Assignment 2: All the PDF files, in one.

Name: David Samuelson
ID: 208788851

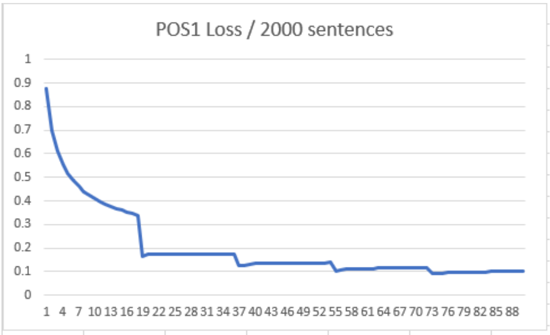
Part 1:

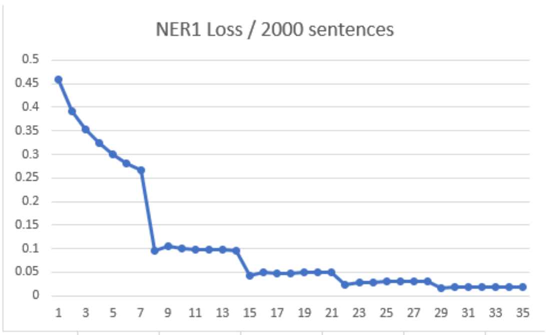
The only parameter I have really inputted was the hidden dimension size, and I chose 100. Used the Adam Trainer from the DyNet package, reached to dev set accuracy of 95-96%.

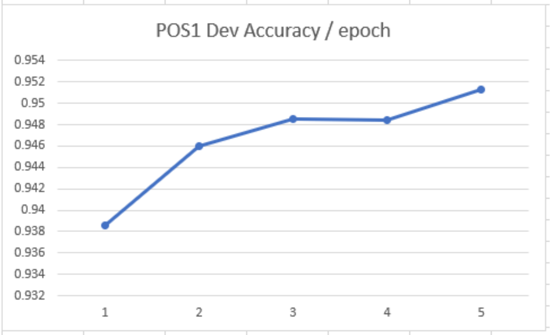
For words that didn't appear in the TRAIN dataset – I just gave them the the embedding vector of the 'UNK' word (unknown word). Once you add the UNKNOWN word to the tagset and wordset, you can just map every unknown word to the UNKNOWN word embedding. Results are still OK.

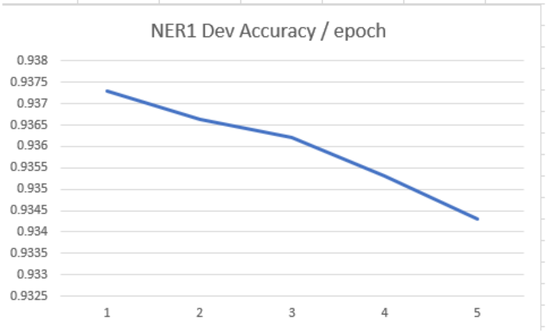
For the first and last words – I padded each sentence with two 'START' signs in the beginning, and two 'END' signs in the end, and added both start and end to the embedding layer, and to the tag set.
See it in the code – every tag set and word set are being extended to contain the 3 special signs, START END and UNK.
Every sentence is padded, so the first window would be 'START START w nw nnw', and the last window would be 'ppw pw w END END'.

Graphs:









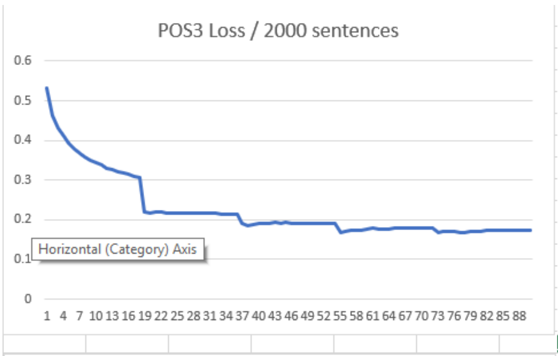
Part 2:

