Jigsaw Unintended Bias in Toxicity Classification

Yao Xiao	Yaoyao Chang	Cheng Peng	Siyu Li	Zhiyu Yuan
2018214183	2018214139	2018214152	2018214124	2018214168

Abstract

Toxic comment detection has recently become an active sub-topic of text classification in NLP field. The unintended prediction bias is one main challenge for prediction model in this problem. In the Jigsaw Kaggle competition, we tried to construct prediction models that could both detect the toxic comments effectively and minimize the prediction bias for identity mentions. We performed elaborative data pre-processing work and feature engineering process. Then designed and trained bi-directional LSTM-based and BERT-based models respectively. Various optimization methods like custom loss function and multi-task learning are employed to enhance the model performance. Several strategies including sequence bucketing and dynamic mini-batch are used to speed up the model training process. We have carried out up to 50 groups of contrast experiments in total to find the optimal solution. The final highest score is achieved by ensemble LSTM-based and BERT-based models. Our best history ranking is top-1% and current best score is 0.94274 being top-5% in competition

1 Introduction

We choose the Kaggle competition, Jigsaw Unintended Bias in Toxicity Classification, as the final project in this Deep Learning class. The goal in this competition is to build a model that both recognizes toxicity in text comments and minimizes unintended bias with respect to mentions of various identities.

In on-line conversation system, toxic comments that are rude, disrespectful or otherwise usually make people leave a conversation. When the Conversation AI team of Jigsaw and Google first built toxicity prediction models, they found that the models incorrectly learned to associate the names of frequently attacked identities with toxicity. That is, models predicted a high likelihood of toxicity for comments containing identities (e.g. "gay"), even when those comments were not actually toxic (such as "I am a gay woman"). This happens because training data was pulled from available sources where certain identities are overwhelmingly referred to in offensive ways. Therefore, the prediction model that can both identify toxicity in on-line conversations and get rid of the bias for identities is highly demanded.

2 Related works

Toxic comment detection has become one active topic in several competitions in recent year. This problem is a sentence-level analysis task in Natural Language Processing (NLP) field. It is a kind of sentence classification task and has many similarities with sentiment classification problem, but its own characteristics make it more challenging.

In 2018, Betty van Aken et al. [1] presented multiple approaches for toxic comment classification mainly based on the Jigsaw dataset. They showed that the approaches make different errors could be effectively combined by ensemble method to achieved obviously better result in this problem. They also pointed out that many highly idiosyncratic or rare vocabularies has bring about challenges

Preprint. Work in progress.

for the prediction model. David Noever [2] systematically evaluated 62 classifiers representing 19 major algorithmic families against features extracted from the Jigsaw dataset. He showed that 28 features including bad word count and capital letter ratio can provide uncorrelated but predictive input for the later prediction model.

Long Short-Term Memory (LSTM) 3 is an Recurrent Neural Network (RNN) architecture used in deep learning especially in NLP tasks. LSTM is good at handling sequence data such as sentence and speech which is suitable for our task.

The self-attention-based BERT [4] has achieved new state-of-the-art results on various natural language processing tasks. BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. The pre-trained BERT representations can be fine-tuned with little additional layers to create state-of-the-art models for a variety of tasks, including question answering, language inference and sentimental classification. BERT has been extensively used as the main model by the participants in a wide range of NLP competitions. Sun et al. [5] have performed extensive experiments to study how to fine-turn BERT effectively for text classification problem.

3 Problem definition

In this competition, a dataset labeled with identity mentions and toxicity value is provided and a metric designed to measure unintended prediction bias is required to be optimized.

3.1 Dataset Specification

A large on-line comment Dataset ($\sim$2 million samples) labeled with toxicity values and identity mentions has been organized and provided.

In supplied data, the text of the individual comment is in the ***comment_text*** column. Each comment in Train set has a toxicity label (***target***), and models should predict the ***target*** toxicity for the Test data. This attribute (and all others) are fractional values which represent the fraction of human raters who believed the attribute applied to the given comment. For evaluation, the comments with ***target*** ≥ 0.5 are considered as positive samples (toxic).

