Sequential LSTM-based Encoder for NLI

Ankita Sharma
ICME
Stanford University
Stanford, CA 94305
ankita89@stanford.edu

Yokila Arora
ICME
Stanford University
Stanford, CA 94305
yarora@stanford.edu

Abstract

The ability to perform inference from natural language sentences is a central problem in Natural Language Understanding. For many downstream applications it is important to know that from *some birds fly* we can infer that *some animals fly*. Deep neural networks are achieving extraordinary results in this task with the large Stanford Natural Language Inference (SNLI) corpus. In this project under guidance of **Ignacio Cases**, we investigate ways to improve sequential architectures with attention, which are yielding very strong results on SNLI despite their relative simplicity when compared with more complex models.

1 Introduction

Natural Language Inference is a task to identify relationship of entailment or contradiction between premise and hypothesis. Capturing this reasoning and inference is quite challenging but fundamental to language understanding. The recent release of a large corpus (SNLI [1]) with about 570k sentence pairs labeled for inference has enabled neural network training to yield good results. Some examples from the corpus include:

Entailment
P: *A man, woman, and child get their picture taken in front of the mountains.*
H: *A family on vacation is posing.*

Contradiction
P: *People are throwing tomatoes at each other.*
H: *The men are covered in tomatoes.*

Neutral
P: *Two men are cooking food together on the corner of the street.*
H: *The two men are cooking food for the block party.*

Availability of such large-scale annotated data to estimate parameters made complicated neural network models (e.g. Munkhdalai and Yu 2016) achieve state-of-the-art performance. The goal was to have deep networks-consisting of LSTM based encoding, recursive networks and combination of attention models- for better sentence comprehension. Wang, and Jiang [6], have explored using a matching-LSTM that performs word-by-word matching of the hypothesis with the premise rather than deriving sentence embeddings for the premise and the hypothesis to be used for classification which performed well on SNLI benchmark. Recent work by Parikh et al.[3] and Chen et al.[2] used

basic sequential models (Long Short Term Memory (LSTM) based architectures) have shown either close or better performance than the more complicated structures.

2 Related Work

Past research in NLI is mainly focused on complex, deep text representation models. In contrast to these traditional approaches, more recent work by Chen et al. [2] and Parikh et al. [3] explores the capabilities of simpler models with fewer parameters and still gets better performance than previous state-of-the-art work.

Parikh et al.[3] used the intuition that local text substructure and subsequent aggregation of these is sufficient for inference between sentences. Parikh et al. [3] relies on parallelizing computation with decomposing complex sentences and applying soft alignment using neural attention; applying intra-sentence attention; compare aligned sub-phrases and predict the relationship between the premise and hypothesis. Parikh mentions their use of attention is purely based on word embeddings and is independent of word order. They achieved one of the best results. Taking this model as baseline, Chen et al. [2] explore the potential of basic sequential models (eg. Long Short-Term Memory (LSTM)) based architectures.

Chen et al. [2] models hybrid inference network which consists of the following components: soft alignment, subcomponent inference collection, inference composition, and extension to recursive structures. They present several neural network models and claim that the potential of sequential LSTM-based models has not been explored fully. The performance of this model has outperformed complicated network architectures, thus affirming the claim that potential of such basic models is not fully explored. Our aim is to re-implement this methodology with good performance and explore enhancements.

3 Approach

We have implemented Bowman et al.[1] model as our baseline from scratch in Tensorflow. Upon exploring the three implementations-sum of words,RNN, LSTM- for context encoding, we find that the model is most effective when used with LSTMs (Hochreiter and Schmidhuber, 1997). As a natural extenstion, we implement BiLSTM (Bidirectional LSTM) to capture the context from both prior and later words in the sentence. Then state-of-the-art model by Chen et al. [2] uses this encoding architecture along with alignment which we use to achieve similar results and experiment with different score functions for attention weights calculations.

