Jigsaw Unintended Bias in Toxicity Classification

Zhuo Chen[1], **Yue Zhang**[1]

[1]Technische Universität Berlin

cristianozhuo@campus.tu-berlin.de, yue.zhang.2@campus.tu-berlin.de

Abstract. *Fairness is increasingly recognized as a critical component of machine learning systems[1]. In this project we built a sentiment analysis model that can measure the toxicity of text. The model is based on BERT and GPT2. By setting sample weights for different data-points, the unintended bias of the model is successfully reduced.*
Keywords*: sentiment analysis, BERT, GPT2, unintended bias.*

1. Motivation

In order to protect voices in conversation, we want to build a model that can identify toxicity of comments in online conversations, where toxicity is defined as anything rude, disrespectful or otherwise likely to make someone leave a discussion.

There are numerous comments on platforms like Youtube, it is impossible to classify the comments and block the toxic ones manually. Applying sentiment analysis model helps finding and blocking the toxic comments, that improves the Internet environment.

In the previous models the unintended bias always exists, which means when a comment contains some sensitive words (e.g. ”gay”), it will be classified as toxic with a high probability, even though this comment is actually non-toxic (such as ”I’m a gay”). In this project, we’re challenged to build a model that recognizes toxicity and minimizes this type of unintended bias.

2. Solution Overview

We are required to build a model, the input of the which is comment-text and the output is a group of scalar ranged between 0 and 1, this scalar describes the toxicity of these comment-text (0 denotes non-toxic and 1 denotes very toxic).

First we split the train-set into real-train-set and validation-set with split ratio 9:1. Then we did some preprocess to clean the data. After that we applied BERT[2] and GPT2[3] models separately. The results were not good enough and the validation scores showed that, these models performed bad on the data-points with some special attributes. We used three methods to improve the results: we defined a new loss function, which could capture more information from the data; we set different sample weights for data-points with different attributes, this helped reducing unintended bias of the models; we made ensemble of multiple models, that reduced the bias too. Finally we built a model composed of 5 BERT models and 1 GPT2 model, this model achieved our goal.

3. Data Format and Metric

The dataset is labeled for identity mentions and we optimized a metric designed to measure unintended bias.

3.1. Data dimension

The train-set is 1804874 samples of comments from the online forum, and the competition organizer Jigsaw has extended annotation of it by human raters for various toxic conversational attributes. The test-set is 97320 samples of comments without any attributes.

3.2. Toxic attributes

Each train sample has 45 toxic attributes, we used only part of them. Table 1 shows the structure of train-data we used.

target: A scalar value with range [0,1], it describes how toxic the comment is. If $value \geq 0.5$, the comment is toxic and we call it positive; otherwise it is non-toxic and negative.

9 subgroups: they are $Male$, $Female$, $Homosexual$, $Christian$, $Jewish$, $Black$, $White$, $Muslim$ and $Psychiatric$. Each subgroup has a scalar value with range [0,1] too. We did binarization for these values, when $value \geq 0.5$ it will be set as 1, which refers that the comment is in this subgroup; otherwise it will be set as 0, which refers that the comment is not in this subgroup.

6 aux_targets: they are $severe\_toxicity$, $obscene$, $threat$, $insult$, $identity\_attack$ and $sexual\_explicit$. They all have scalar value with range [0,1]. It describes what kinds of toxicity a comment may have.

Column name	comment	target	9 subgroups	6 aux_targets
Data Type	string	float	bool	float

Table 1. The column names and data types of train-set.

3.3. BPSN and BNSP

For any dataset X in this project we will use the following definition:

$X^+ = \{x | x \in X, x\_target \geq 0.5\}$

$X^- = \{x | x \in X, x\_target < 0.5\}$

where x_target is the target value of data x.

As mentioned before, we got 9 subgroups. For each subgroup S_i, the rest part of whole dataset(Ω) will be defined as background B_i (i.e. $B_i = \Omega / S_i$). Then S_i and B_i both are divided into two parts by positive and negative, now we got four disjoint sets S_i^+, S_i^-, B_i^+, B_i^-. We use B_iPS_iN to denote $B_i^+ \cup S_i^-$ and B_iNS_iP to denote $B_i^- \cup S_i^+$. As there are 9 subgroups, we got 9 BPSN and 9 BNSP sets in total.

