Avital Rose 318413408

Report Assignment 4

Implementation of Paper from the SNLI competition

1 Paper selection

Implementation of the “Distance-based Self-Attention Network for Natural Language Inference” paper by Jinbae Im and Sungzoon Cho from the Seoul National University.

2 Reasons

The reasons I choose the paper are as follows:

1. I like the attention mechanism, as opposed to other methods such as Bi-LSTM and more.
2. The idea seemed elegant and simple, and well organized into sub sections which were also elegant and simple.
3. The algorithm includes a relatively low number of parameters, and I thought it would mean that the code will run faster. Also, the paper uses an attention mechanism, which overall runs relatively fast.

3 Method used

Summary:

The method proposed in the paper includes encoding each sentence (premise and hypothesis), then concatenating different variations of the encoding and finally passing through a feed forward network to get probabilities of each outcome. The sentence encoding uses a transformer, and the main innovative idea used in the paper is a Directional Multi head self-attention mechanism, where not only the direction of the word in the sentence is considered, but also the distance from the word to the rest of the words.

Further explanation:

The illustration of the overall model architecture can be seen in figure 1. The sentence encoder layout is shown in figure 2.

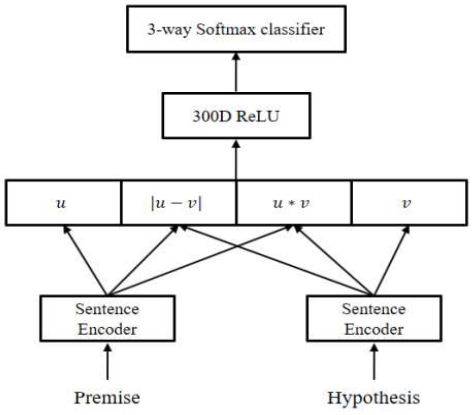


Figure 1: **Overall architecture**

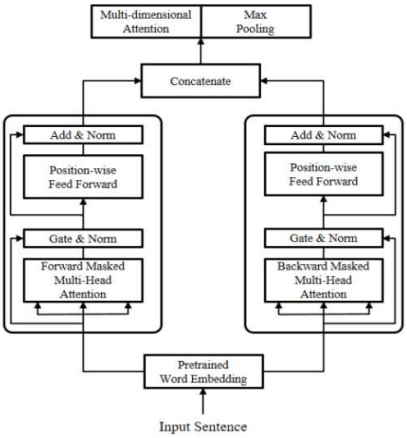


Figure 2: **Sentence encoder**

The attention equation is expressed as following:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

Attention equation with mask:

$$Masked(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}} + M_{dir} + \alpha M_{dir}\right)V \qquad (2)$$

After the attention[1] mechanism was introduced, research [2]showed that considering the direction of the word in a sentence has a significant value. I.e. the model, should not have access to the whole sentence at a time, but rather only the words leading up to its position in the sentence. So, in a forward pass, the mask would be as the mask in figure 3(a), and in a backward pass, the mask would be as in figure 3(b).

This study offers to take a deeper look into the position of the word in the sentence, not only the direction (blocking out words before or after the word) but also the distance between the words in the sentence. This distance is key to learning the local dependency to help understand the context of the text. This is done using a simple distance mask as shown in figure 4. As can be seen in the figure, the (i,j) component of the distance mask is $-|i-j|$ which represents the distance between the $(i+1)^{th}$ word and the $(j+1)^{th}$ word multiplied by -1. The matrix is later multiplied by α, so adding this matrix to the logit results in the attention weight becoming smaller as distance increases. This complies with the idea of local dependency – closer words tend to be more related.

	1	2	⋯	n
1	-∞	-∞	-∞	-∞
2	0	-∞	-∞	-∞
⋮	0	0	⋱	-∞
n	0	0	0	-∞

(b) Forward mask

	1	2	⋯	n
1	-∞	0	0	0
2	-∞	-∞	0	0
⋮	-∞	-∞	⋱	0
n	-∞	-∞	-∞	-∞

(a) Backward mask

Figure 3: **Directional mask**

	1	2	⋯	n
1	0	-1	⋱	-(n-1)
2	-1	0	⋱	-(n-2)
⋮	⋱	⋱	⋱	⋱
n	-(n-1)	-(n-2)	⋱	0

Figure 4: **Distance mask**

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv preprint arXiv:1706.03762

[2] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnnfree language understanding. arXiv preprint arXiv:1709.04696.

4 Results

4.1 Paper results

The paper reported the following results:

Model	T(s)/epoch	Train acc (%)	Test acc (%)
Our Distance-based Self-Attention Network	693	89.6	86.3

4.2 Code results

My implementation of the code had the following results:

Model	T(s)/epoch	Train acc (%)	Test acc (%)
Final Distance-based Self-Attention Network	779	84.0	77.9
Model which was saved and can't recreate	N/A	87.2	N/A

Note: time is according to time for program to run on google colab (gpu). Otherwise about 36 hrs (personal pc is cpu).

Figure 5 presents the learning curves for the train (blue), dev (orange) and test (green)data sets for the final model which was handed in. The accuracy is plotted as ratio of correct labels to all labels. The model is learning and performing not too bad on the datasets.