The dataset is preprocessed by reading in all samples, padding with a symbol to make lengths equal and adding start-of-the sentence indicators for each sentence. Some samples (2%) for which no consensus could be achieved on the gold label are neglected and a smaller vocabulary is built using the unique words in the dataset. 840B token version of GloVe embeddings is used to construct word embedding for the sentence words and characters, thereby giving the initial vectors for each sentence pairs.

3.1 Baselines

This section describes the models used as baseline for comparing our results. As mentioned in Bowman et al. [1], initial sentence embeddings is created using either *sum of words, RNN or LSTM* layer who result is fed to a *tanh* layer so as to map 300 dimensional embeddings into lower-dimensional space (specifically, 100d). The hypothesis and premise projected vectors are concatenated and passed through three additional *tanh* layers to encode semantic relationship in multiple non-linear layers. Finally, the output is fed into a 3-way softmax classifier.The sentence embedding models are as follows:

Sum-of-words: All of the word embeddings in each pair of premise and hypothesis are summed and passed on to subsequent layers. This does not capture intra-sentence information and word wise meaning alignment between sentence, and hence does not perform well.

RNN: In this model, instead of sum-of-embeddings, basic RNN layer is applied on both the premise and hypothesis to generate the initial context vectors but the network needs lot of hyper-parameter

tuning to generate reasonable results (handle diminishing gradients with change of activation functions and hyper-parameter tuning).

LSTM: In this model, an LSTM (Hochreiter and Schmidhuber, 1997) layer is applied on both the premise and hypothesis. This is very stable with clipped gradients and gives the best performance out of the three models.

3.2 BiLSTM

Basic components of model proposed by Chen et al. [2] includes soft alignment, subcomponent inference collection and inference composition (named Enhanced BiLSTM Inference Model **(EBIM)**). The premise and hypothesis sentences are denoted as

$$\text{Premise: } a = (a_1, a_2, ..., a_{l_a})$$
$$\text{Hypothesis: } b = (b_1, b_2, ..., b_{l_b})$$

respectively, where l_a and l_b represent the length of premise and hypothesis. Each $a_i, b_j \in R^l$ is the word embedding of dimension *l*.

Soft Alignment To encode the word token and context around it, bidirectional LSTM (BiLSTM) is used instead of the sum-of-words/RNN/LSTM models used in Bowmna et al.[1]. The intra-sentence attention used by Parikh et al.[3] does not improve the model further.

$$\bar{a_i} = BiLSTM_1(a), \forall i \in [1, 2, .., l_a]$$
$$\bar{b_i} = BiLSTM_1(b), \forall j \in [1, 2, .., l_b]$$

Using BiLSTM was considered crucial for EBIM as it runs forward and backward LSTM along each input sequence and captures both past and future context.

For attention, weights are calculated from the hidden state of BiLSTM, and then they are normalized. The new vectors formed $\tilde{a_i}$ and $\tilde{b_j}$ are called *mimic* vectors.

$$e_{ij} = \bar{a_i}^T \bar{b_j}, \forall i \in [1, 2, .., l_a], \forall j \in [1, 2, .., l_b]$$
$$\tilde{a_i} = \sum_{j=1}^{l_b} \frac{exp(e_{ij})}{\sum_{k=1}^{l_b} exp(e_{ik})} \bar{b_j}, \forall i \in [1, 2, .., l_a]$$
$$\tilde{b_j} = \sum_{i=1}^{l_a} \frac{exp(e_{ij})}{\sum_{k=1}^{l_a} exp(e_{kj})} \bar{a_i}, \forall j \in [1, 2, .., l_b]$$

Subcomponent Inference Collection In this step, heuristic matching between original vectors and mimic vectors is used to improve performance. Specifically, three matching heuristics, namely concatenation of original and mimic vectors (captures context), difference (captures the contradiction part) and element-wise product (captures similarity) is used.

