Jigsaw Rate Severity of Toxic Comments

1st place solution

Guanshuo Xu
Data Scientist at H2O.ai

My Idea and Method in General

The goal of this competition is to predict a single value representing the severity of toxicity for each comment. For model validation, the organizers provide a dataset of around 30,000 comment pairs, each pair has a label of which one is more toxic. For model training, we are free to choose any public data available. The most typical data we can use is the training data for the first Jigsaw competition (Jigsaw18 dataset). In the first Jigsaw competition, we need to predict the probability if a comment is toxic or not, and each comment in that dataset has a label representing if the comment is toxic or not. Obviously, the probability of "toxic" and the severity of "toxic" are two different concepts, for example, an absolute toxic comment may not be the most toxic, but intuitively these two goals are close. If the Jigsaw18 data and its toxic label is all we have, then, other than directly using the toxic probability, we may not have better approaches to improving the performance of this competition. Fortunately, besides the "toxic" label, there are five extra labels, namely, "severe toxic", "obscene threat", "insult" and "identity hate", and these extra labels either encode the types or the severity of toxicity, which means training a regular multi-label model on the Jigsaw18 data and further leveraging the six toxicity predictions could give us a chance to have better models for toxicity ranking.

After training models that were able to generate probabilities of the six toxic labels on the validation data, I opted for a linear combination (weighted average) of the probabilities, and the optimal weights were generated according to the performance on the validation set. There are multiple considerations when I decided to fit only linear models on the validation data:

1) Fitting non-linear models require breaking the provided validation data further into training and validation subset. And the training/validation (or cross-validation) split could be tricky because the comments are in pairs and there would be duplicates causing leakage if vanilla splitting method is used.

2) Even if I could have the perfect splits, the validation data is still too small and over-tuning on it could be overfitting.

3) Even if I could train a perfect non-linear model on the validation data, there could also be data/label distribution shift due to for example comments from different platforms or from different time periods. Complex non-linear models could be hit hard when facing distribution difference, simple linear models should be more robust. However, it was revealed after the competition ended that all the test cases were from the Jigsaw18 dataset, which indicated no distribution difference.

4) I personally don't believe there is much more to learn using non-linear models.

Training Dataset and More Implementation Details

For feature extraction, we trained models on the Jigsaw18 dataset using all the six labels and with BCE loss. Depending on the type of models, each model was trained for 1-3 epochs with linear learning rate schedular. To avoid data leaks, around 7000 duplicates between the Jigsaw18 data and validation data were removed before training (in fact, all the validation data were from Jigsaw18 and again I was not aware of that before the end of the competition). After the models were well-trained, they were applied on the validation data so that the comments were transformed into six probabilities as features. Since the validation data was labeled by comparing the toxicity of two comments, the optimization goal was maximizing the total number of correct pair-wise predictions. Genetic algorithms were used to search for the best weights of the six features. Other optimization methods or even brute-force tuning could also work because there were only six variables. The final prediction was then the weighted average of the six features.

Similarly, I trained the feature extractors using the second Jigsaw competition data (Jigsaw19) which has seven labels (there is one additional "sexual explicit" label), and I directly used the provided probabilities as labels rather than binarizing them. Each model was trained for only one epoch on the Jigsaw19 data.

The third and last dataset used is the Ruddit data which already has a severity label ranged [-1, +1], so I simply transformed the label into the range [0, 1] to use the BCE loss. This time there was no need to optimize any weights on the validation data. Each model was trained for 3-6 epochs.

Results and Discussion

Table 1 below shows the performance of all my models. I only used Roberta and Deberta models as they gave the best performance. Other models I tried included Xlnet, Albert, Bert, etc., but none of them had performance close to Roberta and Deberta.

"-l" in Table 1 means I included the duplicates between Jigsaw18 and validation data. Based on the numbers in the table, models trained on Jigsaw18 gave best performance, this is natural because test data were from Jigsaw18. For final ensemble, I blended the predictions from all the 15 models trained. The ensemble method is weighted rank average with weights set manually based on their validation performance.

Table 1

model	data	validation	public LB	private LB
roberta-base	jigsaw18	0.7023	0.7815	0.8052
roberta-large	jigsaw18	0.7035	0.7788	0.8064
deberta-base	jigsaw18	0.7040	0.7598	0.8030
deberta-large	jigsaw18	0.7050	0.7906	0.8139
roberta-base-l	jigsaw18	0.7028	0.7690	0.8070
roberta-large-l	jigsaw18	0.7027	0.7737	0.8013
deberta-base-l	jigsaw18	0.7030	0.7474	0.8013
deberta-large-l	jigsaw18	0.7044	0.7716	0.8085
roberta-base	jigsaw19	0.7008	0.7617	0.8020
roberta-large	jigsaw19	0.6991	0.7468	0.7968
deberta-base	jigsaw19	0.7026	0.7403	0.7958
roberta-base	ruddit	0.6859	0.8108	0.7845
roberta-large	ruddit	0.6865	0.8132	0.7955
deberta-base	ruddit	0.6880	0.7903	0.7880
deberta-large	ruddit	0.6942	0.8296	0.7989
ensemble	jigsaw18		0.7763	0.8103
ensemble	jigsaw19		0.7509	0.8012
ensemble	ruddit		0.8235	0.7983
ensemble	all15		0.7879	0.8139