Assignment 4
Report

Hariharan Kalimuthu
2020115015

Link to Models:
https://drive.google.com/drive/folders/11D3suravTXbqtr4xyCbro-yyk55k13yD?usp=sharing

Datasets

1. Sentiment Analysis:
   Stanford Sentiment Treebank (https://huggingface.co/datasets/sst)
   Train, validation, test split in the dataset itself
2. Natural Language Inference:
   Multi-Genre NLI Corpus (https://huggingface.co/datasets/multi_nli)
   Train set - first 80k lines, validation mismatched & matched picked from datasets, Test set created using these sets

Some More details:

The Stanford Sentiment Treebank is a dataset for sentiment analysis, which provides fine-grained sentiment analysis of movie reviews. The dataset consists of over 11,000 sentences extracted from movie reviews and human annotations of the sentiment of each sentence.

The Multi-Genre NLI Corpus is a dataset for natural language inference (NLI) tasks. It consists of over 400,000 sentence pairs, where each pair consists of a premise and a hypothesis. The dataset is extracted from a wide range of sources, including fiction, non-fiction, telephone conversations, and government reports. Each sentence pair is annotated with a label indicating whether the hypothesis is entailed, contradicted, or neutral with respect to the premise. This dataset is useful for evaluating the ability of models to understand the relationships between sentences.

Preprocessing

- ▼ Stanford Sentiment Treebank
  - Tokenized
  - stopword removal
  - punctuations and non-alphanumeric characters handled
- ▼ Natural Language Inference
  - Hypothesis and Premise were joined together
  - Tokenized
  - Stopword removal
  - punctuations and non-alphanumeric chartacters

Using batch size = 32, data loaders were created after padding.

After this GLoVE embeddings with 6B parameters and 100 dimensions were used to get static weight matrix on the vocabulary set. This was used as the initial embeddings for the model.

PreTraining on Word Prediction

The ELMo Architecture

Model for SST:

ELMo(
(embedding): Embedding(5264, 100)
(lstm_up): LSTM(100, 100, num_layers=2, batch_first=True, dropout=0.2)
(lstm_down): LSTM(100, 100, num_layers=2, batch_first=True, dropout=0.2)
(dropout): Dropout(p=0.2, inplace=False)

(fc): Linear(in_features=100, out_features=5264, bias=True)
)

Model for Multi_NLI:

ELMo(
(embedding): Embedding(29116, 100)
(lstm_up): LSTM(100, 100, num_layers=2, batch_first=True, dropout=0.2)
(lstm_down): LSTM(100, 100, num_layers=2, batch_first=True, dropout=0.2)
(dropout): Dropout(p=0.2, inplace=False)
(fc): Linear(in_features=100, out_features=29116, bias=True)
)

The code defines a PyTorch implementation of the ELMo model for natural language processing. The architecture consists of an embedding layer, Bi-stacked LSTM layers, a dropout layer, and a fully connected layer. The embedding layer is initialized with pre-trained word vectors. The LSTM layers use the input from the embedding layer to create a sequence representation, and the fully connected layer produces the output of the model. The model is trained using the cross-entropy loss function and the Adam optimizer.