3.4. Metric

The metric is provided by competition holder Jigsaw[4]. It takes account of unintended bias by calculating the roc_auc on different subsets:

$$\begin{aligned} score =& \frac{1}{4}[auc\_overall + (\frac{1}{9}\sum_{i=1}^{9} auc\_S_i^{-5})^{-1/5} \\ &+ (\frac{1}{9}\sum_{i=1}^{9} auc\_B_iPS_iN^{-5})^{-1/5} + (\frac{1}{9}\sum_{i=1}^{9} auc\_B_iNS_iP^{-5})^{-1/5}] \end{aligned} \tag{1}$$

Where $auc\_overall$ is the roc_auc value on the whole test-set, auc_S_i, $auc\_B_iPS_iN$ and $auc\_B_iNS_iP$ are the roc_auc values on the corresponding subsets of the test-set.

The score ranges from 0 to 1. The higher the score is, the higher accuracy the model has and the better it deals with unintended bias.

4. Approach

4.1. Preprocess

The aim of preprocess is to clean the text and reduce the vocabulary of the dataset. It consists of 7 steps.

Remove URLs: More than 51,000 comments in the train-set contain URLs, however, almost all the text-string of these URLs make no sense in sentiment analysis. After removing URLs we reduced the vocabulary of the dataset quite a lot.

Abbreviation mapping: Sometimes people would write words in different forms. For example, “USA”, “U.S.” and “the United States of America” have the same meaning, they will be regarded as three absolutely different groups of token. Mapping abbreviation to their original forms can reduce the vocabulary of the dataset and strengthen the association between comments.

Contraction words mapping: In this step we transformed the contraction words to their original forms (i.e. from “don’t” to “do not”). This reduced the vocabulary too.

Dirty words mapping: When people spell dirty words on forums, in order to avoid being blocked, the dirty words would be written in “clever ways” (e.g. “shit” would be written as “sh*t” or “$hit”). By applying RegEx we successfully mapped these types of dirty words to their original forms. Then they could be recognized by model.

Delete symbols: There are many symbols(e.g. emoji symbols) which cannot be recognized by models. After tokenization all these symbols will be transformed into the same token–“unknown token”. When a comment-text contains this kind of symbols it will destroy the coherence of the comment. Thus we removed all these symbols.

Isolate symbols: There are also some symbols which may be mis-recognized, unless we isolate them (add one space char on each side of the symbol). By isolating the symbols, we kept the correct information of the text.

Additional mapping: Some words appears more than 200 times in train-set while they are out of vocabulary of LSTM or BERT model, we mapped them to words in vocabulary of models (e.g. from “Trumper” to “Trump supporter”). This reduced the number of “unknown token” and kept more information of the text.

After treating comments with preprocess methods, all the samples were converted to tokens of integers which would be used for model input.

4.2. Setting sample weights

Define $S = \bigcup_{i=1}^{9} S_i$, i.e. S is the set of data which belongs to at least one subgroup. Then $B = \Omega/S$ denotes the set of data which belongs none of subgroups. We divided both S and B into S^+, S^-, B^+,B^-.

In order to reduce the unintended bias, we set different sample weights for these 4 subsets: B^- was set with low sample weights (0.25), because they are most common comments; S^+ and B^+ were set with middle sample weights (0.5), these comments are toxic, it helps model learn that, which words makes comments toxic; S^- was set with a high sample weight (0.75), because the size of S^- is much less than the other three, and this dataset can help model learn that, sometimes a comment contains dirty words, but it is non-toxic.

After checking the validation score, we found the model performed bad on the data in 4 subgroups (black, white, muslim and homosexual). We use S_m to denote the set of train-data which belongs to at least one of these 4 subgroups and define $S_n = S/S_m$. Then we added 0.125 to the sample weights of S_m^-, S_n^+, B^+.

The final sample weights we used is shown below:

	S_m	S_n	B
+	0.5	0.625	0.625
-	0.875	0.75	0.25

Table 2. final sample weights

4.3. Loss function

For any data $x \in X$ we use w_x to denotes the sample weight of data x. Then define w as the average sample weight of the train-data.

Now we define the following functions:

$$H(p,q) = -p * log(q) - (1-p) * log(1-q) \quad s.t. \quad p \in [0,1], q \in (0,1) \tag{2}$$

$$Loss\_target = \frac{1}{w} \sum_{x \in \Omega} [w_x * H(x\_target, x\_predict)] \tag{3}$$

$$Loss\_aux = \sum_{x \in \Omega} \sum_{i=1}^{6} H(x\_aux\_target_i, x\_aux\_predict_i) \tag{4}$$

Then the loss_function of our model is:

$$Loss = Loss\_target + \alpha * Loss\_aux \tag{5}$$

where α is a scalar used to adjust the ratio. We set $\alpha = 1$ for BERT and $\alpha = 0.5$ for GPT2.

4.4. Fine-tuning models

4.4.1. BERT

Bidirectional Encoder Representation from Transformers (BERT) is a pre-trained NLP model which is proposed by Google AI team. BERT applies bidirectional training of Transformers[5]. It is designed to pre-trained deep bidirectional representations by jointly conditioning on both left and right context in all layers.

