NLP - Wet Exercise 1

Noam Wolf 318556206
Nir Diamant 304823560

Train

During the training, for each input of quadruples of consecutive 4 words (the two previous words the current and the next one) and their corresponding tags (ignoring the tag of the next word), analyzing which features it matches. The features we took under examination were all the features mentioned in [Ratnaparkhi, 96] ($f_{100} - f_{107}$), and the following:

$$f_1 = \begin{cases} 1 & \text{if previous word } w_{i-2}\text{= X and previous word } w_{i-1}\text{=Y and } t = Vt \\ 0 & \text{otherwise} \end{cases}$$

$$f_2 = \begin{cases} 1 & \text{for all words in corpus: if } word = W \text{ and } t = Vt \text{ and } w_{i-2} \neq W \text{ and previous word } w_{i-1} \neq W \\ 0 & \text{otherwise} \end{cases}$$

$$f_3 = \begin{cases} 1 & \text{if the word contains capital letter and } t = Vt \\ 0 & \text{otherwise} \end{cases}$$

$$f_4 = \begin{cases} 1 & \text{if previous word } w_{i-1}\text{= X and current word } w_i\text{=Y and } t = Vt \\ 0 & \text{otherwise} \end{cases}$$

To better eliminate poor features that might lead the model to over-fit on the training data, we checked different appearance threshold levels (leaving all the features that were below the thresholds out of the model). To search for the threshold feasibly we use the same threshold for all of the features. We also both constant and tested relative thresholds (i.e. taking only the feature above the q quantile of the number of appearances from each group) however for the thresholds that worked best the quantiles of all the feature where almost identical so we chose to use the nonrelative threshold described earlier.
Furthermore, we tested the effect of each feature group on the model (by inactivating feature groups and testing the model) and used only features with positive effects. We also conducted a grid search over the parameters: λ (regularization term), the threshold, beamwidth (see Inference) - to maximize the model accuracy performance over an unseen new text.

the final model for competition number 1 (trained on train1.wtag) contains the following parameters: $\lambda = 0.35$, thresholds are 0 for all features, beam width = 2. the chosen features and their amounts:
$f_{100} : 15395,\ f_{101} : 9904,\ f_{102} : 18954,\ f_{103} : 7610,\ f_{104} : 1018,\ f_{105} : 44,\ f_{106} : 30068,\ f_{107} : 30744,\ f_3 : 42$

the final model for competition number 2 (trained on train2.wtag) contains the following parameters: $\lambda = 0.1$, thresholds are 2 for all features, beamwidth = 2. the chosen features and their amounts: $f_{100} : 1804,\ f_{101} : 1871,\ f_{102} : 3282,\ f_{104} : 292,\ f_{105} : 31,\ f_{106} : 2616,\ f_{107} : 2683,\ f_3 : 18$
The training on train1.wtag took a total of 112.12 seconds (with the optimization itself taking 73.88) and the training on the small model took 2.00 (while the optimization taking 0.96) with an i5-8265 1.60GHz CPU and 8GB 2400MHz RAM.
We also tried using ensemble approaches during training, we trained multiple models on a bootstrap sample of the training data and/or feature samples and aggregate the results during the training by using the mean of the weights or during the inference (see Inference). The reason for the change was to introduce a new way of regularization to the model and to increase the predictive power of the model without overfitting. These approaches were *not used* in

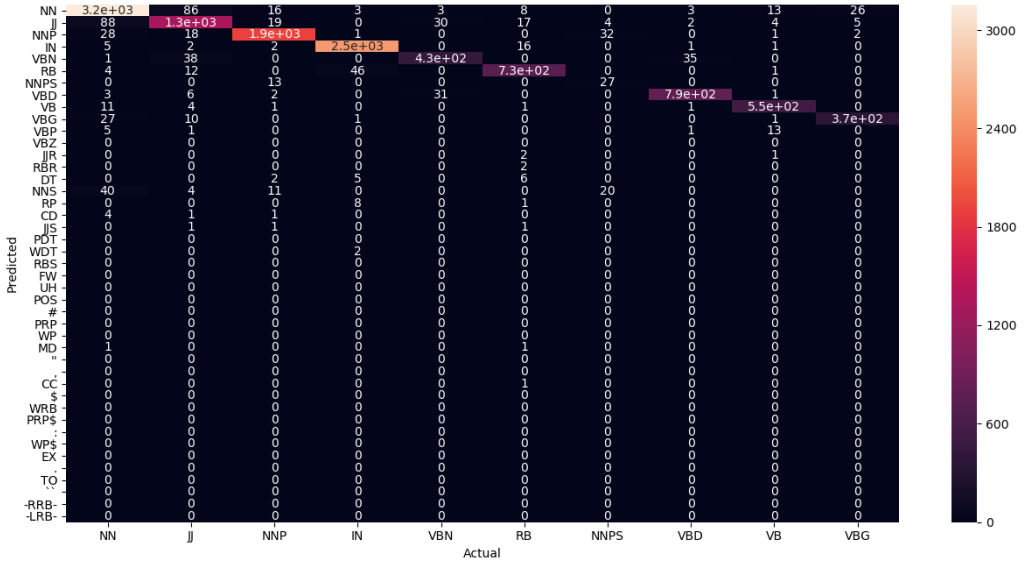
the final model because they provided similar results with a significant increase in the training time. We also tried l_1 regularization since it should generate sparser weights (feature selection during optimization) and uses a smaller number of optimization iterations and then choosing only features with high enough absolute weights. Both of these changes were not used as well due to a lack of contribution to accuracy improvement (finding the first one also being problematic during optimization, due to not being differentiable).

