MATH179 Midterm Project: Natural Language Inference with Enhanced Sequential Inference Model

Rui Zhang
Harvey Mudd College

March 24, 2024

Abstract

This report delves into the exploration of Enhanced Sequential Inference Model on Natural Language Inference using Stanford Natural Language Inference (SNLI) dataset. Enhanced Sequential Inference Model is one of the best models for Natural Language Inference based on Long short-term memory (LSTM) network. For this midterm project, I focus on learning how to use the current implementation with PyTorch by Coet [3] to train the model using SNLI dataset. The next step would be explore how to train the model with another dataset for Natural Language Inference.

1 Introduction

1.1 LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture. LSTM networks are designed to overcome the vanishing gradient problem that occurs when training traditional RNNs on long sequences of data. They are particularly effective in capturing long-term dependencies in sequential data, making them well-suited for tasks such as natural language processing.

Traditional RNNs have a simple structure where each neuron processes input data and passes its output to the next time step in the sequence. However, they struggle with learning long-term dependencies due to the vanishing gradient problem, where gradients diminish as they are backpropagated through time. LSTM networks introduce the memory cell, which retains information over long sequences and selectively updates or forgets information based on the input data.

LSTMs incorporate three types of gates taking the previous hidden state h_{t-1} and the current input x_t to control the flow of information into and out of the memory cell: Forget Gate f_t determines which information to discard from the cell state, outputting a value between 0 and 1. Input Gate i_t decides which new information to store in the cell state, outputting a value between 0 and 1. Output Gate o_t controls which information from the cell state should be exposed as the output.

The cell state is updated using the forget gate, input gate, and output gate. These gates regulate how information flows into and out of the memory cell, allowing LSTMs to retain important information over long sequences while discarding irrelevant information.

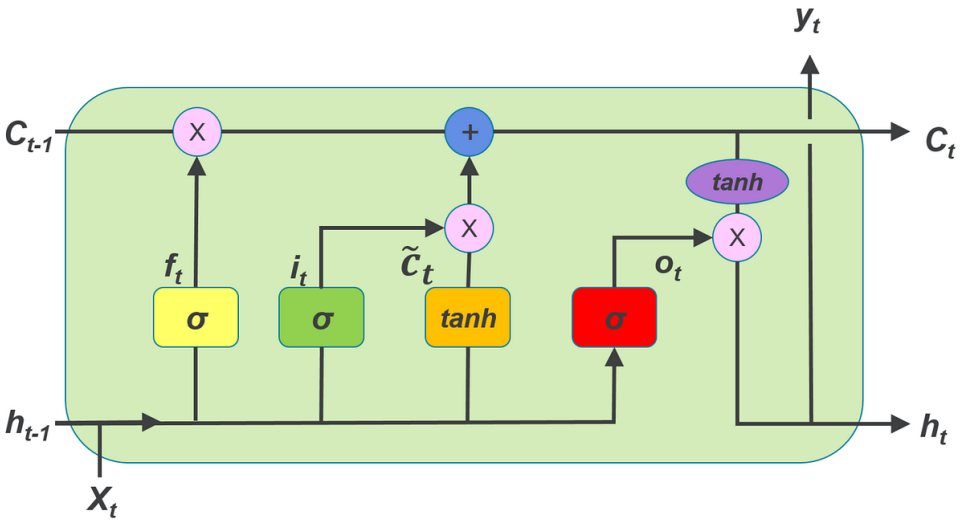

Figure 1: A block for LSTM, source: https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af by Divyanshu Thakur

LSTMs also produce a hidden state h_t at each time step, which is a function of the current input, previous hidden state, and current memory cell state. The hidden state contains information that the LSTM network has deemed relevant for predicting the output at the current time step.

Bidirectional LSTMs (BiLSTM) are an extension of standard LSTMs that process input sequences in both forward and backward directions. In a BiLSTM architecture, there are typically two separate LSTM layers: one layer processes the input sequence in the forward direction and the other layer processes the input sequence in the backward direction. The outputs of these two layers are then typically concatenated or combined in some way before being passed to subsequent layers or output units. BiLSTMs are particularly useful in tasks where understanding the entire context of a sequence is important, such as natural language processing tasks like machine translation.

