Homework 1 - CptS 577 Structured Prediction, Spring 2019
Abhijay Ghildyal - 11632196

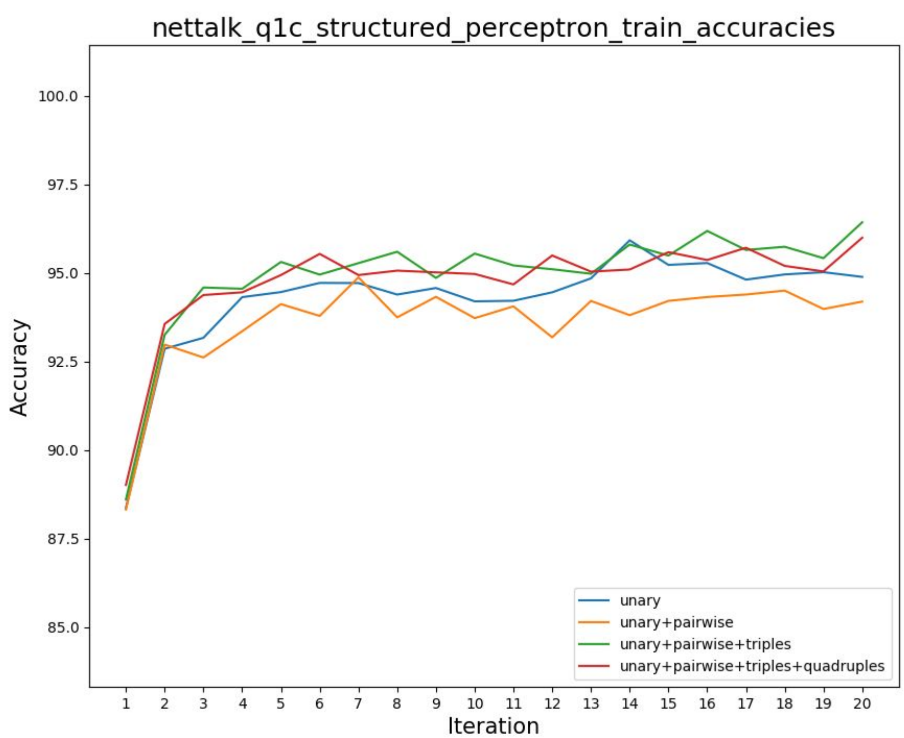
Note: In RGS the way I have implemented it is that it performs one label change at each position and chooses the best. It keeps performing this in a loop until the change does improve from the best score seen in the last pass. Once done it chooses another random start and performs the same as aforementioned.

One of the optimizations I applied while training was that since we know the target labels I use a reduced size weight vector and feature
Eg. ‘mask = self.features_phi > 0’ where self.features_phi are the features for x with known y.
Then I use ‘w_ = w[mask]’ and ‘features_phi[mask]’ (features for x with new predicted y) for the calculation

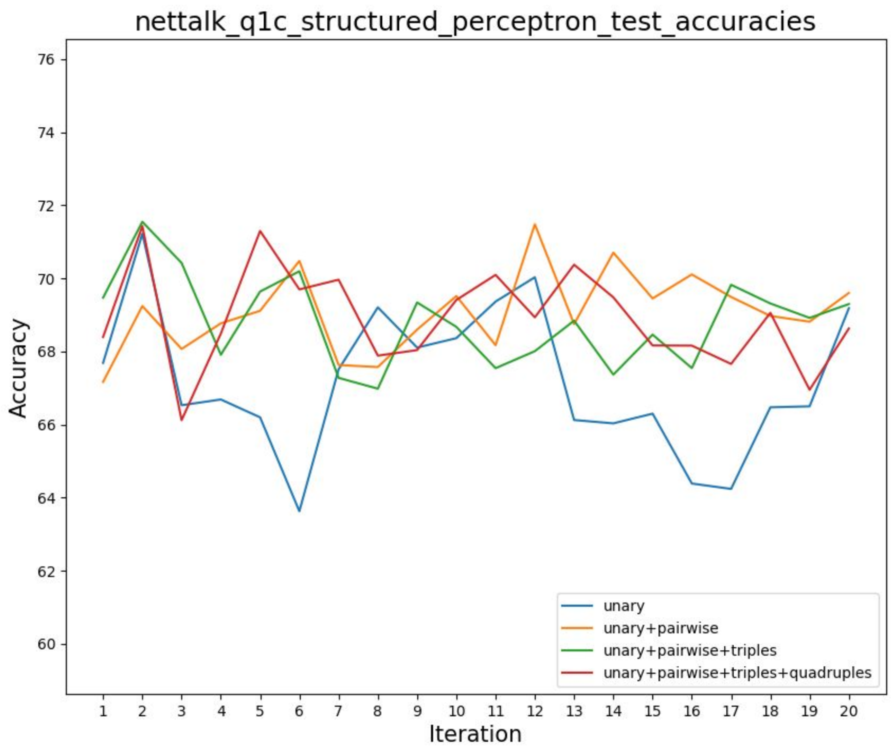
1e) To diagnose the performance of the learning algorithm hamming accuracy is a good measure for comparison of accuracies of models, created at each iteration. Though it is necessary to check for the increase in accuracy (especially on the test data) over several iterations, as more number of updates to the weights would help the algorithm to converge or generalize better. A possible improvement for generalization to the current setup of structured perceptron can be taking a weighted average (based on the performance or accuracy on validation data) of the weights obtained at each iteration.

