בס"ד

Yoel Jasner - 204380992
Steve Gutfreund – 342873791

Part 4

We had two options in mind about what to do with the vocabulary from the pre-trained embeddings (in case one runs our code with the appropriate flag on).
One option, for every word in the vocabulary, including those from the pre-trained, we added to the vocabulary its prefix and suffix, and gave them a separate embedding layer each.
Option two, only for the words from the training set we would do so.
Theoretically, and logically, the first isn't completely right, because for those prefixes and suffixes found in the external vocabulary which are not in the training set, we know for sure that their embedding layer will not be updated (and just stay with their random initialization which doesn't mean much), since during training process we wouldn't access them.
Although, we tried both, and didn't experience much of a difference (option 2 maybe a tiny bit better with max 1%).
We decided to go with option 2, which seems to us more correct.
For every word w in the train set, if the length of w is greater than 3 than we added two more words to the vocabulary; w's prefix and w's suffix. Note, there might be two different words with the same prefix/suffix, in such case the prefix/suffix is added only once. For each one of those prefixes and suffixes we added a corresponding embedding layer.
The whole embedding matrix is being trained and updated during process. When predicting a label, every word is split up into 3 vectors corresponding its prefix, suffix and itself.
By adding sub-words to the vocabulary, we are capable (theoretically) to say something (clever) about words we have never met.
For example, if we get the word 'preboil', even if we did not see this word before, the prefix 'pre' might tell us something, it might have something in common like the same prefix in the word 'preview' which is a word we have already encountered.

Parameters for NER:

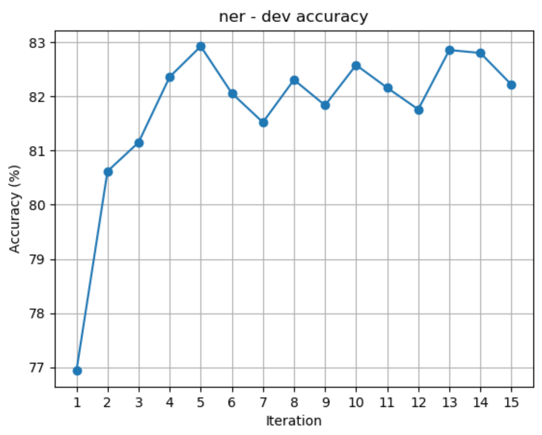
no pre-trained		pre-trained	
epochs:	15	epochs:	15
learning rate:	0.01	learning rate:	0.01
hidden-layer:	150	hidden-layer:	150

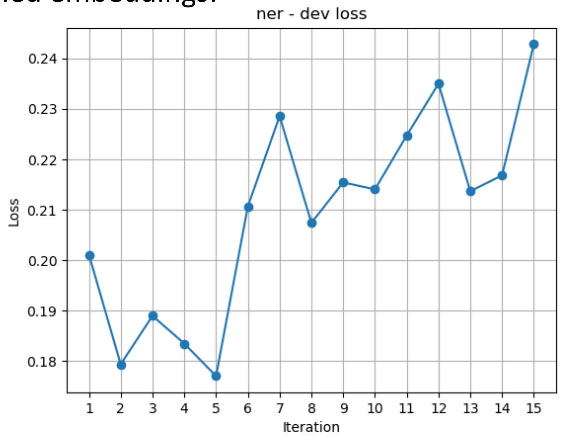
Parameters for POS:

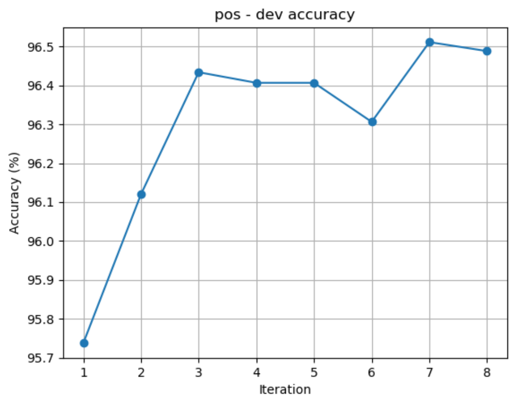
no pre-trained		pre-trained	
epochs:	8	epochs:	10
learning rate:	0.01	learning rate:	0.01
hidden-layer:	150	hidden-layer:	200

