Solution for the Jigsaw Unintended Bias in Toxicity Classification Competition

Yuanhao Wu & Yang Guan

2019.7

CONTENT

1 Background on us

1.1 Competition information

- Competition Name: Jigsaw-Unintended-Bias-in-Toxicity-Classification
- Team Name: Harness the beasts|驯服巨兽
- Private Leaderboard Score: 0.94649
- Private Leaderboard Place: 10th

1.2 Team information

There are two members in our team.

- Name: Yuanhao Wu
- Location: Shanghai, China
- Email: wuyhthu@gmail.com

- Name: Yang Guan
- Location: Shanghai, China
- Email: yang221199@sina.com

1.3 Other information

- What is your academic/professional background?

We both obtained B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 2014 and 2017, respectively. Yuanhao's research interests were biped walking and human robot interface. The research interest of Yang Guan was wireless light communication. Now, we are algorithm engineers in the AI lab of SAIC Motor Corporation Limited (SAIC, formerly Shanghai Automotive Industry Corporation), and focus on developing NLP technology for human-car interface and intelligent customer service system.

- Did you have any prior experience that helped you succeed in this competition?

We have won several medals in previous Kaggle competitions and learnt lots of NLP skills from the previous competitions. They help a lot.

- What made you decide to enter this competition?

Important reason of why this competition attracts us is that it's a typical NLP competition. And we can try a lot of state-of-the-art models to solve this problem. This helps us keep track on the top techniques in the NLP area. We learnt a lot in the competition.

- How much time did you spend on the competition?

May be 80 hours for each person.

2 Summary

- We think the first key to succeed in this competition is using apex to highly reduce the training time on GPU. The second one is further pretraining the BERT models with data of this competition. Last but not least, ensemble different models with diversity.
- We mainly used BERT with different sizes and different input types (cased and uncased). In addition, we use GPT-2 and RNN models to add diversity.
- We trained models with custom loss which use both target and auxiliary label including severe_toxicity, obscene, identity_attack, insult and threat. We also gave each sample different weight according to its label.
- Training all the models should take tens of hours, while the inference procedure is relatively fast. The whole solution including data processing, model loading, model inference and blending needs about 6700s to run on Kaggle GPU kernel.

3 Models and Training Method

3.1 BERT

In this competition, we used six BERT models. They are

1. BERT-large-uncased (further pretrained with competition data),
2. BERT-large-cased (further pretrained with competition data),
3. BERT-large-cased (trained with uncased data),
4. BERT-large-wwm-uncased (further pretrained with competition data),
5. BERT-large-wwm-cased,
6. BERT-based-uncased (further pretrained with competition data).

All BERT models are based on the pretrained model from https://github.com/huggingface/pytorch-transformers (formerly https://github.com/huggingface/pytorch-pretrained-BERT). For some of these models, we further pretrained them with the competition data.

To add diversity to our models, we trained BERT-large-cased model with uncased data and count it as one of our models.

3.2 GPT-2

Our GPT-2 model is also based on the pretrained model on https://github.com/huggingface/pytorch-pretrained-BERT. We didn't have enough time to optimize the hyperparameters of it, so we used the hyperparameters of BERT directly. This model didn't achieve a high score as we expected, but it still improved the score of our ensembled model.

To run the model without bug, we removed any characters whose ASCII value is larger than 255. New version of the package seems to have fixed the problem. You can check the issue 537 in the GitHub repo.

3.3 RNN

For RNN models, we preprocessed the data in a different way from BERT models. We used Spacy to tokenize the data and replaced the email-like/num-like/url-like tokens with some simple words. We used pretrained 300d Glove and the 300d Crawl word embeddings in our RNN models. To further reduce OOV words, we tried to lookup embedding with the lowered/stemmed/lemmatized/capitalized word if the original word does not exist in the lookup table. We also pretrained a 100d word embedding with the Word2vec algorithm and the competition data. In the model, the three embeddings were concatenated to form a 700d embedding.

In addition to token sequence, we also introduced char level count vector and some keyword-based features to RNN models. These numerical features improved the model performance.

The model architectures are ordinary as you can see in the code. Similar with BERT models, we also trained RNN models with multitask loss.

3.4 Blending

The final submission was a blending of six BERT models, one GPT-2 model and thirteen RNN models. We used a two-layer blending strategy to put them together.

In the first layer, we blended models of the same family. For RNN models, we simply averaged all their predictions. For BERT models, the blending weights are as follow:

Model	Weight
BERT-large-uncased (further pretrained with competition data)	0.25
BERT-large-cased (further pretrained with competition data)	0.15
BERT-large-cased (trained with uncased data)	0.2
BERT-large-wwm-uncased (further pretrained with competition data)	0.2
BERT-large-wwm-cased	0.12
BERT-based-uncased (further pretrained with competition data)	0.08

The above weights were determined according to their CV scores and our experience. As you can see, we gave higher weight to the finetuned models and uncased models.