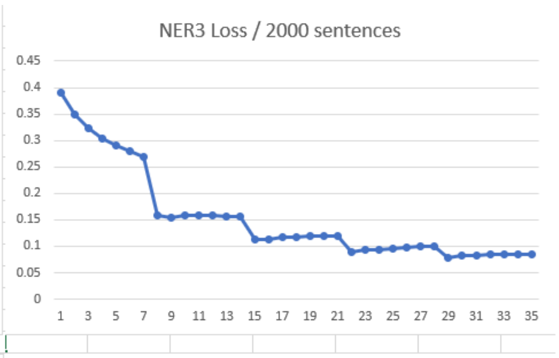
- dog:
    - dog
    - cat
    - rabbit
    - puppy
    - frog
    - kitten
- England
    - England
    - Ireland
    - Scotland
    - Australia
    - Wales
    - Europe
- John
    - John
    - George
    - Robert
    - Charles
    - William
    - James
- Explode
    - Explode
    - Grenadiers
    - Slashed
    - Appendix
    - Monnet
    - Instantaneously
- Office
    - Office
    - Board
    - Court
    - Offices
    - Commission
    - Authority

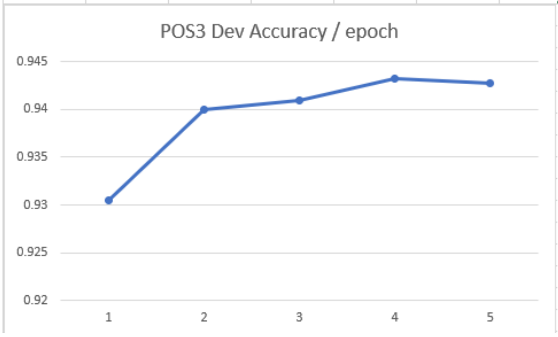
Part 3:

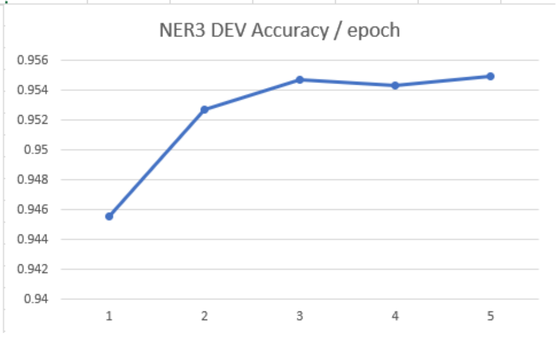
Same logic, but the difference is this – when I load pre-trained vectors, in which I know the vocabulary is lower case, I just lower case the train data when reading it. Same for DEV.

Words that are not in the pre-trained vocabulary are mapped to the UNK vector.









Part 4:

As before, I used the same parameters – using DyNet's AdamTrainer, not initializing it with learning rate, hidden dimension of size 100.

Choices I had to make when including sub words:

- I had to change the network itself, add 2 different lookup parameters, that is, 2 different embedding vectors Esuff and Epref for the suffixes and prefixes.
- Map every unseen suffix/prefix to the UNK vector
- Create two different vocabularies – for the prefixes and the suffixes.

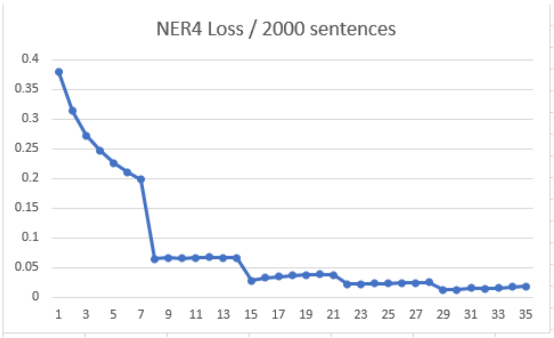
Brief Analysis:

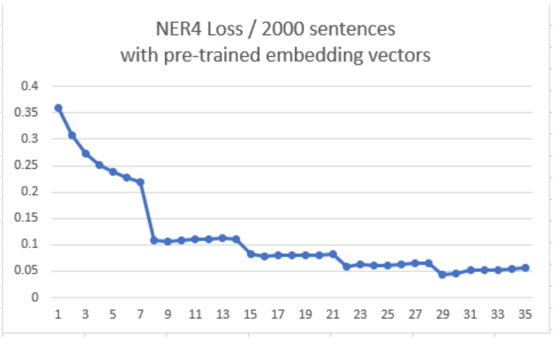
It seems that the sub-word units worked really good for the POS domain, by looking on the graph I can conclude that it could use more training.