BERT has two model sizes: BERT-base with 12 layers, 768 hidden units and 110M parameters; BERT-large with 24 layers, 1024 hidden units and 340M parameters. There are two methods can be applied in BERT: uncased and cased. Uncased means that the text has been lowercased before WordPiece tokenization, e.g., John Smith becomes john smith; Cased means that the true case and accent markers are preserved.

In BERT models, 3 groups of input are required. Token-input is the tokens those are tokenized from the preprocessed comments, it has shape of (batch_size, 220); Seg-input is a zero matrix with shape of (batch_size, 220); Mask-input is a matrix of ones with (batch_size, 220). The BERT corpus output is with shape of (batch_size, 220, hidden_units). We connected BERT corpus with an Extract-layer, which extracts only the first of 220 vectors from BERT corpus output, the shape of extract output is (batch_size, hidden_units). Then we connected Extract-layer with our output-layers, i.e. target_output and aux_output. The target_output is with shape of (batch_size, 1) and the aux_output is with (batch_size, 6).

4.4.2. GPT2

Released on February 14, 2019, the OpenAI GPT-2 (GPT2) models have been released with similar structure like BERT models but with different tokenization method (Byte Pair Encoding) and different pre-trained data. In our project we used GPT2 base model with around 117 million parameters (GPT2_117) as the infrastructure.

Unlike BERT, GPT2 model has only token-input. The token-input of GPT2 has shape of (batch_size, 300), which is also transformed from preprocessed comments. The GPT2 corpus output is with shape of (batch_size, 300, 768). We plugged max-pooling layer and average-pooling layer on the GPT2 corpus output. Then we added one Dropout-layer, after that 2 output-layers which are the same as BERT output-layers were plugged.

Figure 1 shows the difference between the structures of our BERT and GPT2 models.

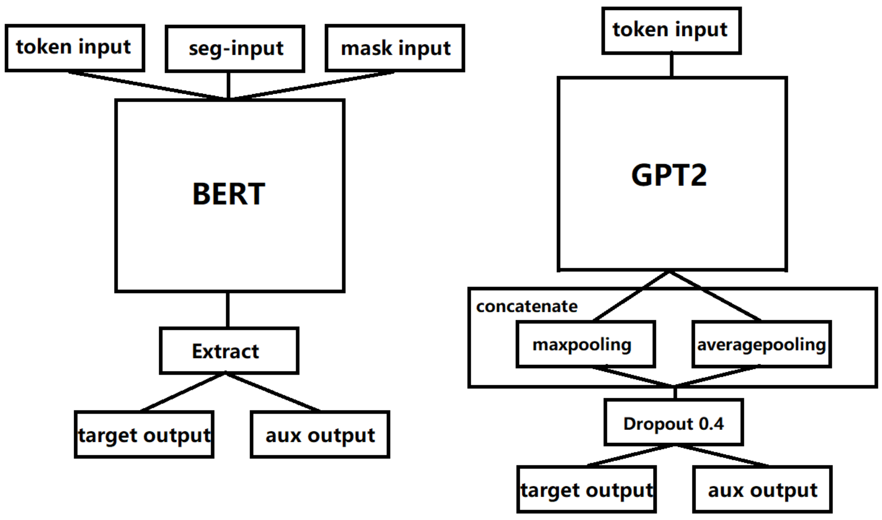

Figure 1. Architecture of our BERT and GPT2 models.

4.5. Setting optimizer

We use the following definitions:

n: an integer can be chosen as 1 or 2

bsz: batch size

$card(\Omega)$: the number of samples in train-data

$total\_steps = n * card(\Omega)/bsz$

$warm\_up\_steps = 5\% * total\_steps$

$actual\_steps$: the number of batches, which has been trained

r: the ratio between $actual\_steps$ and $total\_steps$, ranges in [0,1]

$lr1$, $lr2$: two learning rate we set for training.

Then our real time learning rate is:

$$lr = \begin{cases} lr1 * r/0.05 & r \leq 0.05 \\ lr1 * (1-r)/(1-0.05) & 0.05 < r \leq 0.5 \\ lr2 * (1-r)/(1-0.05) & 0.5 < r \leq 1 \end{cases} \tag{6}$$

We used official optimizer warm_up-Adam[6] for BERT and GPT2. The parameters is shown in Table 3.

