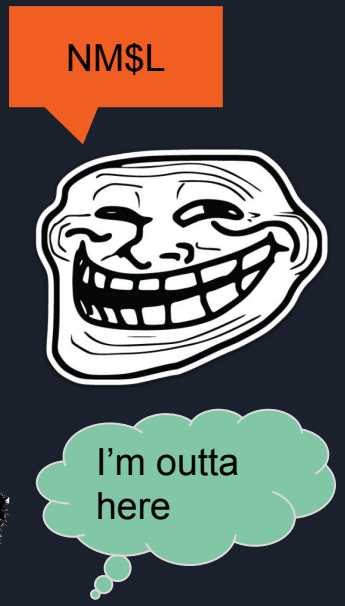
Toxic Comment Classification

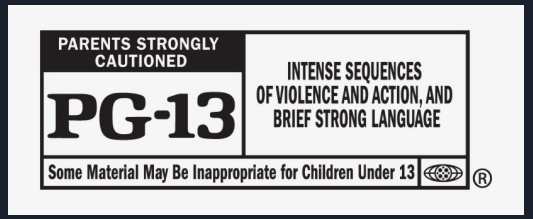
Sertan Akinci, Deyu Kong

The problem with Trolls



- Hostility & aggressive sentiment
- Unconstructive choral
- Intolerance & offensive statements towards minority communities
- Drives out reasonable & committed contributors

Warning: This project contains toxic language including but not limited to....



Kaggle Toxic Comment Classification Challenge:

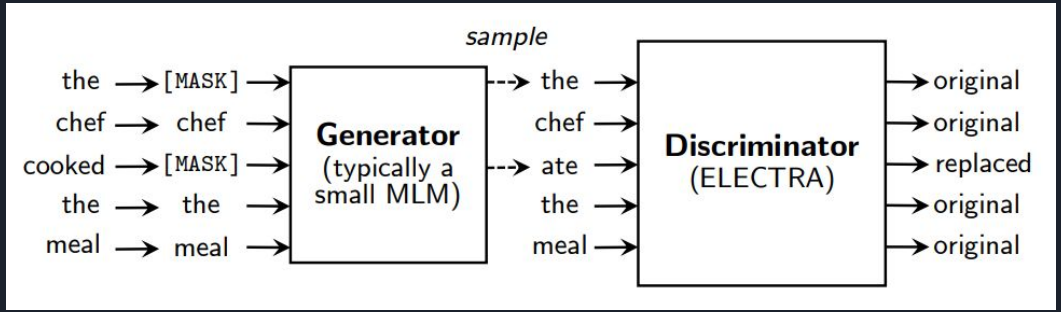
- 159,571 comments from Wikipedia talk page edit in training set
- Multilabel: toxic, severe_toxic, obscene, threat, insult, indentity_hate
- Distribution as follows:

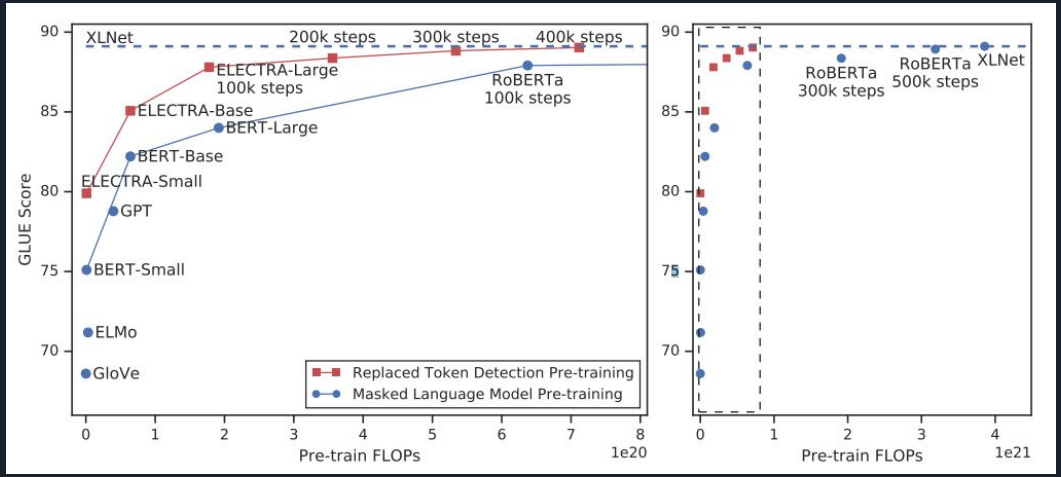
Label	# of Comments that contains this label
toxic	15,294
severe_toxic	1,595
obscene	8,449
threat	478
insult	7,877
indentity_hate	1,405

