Yehoshua Stern 314963927
Eyal Cohen 207947086

Part 4 (tagger 3):

Parameters: (POS with pre-trained embedding)

Learning rate	Epochs	Batch size	Batches per epoch	Middle dimension
0.5	5	256	4096	`(DIM * 5 + out_dim) / 2` = 138

Parameters: (NER with pre-trained embedding)

Learning rate	Epochs	Batch size	Batches per epoch	Middle dimension
0.5 divided by 2 every 5 epochs	10	256	4096	`(DIM * 5 + out_dim) / 2` = 138

The training process was with mini batches, every batch was taken randomly from the training data, we did not make a full pass for every epoch, we just took every epoch 4096 random batches of 256 samples.

The sub words vectors were initiated with random vector with uniform distribution that was normalized to match the mean vector.
We also saw that the pretrained vectors have already special UNK, START, END symbols so we used them.

We used the lower-cased vectors to add new vectors for each of the words we have in the training data.
For example: if we had a pre-trained vector for the word "in", if we saw the word "In" we copied the vector for the lower-cased word and gave it to the new upper-cased word.

Another thing we noticed is that the given vocabulary has no numbers, instead each digit in number was switched to DG (100 turned to DGDGDG), so we did this transformation to the training data too. (which added 2% to the accuracy on the dev set)

Also we divided the vectors by 30, we noticed the model converging much faster when we do it.
We did it to match the standard deviation of the vectors we got in tagger-1.

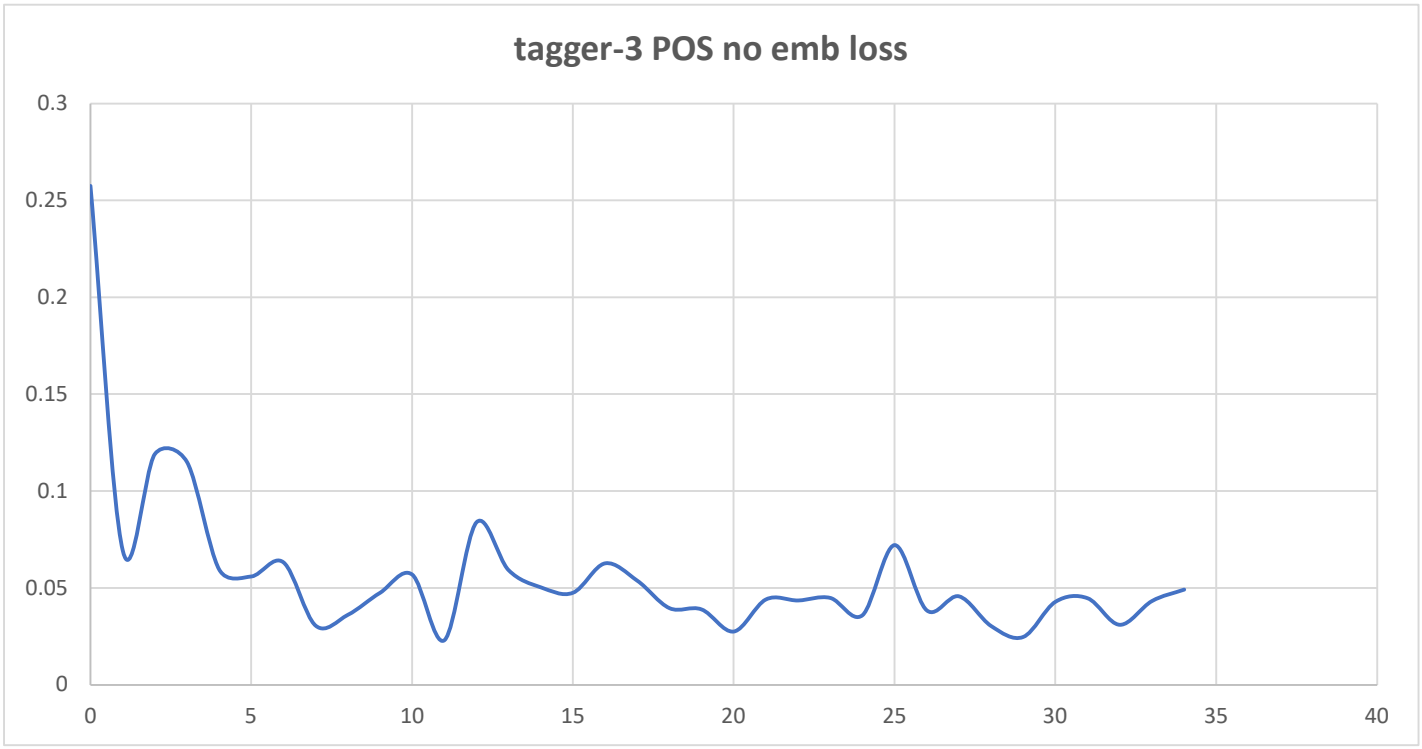
Brief analysis:

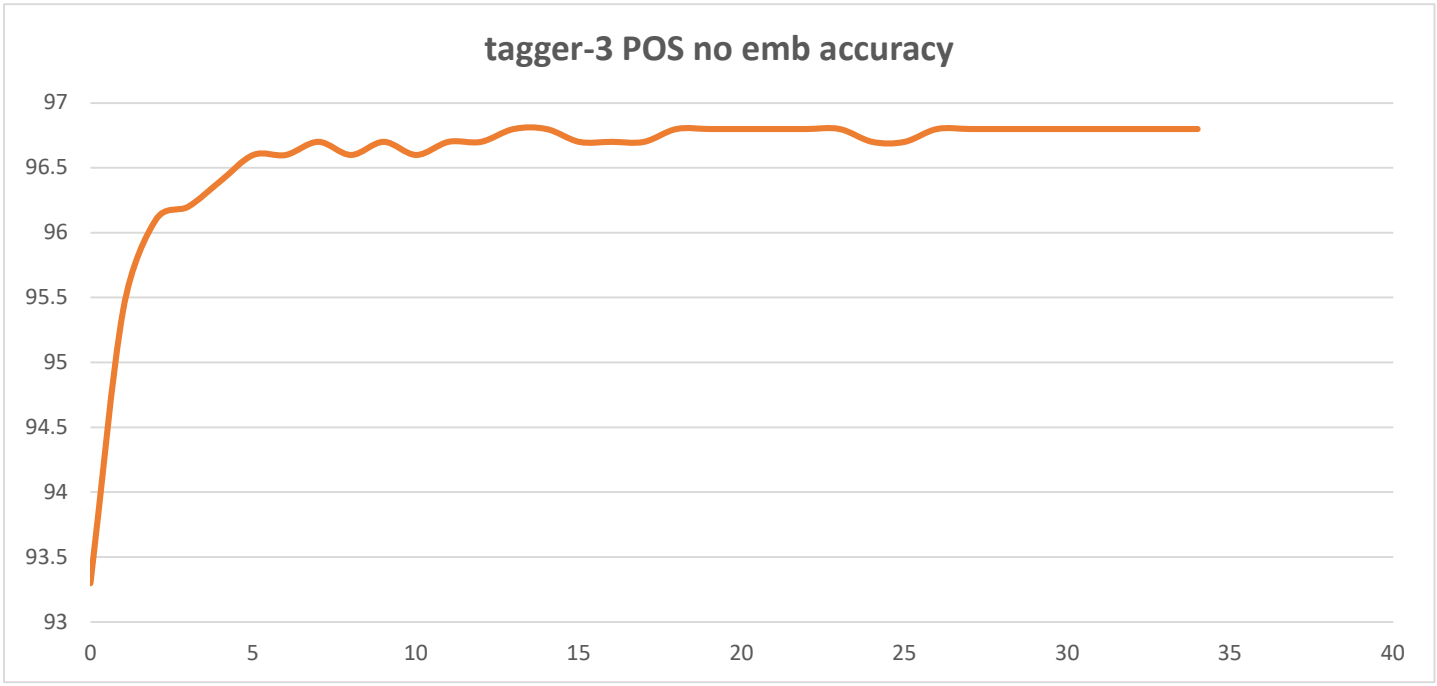
Accuracy on dev POS		
	No subwords	**With subwords**
No pre-trained embedding	**95.8%**	**96.7%**
Pre-trained embedding	**95.9%**	**96.7%**

