<u>Part 1 Answers</u>

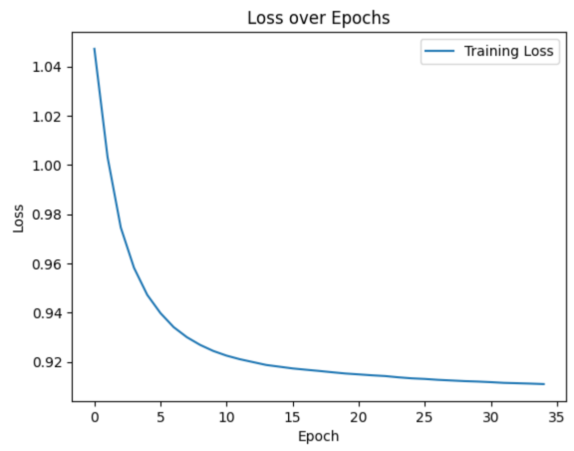
Roee esquire, ID 309840791
Yedidia Kfir, ID 209365188

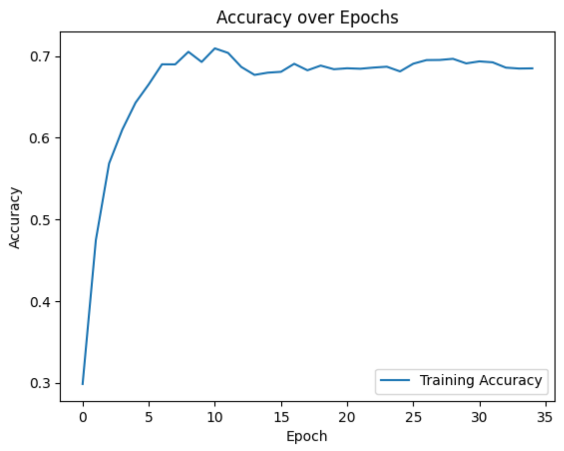
<u>Considerations</u>

1. For a word that is found in test/dev but is not found in train set, we allocated a *glorot* initialized vector for the unknown word. In this way we are using the learned embeddings of the surrounding words, under the assumption that it is likely that some of the surrounding words are found in dev.
2. For the first and second word, and the last and before last word in every sentence, we created padding pseudo-words, and for each an embedding as a vocabulary member. Concretely, let's say the first word of the sentence is in index 0 and the last word is in index n. we created words for index -1, -2, n+1, n+2. The first word in the sentence was padded with word -1 and word -2. the second word in the sentence was padded with word -1. The last word in the sentence was padded with word n+1 and n+2 and the second to last word in sentence was padded with word n+1. These padding words were treated as vocabulary words, with learned embeddings.

<u>NER Results</u>

The best parameters we found was using hidden dimension 40 and 0.001 learning rate. This yielded 0.709 accuracy rate on dev set after 10 epochs. See accuracy and loss function with hidden dimension 40:



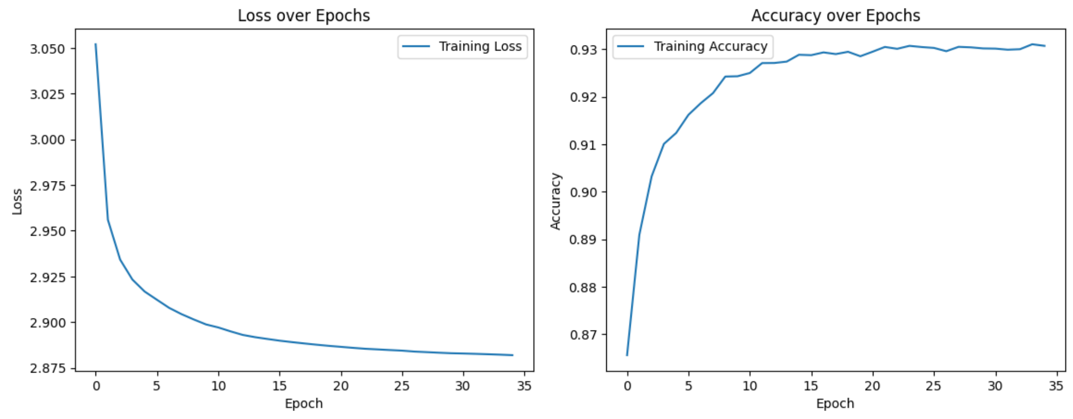


*notice this is accuracy on dev not on train as written in the graph

We also tested with hidden dimension 20, this resulted in peak accuracy of 0.6343071616960844 and hidden dimension 60 that peaked in accuracy of 0.6895493970806008.

<u>POS Results</u>

The best parameters we found was using hidden dimension 60 and 0.001 learning rate. This yielded 0.931069765499384 accuracy rate on dev set after 33 epochs. See accuracy and loss function with hidden dimension 60:



*notice this is accuracy on dev not on train as written in the graph

We also tested with hidden dimension 20, this resulted in peak accuracy of 0.9032972081271273 and hidden dimension 40 that peaked in accuracy of 0.8996846875065255.