Word In Context Disambiguation Homework1-NLP

Mandelli Lorenzo 1747091

1 Introduction

In the following pages I will discuss my work for the proposed task, describing the overall approach I choosed, my intuitions, my solutions and commenting the obtained results.

2 Experiment

Since I had a lot of difficulties finding out a decent layers combination and the corresponding adequate learning rate which I should have used for the training step, I decided to set up an 'experiment' whose purpose is to generate a lot of models and train them using different learning rates and preprocess techniques. I used different combination of layers resulting in several 'architectures' I called for simplicity Nstrct with N varying in 0 to 5. Than I have parametrized each layer by the n-hidden units (varying from 8 to 200 with a pace of 8), and each model by the training learning-rate (varying from 0.3 to 1.3 with a pace of 0.1). I did this procedure both for MLP and LSTM models, resulting in a lot of data to inspect, consisting in accuracy and loss plots as well as model-state-dict file and a text file which contains the best validation accuracy obtained. In this way I obtained a clear picture of how whether my intuitions were correct or not, and in both cases proceed further in the right way; Furthermore the 'experiment' has been very convenient because it let me eventually select and reproduce one specific convincing architecture hyper-parameter combination in order to work further on it or changing/adding something.

3 MLP, LSTM and Overfitting

Before setting up the experiment environment I played around with basic MLP and LSTM architectures using BCELoss with sigmoid prediction and SGD Optimizer for the MLP models and CrossEntropyLoss with softmax prediction and SGD (Adam Optimizer did not enhance performances in both cases so I opted for using SGD only).I started by working on LSTM models since I thought them to be a solid power full tool for the task proposed, but soon I realized they are very sensible to overfit. While working with these networks I used alternatively dropout and batch normalization and I further introduced other different architectures because of performances problems; I also tryed deeper structures as well as GRU and BILSTM layers instead of simple LSTM layers. After dozen of trials, I kept obtaining similar results, the best one coming from a standard training procedure where just the base preprocess was employed. I noticed a strong predisposition to overfit also for MLP models. In this case, as best practice for NLP suggests, working around a smart preprocessing, looking for a specific kind of architecture, adding Dropout layers as well as tuning some hyper-paramters such as the number of units of the layers or the deepness of the network, I successfully reduced the undesired behaviours. Each model have been trained on purpose for 70/80 epochs in order to study always the full behaviour/evolution. After some trial I selected respectively six kind of architectures both for MLP and LSTM which vary from a single linear layer to four linear layers and dropout layers; these architectures are displayed in the colab notebook.

4 Base Preprocess, Words Aggregation and Sequence Encoding

As a general procedure I applied literally every time, I subdivided each training sample into its main components: sentence1, sentence2, lemma and label. Than I proceeded lowering each word composing the sentences and removing the punctuation (briefly BASE). After this common steps I adopted two different preprocessing techniques. The first procedure, I called SCONCAT, consists first in concatenating sentence1 and sentence2, separated by a special character and than transform the result into a tensor via the aggregation function. The second procedure, I named TCONCAT, consists instead in transforming first both the sentences into tensors and than concatenate the two resulting tensors. In first case I ended up with a torch.Tensor[(300)] sample descriptor, while in the second case I ended up with a torch.Tensor[(600)] sample descriptor, which ideally should be more representative.As a best practice for NLP the sentence to tensor transformation is done by an aggregation function called sentence2vec which makes use of a pretrained word embedding(in my case I used GloVe with a feature space of 300 elements). When dealing with LSTM models I used a sequence encoding solution like the one proposed during the lectures using word2index, in order to generate sequence of words and also being able to trace back the word from the index. For the MLP approach I also tryed different kind of word aggregation techniques: originally I used a weighted sum approach, using higher weights for words which are more representative and lower weights for words which are less representative; I considered words which occurs most and least frequently to be less representative and on the contrary words which occurs with a medium frequency to be more representative. Than I also tryed weighted mean, simple sum and simple mean, together with sample normalization (0 mean and 1 std). In the end the simple mean turned out to be the most effective choice.

5 Additional Preprocess

The main idea was to find out a good combination of preprocessing techniques to use before the aggregation in order to refine the data and enhance evaluative performances. As a first one I removed the most frequent words, words which occur more than 500 times (briefly MOST), such as prepositions and conjunctions, since they less contribute to the overall context and sense of the sentence. Working around I noticed that there were a lot of words occurring just once. I thought these words would, similarly, not have given any contribution to the overall context of the sentence than I tryed removing them together with the most frequent ones (briefly mixPre). I also report one last procedure consisting in refining the sentences by eliminating numerical quantities together with randomly eliminating words according to their occurrences plus eliminating most frequent words (briefly VARIOUS), even if it turned out to be a complete failure, therefore I cite it here and I will not consider it later.

