# Module 3: Text Classification with Transfer Learning

Trang Tran

CPS, Northeastern University

EAI 6010 | Applications of Artificial Intelligence

Dr. Dustin Garvey

June 10, 2024

## Introduction

Text classification is a fundamental task in natural language processing (NLP) that finds applications across various domains, including sentiment analysis, spam detection, and topic categorization. In recent years, deep learning has revolutionized the field of NLP, with techniques like transfer learning offering significant improvements in model performance and efficiency. The assignment at hand tasks us with selecting a text classification dataset and applying transfer learning techniques using the AWD_LSTM pre-trained language model in fastai, following the ULMFiT (Universal Language Model Fine-tuning) methodology. Additionally, we are required to develop a more traditional NLP model for comparison.

## Data Selection and Business Applications

I chose the "Tweet sentiment extraction" dataset sourced from Kaggle for its relevance in analyzing tweet sentiments, a critical aspect of modern business strategy. This dataset, originally from Figure Eight's Data for Everyone platform, includes tweets labeled with sentiments (positive, negative, neutral) and supporting phrases. For example: "My ridiculous dog is amazing." [sentiment: positive]. [1]

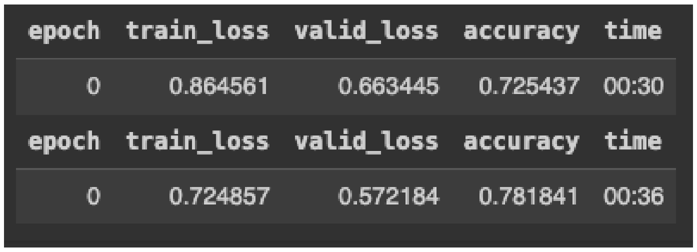
This dataset’s text classification analysis can address several business problems: automating customer service responses, conducting market research to gauge customer satisfaction, managing crises by monitoring public reactions, and measuring the effectiveness of advertising campaigns through sentiment analysis. Using this dataset, businesses can develop models to analyze and respond to their social media sentiments, enhancing strategic decision-making and business intelligence.

## Constraints and Development Efforts

To ensure the feasibility of completing the assignment, we may encounter constraints related to data availability, preprocessing requirements, and computational resources. In this analysis, I needed to preprocess the dataset to handle missing values in the ‘selected_text’ column and convert all values to ‘string’ type; tokenize and normalize text before training the traditional model.

**A comparison of the deep learning model (AMD_LSTM) and the traditional NLP model (Random Forest Classification)**

We achieved a relatively high accuracy score of 78% and a low 'valid_loss' after only a single fine-tuning epoch on the AWD_LSTM model (Figure 1). However, when we added the learning rate and ran a couple of fine-tuning epochs, the accuracy slightly decreased to 76% (Figure 4). Despite this slight decrease, the accuracy of the deep learning model still outperforms that of the traditional NLP model, which achieved 70% accuracy (Figure 6).



```
epoch  train_loss  valid_loss  accuracy  time
    0    0.864561    0.663445  0.725437  00:30
epoch  train_loss  valid_loss  accuracy  time
    0    0.724857    0.572184  0.781841  00:36
```

*Figure 1: A single fine-tuning epoch of AWD_LSTM classifier*

Developing this deep learning model requires less effort and manual tasks, yet it demands more time and computational power compared to the traditional model - the Random Forest classifier. Both models undergo the same data preprocessing steps, including formatting the data into new folders for training. However, the deep learning model involves additional steps such as generating a data block to facilitate dataset normalization and creating a data loader, followed by fine-tuning the language model before building the text classification model.

On the other hand, the RF classification model requires more steps for the train-test convention, tokenizing, normalizing, and TF-IDF training processes.