Nettalk Data

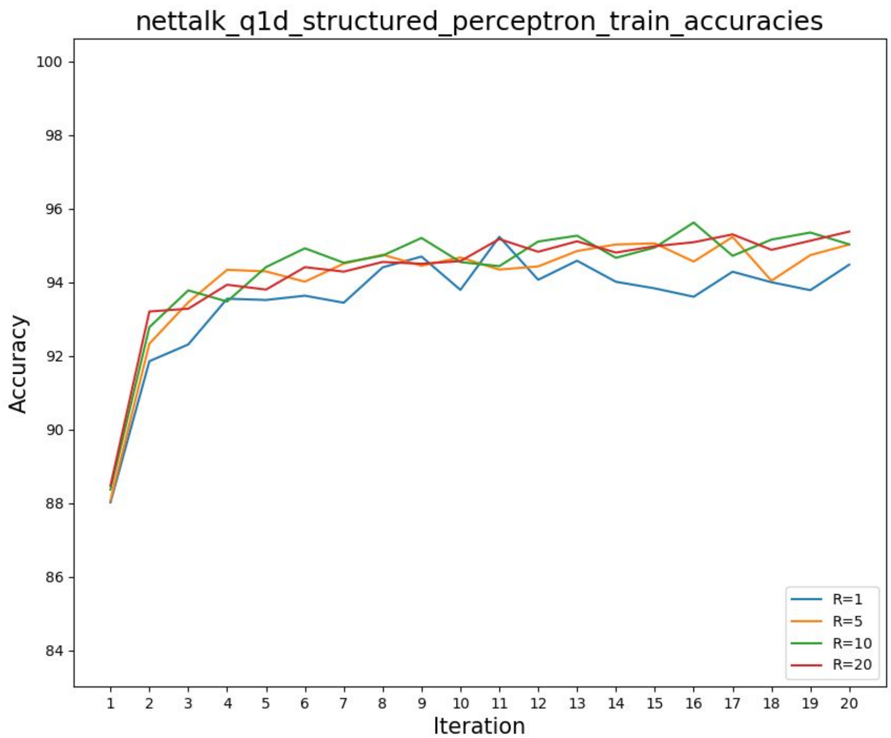
The first plot is training accuracy of the structure perceptron for the setup mentioned in Q1c i.e. plot the hamming accuracy with MAX=20, η = 0.01, R = 10 and varying feature representations



The model learns faster with triples and quadruples included in the feature set and show signs of superior performance.

