CS340 Final Project Report

- Zhiyuan Zhong 12110517
- Yuhang Dong 12011710

Unintended Bias in Toxicity Classification

The goal is to build a model to recognize toxicity while minimizing this type of unintended bias with respect to mentions of groups.

Let's begin with the benchmark.

Note of Structure

Due to the file size, some of the initial files are **omitted**.

- `Data` (missing): Due to data size, we didn't upload all the original data here. The complete data contains `sample_submission.csv`, `test_private_expanded.csv`, `test_public_expanded.csv`, `test.csv`, `train.csv`. We add the complement data `test_private_expanded.csv` to calculate the final score of the model.
- `Embeddings` (missing): The original embeddings used to train CNN.
- `models`: The trained benchmark CNN model and tokenizer.
- `submissions`: The submission files (to calculate the final score).
- `bias`: The bias analysis results.
- `kaggle`: The demonstration notebook for training (on kaggle) and inference (locally). Inside `bert-inference` is the BERT config and model files, checkout the kaggle kernel for details.
- `benchmark.ipynb`: A CNN model training using keras and tensorflow.
- `cnn_predict.ipynb`: Prediction using CNN model.
- `alignment.ipynb`: Quick check of the CNN prediction score to align with human intent.
- `detoxify_predict.ipynb`: Prediction using model from Detoxify.
- `metrics.py`: Functions to calculate final score.
- `eval_bias.ipynb`: Bias evaluated by Demographic Parity and Equalized Opportunity for all models.
- `final_score.ipynb`: Calculate the final score for all models.
- `tradeoff.ipynb`: Calculate the accuracy of models.

Benchmark

We trained the benchmark `CNN` model from ground up, the saved model and tokenizer are in the folder `/models`. The final score for `CNN` is `0.8811`.

Value Alginment

Checkout `alignment.ipynb`.

- Example comment 1:

> [Integrity means that you pay your debts.] Does this apply to President Trump too?

score: 0.0163

It is a common text, without toxicity.

- Example comment 2:

> Our oils read; President IS taking different tactics to deal with a corrupt malignant, hipoctitical , one way press! Idiots forget what witnessed them doing during the last election process.

score: 0.9308

It is toxic for the word `idiots` used, and the intense tone of speaking.

- Example comment 3:

> 'Work together'? Dream on. The liberals are already busy creating the next scandal/crime to accuse President Trump of since they lose the policy debate every time. They are sniveling statist traitors and they should be ground into the dirt as they will never ever let up in their qwest to destroy this country by any means available. Of that you can be sure.

score: 0.3644

It expressed an extreme viewpoint on politics, which is a bit aggressive, but not too toxic.

- Example comment 4:

> Well here we go again. Let's continue to subsidize the cost of power, costs that are out of control due to decisions made by the Liberal government. How nice to know the government has such contempt for the citizens of Ontario that it thinks we won't understand we are being bribed with out own tax dollars - tax dollars we will have to pay back with interest - to make re-election of these corrupt fools possible. Not enough that the deficit remains large, we will now increase that deficit to buy votes. This stinks, just like the power plant cancellation last election. Ontario voters, wake up and throw these bums out!!

score: 0.59185725

It a strong negative opinion towards the Liberal government using words like `corrupt fools`, which might be considered toxic.

Bias Assessment

The sensitive groups includes `genders`, `religions`, `races` and `illnesses`. Since `illnesses` has only one subgroup, we omit it here.

```
identity_columns = [
    'male', 'female', 'homosexual_gay_or_lesbian', 'christian', 'jewish',
    'muslim', 'black', 'white', 'psychiatric_or_mental_illness']
```

We compute the Demographic Parity and Equalized Opportunity concerning the above subgroups.