In the second layer, we blended the blended results of BERT, GPT-2 and RNN with the weights of 0.88, 0.06 and 0.06 respectively. The second-layer weights were also determined according to the CV score and our intuition of the strength of the models.

3.5 Training method

Thanks to the power of advanced language models, model design is very straightforward in this competition. However, training method is critical. Better training method means better score. We will introduce our training method in this section.

First, we used Nvidia Apex (https://github.com/NVIDIA/apex) and bucket sampler to highly reduce the training time. This trick allowed us experiment rapidly. The typical training time of one epoch BERT-base model is about 2 hours with a single V100 GPU.

Second, inspired by the discussions and kernels on Kaggle, we used custom loss function to exploit the information of data. We used auxiliary labels including severe_toxicity, obscene, identity_attack, insult and threat.

For the target of this competition, we used 0.5 as threshold to get the binary value, and then calculated the binary cross entropy with our prediction as loss1. For auxiliary target, we directly calculated the binary cross entropy with our prediction as loss2 (both labels and logits were float numbers in the range of [0,1]). The final loss equals loss1 * 0.02 + loss2.We tried a lot of other loss functions to fit the metric, and this one worked best.

Third, we used sentence clipping to handle long sentence. If a sentence is longer than L_{max}, we concatenated the $0.25L_{max}$ tokens in the beginning and $0.75L_{max}$ tokens in the end, for the most important information often exists in the tail and the head. The max length L_{max} was set to 220.

Fourth, we used different learning rate among the layers of BERT. We set the learning rate of last layer of BERT encoder and the task-specific classifier to a base learning rate, such as 5e-5; and we reduced the learning rate by a ratio of 0.98 for the preceding BERT layers. As a result, the learning rate of the first layer of a BERT base model is 5e-5 * 0.98^{12} . We learned this trick from [1]. We think different BERT layers express different levels of semantic information. The information of lower levels are more general, so smaller learning rates were used to avoid catastrophic forgetting. The higher layers are expected to do the classification job, and the higher learning rates were used to speed up convergence. We used the BERTAdam optimizer and the learning rate schedule was *warmup-linear*. The warmup ratio was set to 0.05.

Finally, we used gradient accumulation to stable the training. For BERT-base and GPT-2, we set batch size to 48 and accumulating step to 3; and for BERT-large the batch size and accumulating step were 12 and 20 respectively.

4 Interesting findings

- What was the most important trick you used?

The most important tricks are described in the training method section.

- What do you think set you apart from others in the competition?

We think the following things are very important:

1. Use Apex O2 and bucket sampler to highly reduce the training time;
2. Models with good capacity and diversity;
3. finetune language model with competition data.

- Did you find any interesting relationships in the data that don't fit in the sections above?

No.

5 Simple Features and Methods

- What model that was most important?

The best model is BERT-large-uncased, and it can get about 0.944-0.945 on the public leaderboard.

A blending of six BERT models can get 0.94636 in the private leaderboard, compared to 0.94649 for the blending of six BERT models, one GPT-2 model and thirteen RNN models. The scores of RNN and GPT-2 models are around 0.940 on public leaderboard, which are marginally lower than the BERT models.

6 Model Execution Time

- How long does it take to train your model?

With a single V100 GPU, training a BERT base model took about 1.5 hour, and a BERT large model took about four to five hours.

- How long does it take to generate predictions using your model?

It takes about 6700s to run the whole inference script on Kaggle GPU kernel.

7 Reference

[1] Sun C, Qiu X, Xu Y , et al. How to Fine-Tune BERT for Text Classification? [J]. 2019.

[2] Yuval reina, "Toxic BERT plain vanila," 2019. [Online]. Available: https://www.kaggle.com/yuval6967/toxic-BERT-plain-vanila

[3] Abhishek Thakur, “Pytorch BERT Inference," 2019. [Online]. Available: https://www.kaggle.com/abhishek/pytorch-BERT-inference.

[4] Tanrei(nama), “Simple LSTM using Identity Parameters Solution" 2019. [Online]. Available: https://www.kaggle.com/tanreinama/simple-lstm-using-identity-parameters-solution.

[5] Pronkin Nikita, "BERT is not the only one," 2019. [Online]. Available: https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/discussion/95220#latest-577782

[6] Yuval reina, "Lets prevent a heart break," 2019. [Online]. Available: https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/discussion/95054#latest-557823.

[7] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.

[8] Howard J, Ruder S. Universal Language Model Fine-tuning for Text Classification [J]. 2018.