6 Training Procedure and Model Selection

Each model we previously introduced has been trained in different experiment sessions using both the SCONCAT technique and the TCONCAT technique together with each different additional preprocessing techniques; in this way I ended up with several different models coupled with a specific preprocessing combination. First I trained the various models only with BASE and I considered the validation results as a baseline; I will call the models trained in this way 'model-noPre'. Than, I trained the same models in three ways: BASE + MOST calling them 'model-MOST', BASE + MIX calling them 'model-MIX' and BASE + VARIOUS calling them 'model-VARIOUS'. Proceeding by inspection of the experiments results I selected the best models generated, consolidating some intuition and undermined others.

7 Conclusion and Observations

After the result collection from the different experiments it turned out that for MLP models, architecture 3, 4 and 5 gave generally the best results; this lead me concluding that narrow bottlenecks and consequent smaller feature spaces are the best choice according to the dataset dimension and potentially for the task itself. Moreover too small learning rates helps avoiding overfitting but at the same time brings the model to underfit; therefore the best results clusters around models trained with a learning rate between 0.7 and 0.11 and a considerable number of input units which quickly reduce toward the output layer. As I expected the sample TCONCAT strategy turned out to be particularly better with respect to the SCONCAT one, this is due to the fact it better represents the samples. Another fact that can be observed is how a smart preprocessing can in effect enhance the performances, in my case the mix-Pre, as I expected turned out to be the best one. It can also be observed that dropout layers helped a lot avoiding overfitting and therefore let the model achieve a more general prediction power. For LSTM/BILSTM models it is always observable a strong overfitting behaviour.

8 MLP and LSTM Tables and Plots

For MLP I will display in the following order, the tables which contains the best accuracies obtained subdivided in colors, and only the respective accuracy plot of the best models (the green ones), while for LSTM models, since i got always very similar results, I will display just one table containing the best results and some plots, even if they are all similar. In the end I summarized my results for MLP models in two different tables showing how the preprocessing choices led to very different results.

MLP SCONCAT+noPre	
Structure-units-lr	Accuracy
0strct-128-0.07	0.630
1strct-96-0.11	0.631
2strct-88-0.11	0.634
3strct-152-0.09	**0.643**
4strct-144-0.11	**0.645**
5strct-128-0.11	0.602

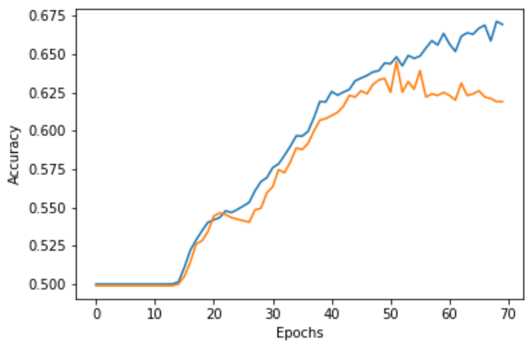

Figure 1: ACCURACY PLOT for MLP strct 4 with 144 units trained with SCONCAT+noPre

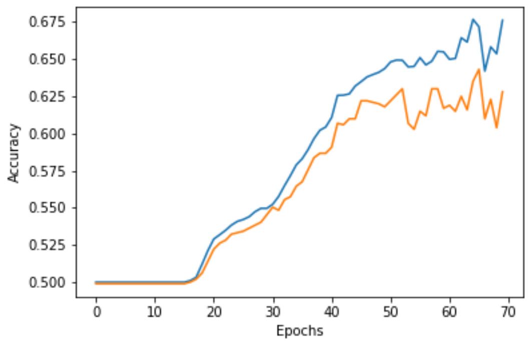

Figure 2: ACCURACY PLOT for MLP strct 3 with 152 units trained with SCONCAT+noPre

MLP SCONCAT+MOST-Pre	
Structure-units-lr	Accuracy
0strct-224-0.09	0.632
1strct-104-0.11	0.649
2strct-48-0.1	0.639
3strct-48-0.11	**0.654**
4strct-76-0.04	0.648
5strct-148-0.09	**0.659**

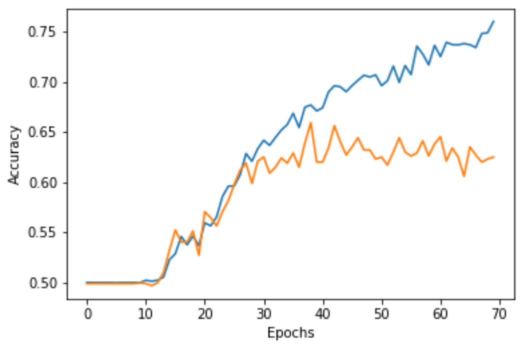

Figure 3: ACCURACY PLOT for MLP strct 5 with 148 units trained with SCONCAT+MOST-Pre