Demographic Parity

> Demographic Parity measures the equality of prediction outcomes across different sensitive groups. It is achieved when the probability of a certain prediction is not dependent on sensitive group membership.

```
genders
--------
max difference: ('female', 'homosexual_gay_or_lesbian') | 0.0526
min difference: ('male', 'female') | 0.0172
smallest ratio: ('female', 'homosexual_gay_or_lesbian', 0.4947567804985168)
largest ratio: ('female', 'male', 0.7501073444830227)

religions
--------
max difference: ('christian', 'muslim') | 0.0460
min difference: ('jewish', 'muslim') | 0.0110
smallest ratio: ('christian', 'muslim', 0.43583975652519097)
largest ratio: ('jewish', 'muslim', 0.8647654614383528)

races
--------
max difference: ('black', 'white') | 0.0023
min difference: ('black', 'white') | 0.0023
smallest ratio: ('white', 'black', 0.9838320275385117)
largest ratio: ('white', 'black', 0.9838320275385117)
```

We can observe a huge gap of ratios for around `0.3` between different subgroups. The model is biasd on `homosexual_gay_or_lesbian` for genders, `muslim` for religions, which are both underrepresented groups in reality.

Equalized Opportunity

> It means the protected and unprotected groups should have equal true positive rates.

```
genders
--------
max difference: ('male', 'homosexual_gay_or_lesbian') | 0.1165
min difference: ('male', 'female') | 0.0266
smallest ratio: ('homosexual_gay_or_lesbian', 'male', 0.6611570247933882)
largest ratio: ('female', 'male', 0.9225974025974025)

religions
--------
max difference: ('jewish', 'muslim') | 0.0384
min difference: ('christian', 'muslim') | 0.0106
smallest ratio: ('muslim', 'jewish', 0.8755364806866953)
largest ratio: ('muslim', 'christian', 0.9623917945733618)

races
--------
max difference: ('black', 'white') | 0.0123
min difference: ('black', 'white') | 0.0123
```

```
smallest ratio: ('black', 'white', 0.96448087431694)
largest ratio: ('black', 'white', 0.96448087431694)
```

We can still observe a gap of ratios between different subgroups. The model is biasd on `homosexual_gay_or_lesbian` for genders, `jewish` :) for religions.

Bias Mitigated model -- BERT

To get a more in-depth contextual embeddings of text, we replace the `CNN` model with a `BERT` model, which stands for Bidirectional Encoder Representations from Transformers.

Model Details

`BERT` is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labeling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pretrained with two objectives:

- Masked language modeling (MLM): taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally masks the future tokens. It allows the model to learn a **bidirectional representation** of the sentence.
- Next sentence prediction (NSP): the models concatenates two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not. This way, the model learns an inner representation of the English language that can then be used to extract features useful for downstream tasks: if you have a dataset of labeled sentences, for instance, you can train a standard classifier using the features produced by the BERT model as inputs.

We use the `bert-base-uncased` model from huggingface as the base model in this project.

Model	#params	Language
`bert-base-uncased`	110M	English

The `BERT` model was pretrained on `BookCorpus`, a dataset consisting of 11,038 unpublished books and English Wikipedia (excluding lists, tables and headers).

Model Structure

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): BertLayerNorm()
      (dropout): Dropout(p=0.1, inplace=False)
```

```
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): BertLayerNorm()
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): BertLayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=1, bias=True)
)
```

- **Embedding**: it contains a `word_embedding` and a `position_embedding`, to obtain dense vector representations of words and their relative location in the sentence. The vocabulary size is 30522, the maximum sentence length is 512.
- **Encoder**: it contains 12 `BertLayer`, each of which has an `attention` layer, a `intermediate linear` layer, and a `output linear` layer. These encoder layers get thick and meaningful representation inside the model.
- **Pooler**: it generates a fixed-length sentence representation by applying a linear and a non-linear transformation to the output of the last layer of the BERT model.
- **Classifier**: to map the output features to a single output value, used for binary toxicity classification. This output value can be interpreted as the probability of belonging to the positive class (i.e., the probability of a comment being toxic).

Model configuration


```
{
  "architectures": [
```

```
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.6.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```


Model Fine-tuning

We fine-tined the `BERT` model on the toxicity dataset, using `1200000` comments samepled from `train.csv` as training data, and `100000` comments as validation data. The text is first converted to the format `[CLS] Sentence A [SEP] Sentence B [SEP]`.

