**Deep Learning for text and sequences (89687)**

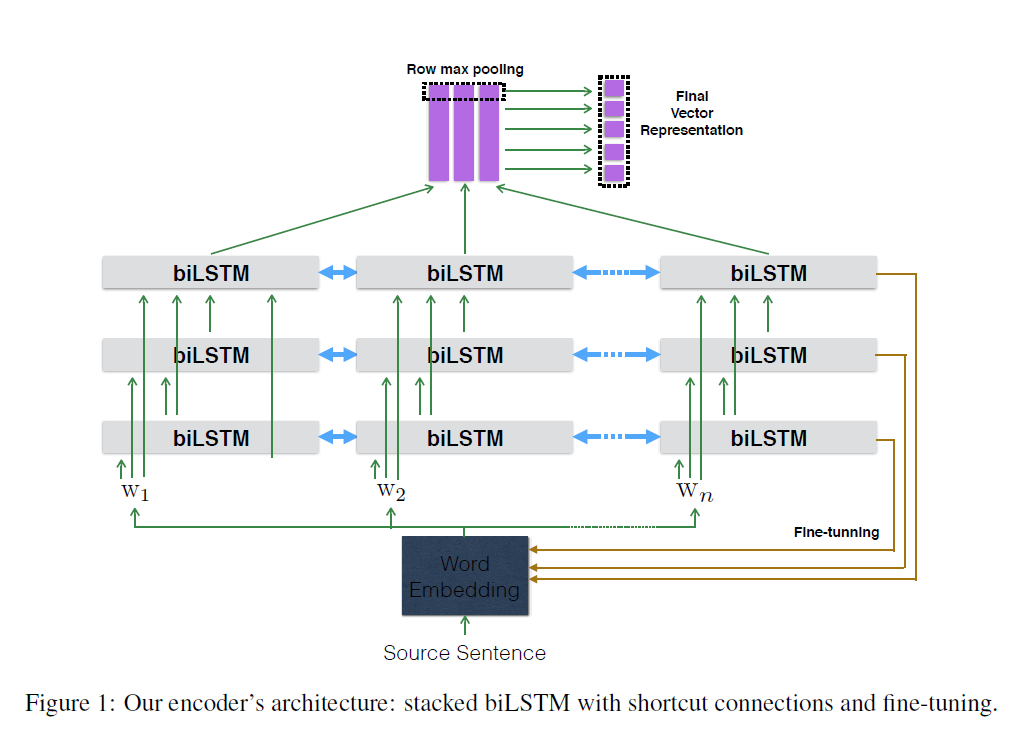
**Assignment 4 – SNLI model implementation report**

*Ido Natan | 305727802*

*Tzach Cohen | 208930842*

Part 1 - The paper

After trying some alternatives we chose [Shortcut-Stacked Sentence Encoders for Multi-Domain Inference](https://arxiv.org/abs/1708.02312) by Yixin Nie and Mohit Bansal.

In the last exercises we have implemented bilstms but wanted to try the shortcuts feature that   
was taught in the lecture. When we arrived at this paper we knew we wanted to experiment with   
this important feature which is one of it’s core innovating building blocks. Moreover, the structure of the model seemed very simple to us. The whole structure can be summed up to three familiar simple parts - and simple is good (Occam’s razor). Not all roses though, We had to compromise on the fact that this model is fed with 9.7m parameters, but, sooner or later we all need to learn to deal with such vast amounts of parameters if we want to develop deep learning models. So, we took this as a challenge of memory management and efficiency as well.  
  
Paper’s Model  
The paper is suggesting a simple method to construct a strong encoder which can work in any genre or domain of language inference. Using the pre trained [300D Glove 840B](https://nlp.stanford.edu/projects/glove/) to initialize the word embedding to process two sentences (the premise and hypothesis sentences) at input it feeds a network of 3  
layers of N biLSTMs which are connected with shortcuts (feeding all previous layers’ outputs and word embeddings to each layer) and fine tuning to end up with row-max pooling and finally creates a fixed length vector representation of these sentences.