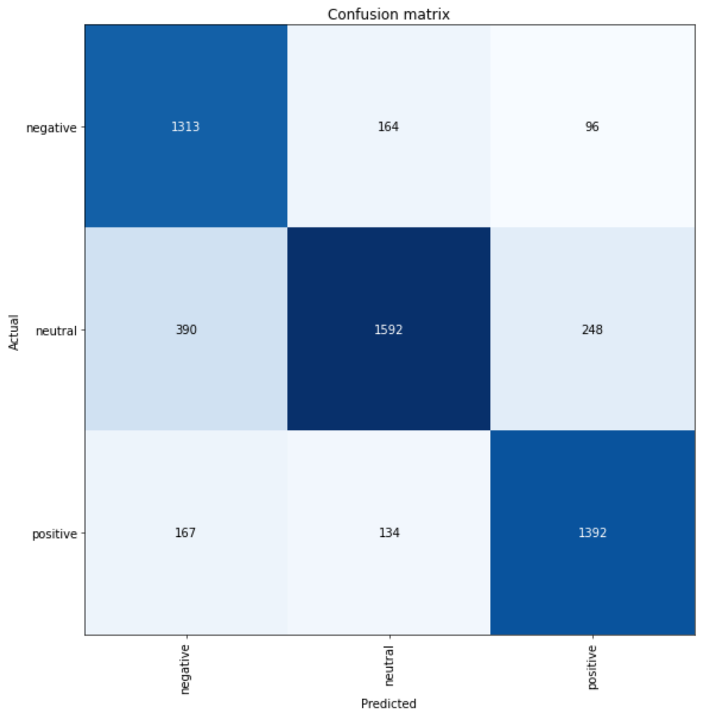


*Figure 2: Confusion matrix after a single epoch*

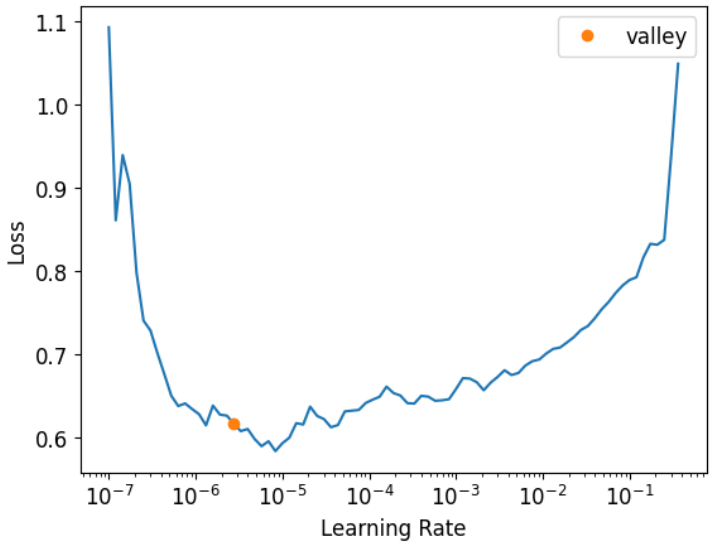
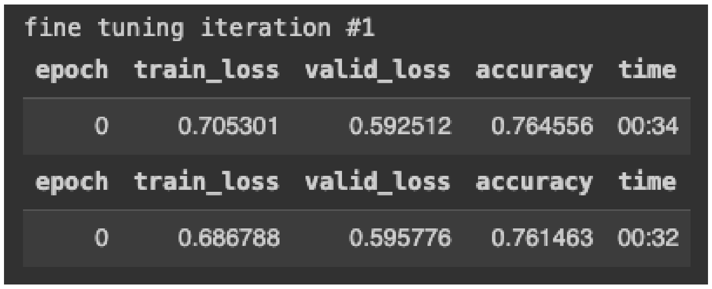


*Figure 3: Finding a good learning rate*



```
fine tuning iteration #1
epoch  train_loss  valid_loss  accuracy  time
    0    0.705301    0.592512  0.764556  00:34
epoch  train_loss  valid_loss  accuracy  time
    0    0.686788    0.595776  0.761463  00:32
```