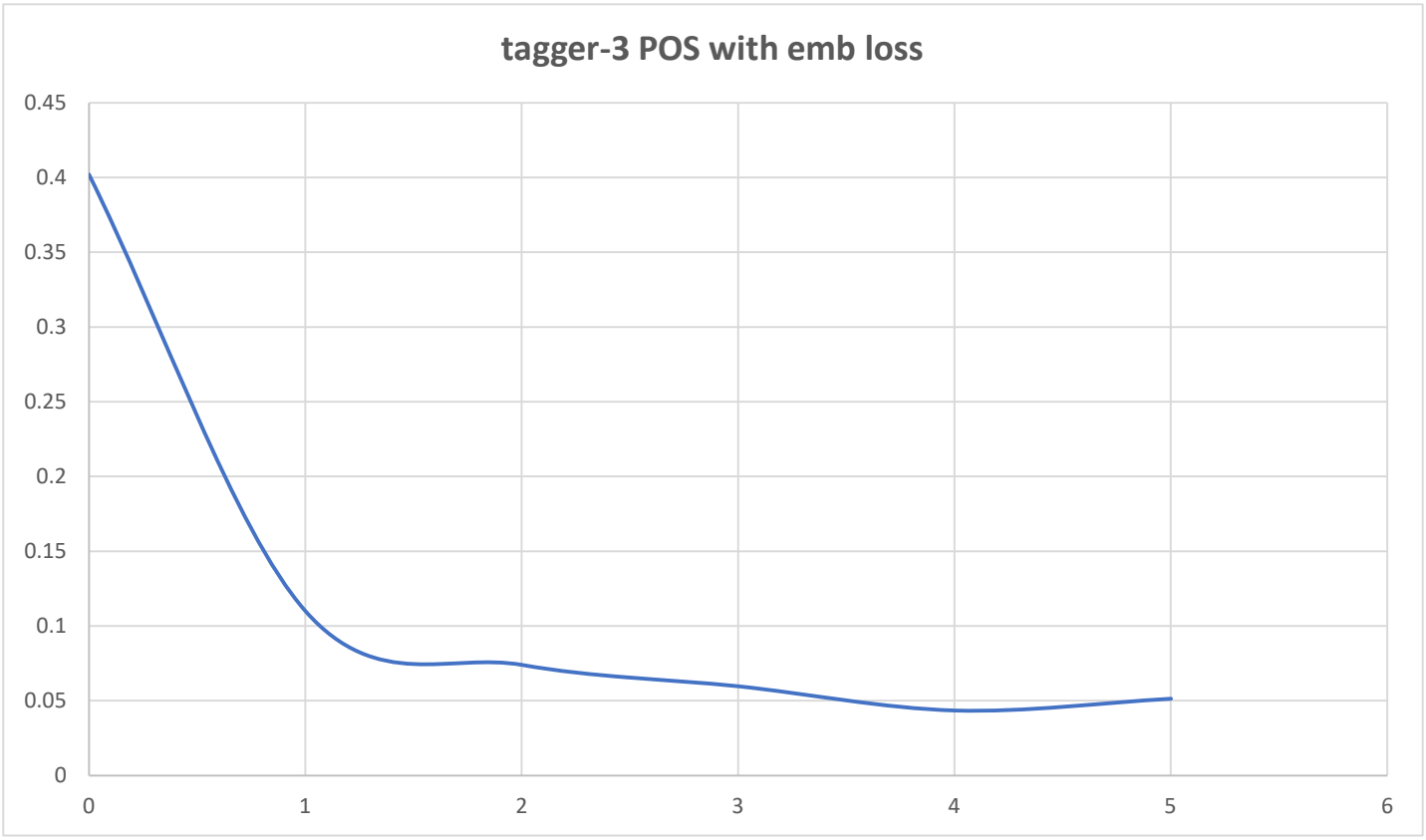
Accuracy on dev NER		
	No subwords	**With subwords**
No pre-trained embedding	**80%**	**86.2%**
Pre-trained embedding	**81.1%**	**86.2%**