Inference

The first change we introduced was the beam search heuristic i.e. at each time step passing only the *beamwidth* combination of current and previous tags with the highest probability instead of all the possible combinations. The purpose of this change is to improve the inference running time: testing on the small training file, the running time of the feature extraction, and the probability matrix computation were $14\frac{1}{2}$ minutes seconds and the Viterbi logic took only 1 second. Indeed our model performed best with $beamwidth = 2$ which reduces the total running time of the inference on the small file from $14\frac{1}{2}$ minutes to 10 seconds (we also believe using beam search added minor regularization differently from the l_2 regularization and for that reason, $beamwidth = 2$ performed slightly better in term of accuracy).
Later, we tried to introduce the aggregation of multiple models during the inference using the mean of the q matrix of the model or the majority voting on the tags (we also tried to give more weight for more trusted models). As said earlier, the purpose of these changes is to introduce a new way of regularization of the model to increase the predictive power of the model without overfitting. However, we did not use these changes at our final model as they produced similar results with a significant increase in training time and inference time.

Test

Testing our big model on the test1. wtag file yielded 95.75% accuracy while the tagging process took 45.62 seconds (including inference).



For the confusion matrix, we chose the ten tags our model was most confused upon by the number of errors in the classification of that tag. The reason we chose this metric is the fact that we are measured by the accuracy of our model making the errors shown in the confusion matrix to be the most significant to our model score.

We will propose a solution the deal with the confusion between NN (noun) and JJ (adjective). We believe that the confusion accrues because our model does not have enough information of the next tag and most importantly if it is a noun class or does not (f_{107} contain some information. However, it will not help for unseen cases in the training or those who were seen only a small amount of times). For example in the sentence: "the_DT *giant_NN* closely_RB held_VBD..." (test1.wtag, line 28) without knowing the next word, "giant" is more likely to be an adjective as it is used more times that way (as our model predicted). However, knowing that the next word is not a noun would allow predicting that "giant" is a noun.
Therefore we propose the following pair of features:

$$f_4 = \begin{cases} 1 & \text{if the current tag is JJ and next word most frequent tag is not in \{NN, NNS, NNP, NNPS\}} \\ 0 & \text{otherwise} \end{cases}$$

$$f_5 = \begin{cases} 1 & \text{if the current tag is JJ and next word most frequent tag is in \{NN, NNS, NNP, NNPS\}} \\ 0 & \text{otherwise} \end{cases}$$

These features use a unigram model to approximate if the next word should be a noun an either decreasing or increasing the probability of adjective based on the result.
For the small model, we started by trying similarly to the big training file and its test to test the small model and our results were between $62\% - 65\%$, however (due to counter-intuitive results) we decided to look into files and discovered that the files are from different domains (the small file and its competition are derived from medical researches and the big training file, its test, and competition are from more general domains). Therefore we decided to use 5-fold cross-validation for the testing (dividing the training file into five equally-length sets of sentences and then for training on 80 percent and testing on the remaining 20 percent at a time, returning the mean of the results).

Competition

Our model for both competitions is the model described above.
For the big model, we used the accuracy on the test1 for as our approximation for the approximation on the competition accuracy, which is **95.75**%. There might be a difference between the accuracy on the competition and the test because we determined the hyperparameters based on the accuracy on the test, what might make some bias to our estimation (the accuracy on the test should be slightly higher if it is indeed representative since we chose the optimal hyperparameters for it). To check how much our results may vary, we divided the test to ten segments (without shuffling - to make the segments less dependent, since the test seems to have connected segments) and computed the accuracy on each of them with the maximum accuracy being 96.8% and the minimal being 94.5% so we believe our accuracy on the competition would likely be between the two.
For the small model, we used 5-fold cross-validation. Our estimation error margin might be different from the results on the competition file because in cross-validation we compute the accuracy for smaller and different training samples (and the fitting of the hyper-parameters that were discussed earlier). Our estimate for the accuracy of the competition (the mean accuracy of the folds) is **93.70**%. We also looked at maximum and the minimum accuracy of the folds to see how the result may vary, with the maximum accuracy 95.65%, and the minimum being 91.81%.

Work Partition

Nir worked on the feature implementation (both the demanded and the new ones) their extraction, the tagging of the file (given the predicted tags) and the parameter search, while Noam worked on the optimization, the inference and the changes that were tried in both of those sections (the beam search, the ensembling, etc.) We also discussed together our ideas of implementation, improvements, and complicated bugs and weird behaviors of our codes.