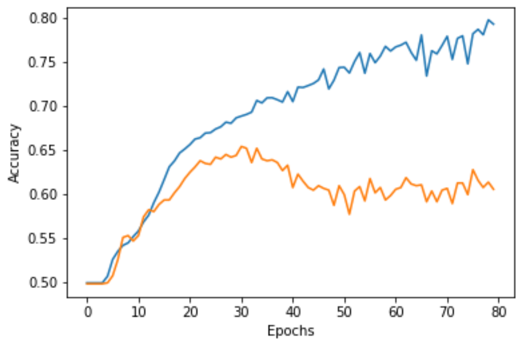

Figure 4: ACCURACY PLOT for MLP strct 3 with 48 units trained with SCONCAT+MOST-Pre

MLP SCONCAT+MIX-Pre	
Structure-units-lr	Accuracy
0strct-152-0.11	0.632
1strct-152-0.11	**0.645**
2strct-64-0.09	0.644
3strct-132-0.11	**0.646**
4strct-156-0.07	**0.648**
5strct-156-0.11	0.644

MLP TCONCAT+noPre	
Structure-units-lr	Accuracy
0strct-136-0.09	0.673
1strct-176-0.07	0.670
2strct-96-0.05	0.671
3strct-48-0.05	**0.681**
4strct-96-0.07	**0.680**
5strct-152-0.11	**0.680**

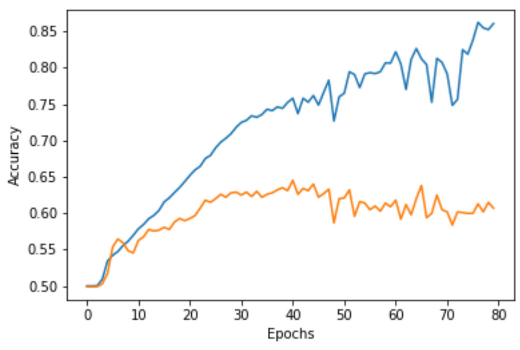

Figure 5: ACCURACY PLOT for MLP strct 1 with 156 units trained with SCONCAT+MIX-Pre

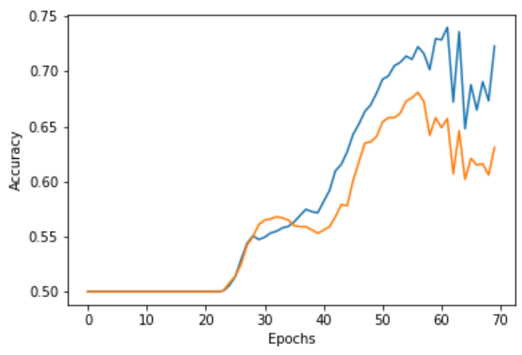

Figure 8: ACCURACY PLOT for MLP strct 3 with 48 units trained with TCONCAT+noPre

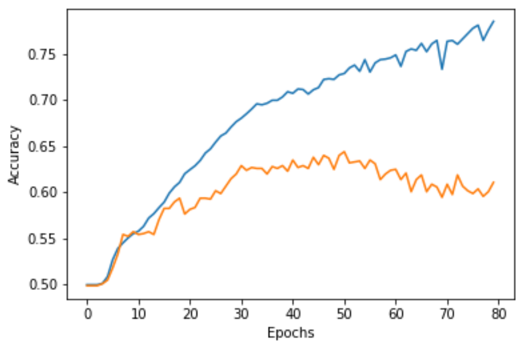

Figure 6: ACCURACY PLOT for MLP strct 2 with 132 units trained with SCONCAT+MIX-Pre