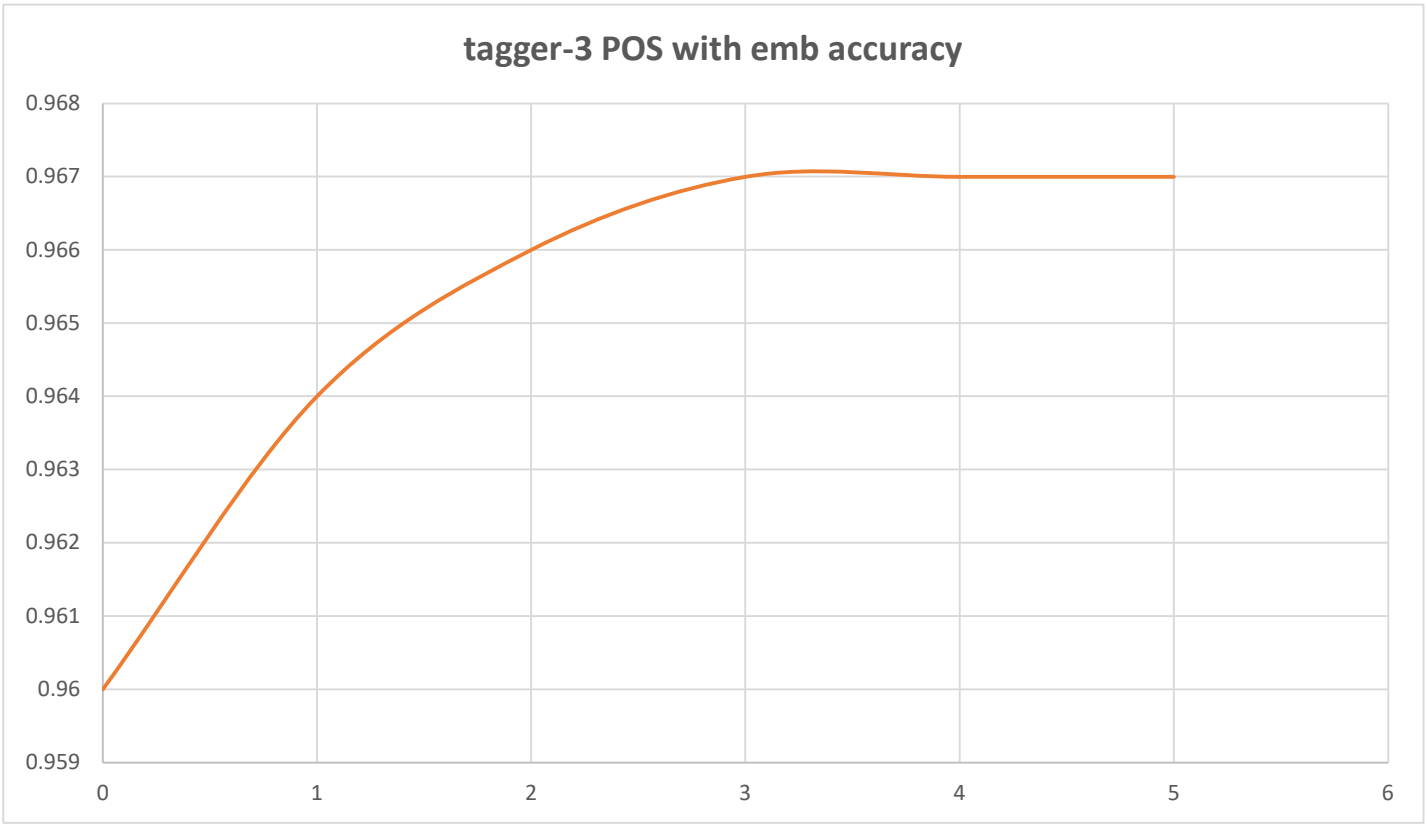
We noticed that the sub-words units increased the accuracy on the NER by 5.7%, on the pos
As for the pre-trained embedding layer in both cases the accuracy didn't increase by much.
With these results we can conclude that for this specific task, this specific pre-trained embedding data did not help us to increase the accuracy, but we did noticed that the convergence if the model is much faster when we use the pre-trained data.
We are not sure if their contribution is complementary, all we can say that we didn't see any loss of accuracy when using them both together.
In the NER tagging task specifically we can see that the accuracy increased when using the pre-trained embedding layer by 1.5% but when we used the sub-words tagger the accuracy did not increase after using the pre-trained embedding.

