Interim Report – Natural Language Inference
Team 25 – YeePM

Timeline

Date	Objectives
15th February	- Researching on, and finalizing the scope of the project - Project outline submission
15th February - 25th February	- Read up and explore the different concepts and implementations of NLI Models. - Look into the different methods of preprocessing and generation of word embeddings, such as GloVe, Word2vec, and ELMO - Explore the suggested datasets, their syntax, and previous optimal NLI implementations on them
25th February – 7th March	- Understand the fundamental concepts involved in Bi-LSTMs and MLPs, as well as the alternate attention model suggested. - Read up on optimized BERT implementations, their working, and implementation, such as RoBERTa.
7th March – 15th March	- Create a baseline implementation of an RNN-based model and run it over the 3 datasets. - Follow the implementation of the MLP attention model as an alternate baseline, and generate scores for the 3 datasets - Interim report submission

16th March – 31st March	- Expand on the baseline implementation and create a resolute Bi-LSTM model with scores better than the alternate baseline. - Perform hyper-parameter optimization on the LSTM implementation, and fine-tune results for the 3 datasets
1st April – 14th April	- Explore the RoBERTa implementation for entailment, and benchmark its performance against that of the LSTMs
14th April – 21st April	- Verify and document the obtained results. - Final optimizations, code refactors, and documentation.

Work Done

For our baseline, we have followed two different tracks of implementation:

- The first method is the baseline version of the Bi-LSTM approach, as described in the project outline, which is effectively a summation model. This method employs GloVe vector embeddings and uses TensorFlow to implement the neural networks.

- The second, alternative method that we have implemented is the 'Decomposable Attention Model', which is based on multi-layered perceptrons, and an implementation that differs from LSTMs. It has been implemented using PyTorch.

Implementation Specifics:

a) Attention Model

The baseline for both implementations used the SNLI dataset while developing. The train and test files of the SNLI Dataset were read, and the premises, hypotheses and labels were loaded. PyTorch's DataSet class was used for further organization. Each of the sentences' length was trimmed to an appropriate hyperparameter, and shorter sentences were padded with a reserved token. For the vocabulary, a library called 'd2l' was used. Finally, PyTorch's DataLoader class is used for the efficient

shuffling and batch processing of the train data and fetching the samples from the train set during the training process.

The architecture of this model consists of the usage of a Multi-Layer Perceptron and an Attention model. The MLP consists of the following components twice in a Sequential object:

- A Dropout layer with probability of 0.2, which helps in the regularization.
- A Linear layer to which the input size and output (here, the hidden layer) size is provided.
- A ReLU activation function.

For a particular pair of premise and hypothesis, the next step after preprocessing is to retrieve embeddings out of them. For this, the GloVe 6 billion token 100 dimensional vectors are used in d2l to get the token embedding data. This embedding weight data is passed onto to the model.

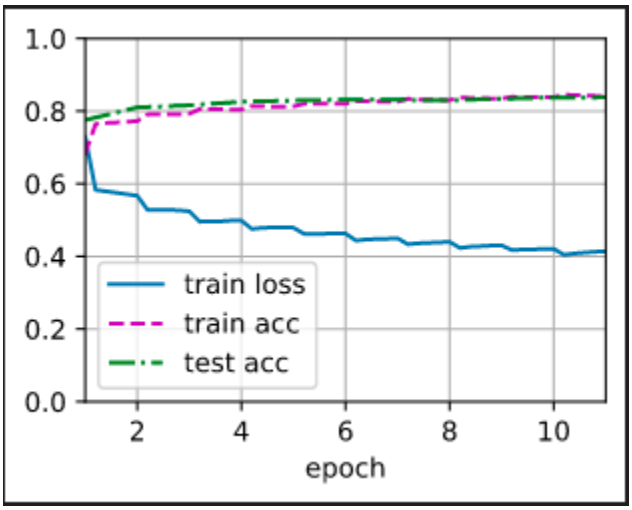
The next part of the model is the Attention itself, and as per the paper, there are three stages for it: Attend, Compare, Aggregate.

- **Attend:** The two embedding vectors are aligned such that tokens with similarity and potential relationship are assigned to each other. This is a soft alignment, with every token of sentence A given a weight for every token in B. Large weights are given to the most useful pairs. An MLP f is defined, and for each pair the weight is computed as the dot product of $f(A_i)$ and $f(B_i)$. When all of them are computed, the softmax of all these (where each weight is raised as power of exponential and averaged) values are computed for both sentences to obtain beta and alpha, which are the return values of this stage.
- **Compare:** This step compares each token from one sequence to every token in another sequence. A concatenation operator is applied to sentence A tokens and the tokens from sentence B which got aligned with it, and vice versa. This set of new vectors is passed onto another MLP g, identical to f. This stage returns two final sets of vectors called comparison vectors, one of length m and the other of length n which denote the number of tokens in each of the sentence.
- **Aggregate:** First, the two comparison vectors are summed up to get two final vectors V_a and V_b. These both are then concatenated and sent to yet another MLP h, whose output is in turn fed into a Linear object to get the prediction for that particular sample.

Results:

The following results are some of the runs (with the batch size hyperparameter always set as 256):

Dataset	Learning Rate	Epochs	Test Accuracy
SNLI	0.001	4	81.5%
SNLI	0.001	11	83.9%
SNLI	0.01	11	61.5%
MultiNLI (Matched)	0.001	4	65.4%
MultiNLI (Mismatched)	0.001	4	66.7%
SICK	0.005	10	62.5%
SICK	0.005	20	66.4%
SICK	0.005	30	67.7%



SNLI – Learning Rate: 0.001, Epochs: 11

b) Summation RNN Model

Our second implementation within our work done is that of a model based on **Recurrent Neural Networks** and GloVe vector embeddings.

All input vectors are initially converted into GloVe embeddings and then normalized. The embeddings of the premise hypothesis pairs are concatenated and then are passed through three layers of tanh along with some L2 regularization. The RNN is then run.

Results:

Dataset	Learning Rate	Epochs	Test Accuracy
SNLI	0.001	10	66.4%
SNLI	0.001	20	70.5%
MultiNLI (Matched)	0.001	10	60.1%
MultiNLI (Matched)	0.001	20	65.3%
SICK	0.005	10	58.5%
SICK	0.005	20	60.9%

Limitations & Upcoming Work

While our attention model is giving sufficient results for the datasets, with a small amount of hyperparameter optimization, the summation RNN model isn't nearly as optimal. Thus, long short-term memory should be incorporated in this process so as to improve the score and increase the context sustained by the model.

As explained in the timeline, future work on the project will be centered towards developing an optimal bi-directional LSTM model that is able to outperform both our current baselines, and tune its hyperparameters for the three datasets separately.

In addition to this, the RoBERTa model, which has the highest score on the SNLI dataset as of now, will also be worked upon as a parallel to the LSTM model, and the accuracy, precision, and recall for the two cases will be benchmarked and contrasted appropriately.