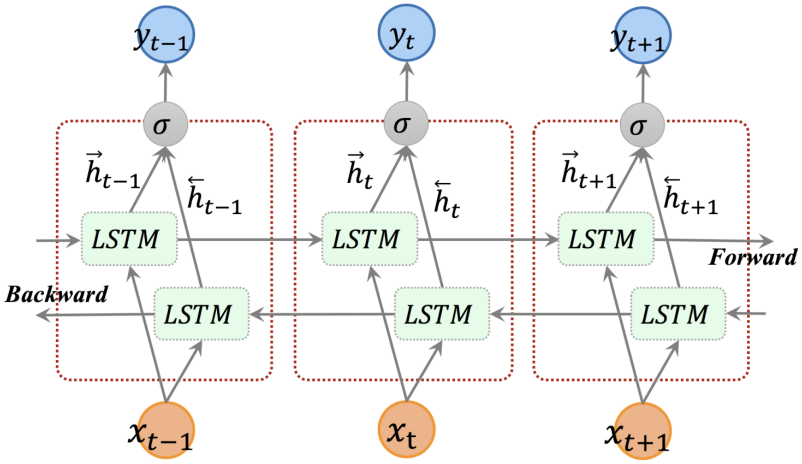

Figure 2: BiLSTM schematic diagram, source: Cui et al., 2018 [4]

1.2 ESIM

The Enhanced Sequential Inference Model (ESIM) is an enhanced LSTM model introduced by Chen et al. in Enhanced LSTM for Natural Language Inference [2]. The ESIM model operates by sequentially processing pairs of input sentences, typically referred to as the premise and hypothesis. It aims to determine the logical relationship between these two sentences, classifying them into one of three categories: entailment (the hypothesis logically follows from the premise), contradiction (the hypothesis contradicts the premise), or neutral (there is no logical relationship between the two sentences).