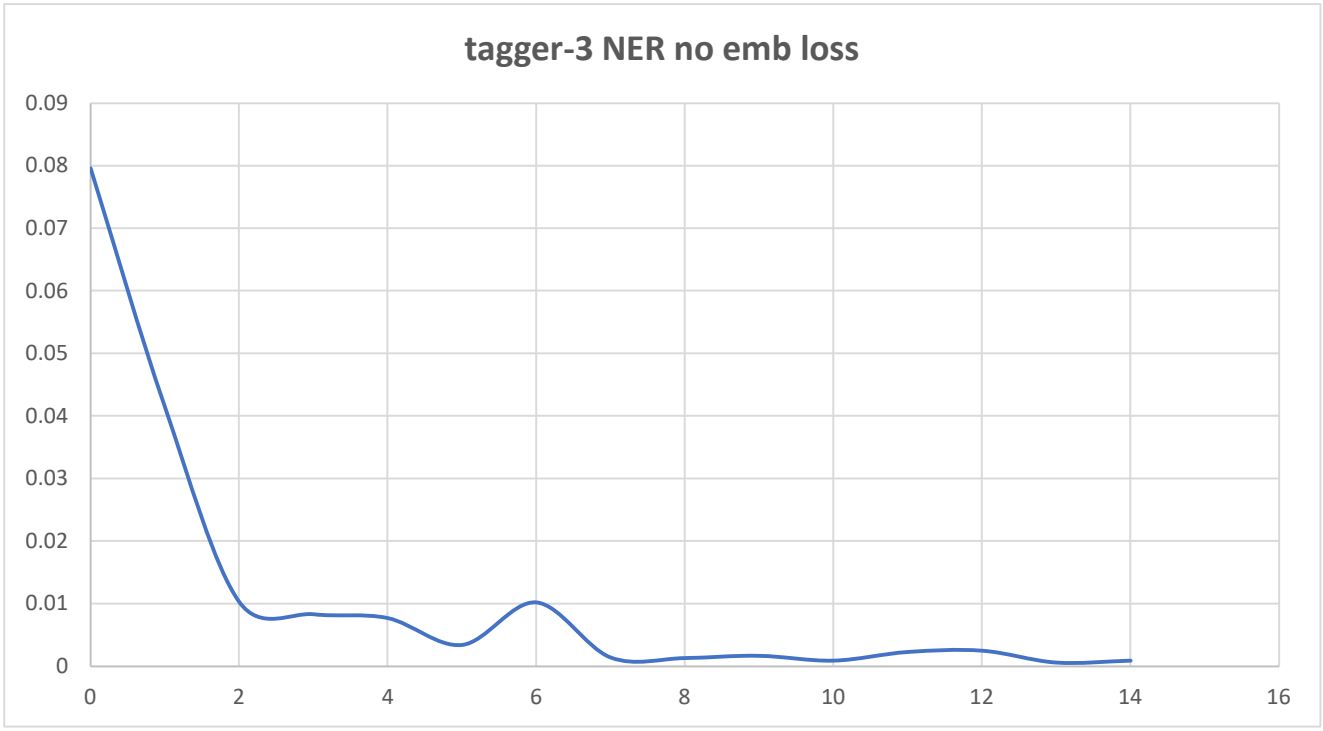
Graphs: (x is epochs, y is value)