The Train data also has six additional toxicity subtype attributes whereas the Test data does not: ***severe_toxicity***, ***obscene***, ***threat***, ***insult***, ***identity_attack***, and ***sexual_explicit***. Models do not need to predict these attributes for the competition, thus they are included as an additional avenue for research. Additionally, a subset of comments in Train set have been labeled with a variety of identity attributes: ***male***, ***female***, ***transgender***, ***other_gender***, ***heterosexual***, ***homosexual_gay_or_lesbian***, ***bisexual***, ***other_sexual_orientation***, ***christian***, ***jewish***, ***muslim***, ***hindu***, ***buddhist***, ***atheist***, ***other_religion***, ***black***, ***white***, ***asian***, ***latino***, ***other_race_or_ethnicity***, ***physical_disability***, ***intellectual_or_learning_disability***, ***psychiatric_or_mental_illness***, and ***other_disability***. The additional feature columns in the Train set are useful in the feature engineering process.

3.2 Evaluation Metrics

The Evaluation Metric of final model score is defined as a combination of the overall AUC in Test set with the generalized mean of three Bias AUCs as showed in Eq.(1). The Bias AUCs are introduced to measure unintended bias for various identities.

$$\text{score} = 0.25\text{AUC}_{\text{overall}} + \sum_{a=1}^{3} 0.25(\frac{1}{N}\sum_{s=1}^{N} m_{s,a}^{-5})^{-\frac{1}{5}} \tag{1}$$

In Eq.(1), $\text{AUC}_{\text{overall}}$ represents ROC-AUC value in the full Test set, and $m_{s,a}$ means bias metric for identity subgroup s using sub-metric a. There are N identity subgroups involved in the final metric. The explanation of all three sub-metrics are provided as follows.

Subgroup AUC: Here, restrict the data set to only the examples that mention the specific identity subgroup. A low value in this metric means the model does a poor job of distinguishing between toxic and non-toxic comments that mention the identity.

BPSN (Background Positive, Subgroup Negative) AUC: Here, restrict the test set to the non-toxic examples that mention the identity and the toxic examples that do not. A low value in this metric means that the model confuses non-toxic examples that mention the identity with toxic examples that do not, likely meaning that the model predicts higher toxicity scores than it should for non-toxic examples mentioning the identity.

BNSP (Background Negative, Subgroup Positive) AUC: Here, restrict the test set to the toxic examples that mention the identity and the non-toxic examples that do not. A low value here means that the model confuses toxic examples that mention the identity with non-toxic examples that do not, likely meaning that the model predicts lower toxicity scores than it should for toxic examples mentioning the identity.

4 Solution

4.1 Overall review

First, we carried out carefully exploratory data analysis (EDA) and performed data cleaning works on comment data. Second, we constructed many statistically features to help the model prediction process. Third, we elaborately designed LSTM-based and BERT-based models and ensemble them to achieve the best performance. Forth, many optimization methods including custom loss, multitask learning are employed to enhance our model. Besides, we listed several valuable strategies which speed up our training process in this competition.

The following parts will be explained in detail in Section 5.

- Data analysis and pre-processing
- Feature engineering
- Model architecture
- Optimization techniques
- Training speed up

4.2 Model Selection

Their are two kinds of models widely used in this competition: RNN-based models and BERT. These two kinds of models both have advantages and shortages.

- For RNN-based models, 1) flexible architectures could be designed; 2) the training process is relatively time-saving compared with the training or fine-tuning process of BERT and 3) various pre-trained word embeddings like wor2vec, glove, and fast-text could be utilized. The RNN-based architectures are the mainstream in various NLP tasks. The basic solutions in public kernels in this competition are commonly designed as multi-layer LSTM models. But mere RNN based models are not competent to achieve the top score in our competition.
- BERT is a very powerful language representation model, which is proved to achieve state-of-the-art performance in various NLP tasks. Commonly, only fine-tuning process is needed on the pre-trained weights. But even the fine-tuning process is highly time-consuming in our experiments, and the model main architecture is not flexible for adjustment.