$$m_a = [\bar{a}, \tilde{a}, \bar{a} - \tilde{a}, \bar{a} \cdot \tilde{a}]$$
$$m_b = [\bar{b}, \tilde{b}, \bar{b} - \tilde{b}, \bar{b} \cdot \tilde{b}]$$

Inference Composition Another layer of BiLSTM is used to model interaction between subcomponent inference created above.

$$v_{1,i} = BiLSTM_2(m_a), \forall i \in [1, 2, .., l_a]$$
$$v_{2,j} = BiLSTM_2(m_b), \forall j \in [1, 2, .., l_b]$$

Then average pooling and max pooling results are concatenated to get fixed length vector (in comparison to summation used in Parikh et al. [3]). This vector goes into a multilayer perceptron (MLP) classifier (hidden layer with tanh activation and softmax output).

3.3 BiLinear Form for Attention score

Luong et al.[4] proposed additional score functions for generating the context vector alignment weights of the two sentences in this context. Chen et al.[2] used the simplified dot product score, and we modified the score function to use the *general* modification introduced in [4], also known as bilinear form.

$$e_{ij} = \begin{cases} \bar{a}^T \bar{b} & dot \\ \bar{a}^T W_a \bar{b} & general \\ v_a^T \tanh(W_a[\bar{a}; \bar{b}]) & concat \end{cases}$$

According to the findings in [4] in the context of Neural Machine Translation, the *concat* implementation does not perform well, *dot* works well for global attention and *general* is better for local attention. We explore the performance for *general* score function in comparison to *dot* for NLI.

4 Experiments

We vary learning rate, dropout, l2 regularization, clip-norm, batch-size, optimizer, token size of GloVe emebeddings, dimensions of the word vector and models to generate following insights. We implemented the code in Tensorflow and overcame several challenges for instance, figuring out where to apply fine-tuning techniques like dropout (which layers/ stage) and for the language especially with debugging the saving of model correctly and running the validation calculations on restored models.

4.1 Dataset

We use the Stanford Natural Language Inference (SNLI) corpus [1], which is a large annotated dataset having 570k sentence pairs labeled for entailment, contradiction and semantic independence. Each entry is a pair of hypothesis and baseline with gold-label, and the parse structure of sentences in Penn Treebank format. Some key statistics about this corpus are listed in Table 1. About 2% of the sentence pairs are labeled '-' in the corpus due to lack of consensus. These were not included in either the training or evaluation process. This results in 549,367 training pairs, 9842 development pairs and 9824 test pairs.

Table 1: Some key statistics of SNLI corpus

Data set sizes:	
Training pairs	550,152
Development pairs	10,000
Test pairs	10,000
Sentence length:	
Premise mean token count	14.1
Hypothesis mean token count	8.3

We used 840B token version of GloVe embeddings (Pennington et al. 2014) and 6B token version with 100 and 300 dimensions for each. The former performed better with 300d. There were several typos and spelling errors in dataset, so the larger word set might have helped capture that. The unknown and padded words, and start-of-sentence identifier is used with random initialization and are stored with the model to use for the test accuracy calculations. Samples with no consensus of gold labels are ignored (about 2%). We have not updated the word embeddings while training due to limited time and multiple results to run but that will definitely bring performance closer to the published results.

4.2 Results

For baselines, dropout keep of 0.8, learning rate of 0.001, clipped norm of 5, and Adam optimizer is used to get good accuracy (percentage of correctly predicted labels). More experimentation with hyper-parameters could lead to results very close to those indicated in the paper. For BiLSTM and BiLSTM with Bilinear form score function, we used learning rate of 0.0004, dropout 0.5, clip-norm 5, and Adam optimizer (performs better than Adadelta). Varying batch size also affected the performance (too big batch size was not suitable for dataset of this size), so we used batch-size of 32 to get consistent decrease in loss. L2 regularization provided no particular benefit and was kept to 0. For our scenario, accuracy is the total percentage of correct predictions.