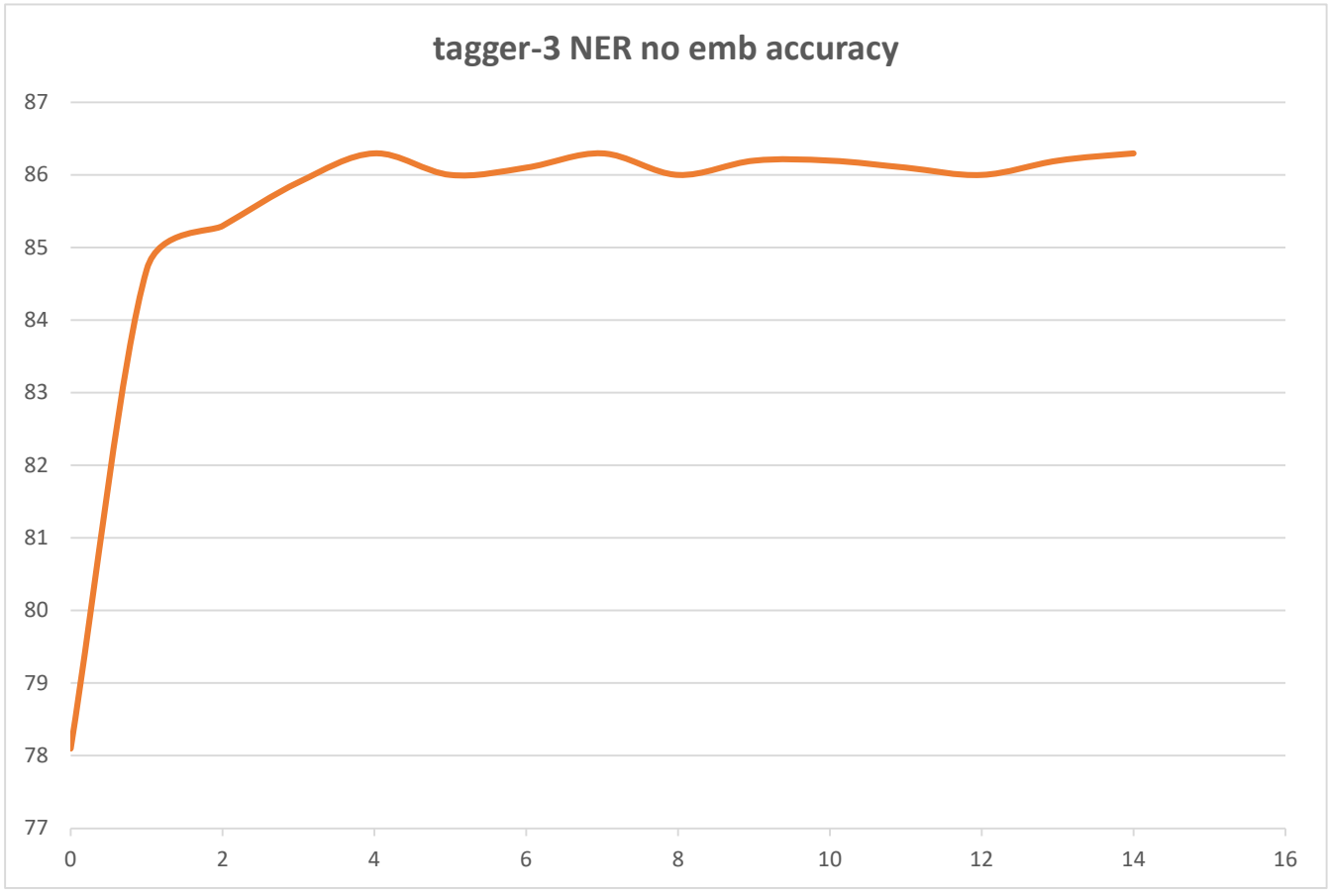


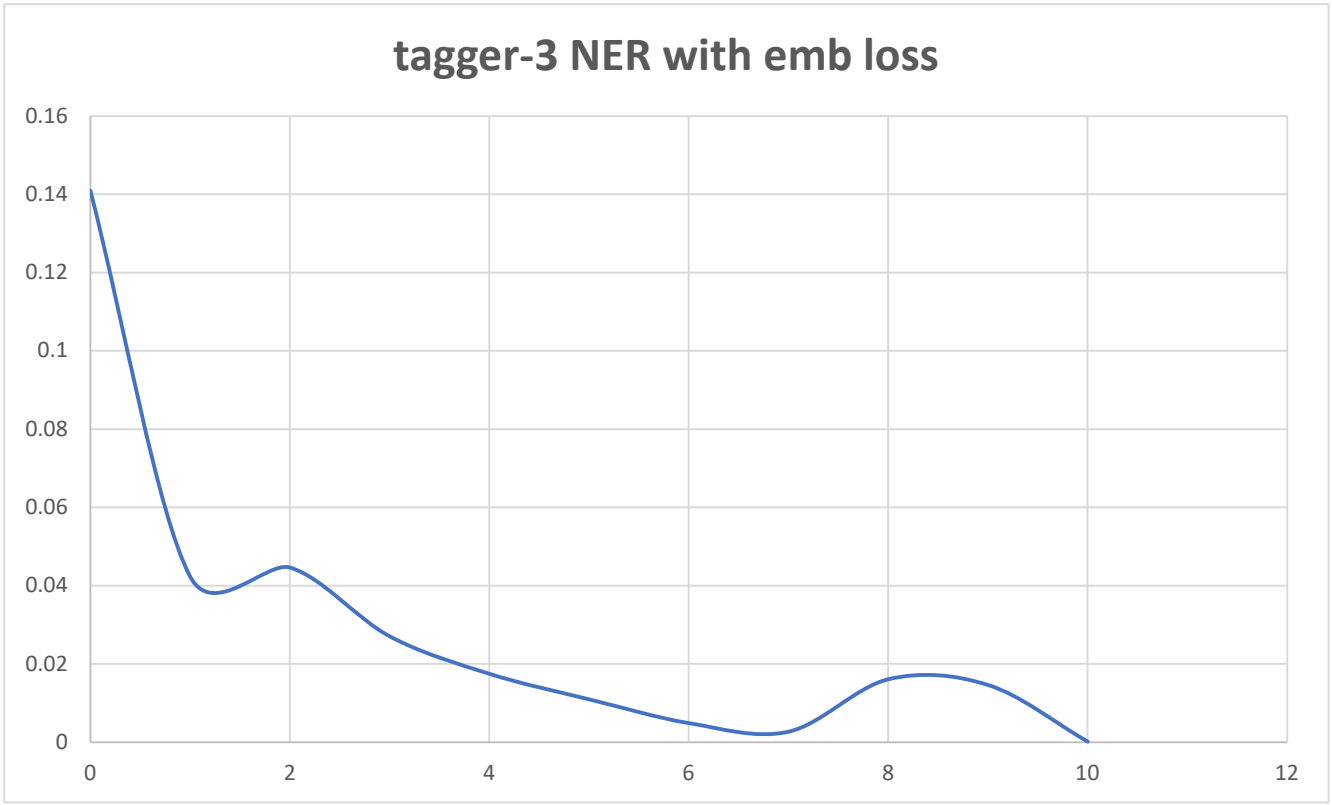


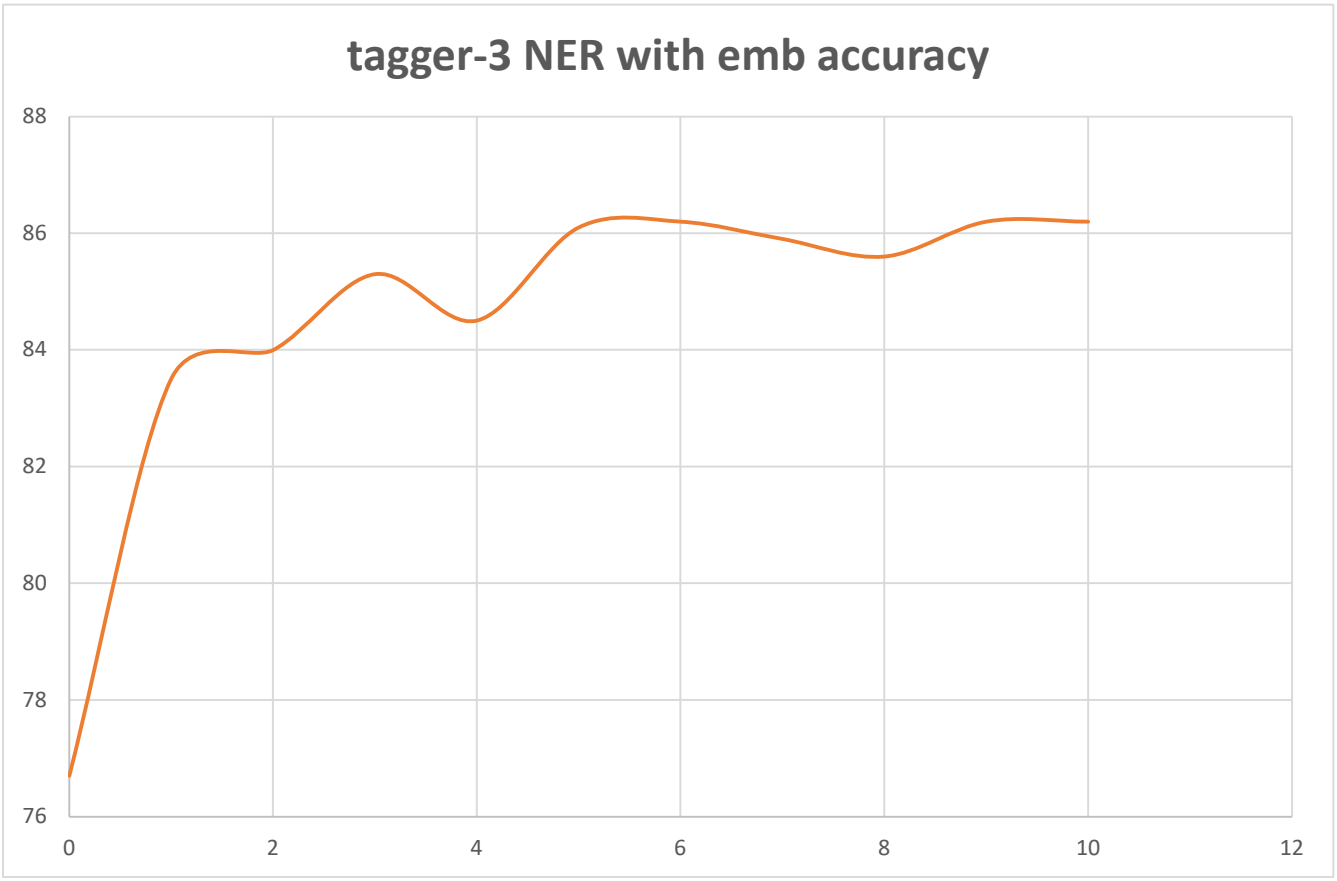

tagger-3 POS with emb loss
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
1
2
3
4
5
6


tagger-3 POS with emb accuracy
0.968
0.967
0.966
0.965
0.964
0.963
0.962
0.961
0.96
0.959
0
1
2
3
4
5
6


tagger-3 NER no emb loss
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
2
4
6
8
10
12
14
16


tagger-3 NER no emb accuracy
87
86
85
84
83
82
81
80
79
78
77
0
2
4
6
8
10
12
14
16


tagger-3 NER with emb loss
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
2
4
6
8
10
12


tagger-3 NER with emb accuracy
88
86
84
82
80
78
76
0
2
4
6
8
10
12