Key Challenge:

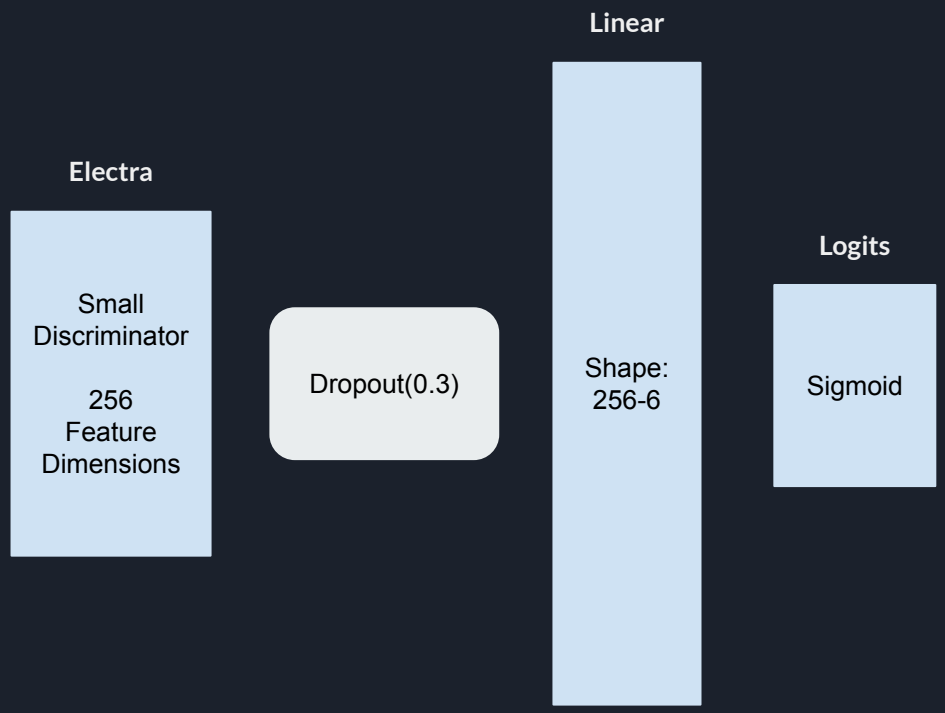
- Discern sentiments with slightly different aspect of negativity
- Differentiate constructive criticism from toxic trolling
- Maybe one, maybe more, maybe none
- Class Imbalance

Electra: Training MLM like GAN





Model


Electra
Small Discriminator
256 Feature Dimensions
Dropout(0.3)
Linear
Shape: 256-6
Logits
Sigmoid

```python
class CustomDataset(Dataset):

    def __init__(self, dataframe, tokenizer, max_len):
        self.tokenizer = tokenizer
        self.data = dataframe
        self.comment_text = dataframe['comment_text']
        self.targets = self.data['labels']
        self.max_len = max_len

    def __len__(self):
        return len(self.comment_text)

    def __getitem__(self, index):
        comment_text = str(self.comment_text[index])
        comment_text = " ".join(comment_text.split())

        inputs = self.tokenizer.encode_plus(comment_text, add_special_tokens=True, truncation=True,
            padding='max_length',
            return_token_type_ids=True
        )
        ids = inputs['input_ids']
        mask = inputs['attention_mask']
        token_type_ids = inputs["token_type_ids"]

        return {
            'ids': torch.tensor(ids, dtype=torch.long),
            'mask': torch.tensor(mask, dtype=torch.long),
            'token_type_ids': torch.tensor(token_type_ids, dtype=torch.long),
            'targets': torch.tensor(self.targets[index], dtype=torch.float)
```

```python
class ElectraClass(torch.nn.Module):
    def __init__(self):
        super(ElectraClass, self).__init__()
        self.l1 = ElectraModel.from_pretrained('google/electra-small-discriminator', return_dict=False)
        self.l2 = torch.nn.Dropout(0.3)
        self.l3 = torch.nn.Linear(256, num_classes)
        self.act = torch.nn.Sigmoid()

    def forward(self, ids, mask, token_type_ids):
        output_1 = self.l1(ids, attention_mask=mask, token_type_ids=token_type_ids)
        output_2 = self.l2(output_1[0])
        output = self.l3(output_2)
        pool = output.mean(1)
        return self.act(pool)
```