*Figure 4: A lower result on more fine-tuning epochs adding the learning rate*

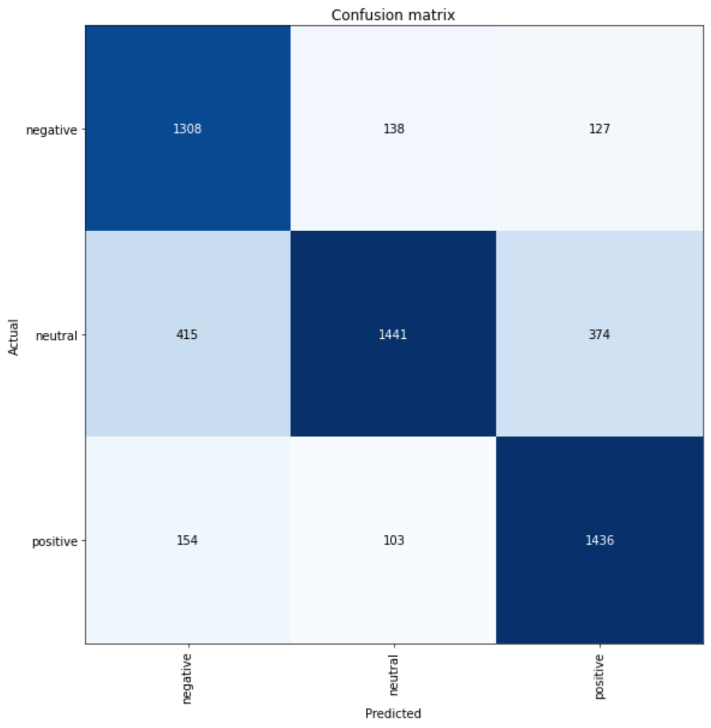
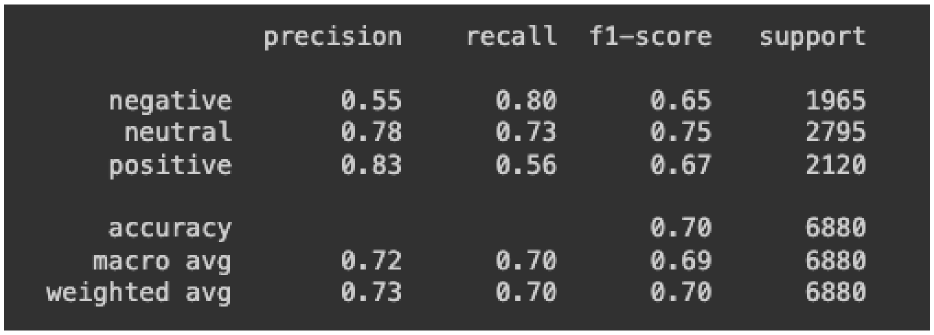


*Figure 5: Confusion matrix after more fine-tuning epochs*



```
              precision    recall  f1-score   support

    negative       0.55      0.80      0.65      1965
     neutral       0.78      0.73      0.75      2795
    positive       0.83      0.56      0.67      2120

    accuracy                           0.70      6880
   macro avg       0.72      0.70      0.69      6880
weighted avg       0.73      0.70      0.70      6880
```

*Figure 6: Test Accuracy of RF Classifier*

If I had to productize this solution, I'd recommend the AWD_LSTM deep learning model for the following reasons:

- Achieved a high accuracy of 78% compared to 70% of the Random Forest classifier.
- More adaptable to varied input conditions and robust with complex data patterns.
- Easy processing and fine-tuning, simplifying deployment and scale-up.

**References**

1. Kaggle. Tweet Sentiment Extraction. [Dataset]. Retrieved June 9, from https://www.kaggle.com/competitions/tweet-sentiment-extraction/overview
2. EAI6010.81433.202435. Module 3: Text Classification with Transfer Learning