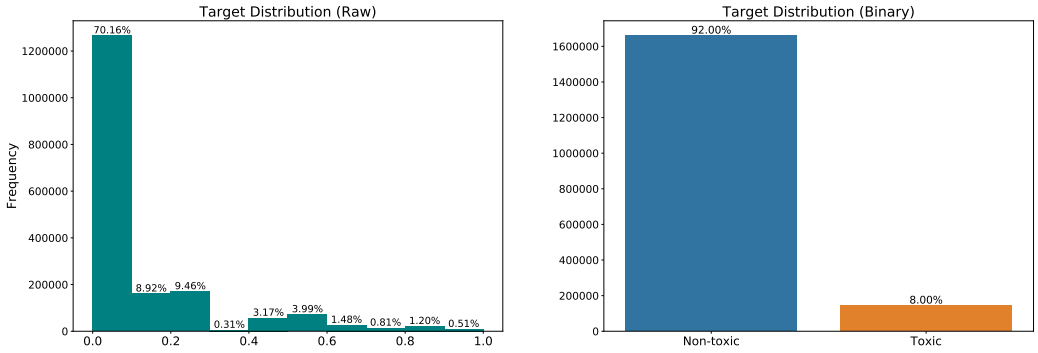
Therefore, we ensemble BERT and LSTM based models to combine their advantages and obtain the best performance.

5 Technique Details

5.1 Data Analysis And Pre-processing

Great efforts have been paid to explore and analyze the given data set, and many useful results have been obtained.

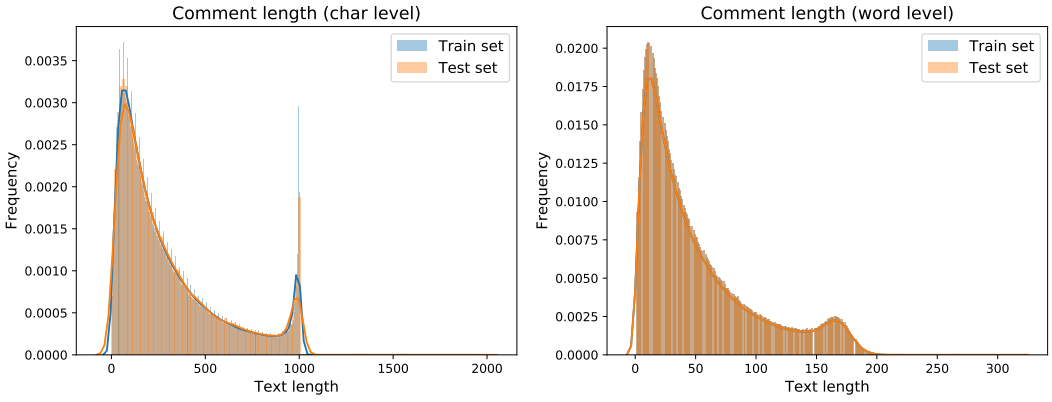
The target distribution of provided data is shown in Figure 1. The dataset is extremely imbalanced. In sub-figure 1(a), there are ∼70% comments with target values merely between 0.0 and 0.1. For the ∼7% comments with target values around 0.5 (0.4 to 0.6), they are likely to be misclassified by the machine learning models. From sub-figure 1(b), it could be found only 8% positive samples are available.



(a) (b)

Figure 1: The data set is extremely imbalanced with only 8% toxic comments as positive samples. (a) Raw target distribution; (b) Binary target distribution with 0.5 as threshold.

The comment length distribution at both word and character level is showed in Figure 2. The distribution of word and character length is somewhat similar. In sub-figure 2(a), there is a obvious peak at character length = 1000. The reason is perhaps that the on-line conversation website has a maximal comment length limit bing 1000 characters. From sub-figure 2(b), it could be found the word number of most comments are below 200. The maximal word number in training LSTM-based models and BERT is set as 220.



(a) (b)

Figure 2: The comment length distribution. (a) Char level; (b) Word level.

For RNN-based models, two embedding dictionaries of FastText [6] and GloVe [7] are employed. To improve the coverage rate of word embedding dictionary and provide high-quality vectorized text for later network model, careful text cleaning works are performed on the input comment texts.

- Isolate punctuations. Add spaces before and after the punctuations could separate them from words and improve the coverage rate of word embedding dictionary.
- Remove unknown symbols. Many special punctuations, letters with special font, emotion symbols and so on have been removed for having no mapped embeddings.
- Handle contractions in Tokenizer. The tokenizer could deal with some contractions and beyond.
- Some other processes like manually correction for misspelled word that have no obvious changes on model performance are omitted.

Notably, BERT already has input pre-processing pipeline, thus no additional operations are performed.