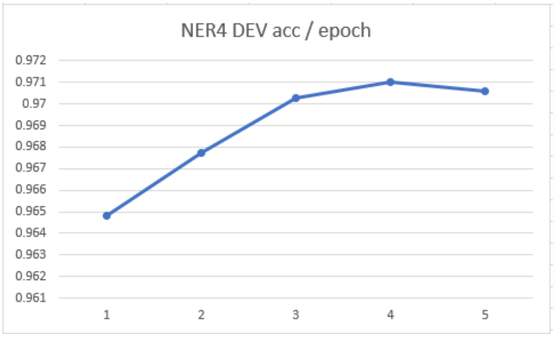
On the other hand, the combination of pre-trained word vectors and the sub-word units wasn't so good, as you can see in the graph. The 5th iteration caused it to become worse.

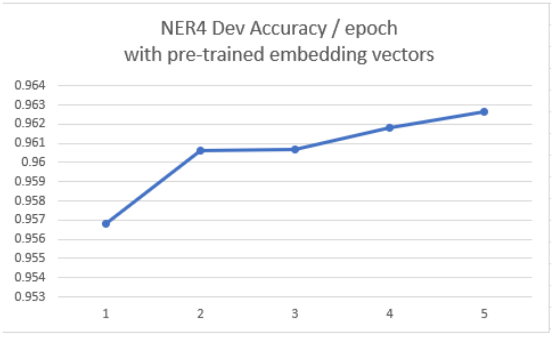
By far – the best result was the POS tagger with sub-word units, without pre-trained word vectors, reaching 96% accuracy on DEV, which is amazing.

NER graphs:

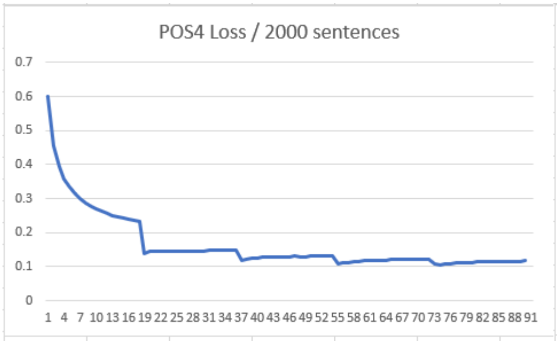


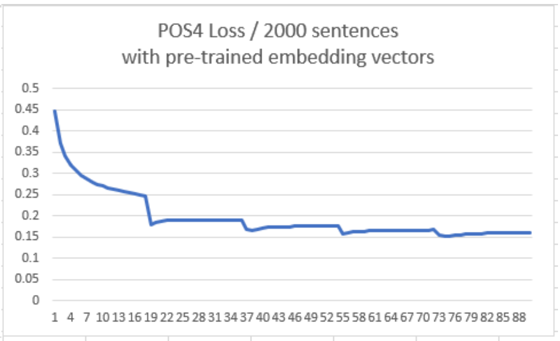


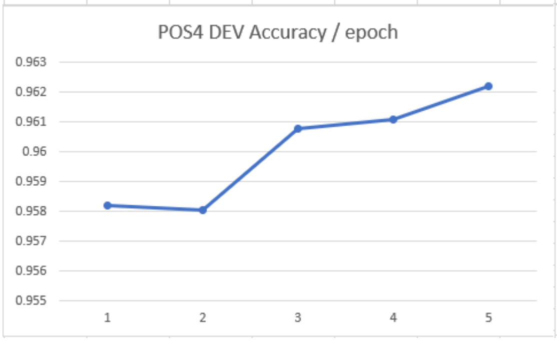


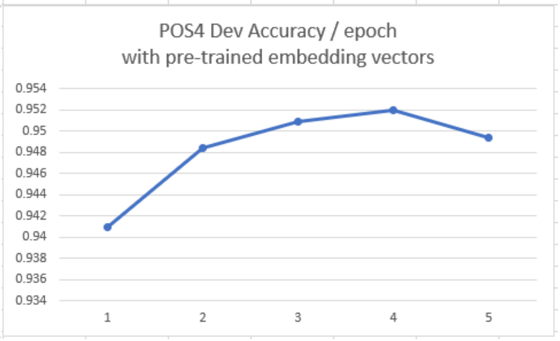


POS graphs:









I am sorry for lack of content in this document, I feel I could write more, but sleep deprivation in the last few days do its thing.. ☹

Anyway, The code is in decent condition, so it might overcome this.