A primer of transformer models and fine-tuning on language tasks

Virender Singh
Guide : Dr. Hongyang Zhang
[1]Khoury College of Computer Sciences, Northeastern University

Abstract

Transformer-based BERT Language Models have become very popular recently and have shown outstanding results on some popular Natural Language Problems. However, there is very little understanding of what makes them so successful and what kind of information is learned, and what are the limitations of such models. In this research work, we study the properties of BERT and its applications in different fine-tuning tasks. We also explore the modification of BERT for the distributionally Robust Neural Networks for group shifts and work on some famous datasets like Movie Reviews, Civil Comments, and the MultiNLI dataset. We show how such robust algorithms for training can improve the performance of the BERT on worst group accuracies.

Introduction

Bidirectional Encoder Representations from Transformers or BERT are the state of the art Natural language Models. The BERT framework was pre-trained using the text from Wikipedia and can be fine-tuned with questions and answer datasets. BERT is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection; this connection is called attention. Older Language Models before transformers used recurrent neural networks (RNN) and convolutional neural networks (CNN) to carry out NLP tasks. The transformer is considered a significant improvement because it does not require sequences of data to be processed in any fixed order, whereas RNNs and CNNs do. With the help of Transformers, we can process data in any order and this enables training on large amounts of data than ever was possible before their existence. This helped in the development of pre-trained models like BERT, which was trained on massive amounts of language data before its release. Other language models read text input sequentially i.e. either left to right or right to left but cannot do both at the same time. BERT uses this bidirectional capability and is trained on two different Language tasks: Masked Language Modeling and Next Sentence Prediction. Masked Language Modeling trains to hide a word in the sentence and at the time of evaluation predict the word that has been hidden based on the context. The sentence has approximately 15% of the words hidden in the sentence. The Next Sentence Prediction task is trained on sentence pairs and the task is to predict whether two sentences have a logical, sequential connection or whether their relationship is totally random. BERT achieved state-of-the-art results in 11 natural language understanding tasks e.g. sentiment analysis, semantic role labeling, sentence classification, etc. The architecture of the Transformer based BERT model is discussed in detail in the Architecture section. Just having a good model like BERT is probably not enough to judge the learning of a model. Good models work generally well but fail on atypical examples. Such models rely on spurious correlations: misleading heuristics that work for most training examples but do not always hold. An example of spurious correlation would be classifying land and water birds. Here the model learns to classify land birds based on the land background and water birds based on the water body in the background. So the background in the image is a spurious feature. A model that learns the spurious correlation would be accurate on average on an i.i.d. test set but suffers high error on groups of data where the correlation does not hold. To avoid learning models that rely on spurious correlations and therefore suffer from high loss on some groups of data, we instead train models to minimize the worst-case loss over groups in the training data. Choice of how to group the training data allows us to use our prior knowledge of spurious correlations. We discuss the Distributionally Robust Optimization (DRO) algorithm that allows us to learn models that minimize the worst-case training loss over a set of pre-defined groups. We discuss the algorithm in detail which is a stochastic optimization algorithm with convergence guarantees to efficiently train group DRO models. We finally discuss our BERT model implementation and the fine-tuning on three different datasets: movie review dataset, Civil comments dataset, and MultiNLI dataset. Our first experiment shows that the BERT model is indifferent to the negation in the sentences using different proportions of positive and negative samples and proving that the in-group accuracy increases with the proportion of the group elements

in the dataset. In the second experiment, we work on the civil comments dataset and show that the in-group accuracy performance of the different groups present in the dataset improves by using the state-of-the-art BERT models and beating the top Kaggle results. Finally, we implement the group DRO algorithm on the MultiNLI dataset and report the worst group accuracies.

PRELIMINARY

Architecture

BERT model is a stack of multiple self-attention heads and for every input token in a sequence, each head computes key, value, and query vectors for creating weighted representations. BERT's key technical innovation is applying bidirectional training of transformers to language modeling. So a transformer is sequence agnostic and thus it is indifferent to input order sequences. Using this property it need not train from left to right or right to left, it can be trained bidirectionally. Thus can have a deeper sense of language content and flow than single direction language models. BERT is a Transformer based model and we discuss that as follows:
Transformer :
Layer Normalization and skip connections in the Transformer Short residual skip connections. It gives a transformer a tiny ability to allow the representations of different levels of processing to interact. The idea is similar to that of resnet: With the multiple paths, we can pass over a high level of understanding of the last layer of the previous layers. Idea of linear layer after multi-head self attention is to project the representation in a higher space and then back in the original space. So Overall process looks like this :
We have a sentence: Word embeddings of the input sentence are computed simultaneously. Positional encodings are applied to each embedding resulting in word vectors. Word vectors are passed to the first encoder block. Each encoder block has following layers, as shown in figure 1 :

- A multi-headed self attention layer to find correlations between each word.
- A normalization layer.
- A residual connection around the previous two sublayers.
- A linear layer.
- A second normalization layer.
- A second residual connection.

