Homework 3

Βησσαρίων Μουτάφης sdi1800119@di.uoa.gr
1115201800119

1. NLP with RNNs

We provided 4 models in our NLP problem solution with the following architecture features

1. LSTM layers, dropout of 0.5 or lower, single linear layer to produce final predictions from final RNN hidden state
2. GRU layers, same dropout, same output layer logic
3. LSTM/GRU layers with skip connections enabled
4. Attention mechanism for the best model. Attention will be applied in the final RNN layer with input, RNN's hidden states and with respect to the final hidden state.

The models were provided with specific pytorch wrappers that enabled stacking RNN layers of different hidden sizes, dropout rates. The skip layers and the attention layer are just add ons to the general class and are provided on the respective models.

1.1 Notes on Data Preprocessing

Data Preprocessing is the same as the previous assignments with the only change some code cleaning and total eradication of http links. To handle dataset in a more effective way we overloaded the torch's dataset class and used dataloaders.

1.2 Notes on Embedding Layer

We created an overload of the embedding layer, by deriving the original class to enable it transform a whole tweet to a embedding sequence. To use embeddings we created a vocabulary from the train corpus, used it to create a id mapper for all words and an embeddings weight matrix with respect to the pretrained glove embeddings. During the embeddings layer creation we used the ready weights and froze the gradients. After that we transformed all the sentences in both train and test data-set into embeddings sequences and used a Dataloader to distribute the data in shuffled batches.

1.3 Notes on RNN Model

The RNN model contains all the techniques we used in previous assignments, plus the one needed for this one. We provided the users with a configuration list, in order to add different layers of various hidden sizes. Also we added skip layers with the change of a switch in the constructor, to use later and gave opportunity to the caller to add his own final attention layer before the actual Linear output Layer that gives prediction labels. We also use private variables to keep the hidden states of the RNNs and we provide an hidden init routine and a detach section during training steps, to ensure calculating the gradients with respect of the current batch.

2. Model 1 and 2

Simple RNN layers did not seem to solve the problem in an appropriate way since they overfitted. Specifically we tried the following for both LSTM and GRU models

- Dropout rates between stacked RNNs in the range of $[0.1, 0.7]$, with dropout rate of 0.2-0.3 yielding the best results of both models.
- Sequence sizes in range of $[10, 30]$ for the input samples, with max sequence length of 20 yielding most accurate results. In other cases we either miss information or introduced to many padding in the end of the sequence to intervene with models' training.
- Hidden sizes in range $[10, 30]$, with hidden size from 15 to 20 working better than others. We also tried stacking some layers with hidden sizes in increasing powers of 2, but this failed with quick model overfiting.
- Output dropout layer in the range of $[0.1, 0.8]$, with 0.3 and 0.4 working best for all models.

While the first 2 models were already too complex to solve the problem we acquired f1-scores in the range of $[0.50, 0.70]$ with the GRU model of 2 stacked layers to be the best performing model. This can be seen by the ROC curves and the classification report (check notebook), since it was the only model to decently classify instances of the "anti-vax" class, that is clearly undersampled in both the training and the testing sets.

3. Skip Layers

The addition of training layers helped only in certain occasions and only when the model under study was already performing more poorly than usual. We specifically tried putting skip layers in models 1 and 2 that did not introduced any additional predictive power, while we also tried putting skip layers in a three RNN-stacked network with

- 1 layer per stack
- 2 to 4 layers per stack and additional dropout

The above models proved to be way more complex and overfitted in the first 10 epochs while the test error never fall under 0.95, when the train error was freezing around 0.35 or lower.

In conclusion the addition of skip layers might have become more handy if we had a more balanced dataset and a more complex task to perform.

4. Attention Addition in Model2

Since the best model was the GRU model, we added one attention layer in the output RNN, to score better at classification report. The results in among 10 to 15 runs we performed, were similar, with attention layer addition to slightly improve classification scores (f1, precision, recall) for the "anti-vax" class, while lowering the scores for the other classes by a small quantity.

Once again we see that the dataset's label imbalance might mess out classifiers training, but with the addition of attention we noticed an important increase in the balance among predictions, that sometimes interfered with the general classification performance.

Here we should note that the classifier with attention layer, converged in almost 10 epochs, hence way faster than the original classifier and had, most of the time, a more balanced ROC curve plot.

5. Comparison With Previous Best Models

The conclusions from our study is summed up by the following

- The Logistic Regression model, was quite simple but yet scored quite good in this classification task, on the specific data-set
- The Simple Feedforward NN was also quite simple but even more powerful when glove embeddings were introduced, while it seemed to prefer the simplicity of BoW sentence representation.
- The RNN models were the best fit for Glove embeddings but were also a little more complex than needed and resulted in overfitting quite easily when trained for too long.
- The best model of all is the RNNs since there is a high predictive power, with respect to the "anti-vax class".However, RNNs with embeddings are quite complex, solving technique so we should stick with simple logistic regression, since the simpler the model the more robust to overfiting.
- Data-set's label imbalance will mess our classifiers no matter how complex the models are, how long they are trained, how much dropout we add between layers.
- Extra tricks of attention and skip connections can be useful to reach the peak performance faster, but if we don't take precautions they will lead to overfiting way sooner than conventional models.

6. References

- Bidirectional RNNs
- Lecture on attention
- Medium article on RNN attention mechanism
- Module List usage in pytorch

- Skip connections-All you need to know with examples in pytorch CNN layers
- Deep Residual Learning for Image Recognition
- Text Summarization with Glove Embeddings