Model	Num	Pre	bsz	$lr1$	$lr2$	n	$Seqlen$
BERT-base-uncased	2	Yes	128	2e-5	3e-5	2	220
BERT-base-cased	2	Yes	128	3e-5	3e-5	2	220
BERT-large-uncased	1	Yes	64	3e-5	1.5e-5	1	220
GPT2_117	1	No	128	8e-5	10e-5	2	300

Table 3. Parameters for all models. "Num" **means number of models.** "Pre" **means doing preprocess.** "$Seqlen$" **means length of input sequence.**

4.6. Ensemble

Compared with BERT or GPT2 single model, the average of outputs of any multiple models could reach a much higher score. Thus we made ensemble of multiple models with different proportions. Finally we chose the combination which achieved the best validation score (i.e. BERT+GPT2).

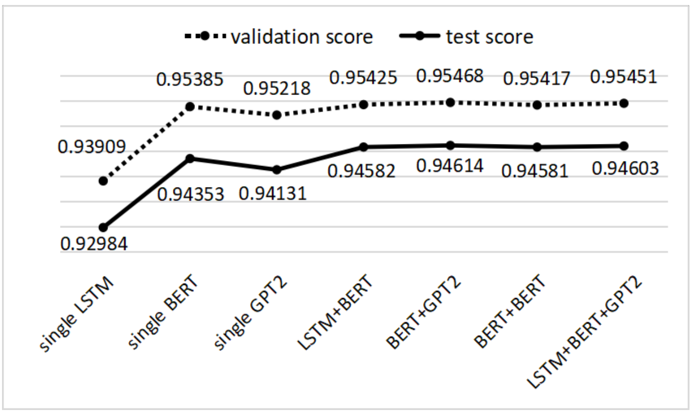

Figure 2. Best scores of different combinations of models

5. Result

We trained 2 BERT-base-uncased models, 2 BERT-base-cased models, 1 GPT2 and 1 BERT-large-uncased model. Table 4 shows the results:

model name	validation score	test score
BERT-base-uncased_1	0.95176	0.94316
BERT-base-uncased_2	0.95262	0.94442
BERT-base-cased_1	0.95089	0.94263
BERT-base-cased_2	0.95109	0.94265
GPT2	0.95218	0.94131
BERT-large-uncased	0.95385	0.94353

Table 4. Results of single models

When we took average of all model predictions, we achieved the ensemble validation score of 0.95468 and the score-table is like below:

subgroup	bnsp_auc	bpsn_auc	subgroup_auc	subgroup_size
white	0.964645	0.924574	0.905582	1363
muslim	0.954981	0.940172	0.905847	1097
black	0.965406	0.919588	0.907722	764
homosexual	0.967714	0.911820	0.908773	550
jewish	0.955496	0.953983	0.941024	407
christian	0.946848	0.973144	0.949919	1978
male	0.963214	0.960471	0.952322	2238
female	0.962045	0.963863	0.953975	2838
psychiatric	0.988752	0.946086	0.980526	248

Table 5. score of ensemble model

The final private score (score of test data) is 0.94614, which can reach 14^{th} highest in this competition.

6. Conclusion

Data imbalance problem is commonly seen in current data-analysis tasks. Until now there is no such systematic or universal approach to solve this problem. So doing research in bias problem is with big opportunities and with potentials in real cases.

In this project both of us has spent more than 800 hours. When the competition ended, our team ranked only 77th. At that time our model was based on BERT and LSTM. We thought that maybe GPT2 could improve our results. So we kept trying to improve our model after the competition and finally built a model could reach the 14^{th} place.

Also, we have learned some lessons like followings:
1. We learned a lot about BERT, GPT2, LSTM and other NLP knowledge(e.g. data cleaning). We hope to have further research in NLP field.
2. We learned to deal with data science problem scientifically with good arrangement.
3. We should trust not only public leaderboard scores but also validation scores.
4. We need to keep learning from the new inventions from the data-science communities.
5. Good features make more improvement than fine-tuning.
6. Keep calm and change the way of thinking when we stuck into a problem.

In the future we would like to implement all our models with pytorch and try to use XLNet[7] for our task.

7. Acknowledgement

First of all, we want to show our gratitude to Prof. Dr. Ziawasch Abedjan for his support of our project with high-performance GPU resources and his guidance of our research directions. Also, we want to express appreciation to Mahdi Esmailoghli for his patience of instructing us and answering our questions. Finally, we would like to thank Martin Kiefer for his enthusiasm in helping us building environment on the server.

References

[1] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. Capuchin: Causal Database Repair for Algorithmic Fairness. arXiv e-prints, page arXiv:1902.08283, Feb 2019.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv e-prints, page arXiv:1810.04805, Oct 2018.

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification. arXiv e-prints, page arXiv:1903.04561, Mar 2019.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. arXiv e-prints, page arXiv:1706.03762, Jun 2017.

[6] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv e-prints, page arXiv:1706.02677, Jun 2017.

[7] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv e-prints, page arXiv:1906.08237, Jun 2019.