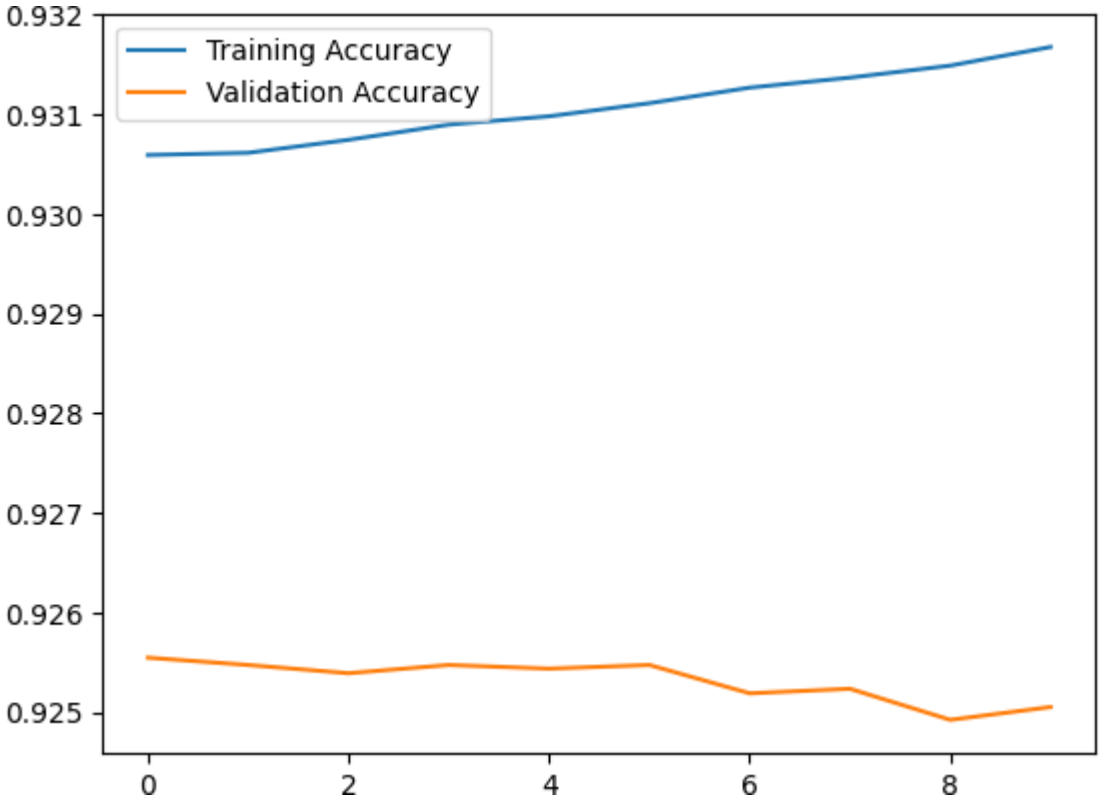
Scores in Pre-Training :

Hyperparameters:

1. epochs = 10
2. dropout = 0.2
3. hidden dim = 100
4. embedding dim = 100
5. learning rate = 0.001

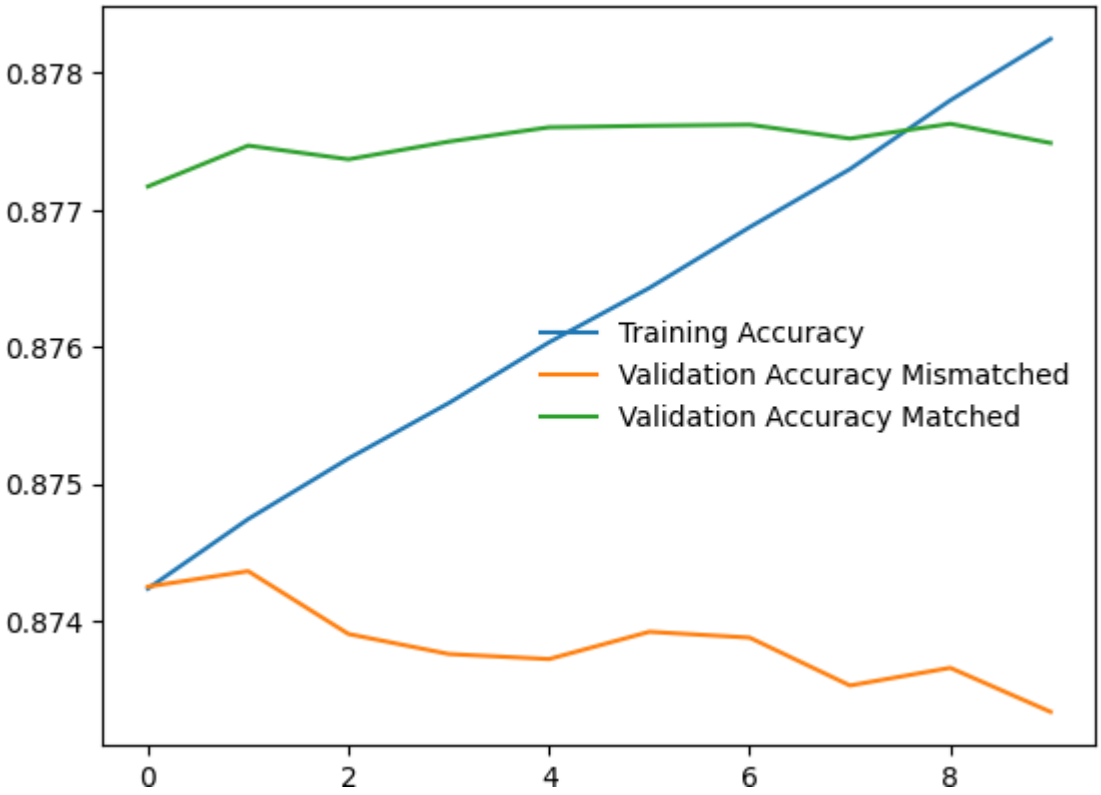
For SST:

Accuracy:



For NLI:

Accuracy:



Training For Downstream Task: Classification

This task was pretty much similar for both datasets. The labels of the SST dataset were converted for binary classification by assuming scores < 0.5 as negative and ≥0.5 as positive. The Multi_NLI dataset had 3 labels: 0,1,2.

Architecture Used

The model takes as input vocabulary size, elmo_embeddings (pre-trained word embeddings), embedding dimension, hidden size, number of layers, dropout rate, and number of output classes.

The model architecture consists of an embedding layer, two LSTM layers (lstm1 and lstm2), a dropout layer, and two fully connected layers (fc1 and fc2). The embedding layer takes in the input text and maps each word to its corresponding elmo embedding. The two LSTM layers process the input sequence in both forward and backward directions, with the output of each layer being fed as input to the next. The output of the last LSTM layer is then combined with the output of the embedding layer and fed through a weighted sum layer to obtain a single embedding vector.

The max-pooling layer is then applied to the resulting embedding vector to select the most important features. The output of the max-pooling layer is then passed through

a dropout layer to reduce overfitting, and then fed through two fully connected layers to obtain the final output, which is a tensor of size (batch_size, num_classes).

The loss function used is Cross Entropy Loss, and the Adam optimizer is used to optimize the model parameters during training.

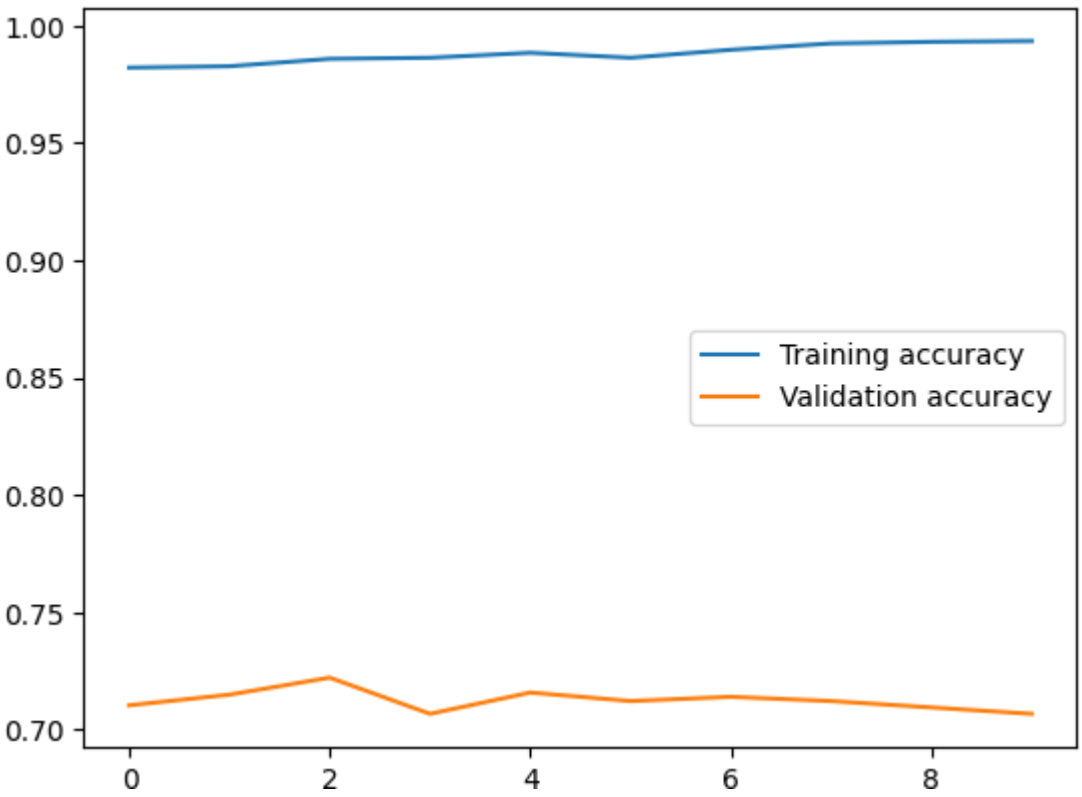
Scores on Train and validation Sets

Hyperparameters:

1. epochs = 10
2. dropout = 0.2
3. hidden dim = 100
4. embedding dim = 100
5. learning rate = 0.001

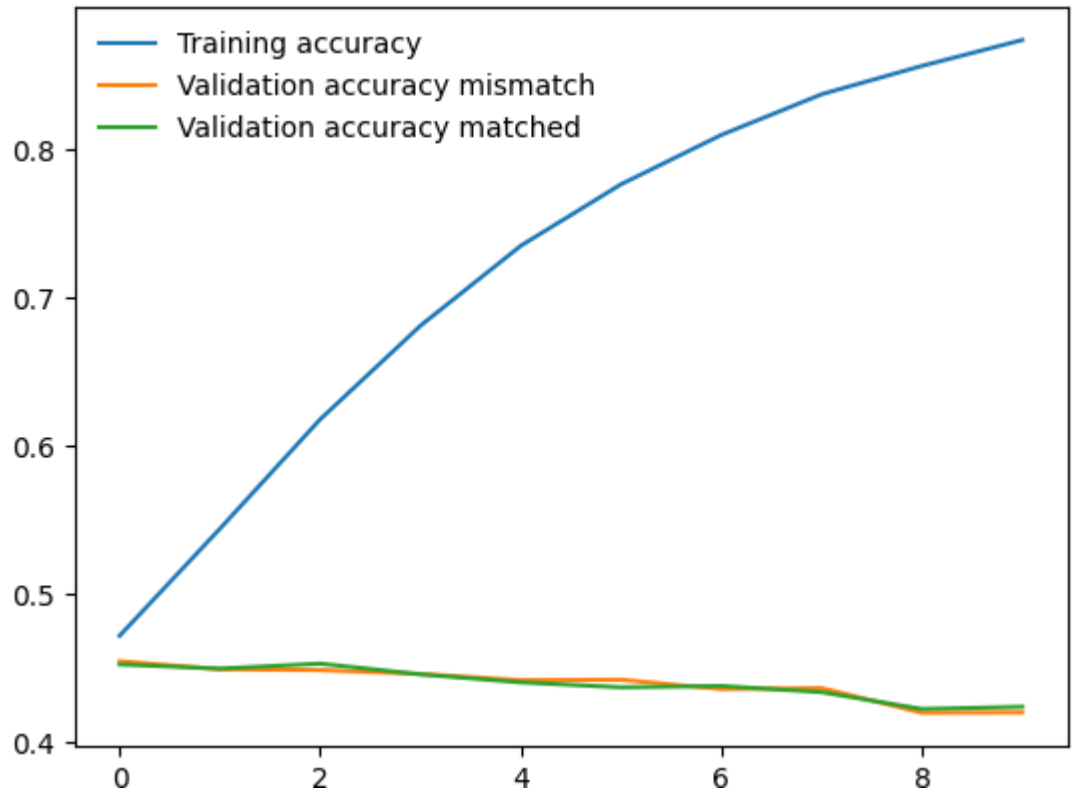
For SST:

Accuracy:



For NLI:

Accuracy:



Evaluation Scores

1. Test Set Accuracy for SST : 70%
2. Test Set Accuracy for NLI: 93% (using train test split) otherwise 47% on the validation sets.

Hyperparameter Analysis (NLI & SST both)

- Increasing epochs to 20 does lead to better results but is computationally heavy for the NLI dataset.
- The validation set mismatched wasn't taken as the test set since we don't want the results to be purely on a different kind of dataset. Moreover, it was used to check overfitting.
- Increasing epochs to more than 20 leads to overfitting in the SST dataset.

- Increasing the dropout rate to 0.5 led to a decrease in accuracy in validation sets during Pre-training and Training both.
- Taking the hidden dimension to be 200 gives better results than 100 on both datasets.
- Removing stopwords has led to better accuracy
- Increasing Max sentence length from 100 to 120 gave 2% more accuracy in the NLI dataset.