The transformer learn the entire sequence of words at once.

Pre-training Tasks

BERT uses two tasks for training: Masked Language Modeling and the Next sentence prediction task. Let's discuss these fine-tuning tasks in detail:

Masked Language Modelling :

- 15% of the words in each sequence are replaced with a [MASK] token as shown in the figure 2.

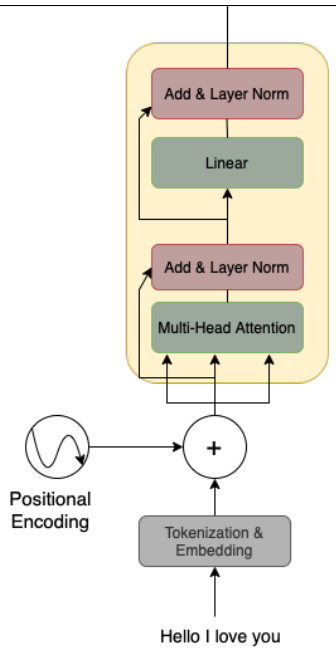

Figure 1: A transformer encoder

- The model attempts to predict the original value of the masked words, based on the context provided by the other non-masked words in the sequence.
- Here as shown in the figure: We add a classification layer on top of the encoder output and change the output vector to vocabulary by multiplying with embedding matrix. Calculate the probability of each word in the vocabulary with the softmax function. Here the BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words

Next Sentence Prediction :

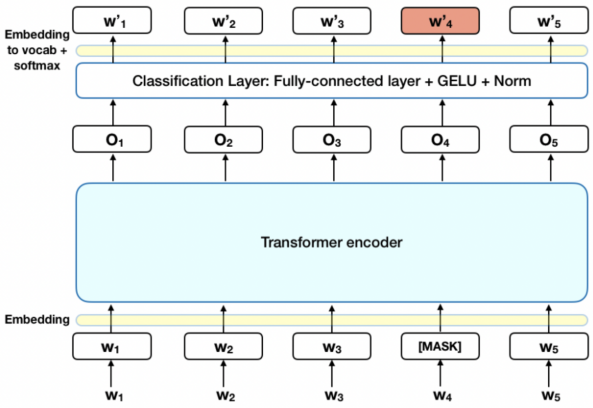

Figure 2: Masked Language Modelling task.

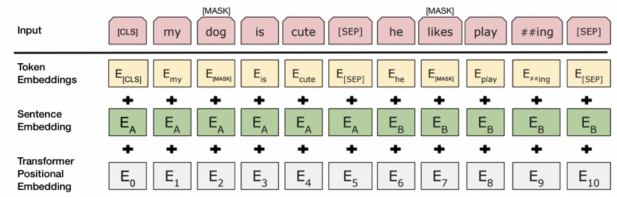

Figure 3: Next Sentence Prediction task.

- In this training, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document.
- During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document while in the 50% a random sentence from the corpus is chosen as the second sentence. "Assuming that the random sentences will be disconnected from the first sentence."
- A CLS token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
- A sentence embedding indicating sentence A or sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
- A positional embedding is added to each token to indicate its positional embedding. Input sequence goes into the transformer model [CLS] token output is transformed into a 2 x 1 vector using a simple classification layer as shown in figure 4.
- Calculate the probability of IsNextSequence with Softmax.

Downstream Applications

BERT has shown its effectiveness in almost all of the fine tuning tasks. Here we present our work on three most famous datasets. Movie Review Dataset, Civil Comments Dataset and the MultiNLI dataset. In the movies review dataset, we show the effectiveness of the BERT on predicting the positive or negative sentiment of the movie reviews. Civil comments task is to predict how effective BERT is in finding which comments are toxic and which comments are not. Finally in the MultiNLI dataset, we predict the model's efficiency in finding whether the second sentence is in support of, violates or neutral with the first sentence. We discuss the experiments in detail in the following sections.

Experiments

Movie review dataset : Reviews Sentiment Analysis

Although the BERT does understand the syntactic structure very well and somewhat of the semantics as well, it is indifferent to the meaning of the sentence overall. For example, an affirmative sentence and its negation won't really affect the understanding of the BERT. To understand this property, we design an experiment on the Movie Review dataset. Here, we have used different proportions of binary labels in different experiements on the dataset and try to see the results at both ends of the spectrum.
Results :

property	precision	recall	F1-score	Support
positive	0.97	0.98	0.98	10800
negative	0.79	0.75	0.77	1200
Macro Avg	0.88	0.87	0.87	12000
Weighted Avg	0.96	0.96	0.96	12000

Table 1: 90% positive and 10% negative

property	precision	recall	F1-score	Support
positive	0.91	0.92	0.91	6000
negative	0.92	0.90	0.91	6000
Macro Avg	0.91	0.91	0.91	12000
Weighted Avg	0.91	0.91	0.91	12000