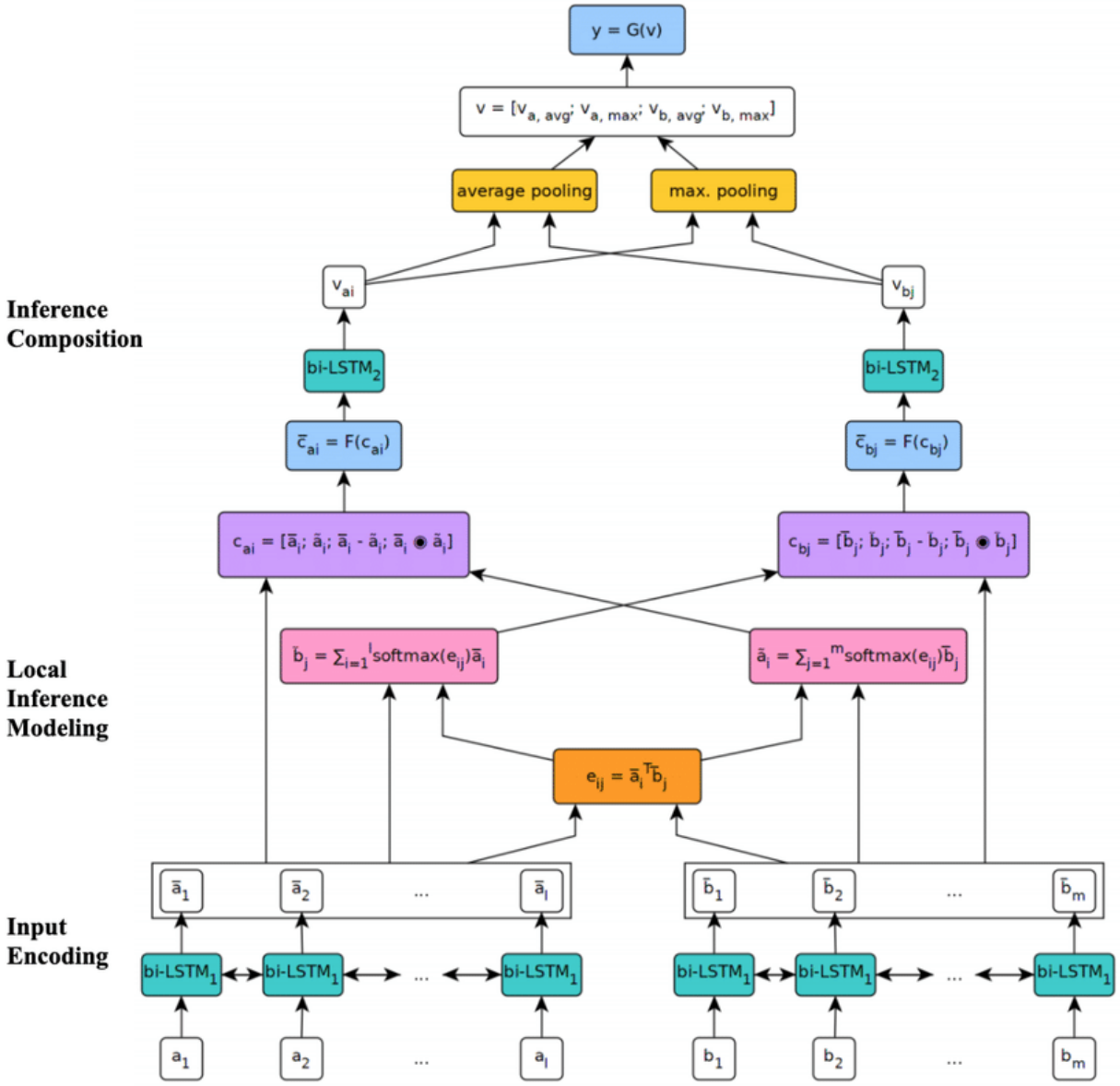

Figure 3: ESIM schematic diagram, source: Farhad Nooralahzadeh, 2020 [5]

Initially, the input sentences are tokenized and embedded into dense vector representations, typically using techniques like word embeddings or contextual embeddings. The embedded representations of the premise and hypothesis sentences are then fed into bidirectional LSTM (BiLSTM) layers so the model can learn to represent a word and its context. Then, a soft alignment layer is used to compute the attention weights to obtain

the local relevance between a premise and hypothesis. For the hidden state of a word in a premise, the relevant semantics in the hypothesis is identified and composed using the attention weight. Then, to further sharpen local inference information between elements and capture inference relationships, the difference and the element-wise product for the tuples of BiLSTM result and weighted summation for relevant semantics are computed and concatenated with the original vectors. At last a composition layer is applied to determine the overall inference relationship between a premise and hypothesis. In this layer, BiLSTM is used to capture local inference information and context for local inference results. Then, both average and max polling are used and concatenated to form the final fixed length vector. The vector is then put into a final multuplayer perceptron classifer, which has a hidden layer with tanh activation and softmax output layer. The model is trained using multi-class cross entropy loss.

ESIM effectively model the complex interactions and dependencies within the input sentences, leading to improved performance on tasks like natural language inference. It demonstrates strong performance on benchmark datasets like the Stanford Natural Language Inference (SNLI) dataset and the MultiNLI dataset. Its modular architecture and reliance on standard neural network components make it widely applicable and adaptable to various NLP tasks.

2 Data

2.1 SNLI Dataset

The Stanford Natural Language Inference (SNLI) dataset, published by Bowman, et al. in 2015 [1], has been widely used as a benchmark dataset in natural language processing. SNLI consists of 570k pairs of "premise" and "hypothesis" sentences with labeling "entailment" (the hypothesis logically follows from the premise), "contradiction" (the hypothesis contradicts the premise), or "neutral" (there is no logical relationship between the two sentences). SNLI includes sentences with varying degrees of complexity, covering a wide range of linguistic phenomena such as lexical semantics, syntax, and pragmatics. This diversity makes it a challenging benchmark for evaluating the performance of natural language processing models.

2.2 Data Preparation

The SNLI dataset provide train, dev, and test set. For preprocess the data for training, I compute worddict, a dictionary associating words to unique integer indices for some dataset. Then, using the worddict I transform the words in the premises and hypotheses of a dataset, as well as their associated labels, to integer indices. Then, I build an embedding matrix from pretrained word vectors.

3 Experiment and Results

In this part of the project, I followed Coet's instruction on his github repository for ESIM model and trained the model with the SNLI dataset. I trained the model for 16 epochs because the validation loss had stopped decreasing. The loss and accuracy are plotted in the figures below. The accuracy after 16 epochs are training accuracy 89.7%, validation

accuracy 88.2%, and test accuracy 87.7%, which has a small difference from Coet training result of training accuracy 93.2%, validation accuracy 88.4%, and test accuracy 88.0% [3].