Hyperparameters and settings:

- seed = 1234
- learning rate = 2e-5
- batch size = 32
- epochs = 1
- weight decay = 0 for `['bias', 'LayerNorm.bias', 'LayerNorm.weight']`
- weight decay = 0.01 for the other layers
- optimizer = BertAdam
- loss function = F.binary_cross_entropy_with_logits()
- `amp.initialize()` is used for mixed precision training, reducing memory usage and computation time.

We trained the model on `kaggle` kernel, which takes around 8h. Checkout the kaggle kernel to see the training process and the details.

Final Score

In total, we compared **5** models: `cnn` from benchmark; `roberta-base-unbiased-small` and `roberta-base-unbiased` from Detoxify, `kaggle_bert` from kaggle, and `my_kaggle_bert` (our final model). Our model achieved **0.9351** in the final score, which is at the same level of well-trained open source models.

Checkout `final_score.ipynb`.

Final score:

- `cnn` : 0.8811
- `roberta-base-unbiased-small` : 0.9336
- `roberta-base-unbiased` : 0.9374
- `kaggle_bert` : 0.9383
- `my_kaggle_bert` : **0.9351**

Bias Assessment

We will do detailed comparison in later section, we first list the similar score below:

Demographic Parity

```
genders
--------
max difference: ('female', 'homosexual_gay_or_lesbian') | 0.0752
min difference: ('male', 'female') | 0.0222
smallest ratio: ('female', 'homosexual_gay_or_lesbian', 0.5068240190472617)
largest ratio: ('female', 'male', 0.7768968925002747)

religions
--------
max difference: ('christian', 'muslim') | 0.0693
min difference: ('jewish', 'muslim') | 0.0174
smallest ratio: ('christian', 'muslim', 0.4204354730670516)
largest ratio: ('jewish', 'muslim', 0.8548256285482564)

races
--------
max difference: ('black', 'white') | 0.0041
min difference: ('black', 'white') | 0.0041
smallest ratio: ('black', 'white', 0.97605561777024)
largest ratio: ('black', 'white', 0.97605561777024)
```

Equalized Opportunity

```
genders
--------
max difference: ('male', 'homosexual_gay_or_lesbian') | 0.0904
min difference: ('male', 'female') | 0.0226
smallest ratio: ('homosexual_gay_or_lesbian', 'male', 0.848575508261372)
largest ratio: ('female', 'male', 0.9621540762902019)

religions
--------
max difference: ('christian', 'jewish') | 0.0333
min difference: ('christian', 'muslim') | 0.0140
smallest ratio: ('jewish', 'christian', 0.9375)
largest ratio: ('muslim', 'christian', 0.9737124463519312)

races
--------
```

```
max difference: ('black', 'white') | 0.0072
min difference: ('black', 'white') | 0.0072
smallest ratio: ('black', 'white', 0.9874225227400789)
largest ratio: ('black', 'white', 0.9874225227400789)
```