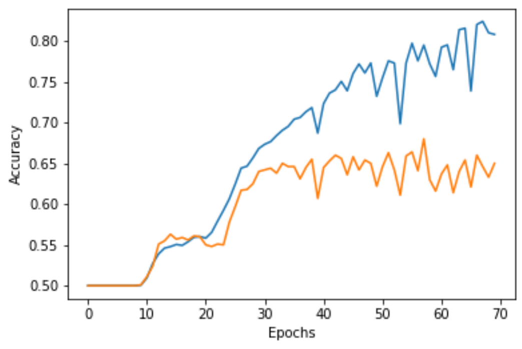

Figure 9: ACCURACY PLOT for MLP strct 4 with 96 units trained with TCONCAT+noPre

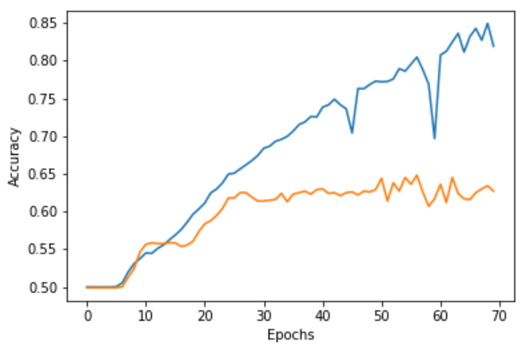

Figure 7: ACCURACY PLOT for MLP strct 4 with 156 units trained with SCONCAT+MIX-Pre

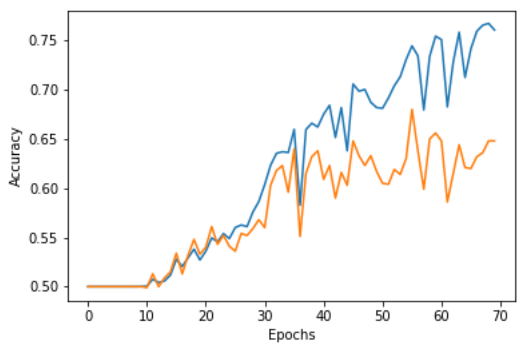

Figure 10: ACCURACY PLOT for MLP strct 5 with 152 units trained with TCONCAT+noPre

MLP TCONCAT+MOST-Pre	
Structure-units-lr	Accuracy
0strct-128-0.07	0.670
1strct-112-0.09	**0.681**
2strct-120-0.09	0.670
3strct-120-0.11	0.676
4strct-128-0.11	**0.682**
5strct-248-0.07	0.650

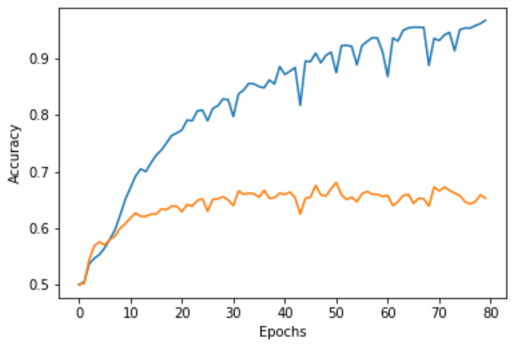

Figure 11: ACCURACY PLOT for MLP strct 1 with 112 units trained with TCONCAT+MOST-Pre

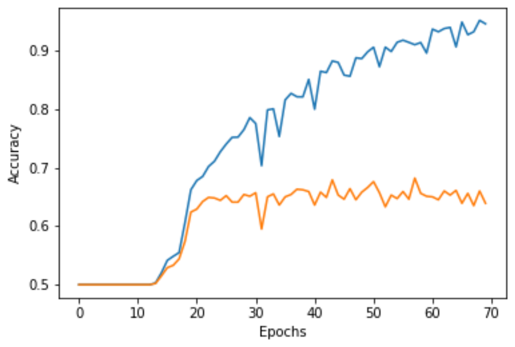

Figure 12: ACCURACY PLOT for MLP strct 4 with 128 units trained with TCONCAT+MOST-Pre

MLP TCONCAT+MIX-Pre	
Structure-units-lr	Model-0
0strct-184-0.05	0.670
1strct-72-0.09	0.677
2strct-192-0.05	0.677
3strct-56-0.07	0.680
4strct-120-0.09	0.679
5strct-120-0.07	**0.688**

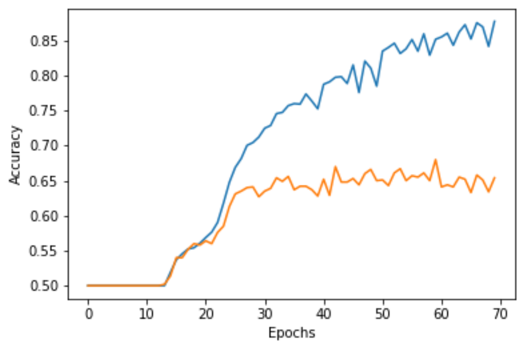

Figure 13: ACCURACY PLOT for MLP strct 3 with 56 units trained with TCONCAT+MIX-Pre