After obtaining the vector representation for the premise and hypothesis sentences, three

matching methods are applied to the two vectors :

(i) concatenation

(ii) element-wise distance

(iii) elementwise product for these two vectors

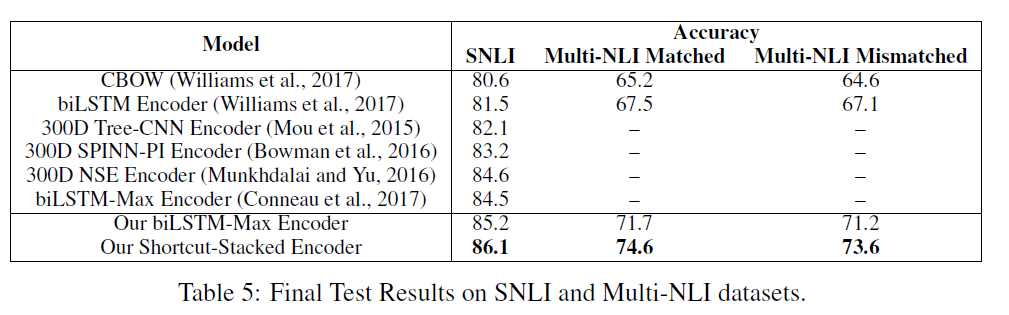
Finally concatenate these three match vectors  
Once the process of creating this complex vector is complete a 3 way MLP entailment classifier is applied to label the connection between the vectors as entailment (true hypothesis), contradiction (false hypothesis) or undetermined (neutral or can’t say). The classifier is constructed of MLP with Softmax for the final calculations.

Paper’s Results  
The data: [snli](https://nlp.stanford.edu/projects/snli/snli_1.0.zip) : train-set dev-set and test-set in the jsonl format.

Parameter settings:

|  |
| --- |
| *Loss function*: Cross-Entropy |
| *Optimizer:* Adam |
| *Batch size:* 32 |
| *Learning rate:* 0.0002 |
| *Learning rate Decay:* half decay every two epochs. |
| *MLP hidden* units*:* 1600 |
| *Dropout:* applied on the output of each later of the MLP. |
| *Dropout rate*: 0.1 |
| Pre-trained vectors: [300D Glove 840B](https://nlp.stanford.edu/projects/glove/) |
| biLSTM layers: 3 |
| biLSTM dimensions: 512, 1024, 2048 |
| Epochs: was not mentioned. |

The reported results (against other methods, only the SNLI column is relevant of course):



Part 2 - Our efforts on implementing the paper

our model was trained for \_\_\_\_\_\_\_\_\_ and managed to achieve some success but unlike the authors. We suspect this has to do with our shortage of computing power that until the model was finished we couldn’t have known. In a private computer with 32GB RAM and cuda we received the notorious :   
cuda out of memory. With less glove vectors and smaller batch size the problem was fixed, the cost : compromising the results. Regarding the label ‘-‘: the authors have not mentioned what they did with this and so after counting samples with “-“ label which was very small comparing to the rest of the labels we decided to ignore it.

Compared to theirs

To elaborate we have implemented their exact architecture and hyperparameters as described above.  
Then we tuned the learning rate and changed the optimizer. Our tunings and results are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test : 1 | Optimizer: Adam | Lr: 0.0002 | Batch size: 32 | Epochs: 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test : 2 | Optimizer: Adagrad | Lr: 0.0002 | Batch size: 32 | Epochs: 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test : 3 | Optimizer: Adam | Lr: 0.002 | Batch size: 32 | Epochs: 4 |

Success, failure etc.

We had problems with dimensions in processing the data through the bilstm stacks. We had to use pack\_pad\_sequence and pad\_pack\_sequence in order to prepare the data correctly in the shortcuts part. We succeeded in making the model run and learn we don’t know how many epochs the original paper’s authors used, we tried from 4 to 25 epochs. Not all we managed to run in time to submit.  
Close to the end of the implementation we came across memory problems when using the full glove vectors even in Google-Colab and so we tried using less than the 400k given.

Out of the box when training the model some loss decreases but accuracy was hardly increasing, it was pretty random as can be seen in the graphs. We tried to tune the amounts what gloves pretrained and random vectors were given as embedding matrix. We looked for a requires\_grad=False value In any of the components but found all turned True. We tried with learning in the pretrained embedding and without – results were the same: random accuracy between 0.35 to 0.75.

Part 3 – Suggestions - improvements to the algorithm

As suggested by the authors themselves three layers of stacked biLSTMs are somewhat minimalistic. So the first thing to try in order to improve the model is increase the number of layers from three to four for example.