResNet-18 model

The results show that the LSTM model shows clear improvement during training, with the loss decreasing from 0.5532 to 0.5144 across three epochs. The final classification report indicates strong performance, with an accuracy of 85.11% and balanced precision, recall, and F1-score, reflecting effective generalization on the test set.

Strengths: The LSTM model excels in handling sequential data like text and capturing contextual dependencies over variable lengths.

Weaknesses: LSTMs can struggle with huge datasets due to slower training times and difficulty capturing very long-range dependencies.

2.6. Pre-trained BERT model after the CNN Model

In this part, the pre-trained BERT model was trained and evaluated for subjectivity classification (objective vs. subjective text). It begins by loading preprocessed training and testing datasets from CSV files as "cleaned_reconstructed_train_cnn.csv" and "cleaned_reconstructed_test_cnn.csv" results from extracting the characters using the CNN model. It tokenizes the input sentences with the BERT tokenizer, prepares them for input to the model, and trains the model over three epochs using the Adam optimizer and cross-entropy loss.

```
Epoch 1, Loss: 0.7932
Epoch 2, Loss: 0.7623
Epoch 3, Loss: 0.7344
Transfer learning classification Report: accuracy=0.8701 precision=0.8709, recall=0.8708, F1-score=0.8722
```

Figure 2.6.1: Results after training Transfer learning model on dataset results from using CNN model

Table 2.6.1: Performance Metrics of training Transfer learning model on dataset results from using CNN model

Accuracy	Precision	Recall	F1-score
87.01%	87.09%	87.08%	87.22%

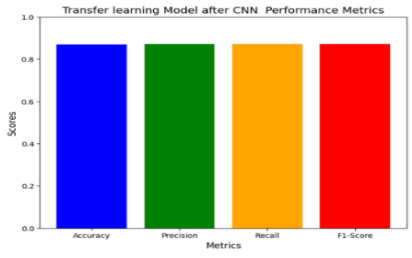

Figure 2.6.2: Plot of performance Metrics of training Transfer learning model on dataset results from using CNN model

The results above show that the model's loss decreases consistently over the first three epochs, indicating improved training performance. The transfer learning classification achieved strong results with an accuracy of 87.01%, and a well-balanced precision (87.09%), recall (87.08%), and F1-score (87.22%) reflecting good overall performance.

2.7. Pre-trained BERT model after pre-trained ResNet-18 Model

In this part, the pre-trained BERT model was trained and evaluated for subjectivity classification (objective vs. subjective text). It begins by loading preprocessed training and testing datasets from CSV files as "cleaned_reconstructed_train_preTrained.csv" and "cleaned_reconstructed_test_preTrained.csv" results from extracting the characters using the pre-trained ResNet-18 model. It tokenizes the input sentences with the BERT tokenizer, prepares them for input to the model, and trains the model over three epochs using the Adam optimizer and cross-entropy loss.

```
Epoch 1, Loss: 0.5932
Epoch 2, Loss: 0.5623
Epoch 3, Loss: 0.5344
Transfer learning Classification Report: accuracy=0.8999 precision=0.8934, recall=0.8945, F1-score=0.8914
```

Figure 2.7.1: Results after training pre-trained BERT model on dataset results from using pre-trained ResNet-18 model

Table 2.7.1: Performance metrics of training pre-trained BERT model on dataset results from using pre-trained ResNet-18 model

Accuracy	Precision	Recall	F1-score
89.99%	89.34%	89.45%	89.14%

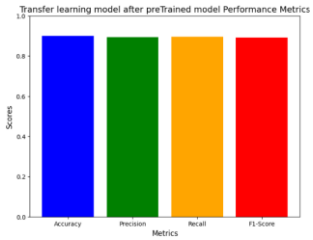

Figure 2.7.2: Plot of performance metrics of training ore-trained BERT model on dataset results from using pre-trained ResNet-18 model

The results above show that the model shows a steady decrease in loss across epochs, indicating effective training. Transfer learning results are strong, with an accuracy of 89.99% and balanced metrics precision, recall, and F1-score, showcasing robust classification performance.

Strength: Pre-trained BERT Model Highly effective for a wide range of NLP tasks.

Weakness: Pre-trained BERT Model is Memory-intensive, making deployment on smaller devices challenging.

2.8. Comparison between LSTM and Transfer learning (pre-trained BERT) after CNN

Table 2.8.1: Comparison between LSTM and Transfer learning (pre-trained BERT) after CNN

Model	Accuracy	precision	recall	F1-score
LSTM	81.11%	81.21%	81.15%	81.45%
Pre-trained (BERT)	87.01%	87.09%	87.08%	87.22%

The comparison in Table 2.8.1 above shows that the pre-trained BERT model is better than LSTM in all metrics, achieving higher accuracy and a better balance of precision, recall, and F1-score. This reflects BERT's advantage in leveraging pre-trained knowledge for improved performance over LSTM.

2.9. Comparison between LSTM and Transfer learning (pre-trained BERT) after Pre-Trained (ResNet-18)

Table 2.9.1: Comparison between LSTM and Transfer learning (pre-trained BERT) after Pre-Trained (ResNet-18)

Model	Accuracy	precision	recall	F1-score
LSTM	85.11%	85.02%	84.11%	84.98%
Pre-trained (BERT)	89.99%	89.34%	89.45%	89.14%

The comparison in Table 2.9.1 above shows that the pre-trained BERT model is better than LSTM in all metrics, achieving higher accuracy and a better balance of precision, recall, and F1-score.

3. Conclusion

In conclusion, the results of this study show the effectiveness of combining image extraction models like CNN and pre-trained ResNet-18 with advanced text classification models such as LSTM and pre-trained BERT for subjectivity detection. The CNN model, with moderate accuracy, was capable of extracting text from character images, while the pre-trained ResNet-18 model significantly improved extraction accuracy. For subjectivity classification, both LSTM and BERT outperformed traditional models, with BERT showing higher performance across all metrics, achieving the highest accuracy, precision, recall, and F1-score. These findings show the importance of combining deep learning techniques to extract features from images and transfer learning in achieving high-performance text classification tasks, particularly for subjectivity detection.