5.2 Feature Engineering

Elaborate feature engineering is of great importance for enhancing the performance of prediction model in many machine learning competition. Besides, David Noever [2] pointed out some features are very useful in this toxic comment detection task. By statistical approaches, 16 features have been constructed. Some of them are listed as follows. The "good" and "toxic" words are generated and scored based on comparing the appearance frequency of each word in toxic and non-toxic comments. The logarithmic function is used in calculating the word score to limit the score value and keep the numerical stability.

- Number and ratio of exclamation marks (!).
- Number and ratio of capitals.
- Number of toxic word.
- Comment toxic score. Sum of toxic word weights.
- Number of good word.
- Comment good score. Sum of good word weights.
- ...

The correlations between newly constructed features and original data information including target value are shown in Figure 3 with the correlation heatmap format. From the heatmap we could note that "num_toxic_words" and "toxic_words_scores" have strong positive correlations with the target information. Meanwhile, "num_good_words" and "good_words_scores" have strong negative correlations with the target information. Besides, the number of capital letters and exclamation marks are more related with the target value compared with other basic features.

5.3 Model Architecture

In sub-figures 4(a) and 4(b), the black part is the baseline LSTM model in competition kernels. It contains an embedding layer, two sequential connected bi-directional LSTMs and latter two kinds of pooling layers and fully-connected (FC) layers. Furthermore, we have tried two ways to take advantages of identity information provided in dataset.

Firstly, in sub-figure 4(a), we try to utilize the identity information in loss to help pre-train the network. But the result is not very good.

Secondly, in sub-figure 4(b), we use some auxiliary columns for multi-task learning and train models to predict the missed identity information in dataset. Then we can get predicted identity information of test data and some training samples. Finally, we use the identity columns as well as other features to improve our model performance together.

For BERT model 5, some additional layers are appended after the model main part, then it is fine-tuned for our toxic comment detection task. As pointed out by Jacob et al.[4], it is fairly expensive to train the BERT model from scratch. Thus, we downloaded the pre-trained BERT weights and fine-tune the model with additional elaborately designed output layers.

Each LSTM model has been trained with exponential weight moving average strategy. Several LSTM models are ensemble together and then combined with the BERT model to make the final prediction.

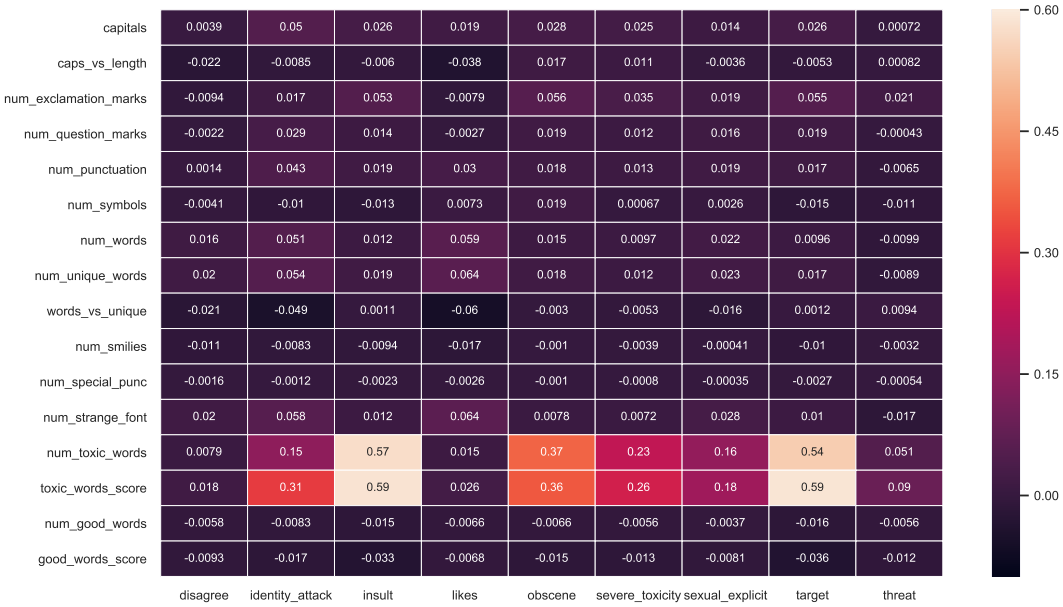
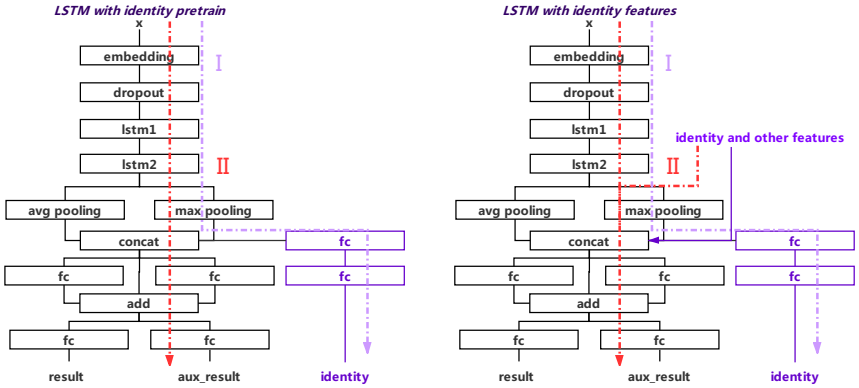