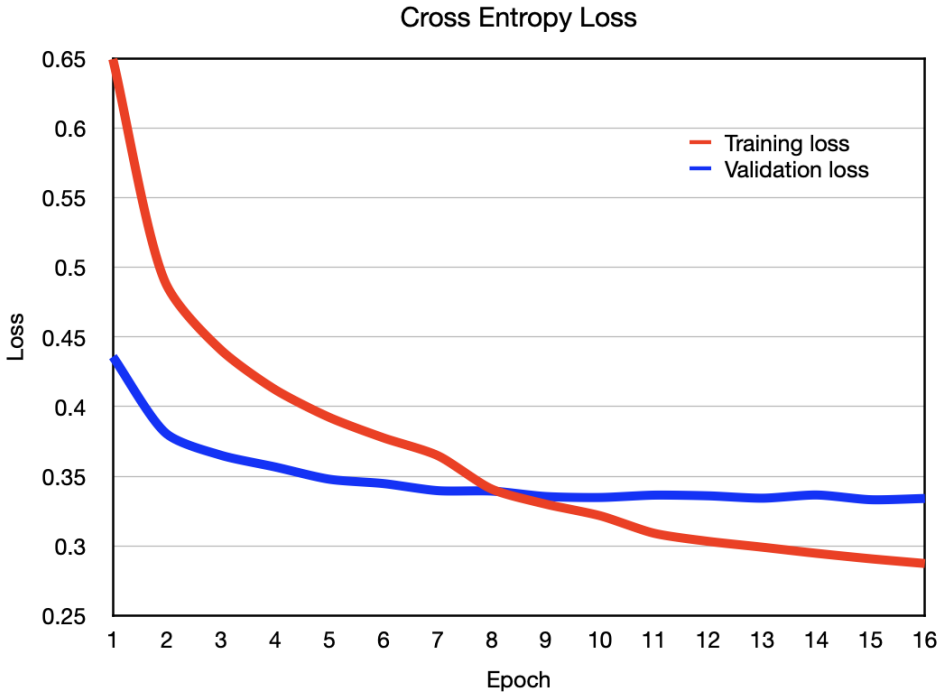

Figure 4: The loss rate for training and validation sets

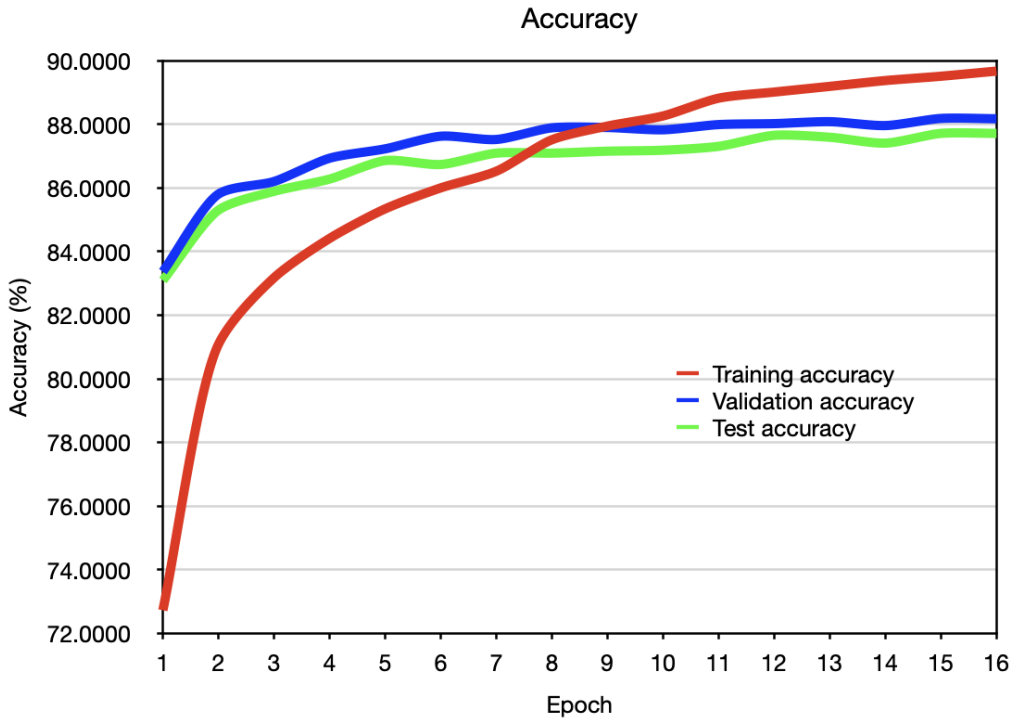

Figure 5: The accuracy for training, validation, and test sets

4 Next Steps

For all the parts above, I am learning the how to train ESIM using SNLI dataset as an example with Coet's implementation. For the following part of the project, I plan to learn how to train ESIM using another dataset, such as FarsTail, a Persian natural language inference dataset, from preprocessing to testing.

References

[1] Angeli G. Potts C.- Manning C.D. Bowman, S.R. A large annotated corpus for learning natural language inference. *Conference on Empirical Methods in Natural Language Processing*, 2015.

[2] Zhu X. Ling Z.-H. Wei S. Jiang H. Inkpen D. Chen, Q. Enhanced lstm for natural language inference. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

[3] A. Coet. Esim. `https://github.com/coetaur0/ESIM`, 2019.

[4] Ke R. Wang-Y. Cui, Z. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. *nternational Workshop on Urban Computing (UrbComp)*, 2018.

[5] Nooralahzadeh F. Low-resource adaptation of neural nlp models. 2020.