Dataloader

- Electra considers ‘input_ids’, ‘attention_mask’, ‘token_type_ids’ as model input

```
return {
    'ids': torch.tensor(ids, dtype=torch.long),
    'mask': torch.tensor(mask, dtype=torch.long),
    'token_type_ids': torch.tensor(token_type_ids, dtype=torch.long),
    'targets': torch.tensor(self.targets[index], dtype=torch.float)
}
```

- Implement a balance batch sampler by Issa Memari,

  etc. to address class imbalance issue

```
elif self.class_choice == "least_sampled":
    min_count = self.counts[0]
    min_classes = [0]
    for class_ in range(1, self.num_classes):
        if self.counts[class_] < min_count:
            min_count = self.counts[class_]
            min_classes = [class_]
        if self.counts[class_] == min_count:
            min_classes.append(class_)
    class_ = np.random.choice(min_classes)
return class_
```

Training

BATCH_SIZE=16

N_EPOCH=10 + 10

LR=1e-5, ReduceLROnPlateau, monitor on test loss

Optimizer: Adam()

Criterion: BCEWithLogitsLoss()

Metrics: accuracy - hamming #save model on this

Perhaps include f1 macro score

Training Results

Model	Epoch	Learning Rate	Test Acc	Eval Acc	Eval hlm
unbalanced	10 + 10	1e-4, 1e-5	0.96842	0.86508	0.03197
sampled_unsampled(least_sampled)	5 + 10	1e-5,1e-6	0.93753	0.89815	0.03842
sampled,unsampled,sampled(cycle)	5+10+5	1e-5,1e-6,1e-6	0.92907	0.89673	0.03880

Results and Demo

```
input1 = ":Dear god this site is horrible."  # 0,0,0,0,0,0
input2 = ":If you have a look back at the source, the information I updated was the correct form. " \
         "I can only guess the source hadn't updated. " \
         "I shall update the information once again but thank you for your message."  # 0,0,0,0,0,0
input3 = "DJ Robinson is gay as hell! he sucks his dick so much!!!!!"  # 1,0,1,0,1,1
input4 = "::No, he is an arrogant, self serving, immature idiot. Get it right."  # 1,0,1,0,1,0
input5 = "== All admins must DIE! == "  # 1,0,0,1,0,0
input6 = ", you stupid retard, or I will kill you!!!   ]]"  # 1,1,1,1,1,0
input7 = "a wild ass contains dick in the pussy"  # 1,1,1,0,0,0
input8 = "There is a special place reserved for you in hell, you nazi piece of shit"  # Original, Toxic, insult
input9 = "Sorry, I shouldn't argue with you. I'm outta here... to sleep with your MAMA!"  # Original, Obscene Sarcasm
```

[toxic, severe_toxic, obscene, threat, insult, indentity_hate]

Limitations & Potential Improvements

- Applying balanced batch sampler lead to “false positive” on neutral comments
  - Resulting in human intervention in responding to appeals, defeat the purpose of ML
- Try with other transformers: DistilBert and FNet
- Data Augmentation: Train on misclassified ones after each loop
- Better Balancing: include each label at least once instead of uniformly random?

Have A Great F*&@%ing Summer !!

Deyu Kong & Sertan Akinci

Reference

- Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020, March 23). *Electra: Pre-training text encoders as discriminators rather than generators*. arXiv.org. Retrieved May 1, 2022, from https://arxiv.org/abs/2003.10555
- Issamemari. *Issamemari/Pytorch-multilabel-balanced-sampler: Pytorch sampler that outputs roughly balanced batches with support for multilabel datasets*. GitHub. Retrieved May 1, 2022, from https://github.com/issamemari/pytorch-multilabel-balanced-sampler

Q&A