Figure 3: Correlations between some constructed features and original data information.



(a) Pre-train by identity information. (b) Final train with all features.

Figure 4: The LSTM architectures designed in our experiments.

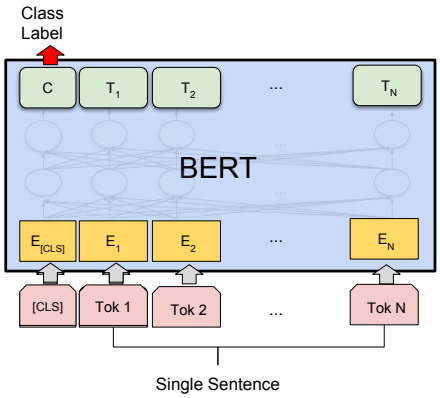


Figure 5: Some additional elaborately designed output layers are added after the BERT basic model, then we fine-tuned the whole model on the toxicity detection comment dataset.

5.4 Optimization Techniques

- Multi-task learning.

  Auxiliary prediction results are utilized in training models. The loss of auxiliary results will be counted to the final loss used in back propagation.

- Custom loss function.
  - Focal loss [8] is used to fight with the data imbalance problem and pay more attention to the tough error-prone samples.
  - Adjust sample weight according to identity information. Because the metric was designed to take sentences with identity mentioned into more important consideration, we need adjust the loss function correspondingly.

- Layer-wise learning rate setting.

  According to the experiment results of Sun et al.[5], we choose their provided layer-wise optimal learning rates of BERT model.

5.5 Training Speed Up

Time is expensive in this kernel only Kaggle competition. For the limitation of only 2 hours GPU time, every time-saving technique is beneficial.

Meanwhile, there are several processes that are time consuming but repeated in every run. Thus, we can run it once and save formated results to help save experiment time.

- Sequence bucketing.
  - For LSTM models, sequences in each batch are dynamically padded to the maximum sequence length in this batch.
  - For BERT models, dynamic mini-batch-based training strategy is employed to speed up the training process.

- Use pickled embedding data.

  The pickled embedding dictionary is more time-saving in data loading process.

- Saving processed variables. It takes $\sim$30 mins for BERT to process the whole training data. Thus, it is well worth saving and load the $numpy$ format processed training data.

6 Experiments and Results

Table 1: Extensive performance comparison of different model architectures and methods.