Comparison with Baseline and other models

Checkout `eval_bias.ipynb`

Demographic Parity

A larger value of `Average ratio` here means the model is **less biased** towards different subgroups. Compared with `cnn`, our model's score is higher in `genders`, lower in `religions`, and around the same in `races`. Also, our model has higher score than some of the other open-source models.

```
======== genders =======

                              smallest ratio   |   largest ratio   |   Average
cnn                         : (0.4947567804985168, 0.7501073444830227, 0.6224320624907698)
roberta-base-unbiased       : (0.4729430058222235, 0.7583332431174286, 0.615638124469826)
roberta-base-unbiased-small: (0.36183704842428877, 0.6669919459947196,
0.5144144972095042)
kaggle_bert                 : (0.5068240190472617, 0.7768968925002747, 0.6418604557737682)
my_kaggle_bert              : (0.4812610988894485, 0.8147007306361423, 0.6479809147627954)

======== religions =======

                              smallest ratio   |   largest ratio   |   Average
cnn                         : (0.43583975652519097, 0.8647654614383528,
0.6503026089817718)
roberta-base-unbiased       : (0.5173602067597618, 0.8307460333778828, 0.6740531200688222)
roberta-base-unbiased-small: (0.36434329065908005, 0.645255474452555, 0.5047993825558176)
kaggle_bert                 : (0.4204354730670516, 0.8548256285482564, 0.637630550807654)
my_kaggle_bert              : (0.4486508039139616, 0.7285445697854457, 0.5885976868497036)

======== races =======

                              smallest ratio   |   largest ratio   |   Average
cnn                         : (0.9838320275385117, 0.9838320275385117, 0.9838320275385117)
roberta-base-unbiased       : (0.9138680690711831, 0.9138680690711831, 0.9138680690711831)
roberta-base-unbiased-small: (0.9755187383917685, 0.9755187383917685, 0.9755187383917685)
kaggle_bert                 : (0.97605561777024, 0.97605561777024, 0.97605561777024)
my_kaggle_bert              : (0.9793236611333098, 0.9793236611333098, 0.9793236611333098)
```

Equalized Opportunity

Again, A larger value of `Average ratio` means the model is **less biased** towards different subgroups. Compared with `cnn`, our model's score is higher in **all** `genders`, `religions`, `races`. Also, our model has almost the **highest** ratio among other open-source models.

```
======== genders =======
```

```
                                  smallest ratio   |   largest ratio   |   Average
cnn                          : (0.6611570247933882, 0.9225974025974025, 0.7918772136953953)
roberta-base-unbiased        : (0.8102589234664707, 0.9315363881401616, 0.8708976558033161)
roberta-base-unbiased-small: (0.8661654135338344, 0.959975696817802, 0.9130705551758183)
kaggle_bert                  : (0.8363963080944213, 0.9717879604672057, 0.9040921342808135)
my_kaggle_bert               : (0.848575508261372, 0.9621540762902019, 0.9053647922757869)

======== religions =======

                                  smallest ratio   |   largest ratio   |   Average
cnn                          : (0.8755364806866953, 0.9623917945733618, 0.9189641376300286)
roberta-base-unbiased        : (0.875036459852494, 0.9894346087949745, 0.9322355343237343)
roberta-base-unbiased-small: (0.8248910675381262, 0.9124331550802138, 0.86866211130917)
kaggle_bert                  : (0.9047210300429184, 0.9861146175208281, 0.9454178237818732)
my_kaggle_bert               : (0.9375, 0.9737124463519312, 0.9556062231759657)

======== races =======

                                  smallest ratio   |   largest ratio   |   Average
cnn                          : (0.96448087431694, 0.96448087431694, 0.96448087431694)
roberta-base-unbiased        : (0.9754498949483876, 0.9754498949483876, 0.9754498949483876)
roberta-base-unbiased-small: (0.9539218403547671, 0.9539218403547671, 0.9539218403547671)
kaggle_bert                  : (0.9685975609756098, 0.9685975609756098, 0.9685975609756098)
my_kaggle_bert               : (0.9874225227400789, 0.9874225227400789, 0.9874225227400789)
```

Tradeoff between Accuracy and Fairness

Concerning accuracy: (checkout `tradeoff.ipynb`)

- `cnn_auc`: 0.8479
- `my_bert_auc`: **0.90998**

Our model achieved both better final fairness score AND accuracy!

Reference

- https://arxiv.org/abs/1810.04805
- https://huggingface.co/google-bert/bert-base-uncased
- https://github.com/unitaryai/detoxify
- https://www.kaggle.com/code/yuval6967/toxic-bert-plain-vanila/notebook
- https://www.kaggle.com/code/abhishek/pytorch-bert-inference/notebook