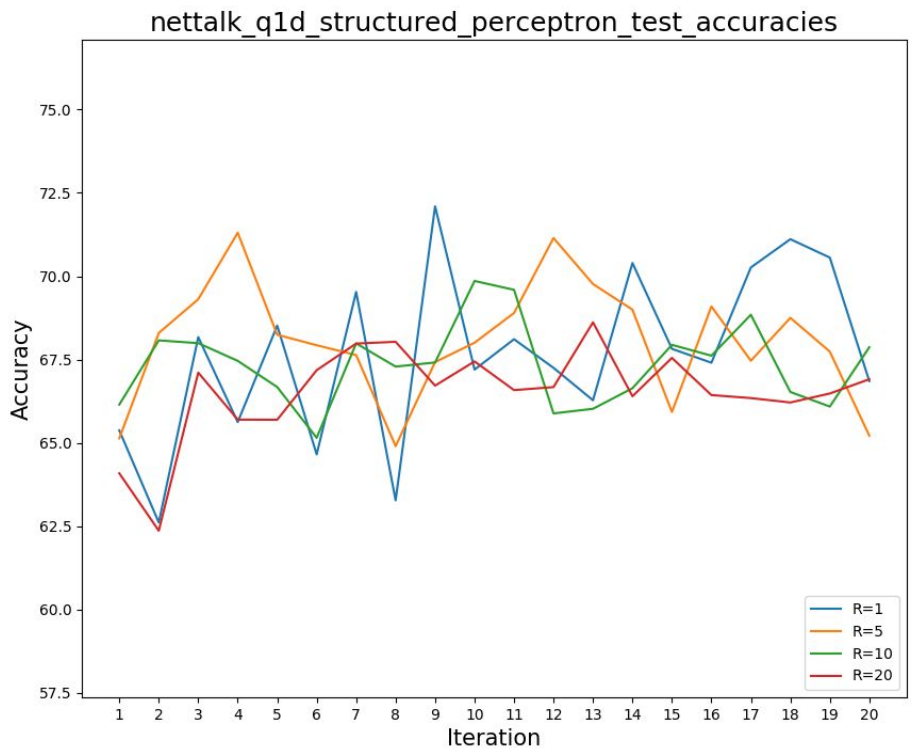


Looking at the test accuracies for the setup mentioned above it is evident that a larger feature set is helping the algorithms to perform better though with more iterations the models seem to have overfit and the performance on the test set is ideal with the model learnt at iteration 2. Though the test accuracies are not as high as training accuracies.



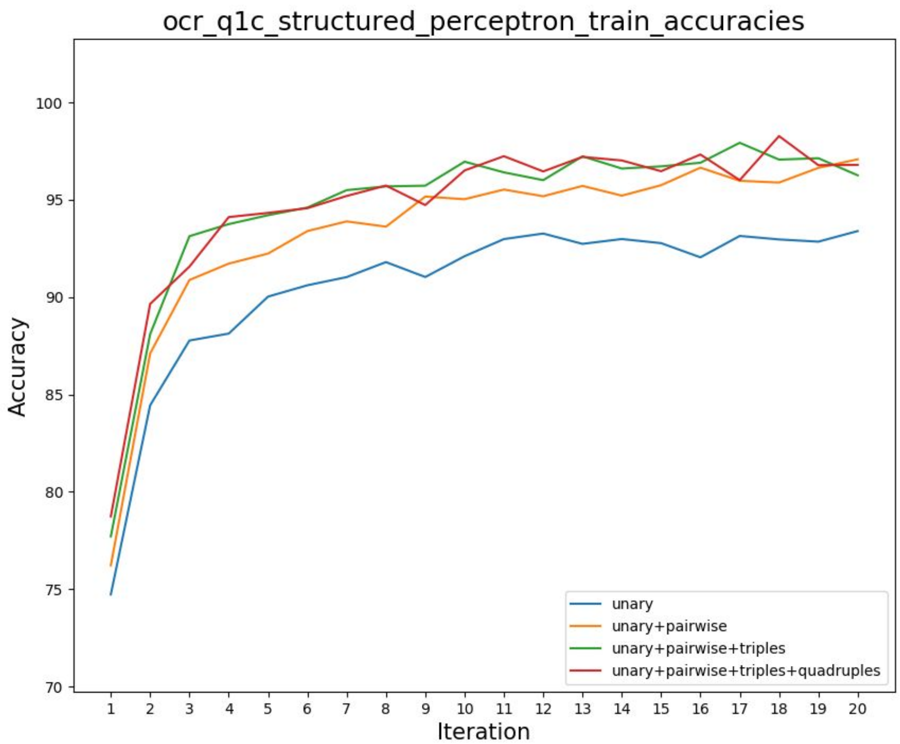
The above plot shows the training accuracies for the setup given in Q 1d, with MAX=20, learning rate η = 0.01, feature representation set to first order i.e. unary + pairwise features and varying number of restarts R=1, 5, 10, 20

From this plot it can be seen that with more number of restarts the model can learn find more optimal solutions during train time