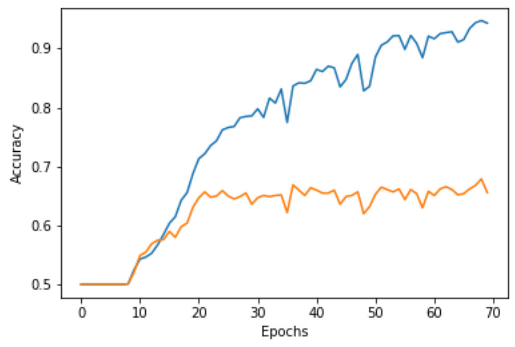

Figure 14: ACCURACY PLOT for MLP strct 4 with 120 units trained with TCONCAT+MIX-Pre

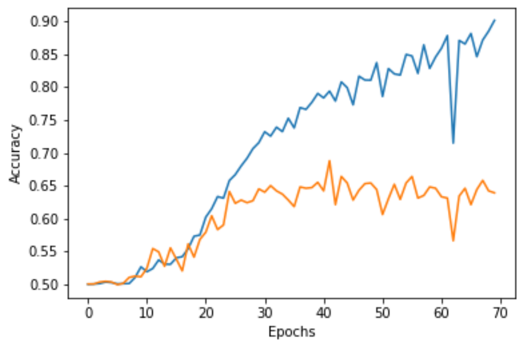

Figure 15: ACCURACY PLOT for MLP strct 5 with 120 units trained with TCONCAT+MIX-Pre

LSTM TCONCAT+noPre	
Structure-units-lr	Accuracy
0strct-88-0.19	0.551
1strct-96-0.16	0.550
2strct-32-0.16	0.530
3strct-56-0.19	0.554
4strct-48-0.16	0.545
5strct-128-0.19	0.567
6strct-96-0.13	**0.579**

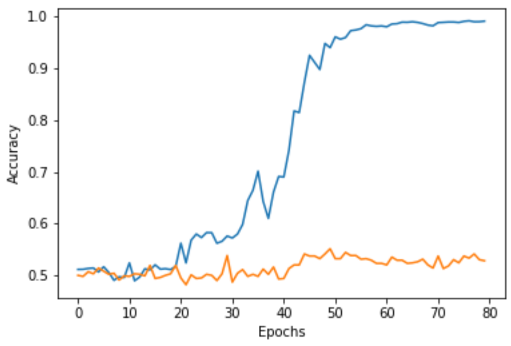

Figure 16: ACCURACY PLOT for LSTM strct 0 with 88 units trained with TCONCAT+noPre

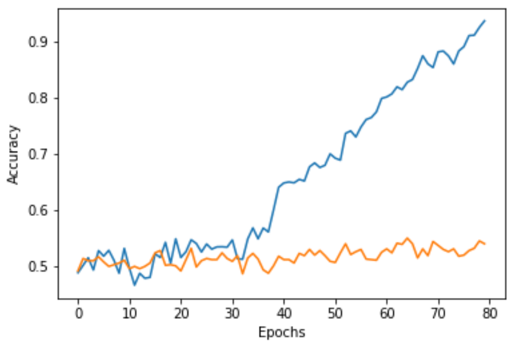

Figure 17: ACCURACY PLOT for LSTM strct 3 with 56 units trained with TCONCAT+noPre

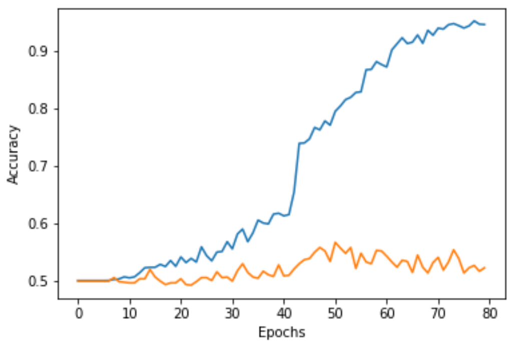

Figure 18: ACCURACY PLOT for LSTM strct 5 with 128 units trained with TCONCAT+noPre

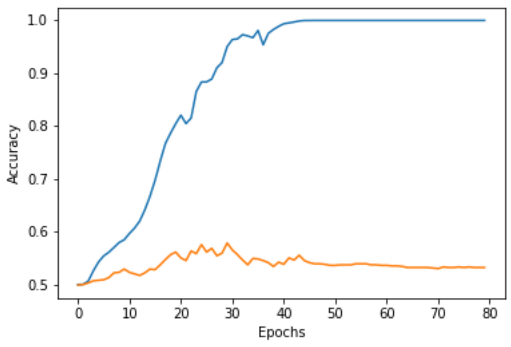

Figure 19: ACCURACY PLOT for LSTM strct 6 with 96 units trained with TCONCAT+noPre