Danny Rechitsky

Cosi 231 – Project 1

Penn Tree Bank NNs

Introduction

Three models were built to perform the task of multiclass classification on a Penn Tree Bank dataset. The dataset included approximately 32,000 sentences, each joined by a connective. In each model, the input features extracted were word vectors for each sentence, while the output was the sense of the relevant connective performing a conjunctive role in each sentence. The output senses of the sentence connectives fell into 21 classes, with a large class imbalance, where the rarest output class was sampled under 10 times in the training data, but the most frequent classes were sampled thousands of times.

Methods

The input sentences were featurized into word embeddings, as GloVe word embeddings and alternatively as randomly distributed embeddings, both of 300 dimensions per word. Two methods were applied to build sentence embeddings from the word embeddings: averaging and concatenation. In the first, the 300 dimensions for each word embedding were arithmetically averaged across the sentence, so that the final sentence embedding had 300 dimensions as well. In the latter, the 300 dimensions of each word in the sentence were concatenated. In order to manage memory, the maximum sentence length was capped at 50 words. The resulting dimensions of each sentence vector were therefore 300*50=15,000. Sentences under 50 words in length were right-padded up a length of 50.

The 21 output classes were highly imbalanced. To mitigate this imbalance, classes were weighted inversely proportional to their frequency in the training data, with a capped weight of 100 in order for the ultra-rare classes not to overwhelm optimization. Thus, the model would effectively be penalized more severely for incorrectly predicting the rarer classes and theoretically improve recall of these rarer classes.

The models used to predict the output class were logistic regression (LR), a multi-layer perceptron (MLP) and a convolutional neural network (CNN). In addition to the encoding of the word embeddings (random vs GloVe) and the format of sentence embeddings (averaged vs concatenated), hyperparameters used to train the models included learning

rate (0.001, 0.005, 0.01), regularization such as dropout rate (0.0, 0.2, 0.5) and weight decay (0.0, 1e-4), as well as input batch size (64, 128).

The output logits of LR and MLP were activated by the softmax function, while the logits of hidden layers of the MLP were activated by the ReLU function.

Experiments

Experiment 1 – Sparse vs Dense Embeddings in MLP

Embedding	Accuracy	F1	Hidden Depth
Sparse			
	0.200	0.067	5
	0.200	0.067	10
Dense (Glove)			
	0.204	0.134	5
	0.200	0.067	10

Experiment 2 – Glove Embeddings in MLP at Various Depths

Embedding	Accuracy	F1	Hidden Depth
Glove	0.200	0.073	50
	0.200	0.106	100
	0.200	0.106	200

Experiment 3 – Best of LR, MLP and CNN

Embedding	Accuracy	F1	Hidden Depth
LR	0.200	0.196	N/A
MLP	0.200	0.106	100
CNN	0.219	0.211	**100**

Results

The results were poor, not exceeding an accuracy of ~20% on the test set. The hyperparameter sweep revealed little variation of this accuracy given different

configurations. An early stop condition was imposed on the training loop once the validation set exhibited an increase in loss for 3 consecutive epochs. Training consistently ended in less than 10 epochs. In fact, the loss on the validation set never significantly decreased, while loss on the training set consistently decreased during training.

Despite attempts to increase regularization to very high values (dropout rate of 0.7 and weight decay of 0.001), the validation set never exhibited a significant decrease in loss during training.

Additionally, various hidden sizes and configurations were experimented with in the MLP model in order achieve non-linear training that might capture the classification patterns in the data better than the linear prediction of LR. Sizes of 3-12 layers in pyramidal e.g. [256, 128, 64, 32, 64, 128, 256, 21] and rectangular configurations e.g. [128, 128, 128, 21] were attempted. However, as in the attempts described above, validation loss never significantly decreased during training and prediction accuracy on the test set never exceeded 15%.

Nor did concatenating the word vectors in the input features instead of a simplified sentence average yield any improvement, although it could have theoretically revealed deeper sequential patterns in the sentence.

Moreover, due to technical difficulties in implementing the CNN, it was never successfully trained.

Furthermore, Experiment 1 above yielded virtually no difference between sparse and dense vectors at two hidden sizes (5, 10). Accuracy remained around 20% and much lower F1 of 7-13%.

Experiment 2 did not yield good results either at MLP hidden sizes of 50, 100 and 200. Accuracy remained at approximately 20%. However, curiously, there were no signs of overfitting. The training and validation loss remained at around 1.6 for all 10 epochs of training. The early stop condition was not triggered like in many other runs of lower depth.

Experiment 3 yielded a slightly higher F1 (21.1%) in the CNN at a hidden size of 100 with the usual accuracy (~21%). However, as these metrics are still rather low and close to random, there are clearly some unresolved training issues.

Analysis

Despite significant regularization, learning rate modification, various hidden sizes and shapes in the MLP as well as experimentation with averaged and concatenated sentence vectors, each model configuration appeared to suffer from overfitting. Given that validation loss never significantly decreased during training under any configuration, the models must have been learning on the noise of the training data rather than the underlying pattern that would result in higher prediction accuracy in the test set. Moreover, the significant class imbalance with the rarest classes in the single digits and the more frequent ones in the thousands, must have complicated the learning process despite attempts to give higher weights to rarer classes in order to penalize incorrect prediction of such. It is also possible, regrettably, that there might have been some flaw in the logic of extracting embeddings and/or building the models despite my best efforts as well as rigorous and persistent debugging to find any such errors in logic. Indeed, Experiment 2 showed no evidence of overfitting, perhaps indicating that there was some other flaw in the composition of the embeddings or training process that prevented proper learning.

Conclusion

The significant class imbalance coupled with potential flaws in implementation led to overfitting the training data despite efforts at regularization, class weighting and numerous hyperparameter configurations. The result was a low accuracy in the GloVe embeddings implementation not significantly higher than that of randomly initialized embeddings. Further work can be done to discover any possible implementation flaws and resample the data to better balance classes in order to achieve improved results in the future.