Deep Learning HW1 - Fakes Detection

Instructions on how to run code:

For FF-NN and LSTM: run the FFNN_LSTM.py file.
For Transformer model and Blind test results: run the transformer.py file.

All requirements and imports are present within the files. The **blind test results** are in the file 'blind_test_predictions.csv'. The data is formatted in two columns: 'text' containing the biography text, and 'target' containing the prediction made by our best model (The RoBERTa Transformer model). The 'target' column contains labels 'REAL' and 'FAKE'.

1. **FFNN Training**

- **Data Preprocessing**: Basic text processing, such as removing HTML tags, extra new lines, unnecessary punctuations, and wiki-specific section headers.
- **Data loaders**: We have defined a collate function, `collate_batch`, which takes in a batch of data, where each data point is a tuple containing a label and a text. The function applies some processing to the text data, converts it into a tensor, and concatenates the tensors into a single tensor. It also converts the label data into a tensor. Finally, it returns three tensors: the label tensor, the concatenated text tensor, and a tensor of offsets indicating the boundaries between the texts in the concatenated tensor. We load the data using a `DataLoader`, passing in our collate function as the `collate_fn` argument. Total training set size - 17900 samples.
- **Model Architecture**: Our FFNN architecture is a simple two-layer feedforward neural network with an embedding layer (`nn.EmbeddingBag`), a hidden layer (`nn.Linear`), and an output layer (`nn.Linear`). The input to the network is the concatenated text tensor, and the output is a tensor of predicted labels. The `nn.EmbeddingBag` layer computes the mean of the embeddings for each text and the `nn.Linear` layers apply a linear transformation to the mean embeddings.
- **Loss function and Hyperparameters**: We have also defined a loss function, `torch.nn.CrossEntropyLoss()` computes the cross-entropy loss between the predicted and actual labels. We use the Adam optimizer (`torch.optim.Adam()`) to update the network weights. We have also defined a learning rate scheduler (`torch.optim.lr_scheduler.StepLR()`) that reduces the learning rate by a factor of 0.1 after each epoch. Params - `{EPOCHS = 10, LR = 0.01, BATCH_SIZE = 64}; Cross-validation split of 20%`
- **Test set performance - 2105 sample size**
    - *Test Accuracy* - 0.785
    - *Confusion matrix for the test set*

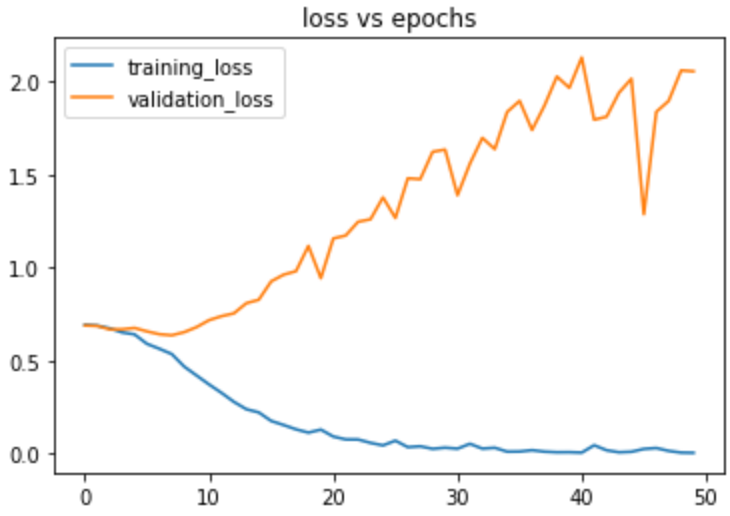
Preds / Truth	1	0
1	890	163
0	289	763

Limitations and Possible solutions: The model lacks enough complexity and clear interpretability (sounds contrasting, I know!). The model is quite simple, and more complex models, such as Transformers, shall outperform these models. On the other hand, because it's a neural net, it's not easily interpretable, which is also a limitation for Transformers. We can use techniques such as LIME or SHAP for interpretation to address these limitations.

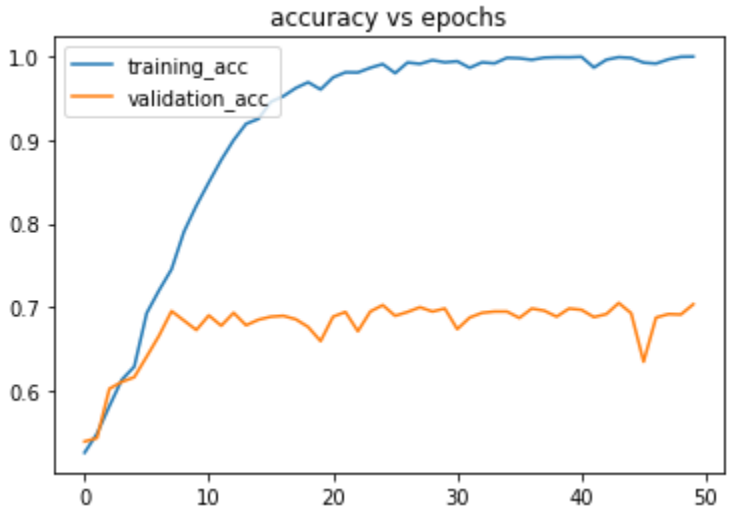
2. **LSTM Model**

- **Data Preprocessing**: Basic text processing, such as removing HTML tags, extra new lines, unnecessary punctuations, and wiki-specific section headers.
- **Data loaders**: After tokenizing the texts, we encode the data by creating a word-to-index and index-to-word dictionary and generating a vocabulary from all 3 files. After encoding each sentence we convert them to Tensors, then use a data loader defined by *torch* for each of the three datasets (train, validation and test). After experimenting, the input sequence length was chosen as 256 - longer sequences were truncated and shorter sequences were padded to this length.
- **Model Architecture**: We used a binary text classification LSTM model. Our LSTM Architecture is as follows:
  Embedding Layer: Embedding(247676, 128, padding_idx=0)
  LSTM layer: LSTM(128, 64, batch_first=True)
  Dropout Layer: Dropout(p=0.2, inplace=False)
  Linear layer for output: Linear(in_features=64, out_features=2, bias=True))

