Ayça Avcı

(s4505972)

October 4, 2021

Assignment 4

NN - 1:

Majority class baseline scores for “person”, “cardinal”, “date”, “gpe” and “org” are 25.7%, 14.9%, 8%, 18% and 33.5% respectively. Looking at the baseline scores, we can say that data is slightly imbalanced. Considering my final model had 77.13% validation accuracy and 76.5% test set accuracy, and majority class scored as 33.5%, model accuracy is considered as good (since model accuracy greater that 33.5%).

NN - 2:

Dropout is a regularization technique that where randomly selected neurons are ignored during training, they are set to 0 with a frequency of rate at each step during training time, which helps prevent over-fitting. This means that their contribution to the activation of neurons in the next layer is temporally removed on the forward pass. Any weight updates are not applied to the neuron on the backward pass. I used Dropout layer with 0.1 rate after applying hidden layer with 32 units (in lecture pdf, it was suggested to use it after hidden layers, so I followed the suggestion). Without the hidden and Dropout layer, accuracy was 77.2%. After adding the hidden layer and Dropout layer afterwards, accuracy decreased to 76.5%. Hence, model performance decreased slightly.

NN - 3:

To stop training, I set epochs to 50. During training, the model is evaluated on a holdout validation data set after each epoch. If the performance of the model on the validation data set starts to decrease (e.g. loss begins to increase or accuracy begins to decrease), then the training process should stop. I tried different epochs to test this out. I tried 10, 30, 50, 100, and 200 respectively. With epochs 100 and 200, loss started to increase after a certain amount of iteration completed before reaching 100 and 200, and accuracy started to decrease at the same time. For 10, 30 and 50, there were no increase in the loss (both training and validation loss) during the training, so I looked at the test set accuracy values. For 10, 30 and 50 epochs, model accuracy on test set were 71.3%, 76.5%, and 77.2%. Since epoch 50 provided the highest accuracy on test set, I used it in my model. I set the epoch manually, but it can be done automatically as well by setting up a stopping criterion. For example, in each epoch, we can calculate the change in the weight vector. If change is below a certain threshold value, meaning that weights are now changing very slightly, we can stop training to prevent over-fitting. There are many regularization techniques that can be used for early stopping to prevent over-fitting: weight decay, L2 etc.

NN - 4:

I experiment with changing the learning rate, epochs, batch size, optimizer, activation function and including/excluding Dropout layer(s). I realized that lowering learning rate slightly (0.005 to 0.004) and lowering the batch size (32 to 16) with increased epoch numbers (10 to 50) increases the model accuracy with 9.3% (67.9% to 77.2%). Adding hidden layers with Dropout layers decreased the accuracy between 1%-3% applying different settings (multiple hidden layers with different units + one Dropout layer, multiple hidden layers with different units + Dropout layer after each hidden layer, trying different hidden units and dropout rate etc.) Changing optimizer (sgd to adam, adagrad, adamax) cause a slight drop in the test set accuracy (73.1%, 76.2% and 75.4% accuracy on test set for adam, adagrad and adamax respectively). I tried different activation functions (instead of softmax, I tried sigmoid, tanh, relu and selu). Using sigmoid function did not change the test set accuracy (77.2%). Using tanh, relu and selu, test set accuracy dropped gradually to 29.1%, 14.5% and 39.5% respectively. I tried all the values manually, passing the parameters as an argument. I tried to use GridSearchCV package from scikit-learn, but it took forever to run all the parameter combinations, so, I quit using it. It would be nice to have a cluster access to experiment with it since my computer is not that powerful.

NN - 5:

In my final model, I set the learning rate, activation function, loss function and optimizer to 0.004, softmax, categorical cross entropy and sgd respectively. Epoch number is set to 50 and batch size is set to 16. I have not used any hidden layers and Dropout layers due to accuracy drop that is explained in NN-2 and NN-4. Other settings remain unchanged compared to initially provided model. All the decisions are already discussed in NN-2, NN-3 and NN-4. Final test set accuracy for this setting is 77.2%.

NN - 6:

Out of 732 data points, my model labeled 172 data points wrong. For “person”, “cardinal”, “date”, “gpe” and “org”, there were 183, 106, 58, 136 and 249 data points on the test set corresponding to the labels respectively. 46, 7, 14, 29 and 66 of them labeled wrong respectively. The interesting thing is, in training set, there were 1225 and 290 data points that have the labels “org” and “date” respectively. When we look at the test accuracy for each class, they are 73.5% and 75.9% respectively. I found it very interesting that my model can guess the label “date” with slightly higher accuracy than “org” since there are very few data points that represent “date” during the training. Another interesting thing was label “cardinal” have 93.4% test accuracy. When we look at the distribution in training set, label “cardinal” has similar distribution (544 data points for “cardinal” and 657 data points for “gpe”) with label “gpe”, but label “gpe” has 71.3% test accuracy. Since we are using embeddings to represent each word, words might have very similar vector representations both in training and test data set for the "cardinal”” label for instance, so that even there are less data points in the training set, test set gives higher accuracy for label “cardinal” than other labels. Same situation may apply for “date” and “org” example that is mentioned above.

WE – 1:

I tried following words in distance tool that have multiple meanings:

letter → there were no words related with the alphabet.

live → there were words that are both related with “watching something live” and “living the life”.

right → most words are related with the politics. “left” word occurred as well which is the opposite of “right”. This case is interesting since as direction, they have opposite meanings. In my opinion, in this case left refers to the political left. There are also words that is related with religion. Additionally, “wrong” word occurred, which is again interesting because they have opposite meanings in the right/wrong concept considering tool is catching the similarities.

capital → Mostly political and economy related words occurred. There were no capital names of the country.

watch → All word was related with the watching action, not with the watch that we wear.

I realized that for the words that have multiple meanings, toll was not successful to detect every meaning in the tool.

WE - 2:

I tried the following analogies in the analogy tool:

fingers toes elbow → knee with 60% at the first place

ying yang white → black with 55.7% at the first place

man woman actor → actress with 86% at the first place

car highway train → railway with 57% at the first place

car road train → railway with 49% at the second place (first place was trains with 50%)

For “car road train” example, analogy was not as expected. In my opinion, it is caused due to selection of words that made by humans. When I say “highway” instead of “road”, analogy was successful as shown in above. I believe tool is not that successful to catch analogies for similar words (like using road instead of highway in the given example), algorithm lacks catching up the similarity between words sometimes as it was the case in “car road train” example.