NLP – 097215 – HW2 – NER – LSTM and FFNN

Hadar Sugarman 206567067, Tomer Grinberg 318155843

Model 1:
Train: For the first model, implemented a logistic regression classifier. We used 'Word2Vec-google-news-300' for the embeddings.
Test: Train F1: 0.634, Dev F1: 0.541

Global processing we performed for the following models:
In our exploration, we experimented with all possible pairs of word embedding models available in the Gensim library. After systematically evaluating the performance of each combination, we identified that concatenating 'glove-twitter-50' and 'Word2Vec-google-news-300' yielded the best results. We did this as we realized that OOV words are quite problematic, and we believe that this type of embedding may yield a more expressive and more informative vector for the model. As for handling out-of-vocabulary words, we initially experimented with two approaches: using zero vectors and using random vectors. After evaluating both methods, we determined that using zero vectors for out-of-vocabulary words produced better outcomes than using random vectors. In addition, we used a weighted loss function as the labels are extremely imbalanced with a bias towards non-entities. We used an Adam optimizer with betas=(0.9, 0.999) and eps=1e-08, and a ReduceLROnPlateau scheduler such that we reduce the learning rate by half if there is no improvement after 3 epochs.

Model 2:
Train: The FFNN we used has 3 fully connected layers, including a residual layer, with a hidden dimension of 128, with Tanh as an activation function, A learning rate of 0.008 and 5 epochs.
Test: Train F1: 0.344, Dev F1: 0.584

Model 3:
Train: The LSTM we implemented is a Bidirectional LSTM with one layer of hidden dim 128, a LayerNorm, 2 linear layers with Tanh activation and a dropout of 0.5, A learning rate of 0.008 and 10 epochs.
Test: Train F1: 0.423, Dev F1: 0.528

Comp Model:
*Train:*in addition to the processing we did for model 2 and 3, In order to capture structural features of words, we created a list of regex classes, and concatenated them into the embedding vector in the form of One-Hot-Encoded. Rigorous and Tedious Hyper parameter Tuning led us to choosing the following architecture:

We continued working on model 3, but instead of the final linear layers, we used model 2, without the dropout, as its performance was great. Thus we're left with a bidirectional LSTM with hidden dim of 128, LayerNorm, 3 fully connected layers with Tanh activation, including a residual layer, where the dimensions are reduced by a factor of 2 after each layer, a Learning Rate of 0.008 and about 5 - 10 epochs, depending on where the best F1 val score was achieved. Finally, we trained the comp model on both the train and the dev files.
Test: Hopefully 0.99999 :)

An illustration of us working on the task :)