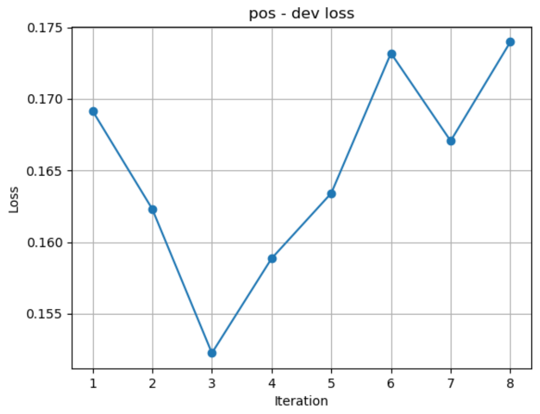
בס"ד

Results for using sub-words units, **without** pre-trained embeddings:

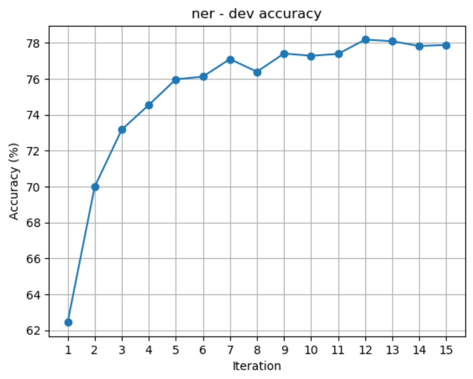


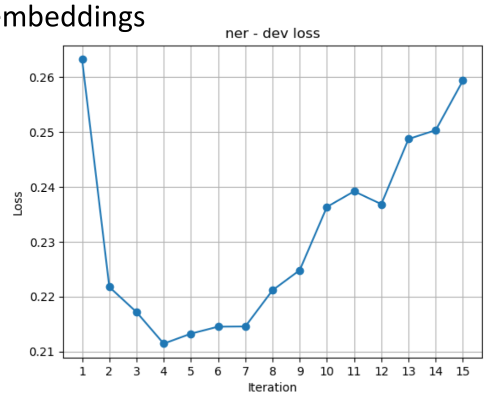


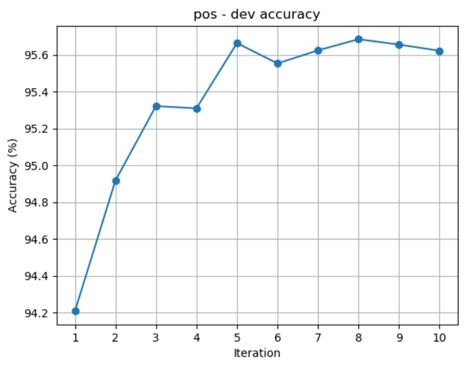


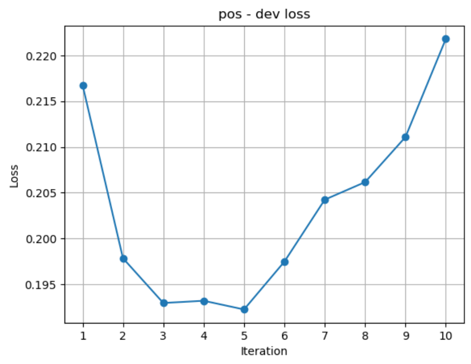


Results for using sub-words units, **with** pre-trained embeddings









Analysis:

When comparing the results we got on this part with the results of the previous parts (part1 and part3) we can see that using sub-words is increasing the accuracy.
From 81% on NER we achieve here almost 83% and from 95% on POS we get here 96%.
But, that's only when we are not using the pre-trained embeddings. When adding the pre-trained embeddings the accuracy drops by 3-4% on NER. On POS it's quite equal (the results above show a difference of 1% in favor of not using pre-trained embeddings).

(The reason that using pre-trained embedding while working with sub-words decreases the accuracy might be because the prefixes and suffixes are not pre-trained, they are 'randomly' initialized and updated during process. So, it might be that summing the three vectors for every word is not the best solution.)