Figure 6 presents the learning curves for the train (blue) dataset of a model that was not saved, and I did not manage to find the exact way it was configured. The accuracy is in percentage. The model did not run on the dev or test sets, only on the train. The model achieved results close to the results reported in paper.

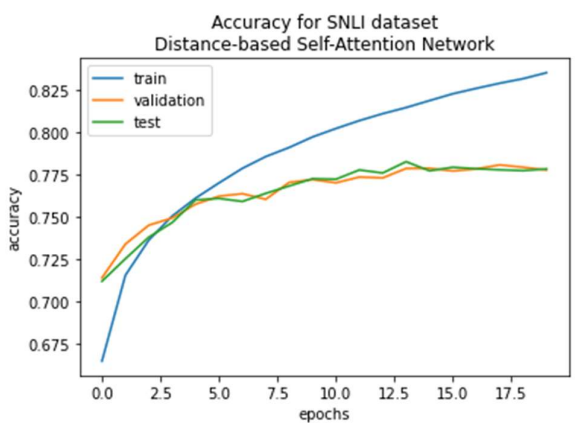


Figure 5: **learning curve final model**

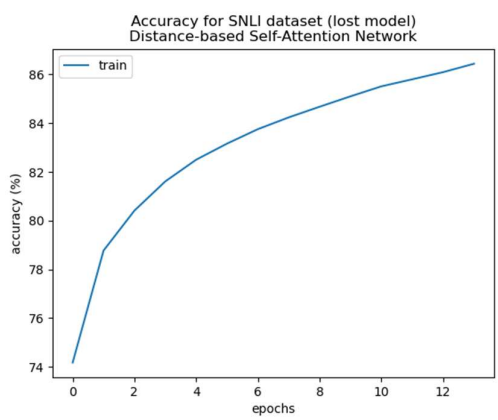


Figure 6: **learning curve lost model**

5 Discussion

5.1 Code performance

The code did not manage to replicate the results. I assume this is either due to partially ambiguous instructions of the article, or due to the data representation.

Possible failure points regarding the ambiguous instructions:

- Complete dropouts performed.
- Complete layer normalizations performed.
- Weight initializations for linear projections.

Possible failure points regarding the data representation:

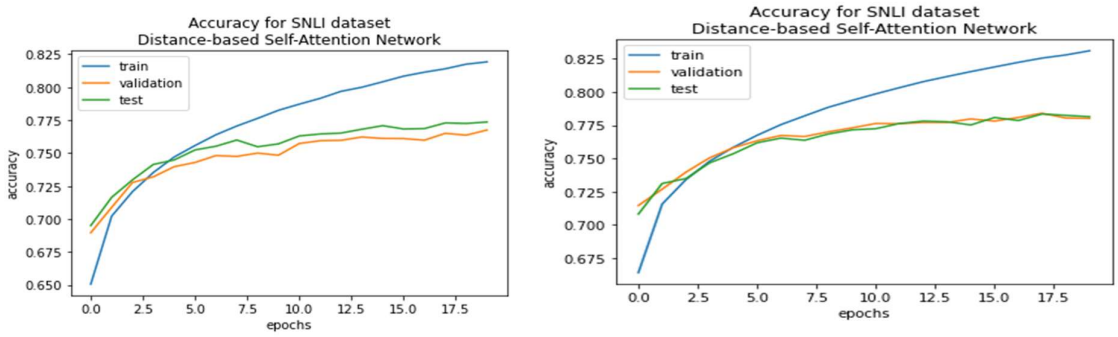
- The sentences in my implementation are padded to the maximum length of the longest sentence in the batch. The sentence pairs are sorted by the premise length to achieve minimum padding per sentence. Maybe a different padding configuration would have been better.
- A constant embedding was given to all unknown words. Maybe a changing embedding, an embedding which is an average the embedding matrix etc. could have been more beneficial.

5.2 Approaches tried

To achieve better results, the following approaches were tried:

1. Trying different weight initializations (Figure 7(a)). By not choosing xavier distribution for weights initialization, weights are initialized to default kaiming_uniform distribution (since using nn.linear).
2. Sorting sentences pairs by premise length to receive less padding overall (similar length- sentences put together)
3. Different dropout and layer normalization approaches (Figure 7(b)).

None of the approaches helped achieve better results.



(a) No weight initialization (b) Less Dropouts run

Figure 7: **learning curves for approaches tried**

5.3 Suggestions for improvement

The algorithm explores the effect of the distance between the words in a sentence on the relationship between the words. Possible improvements can explore either more complex ways to express the physical distance between the words or explore the “distance meaning” between the words.

Avital Rose 318413408

More on a complex distance measure: The algorithm computes the distance between the words as $-|i - j|$. This approach assumes a linear connection between the distance of the words. Maybe a different decay will result in better results: exponential, polynomial etc.

More on a “distance meaning”: The algorithm fucoses on the physical distance between the words. What if we would add a component that will consider also the relationship between the meaning of the words. Words which are closer in meaning (“dog” and “cat” for example) tend to have similar embedding vectors (cosine distance). An improved algorithm can perhaps take that distance as well into account.