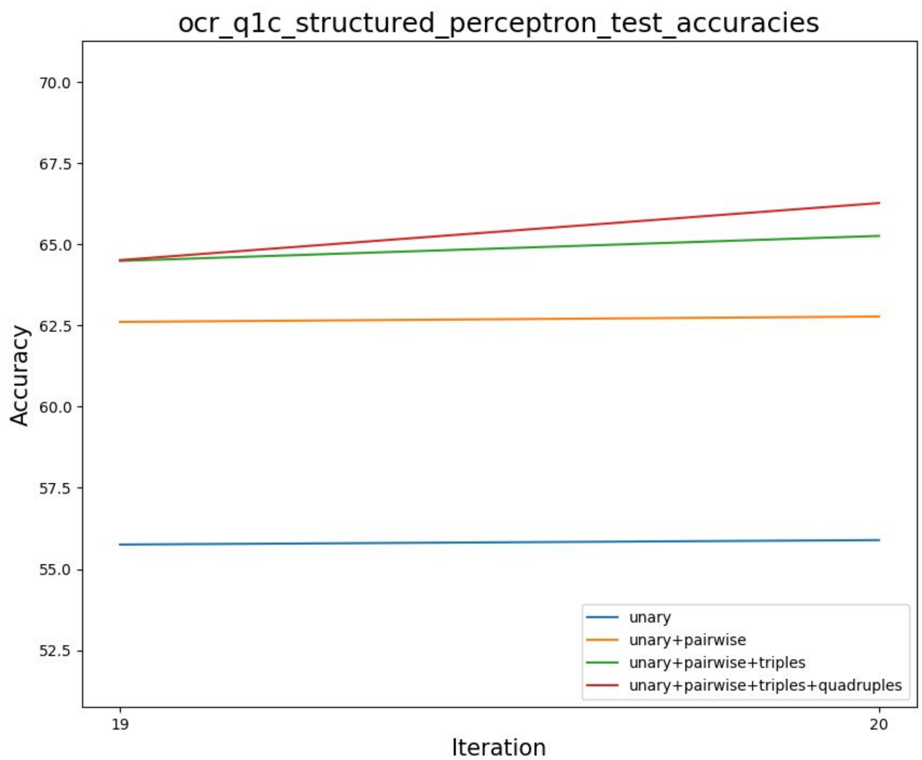


Though in the test accuracies the same as mentioned cannot be stated for this data. With just one restart an optimal solution can be found if the feature set is of first order and learning rate while training was 0.01. Again the test accuracies are not as high as the train accuracies.

OCR Data



This is the setup mentioned in Q 1c. This plot shows that the model is learning well and improves as more features are added.

