NLP Assignement

Data preprocessing

The first important part is to translate all the sentences into number(tensor) usable inside the neural network.

Data embedding

Once the item and the target are loaded, we will use the Word2vec module from gensims to replace word by tensor. For doing that first we need to define a Word2vec model using the data target from our data and then we apply this model to every word that we want to use.

Padding

Since each word uses the word just before and just after as a context we need to add a padding to avoid error so before and after each sentence we add as much padding word as needed. I also added some padding word at the end of each sentence so every sentence are doing the exact same size.

DataLoader

Once we have a tensor representing each sentence, we create a dataset using it and pass it into a dataLoader to improve fluidity and simplicity of the data usage later. For that we use the dataset and daLoader provided by pytorch

Model creation

For the model I choose a Gated Recurrent Unit(GRU) because usually GRU and LSTM are more efficient than a simple RNN. Then since GRU and LSTM have merely the same result I choose to use GRU because it only has two logical gate instead of three so it’s faster.

For this model first I use a GRU with dropout to prevent overfitting, then I add a dropout layer because the built-in dropout of pytorch.GRU doesn’t apply after the last gate. The last layer is a ReLU one to output into the right number of classes.

Training/Evalutation

For the training and evaluation, I used the classical 20 epochs with one validation every 5 epochs (the validation dataset is extracted from dev.txt). During each training epoch, the processed data will pass through the model and then the differents metrics will be computed: first the loss will be computed using nn.CrossEntropyLoss(). After each epoch the optimizer will improve the different parameter passed to the model to improve performance.

Hyperparameter tuning

It seems like all the tuning didn't improved the F1 score or the accuracy at the end but it speed the accuracy growth.

To improve the model result I tried a lot of different hyperparameter and looked at what changed:

- Learning rate : I tried to lower or raise the learning rate a bit and nothing really changed, when I start to go really down like lr = 0.001 the accuracy decrease without any improvement.
- Windows : how far will the embedding look for a context it seems to give the fastest accuracy growth around 5.
- the lower is the BATCH_SIZE the longer it take to compute but with 16 it's still fast enough and increase a lot the accuracy growth speed

Result Improvement

The dataset is highly imbalanced, so there is a huge majority of O class, this led my model to a bad F1 score here are some of the thing.

Text sanitization:

I tried to remove all the word containing only punctuation but it only improved the F1-score by 0.5%

Useless sentence removal:

I also tried to remove all the sentences containing only word from the O class, and I also gained 0.5% on the F1 score.

Conclusion

The F1 score I managed to get in the end is bad and I don’t know how to improve it more, I think that I miss something important but I can’t find what.

If possible, I would love to have some feedback explaining the points I missed so that I can improve for the rest of the semester.