Table 2: Performance (accuracy) of models on the SNLI data

Models	Original		Implementation	
	Training	Test	Training	Test
Sum-of-words [Bowman et al., 2015]	79.3	75.3	68.1	68.9
LSTM [Bowman et al., 2015]	84.8	77.6	72.41	73
Chen-BiLSTM [Chen et al., 2016]	92.9	87.7	85.27	85.1
Chen- BiLSTM + Bilinear	-	-	85.42	85.2

Confusion matrix was a good tool to visualize how the training accuracy improved over with increasing number of epochs. As the model becomes more sophisticated, the incorrect predictions are more concentrated towards Neutral label which is good as it reduces incorrect Entailment and contradiction predictions (better neutral than incorrect one-sided classification). This can be seen below:

Table 3: Confusion matrices of models showing actual and predicted labels

Bowman LSTM

Label\Pred	E	N	C
E	2727	300	341
N	601	2163	455
C	471	484	2282

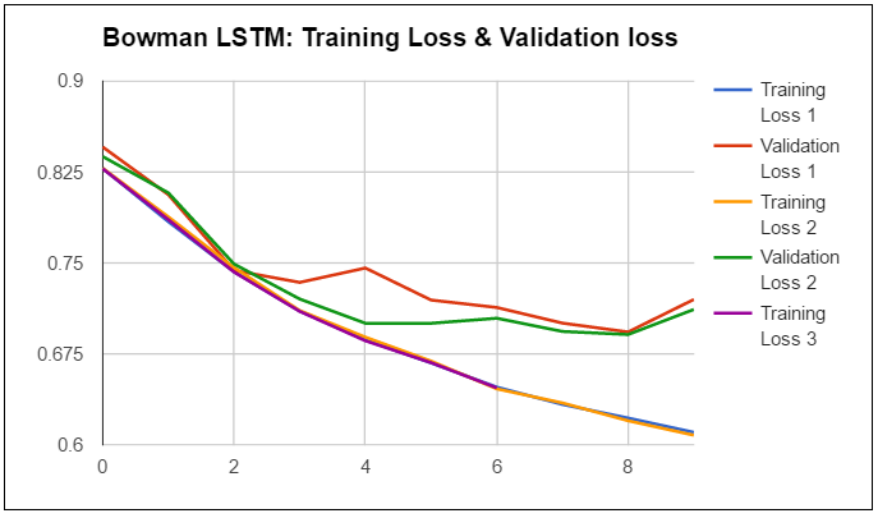
Chen BiLSTM

Label\Pred	E	N	C
E	3024	253	91
N	390	2559	270
C	151	302	2784

Chen BiLSTM + Bilinear

Label\Pred	E	N	C
E	2986	268	114
N	343	2542	334
C	131	267	2839

The loss history is shown below for Bowman LSTM model and we can see that the training loss consistently decreases as we keep training the model but over-fitting will lead to increase in validation loss eventually.



5 Conclusion

Fine tuning of hyper-parameters would get closer results to the paper. We could also explored training and independent evaluation on another dataset like SICK. Updating of word embeddings has known to give better results and this could be added to our implementation for better results.

Overall, the simple modification of of Bilinear form attention on EBIM (Chen et al. [2] model) performs better than the later affirming the claim that potential of sequential LSTM-based models can be explored further.

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[2] Chen, Q., Zhu, X., Ling, Z., Wei, S., & Jiang, H. (2016). Enhancing and combining sequential and tree lstm for natural language inference. arXiv preprint arXiv:1609.06038.

[3] Parikh, A. P., Tckstrm, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. arXiv preprint arXiv:1606.01933.

[4] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

[5] Erick Rocha Fonseca, Decomposable Neural Network Models for Natural Language Inference, GitHub repository, https://github.com/erickrf/multiffn-nli

[6] Wang, S., & Jiang, J. (2015). Learning natural language inference with LSTM. arXiv preprint arXiv:1512.08849.