- **Loss function and Hyperparameters**:
  - *Loss function*: Cross Entropy Loss
  - *Optimizer*: Adam
  - *Epochs*: 50
  - *Dropout Rate*: 0.2
  - *Learning Rate*: 3e - 4 (0.0003)
  - *Training time:* ~10 minutes
- **Learning Curves:**
  - *Loss curve***:**

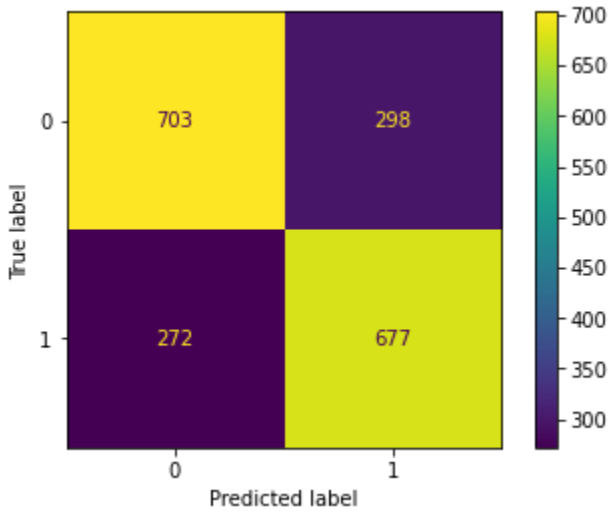


- *Accuracy curve:*



- **Test set performance - 2105 sample size**
  - *Test accuracy* : 0.7041025641025639

- *Confusion matrix for the test set*



- **Limitations and Possible solutions**: By considering the full sequence in some cases, the model seems to underperform, but when the max sequence length was set to the length of the longest sequence in the set, the model did not learn at all and performed extremely poorly. We could experiment with a larger and more complex LSTM model to improve the accuracy. Also, surprisingly a single directional LSTM performed better than a bidirectional LSTM layer.

3. Transformer Model (*RoBERTa-base*) + Blind Set Results

- **Data Preprocessing**: Basic text processing, such as removing HTML tags, extra new lines, unnecessary punctuations, and wiki-specific section headers. Used PyTorch Dataset to be able to load the data quickly as well as to apply some preprocessing steps. The tokenizer is then tokenizing our cleaned up character sequence, preparing it to be fed into a BERT based model (*RoBERTa-base*)

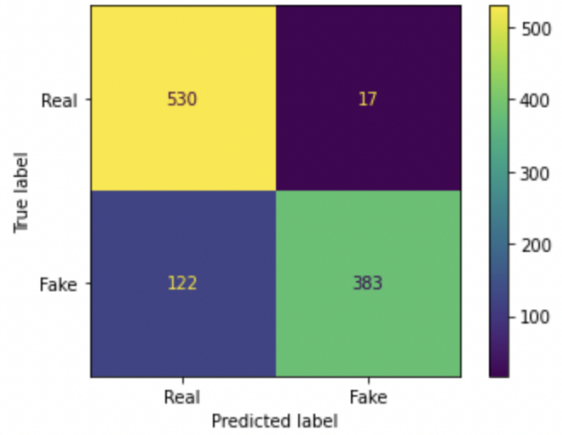
- **Model Architecture**: The RoBERTa-base architecture is a transformer-based neural network model that has 12 layers of transformer blocks, each with 768 hidden units and 12 attention heads.This model takes input as a sequence of tokens and produces a sequence of hidden states, where each hidden state corresponds to one token in the input sequence and the output is predicted label. We use binary cross-entropy loss (BCELoss) as the loss function and the Adam optimizer for updating the model parameters.

- **Why this model was chosen:**

  We used our best performing model , the transformer model on the **blind set**. Apart from the large amount of biographical data it is already pre-trained on, since the transformer model also has the attention mechanism, it seems to capture the ‘fakeness’ of the biographies better than both the other neural networks.

[Results] This model gave us the best accuracy on validation dataset (~0.88), and the test dataset (0.867) out of all the three models.

```
[Transformer Model] Accuracy score on mix test : 0.8678707224334601
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fb39c0649d0>
```



- **Limitations and Possible solutions**:
  Some limitations include,

  1) RoBERTa-base is a transformer model, it requires a large amount of training data to learn complex representations of language. If the training data for the specific classification task is limited, the model may not generalize well to new data. A possible solution is to use transfer learning and fine-tune the RoBERTa-base model on a related task that has more training data.

  2) Inability to detect sophisticated fake content: The RoBERTa-base model may struggle to detect sophisticated fake content that is designed to be highly convincing.Solution: One solution is to use an ensemble of models that incorporate different types of features or representations, such as visual features, social network analysis, and metadata. Additionally, providing additional contextual information, such as the source of the content or the timing of its publication, may also help the model to better understand the context and detect sophisticated fake contents.