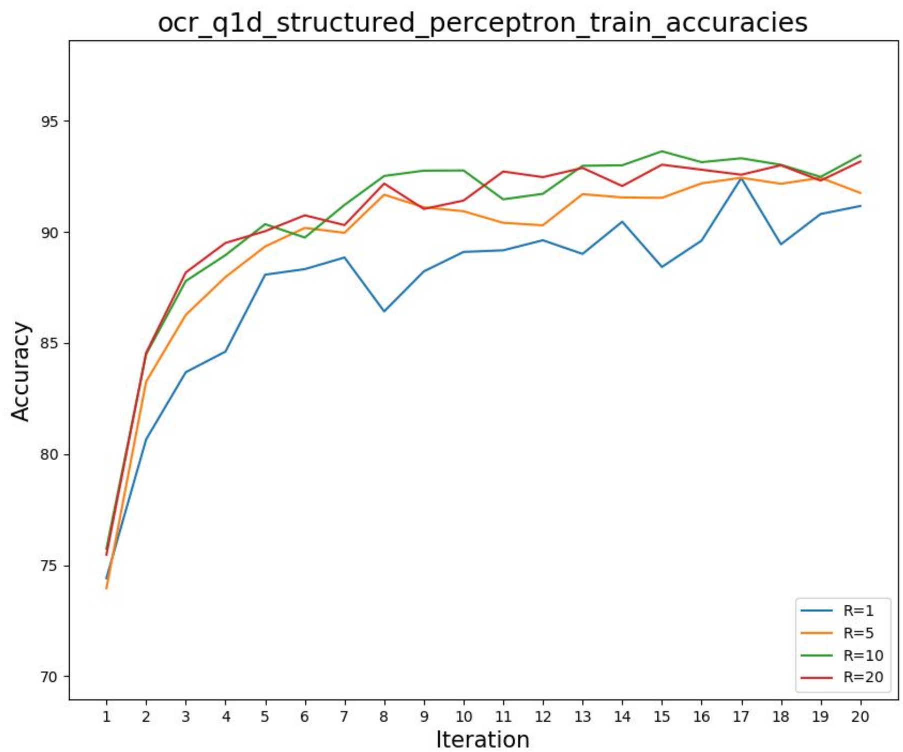


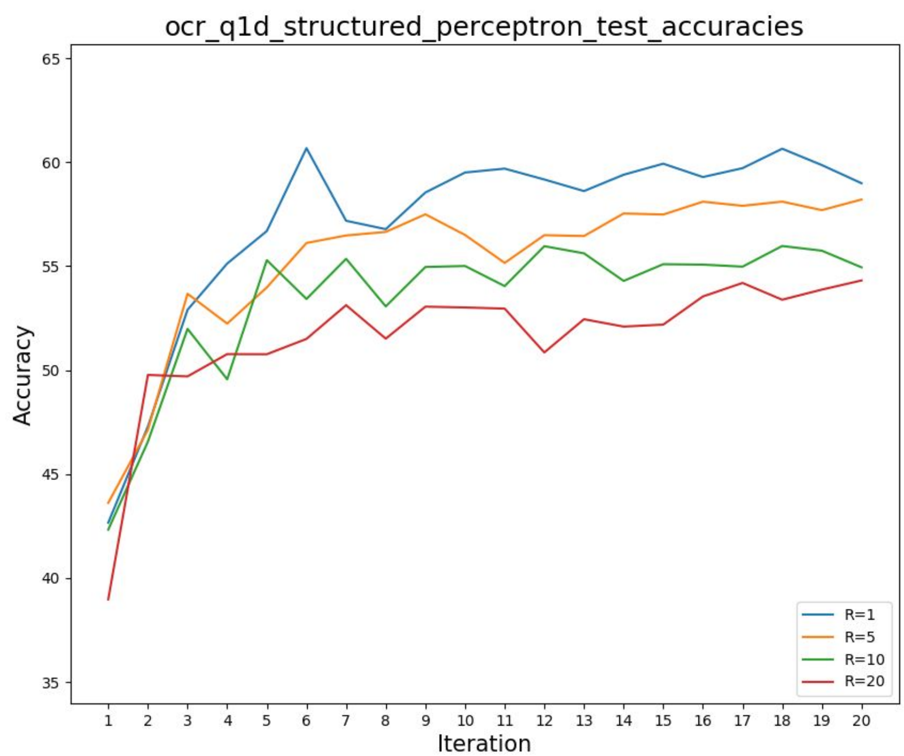
I tried to test the models for all iterations but it was taking a lot of time during the test runs (for triples and quadruples features) and so decided to only plot the results for R=1 (with R=10 the accuracies were coming to be about similar, checked for unary and first order feature representation, mentioned in the log files) and iteration 19-20.

From the plot it can be seen that for the models learnt at iteration 19 and 20, more number of features used for training show superior performance during test.

Q_1c

Experiment: unary Iteration: 1, Accuracy: 0.4177355403799173 Iteration: 2, Accuracy: 0.4827614594651869 Iteration: 3, Accuracy: 0.5325767493140714 Iteration: 4, Accuracy: 0.5220340915949618 Iteration: 5, Accuracy: 0.5337729557454401 Iteration: 6, Accuracy: 0.563773078802674 Iteration: 7, Accuracy: 0.5327734371247409 Iteration: 8, Accuracy: 0.5216408922136005 Iteration: 9, Accuracy: 0.5415641234110279 Iteration: 10, Accuracy: 0.5617576001914507 Iteration: 11, Accuracy: 0.5419533642737929 Iteration: 12, Accuracy: 0.5482877988717054 Iteration: 13, Accuracy: 0.5366245918357581 Iteration: 14, Accuracy: 0.5429657293239121 Iteration: 15, Accuracy: 0.5482283434051151 Iteration: 16, Accuracy: 0.5446983338920629 Iteration: 17, Accuracy: 0.5546995260201147 Iteration: 18, Accuracy: 0.547916016864985 Iteration: 19, Accuracy: 0.560615216650251 Iteration: 20, Accuracy: 0.5581028900177837	Experiment: unary+pairwise Iteration: 1, Accuracy: 0.4113276976138918 Iteration: 2, Accuracy: 0.5209439878452836 Iteration: 3, Accuracy: 0.5136675936268001 Iteration: 4, Accuracy: 0.512510805463933 Iteration: 5, Accuracy: 0.5464266474384855 Iteration: 6, Accuracy: 0.5958613339592382 Iteration: 7, Accuracy: 0.589639020838029 Iteration: 8, Accuracy: 0.6032690979899427 Iteration: 9, Accuracy: 0.6128588860791708 Iteration: 10, Accuracy: 0.6197784720301117 Iteration: 11, Accuracy: 0.635340489599808 Iteration: 12, Accuracy: 0.6355039510694606 Iteration: 13, Accuracy: 0.6104040096841249 Iteration: 14, Accuracy: 0.6251497808968614 Iteration: 15, Accuracy: 0.6330726215784607 Iteration: 16, Accuracy: 0.6301709049229446 Iteration: 17, Accuracy: 0.6375616904292316 Iteration: 18, Accuracy: 0.6363567297097932 Iteration: 19, Accuracy: 0.6273069653288817 Iteration: 20, Accuracy: 0.6292569412670196





From the training plot it can be seen that with the setup given in Q 1d on OCR data, structured perceptron is able to learn well. Though from the test plot it is evident with R=1 best performance is seen at every iteration.