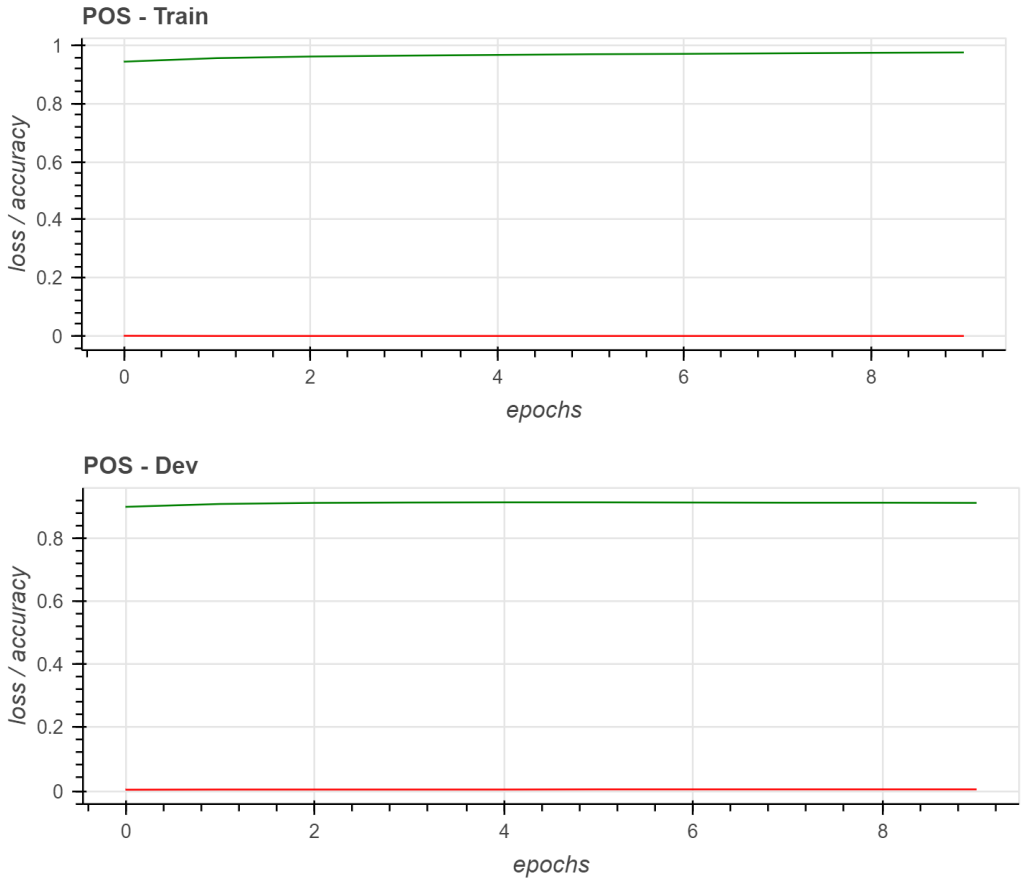
Dafna Magid 315635870 Oved Nagar 302824875

The parameters of the best model for each of the tasks (NER and POS):

	Model POS	**Model NER**
Optimizer	torch.optim.RMSprop	torch.optim.RMSprop
Batch size	512	512
Hidden dim	80	165
Learning rate	0.005	0.005
Number of epochs	10	20
Last epoch accuracy dev	86%	81%

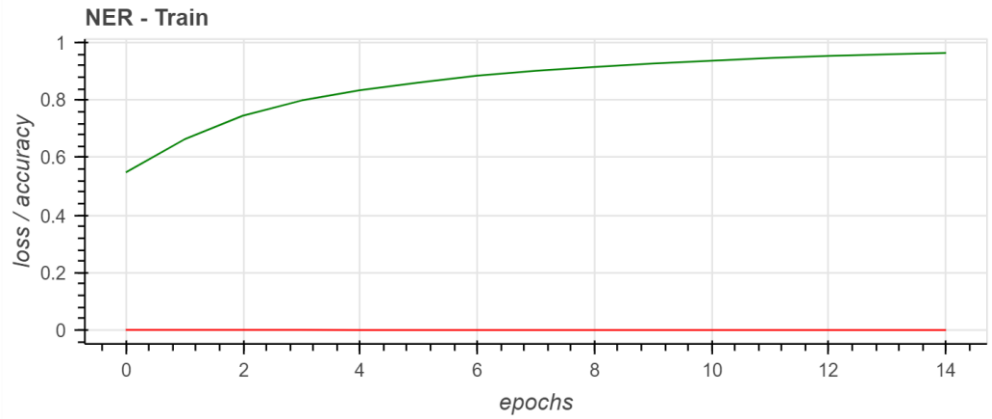
The graphs showing the **accuracy** of the dev set and the **loss** on the dev set as a function of the number of iterations:

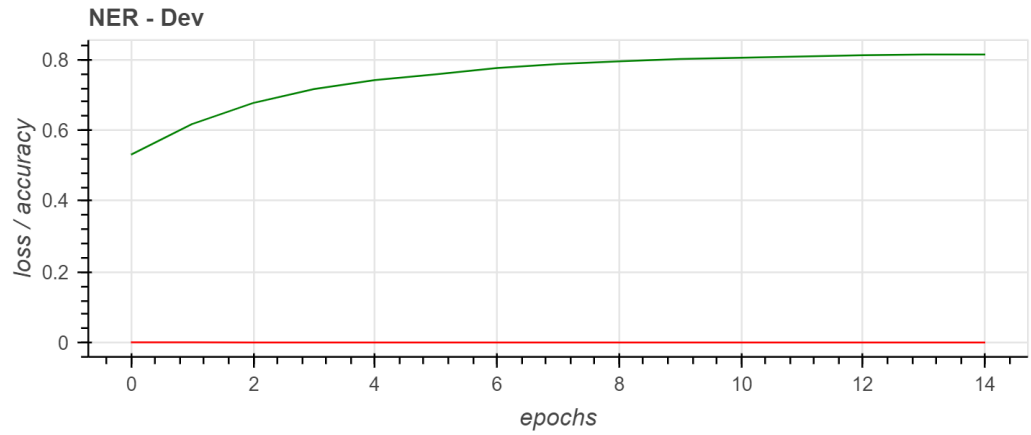
POS-



Dafna Magid 315635870 Oved Nagar 302824875

NER-





Accuracy calculation does not include the 'O' label at all!

description of the logic of using the pre-trained vectors:

If we take pre-trained embedding, there may be other information that is learned from another task that can be useful for our labeling mission.

There's might be a situation in which this information given in the pre-trained embedding is not accessible to us (via learning) because our mission learns something different but with some similarity.

A bad initialization of the pre-trained embedding is equivalent to a bad random initialization of our training, so the worst case is not that bad.

how do you handle lower casing, what do you do with words that are not in the pre-trained vocabulary?

Before we searched each word that needed to be found in the embedding matrix, we turned it into lower case.

It is clear to us that some of the information that can help us label the word is lost in this process.

Another possible solution is to add another embedding layer that signals the presence of a capital letter in the word and sum it with the representation of the word.