NLP - Wet Exercise 2

Noam Wolf 318556206

Nir Diamant 304823560

Training

Basic model

Our basic model is from the exactly same architecture as the paper model (including the speed improvement part and the word dropout which was described in the requested section of the paper) i.e.,

Word embedding dimension	100
POS tag embedding dimension	25
Hidden units in MLP	100
BI-LSTM Layers	2
BI-LSTM hidden	125
α (for word dropout)	0.25

Table 1: Basic model/paper parameters

We trained this model using the ADAM optimizer with $lr = 0.01$ and effective batch size of 50 for 4 epochs yielding 88.51% UAS on the test and 97.17% UAS on the train, the training took 13 minutes an 36 seconds.

Advanced model

The first change we introduced is that by accident (misreading the paper) we use a POS embedding dimension of 100 this yielded an improvement of $0.2\% - 0.5\%$ (the latter models were improved more by this) we believe that though this is counter intuitive to have larger POS embedding dimension then the number of POS tags since there are extrmly low number of tags compere to the dataset and thus not likely to overfit and having larger embedding may give a repsetation that encodes better the information of the POS tag giving better inraction with the word embedding.
The first change we introduced was the use of a pre-trained word embedding since the pre-trained word embeddings was trained on a much larger dataset yielding word vectors that will allow better generalization (it will also prevent the emmeding of the model from overfitting, for those reasons we chose to freeze the pre-trained word embeddings). We used the 100 dimension Glove word embedding (over all our testing we saw that the dimensions from the original paper work quite well) this improvement increased the UAS on the test to 89% (without making significant changes to the train).
Next in order to introdous more regularization to the model we added dropout between the layers (and in the lstm) and a did a grid search on this value and larger value for the word dropout, our best performance in this stage was with 89.7% on the test and 97% on the train.
Then since the model was still at a state of overfit and increasing the dropout did not help we decided try to improve the ability of model to generalizes by changing it structure and the loss.
Firstly we noticed that by the analysis speed improvements section of the paper the MLP can be written as $v^T tanh(W_1 v_h + W_2 v_m + b_1) + b_2$ which is same as additive attention (apart from the biases)

and thus we also tried the following form of attention $v_h^T W_1^T W_2 v_w$ i.e. low rank general attention.
In term of the loss we tried the following loss augmented inference

$$\max(0, \max_{y'} \sum_{part \in y'} (score_{local}(x, part) + \mathbb{1}_{part \notin y}) - score(x, y) + 1)$$

i.e. making the model to keep a margin between tree that get high scores but also very wrong and thus the model will better capture the notion of constricting the tree even if it will still be overfitted and will better generalize, another important reason we wanted to change the loss is that the UAS on test for the basic model and the loss were raising at the same time (between epoch 2 and 4 see the graph in page 3).
Using both of the above improved the model to 90.4% on the test.
Lastly since the model was still at state of overfit we added the regularization term $\lambda \sum_{part \in y} score_{local}(x, part)^2$ we chose this regularization term since it should crate more even distribution of the score btween the parts of the tree. This (and grid search and some manual tuning) gave the following model:

Word embedding dimension	100
POS tag embedding dimension	100
Hidden units in the attention layer	100
BI-LSTM Layers	2
BI-LSTM hidden	125
α (for word dropout)	5
attention dropout	0.25
lstm dropout	0.1
λ for loss regularization	0.5

Table 2: Advanced model

We trained this model using the ADAM optimizer with $lr = 0.005$ for the first 6 epochs and 0.001 for the latter epochs and effective batch size of 50 for 18 epochs yielding 90.86% UAS on the test and 96.26% UAS on the train, the training took one hour and 4 and a half minutes.

Inference

The inference was done via the given code (chu_liu_edmonds.py) setting the weights of the edges entering in to "root" to $-\infty$ so "root" will always be the root of the inferred tree (the self edges do not need to be set to $-\infty$ as the algorithm handles them and dose not include them in the tree) no changes where done to the inference process.

Test

UAS on train basic	97.17%
UAS on test basic	88.51%
UAS on train advanced	96.20%
UAS on test advanced	90.86%

The file took 10.3 seconds to tag with the basic model and 9.7 seconds with the advanced model.

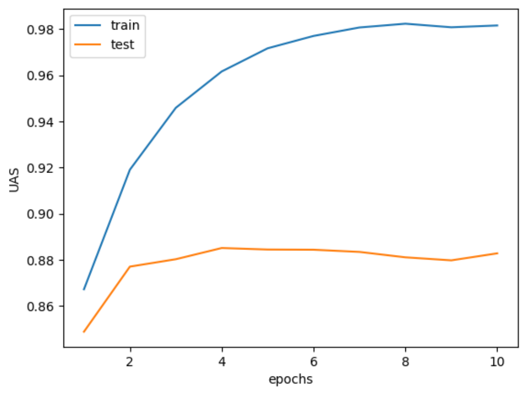

Figure 1: epoch vs UAS for the basic model

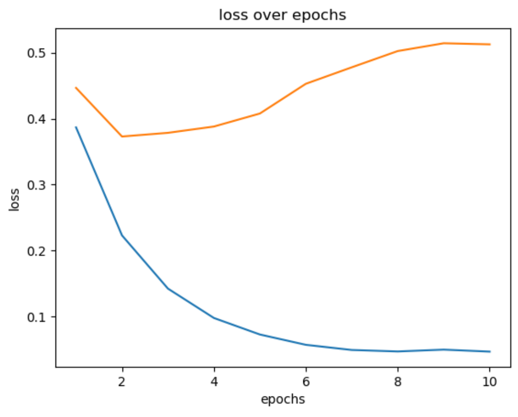

Figure 2: epoch vs loss for the basic model

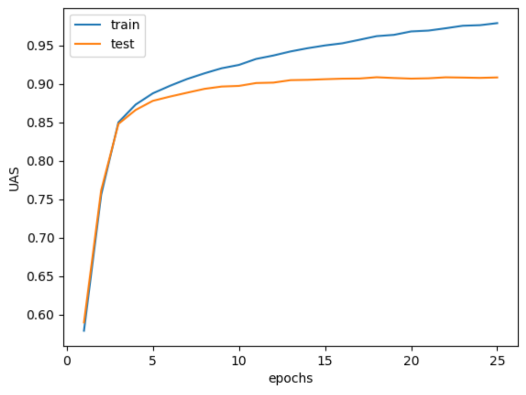

Figure 3: epoch vs UAS for the advanced model

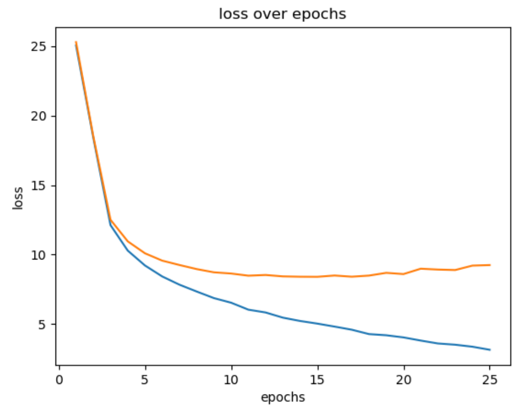

Figure 4: epoch vs loss for the advanced model

Competition

As all our changes in the advanced model was in the train the changes were discarded in the train section.

Work partition

We started from building the base network together and then Noam implemented the loss, infrence and evaluation while Nir implemented the dataset, the file tagger and the graph plotter, for the advenced model Noam implemented the new network and the paper loss while Nir implemented and (ran) the grid sreach. During the work we also consulted with each other on problems or ideas regarding the implementation and our respective parts.