Upskill ISA IM Project 02

Unintended Bias in toxicity classification

Md Raisul Islam, June 25, 2021
Email: shuvo714@gmail.com

Abstract

The Kaggle competition Jigsaw unintended bias in toxicity classification, existing models suffer from unintended bias where models might predict high likelihood of toxicity for content containing certain words (e.g. "gay") even when those comments were not actually toxic (such as "I am a gay woman"), leaving machine only classification models still sub-standard.

I have tried BERT pretrained models to with XLNet to predict the Unbiased outcome of the toxic and nontoxic comments.

Methodology

I have used pretrained NLP models and Word vector embedding to train and predict the unbiasedness to identify the Toxic classification from the test dataset.

It gave me a Kaggle Private Leaderboard Score of **0.94074**.

Methods Used:

- BERT Pretrained Model
  Neural Net
- XLNet
- Gensim Word Embeddings

Data Preprocessing

Obscene Text Preprocessor

As discussed earlier on in the *Data exploration* section, it is common in NLP to remove from the corpus punctuations, very common words, and special characters as generally those do not contribute to the overall meaning of the sentence. However, when dealing with toxic comments, special characters are used to camouflage swear or offensive words to circumvent the filtering mechanism that are in place on most social platforms.

Tokenization

One of the very first steps in building a language model is tokenization, that is transforming a corpus of text into a dictionary of its components words, generally referred to as tokens.

Once a dictionary is built, we can represents each word that is encountered in numerical terms. There are two main approaches to represent a word, a *one-hot-encoding* approach turns a word into a binary vector of the length of the dictionary where it contains all zeros except for a one at the index position of the word in the dictionary, this works well with small dictionaries.

A second approach is to represent a word by its index in the dictionary, this turns a sentence into a vector of dictionary indices making word representation more memory efficient.

Because we are dealing with a large corpus of over 95 million words and a dictionary of nearly half a million words, *one-hot-encoding* is not appropriate.

Bidirectional Encoder Representations from Transformers (BERT)

BERT is a method of pre-training language representations, meaning that we train a general-purpose "language understanding" model on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks that we care about (like question answering). BERT outperforms previous methods because it is the first unsupervised, deeply bidirectional system for pre-training NLP.

For this project, I have used BERT-Large, Uncased (Whole Word Masking): 24-layer, 1024-hidden, 16-heads, 340M parameters release file.

GloVe: Global Vectors for Word Representation Embeddings

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

We used a pre-embedded file which is defined as: Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download)

Result Analysis

The final prediction from the BERT prediction metric gives a test submission score **0.94074** in Kaggle Private leaderboard.

```
To train the model the following metrics were used:
def train_model(model, train, test, loss_fn, output_dim, lr=0.001,
                batch_size=512, n_epochs=4,
                enable_checkpoint_ensemble=True):
   param_lrs = [{'params': param, 'lr': lr} for param in model.parameters()]
   optimizer = torch.optim.Adam(param_lrs, lr=lr)
```

It seems, changing the weights, the Optimizer Adam from torch really makes a difference in the final score for prediction to the classification.

Model Evaluation and Validation

These scores are the Final Metric as detailed in the Metrics section above and calculated as follows:

$$score = w_0 AUC_{overall} + \sum_{a=1}^{A} w_a M_p(m_{s,a})$$

where:

A = number of submetrics (3)

$m_{s,a}$ = bias metric for identity subgroup ss using submetric a

w_a = a weighting for the relative importance of each submetric; all four w values set to 0.25

Conclusion

Providing the models good score, it is also realized that the unbiasedness can be further reduced by finetuning the BERT & Bi-LSTM Models together. Using the given metric by the Kaggle Competition host, combining the overall AUC with the generalized mean of the Bias AUCs to calculate the final model score is the proper key to evaluate the score properly.

References

[1] Big data for better or worse: 90% of world's data generated over last two years. https://www. sciencedaily.com/releases/2013/05/130522085217.htm. Source: SINTEF

[2] Bidirectional Encoder Representations from Transformers (BERT); Google, 2018 by Jacob Devlin and his colleagues.

[3] pytorch-pretrained-BERT-haqishen, Qishen Ha; Kaggle User.

[4] GloVe: Global Vectors for Word Representation, Jeffrey Pennington, Richard Socher, Christopher D. Manning; Stanford University.

[5] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification, 2019.

[6] Jeremy Howard Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification https://aclweb.org/anthology/P18-1031