Model	Core idea	Pre-processing	Traning Data	Batch Size	#Epoch	LB Score	Rank	Ensemble
Bidirectional LSTM	–	YES	ALL	512	5	0.93524	1200+	6 LSTM(v1)
Bidirectional LSTM	More Data Preprocessing					0.93706	900+	2 LSTM(v2)
Bidirectional GRU	LSTM $\rightarrow$ GRU					↓[1]	↓	3 LSTM + 3 GRU
Bidirectional LSTM	Focal Loss (3 versions)					**0.93792**	**800+**	6 LSTM(v3)
BERT	One More Dense Layer	NO	65%	32	1	0.93686	1000+	NO
	More Complex Classifier and Custom Loss	NO	65%	64	1	0.93791	800+	NO
	Change Loss Weight (3 versions)	NO	65%	64	1	↓	↓	NO
	Remove 2 FC Layers							
	Focal Loss (3 versions)							
	Add Aux/Statistics Features (4 versions)							
	Add Test Prediction In Training							
	All Kinds of Data Preprocessing (3 versions)	YES						
	All Data, 128 BS, 2 EN	NO	ALL	128	2	0.9402	400+	NO
	Focal Loss (3 versions)	NO	ALL	128	2	0.94075	300+	NO
	Focal Loss + Layer-wise LR	NO	ALL	48	2	**0.94110**	**300+**	NO
Ensemble	6 LSTM(v1) + BERT(v1), Average					0.93964	400+	
	6 LSTM(v1) + BERT(v1), Weighted(3 versions)					↓	↓	
	6 LSTM(v1) + BERT(v1), Weighted(0.4/0.6)					0.93978	400+	
	6 LSTM(v2) + BERT(v1), Weighted(0.4/0.6)					0.94065	300+	
	6 LSTM(v2) + BERT(v2), Weighted(0.4/0.6)					0.94265	170+	
	6 LSTM(v3) + BERT(v3), Weighted(0.4/0.6)					0.94272	160+	
	6 LSTM(v3) + BERT(v4), Weighted(0.4/0.6)					**0.94274**	**161**	

[1] The symbol ↓ means ranking decline.
There are 3,096 teams in total.

Our main model architectures are bi-directional LSTMs and BERT. Based on them, we have carried out a large number of experiments to improve our model ability and prediction score. The experiment records are briefly listed in Table 1. Our currently best score is 0.94274, ranking $\sim$top-5%. Our best ranking in history is top-1%.

Firstly, it's necessary to explain the basic configuration of the hyper-parameter. The bi-directional LSTM-based models just have a small number of model parameters, so the memory used during training is not large, and the training time could be shorten. That's why the batch size in training process can be set to 512 and epoch number can set to 5. But for BERT, the model parameter size is too large, so the batch size is just set to 64 during training. Meanwhile, since an overall training epoch of BERT takes more than 5 hours, only 65% of the data is used during normal training experiments.

Data preprocessing has different influence on LSTM and BERT models. For LSTM-based models, we performed careful data pre-processing process such as abbreviated expansion and punctuation isolation, which improved $\sim$0.002 score. On contrast, after data preprocessing for BERT, the performance of model declined in fact. Then, we noticed that BERT itself has already have data pre-processing pipeline and no further operations are added later.

Custom loss function could promote the model performance. The loss function in Kaggle baseline kernel is cross entropy of the target and several feature columns. Experiments showed that focal loss could improve our model score. The data set is rather imbalanced and focal loss is suitable for this problem. Meanwhile, paying more attention to the tough samples is also beneficial.

We have tried various minor network structure adjustments on LSTM-based models. The adjustments include adding FC layer to add additional feature information, removing the final FC and adjust the number of neurons, but these changes only have very limited effects on prediction score.

Ensemble could obviously contribute to prediction score. Basically, exponential weight moving average strategy is used to combine the different prediction results of a LSTM model. That is, when training a single LSTM model, the inference results after different epoches are averaged with exponential moving weights. Moreover, the prediction results of different LSTM models have been ensembled and further integrated with the BERT-based model prediction results. We finally trained six LSTM models and one BERT-based model to make ensemble, which makes the prediction score greatly improved.

7 Conclusion

We constructed several prediction models to both detect the toxicity in comments and minimize the prediction bias for different identity mentions. Our overall work mainly includes the following parts:

- Investigate almost all important kernels and discussions in the competition.
- Perform elaborative data pre-processing work and feature engineering process.
- Design and train LSTM-based and BERT-based models respectively.
- Design and try various optimization methods to enhance the model performance and speed up the model training process. Strategies such as custom loss function and sequence bucketing are very useful.
- Carry out up to 50 groups of contrast experiments in total to find the optimal solution.
- Ensemble LSTM-based and BERT-based models to achieve our best score.

The highest score we achieved in history is top-1%. Our current leader-board score is 0.94274 and ranking top-5%.

References

[1] Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. Challenges for toxic comment classification: An in-depth error analysis, 2018.

[2] David Noever. Machine learning suites for online toxicity detection, 2018.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[5] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2019.

[6] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.