Table 2: 50% positive and 50% negative

Property	Precision	Recall	F1-score	Support
positive	0.77	0.76	0.76	1200
negative	0.97	0.97	0.97	10800
Macro Avg	0.87	0.86	0.87	12000
Weighted Avg	0.95	0.95	0.95	12000

Table 3: 10% positive and 90% negative

Table 1 shows performance on 90% positive and 10% negative labels, 2 shows it for 50% each and table 3 shows the other end with 10% negative and 90% positive.
Implementation Code can be found here : code

Civil Comments Dataset

The civil comments dataset has data in the form of comments and the target is toxicity, we have several subgroups like white, black, asian etc. We first convert toxicity into binary features using a threshold of 0.5. Given a comment and presence of various subgroup items, we check how well the BERT performs on the test set.
train size = 1804874,
test size = 97320
valid size = 97320.
Implementation Code can be found here : code.
Reported scores for the experiment on the Civil Comments Dataset are shown in table 4 :

Group_Id	Subgroup	Subgroup_size	Subgroup_AUC	bpsn_auc	bnsp_auc
2	homosexual_gay_or_lesbian	579	0.840644	0.834775	0.973330
7	white	1312	0.848648	0.850010	0.973092
6	black	744	0.860686	0.821019	0.980129
5	muslim	1076	0.875703	0.883754	0.969875
4	jewish	387	0.893896	0.920917	0.958414
1	female	2790	0.924606	0.941426	0.961825
3	christian	1997	0.926684	0.954420	0.951775
0	male	2260	0.928112	0.928827	0.969396
8	psychatric_or_mental_illness	227	0.954249	0.923908	0.981811

Table 4: Results for the BERT model on Civil comments dataset and group wise results

The final metric score for AUC is **0.9685615613437878**
The worst group AUC score is **0.840644**
This score beats the top-ranking results from the Kaggle competition held here: kaggle

MultiNLI dataset

Given premise, determine if given hypothesis entails, neutral with or contradicts with it. The target labels for this task = {Entailed, Neutral, Negation}. We have the data in the form of pair of sentences. The total data size is 433000. Here we apply the group DRO problem described in Appendix which tries to improve the worst group accuracy. We run the algorithm on the MultiNLI dataset. The code can be found here: code Here we report the average and worst group accuracy for the data.
Average Accuracy : **82.8%**, Worst group accuracy : **77.6%**

Practical Implementation

Let's discuss the general pipeline for working in these general Language problems setup consisting of the BERT models. The general pipeline works as follows:

- Preprocess the dataset and handle the different features based on the requirement. For example, in the civil comments dataset, the target column is the toxicity column. The values present are in the range of 0 to 1. We do a thresholding there and set toxicity above 0.5 to true and below 0.5 to false.
- Load one version of the BERT pretrained models like BERT_base_uncased, BERT_base, Distill_roberta_base
- Create data loaders for train, test and validation. The pretrained model will be trained on these dataloader items.
- Load the pretrained models like bert_base_uncased
- Define the objective loss function on which you are going to fine tune the model. For example it might be an empirical loss objective or the DRO objective.
- For a given number of epochs and given batch size, train the model with your custom loss function.
- Get a figure from the evaluation function computing the accuracy metric.
- Save the checkpoint of fine tuning model after each N number of steps.
- One thing to experiment is the number of epochs to use for training as complex natural language models tend to overfit highly on more epochs.

BERT understanding

(Rogers, Kovaleva, and Rumshisky 2020) is a very nice survey paper on the details of the fundamental properties of the BERT. It studies in detail about the various papers on BERT which explain some property of BERT. We have seen the BERT results on various fine tuning tasks and noticed that these models are pretty effective for various kind of tasks. This raises a question about what exactly these models learn, are they intelligent enough to learn some real meaningful features of the data or do they rely on something intelligible. In the following section, we see what exactly the model learns and various components of the understanding.

Syntactic Knowledge

Syntactic knowledge is the knowledge of the grammatical syntax and understanding of the language as whole. For the case of BERT, not only the word order but syntactic tree is learned like a punctuation before noun or verb after object. Enough syntactic information seems to be captured in the token embeddings themselves to recover syntactic trees with evidence of hierarchical structure in ¾ tasks. For cloze tasks like Masked Language Modelling, we see that BERT is able to learn the relation between subject and verb. Moreover, BERT is also able to understand the usage of the negative polarity items, which are the tokens or words that help to detect the presence of a negative tone in the sentence. One interesting result shown by (Ettinger 2020) is that BERT does not understand negation and is insensitive to malformed input which is the opposite of what we learned before. So we might have two guesses here: either the BERT's syntactic knowledge is incomplete, or it does not need to rely on it for solving its tasks so the second option seems more likely.

Semantic Knowledge

One important aspect of any language model is to understand what the content in the sentence says. Like how well the model is able to make out the meaning of the sentence. For BERT, it struggles with the representation of

numbers, addition, and number decoding tasks showed that BERT does not form good representations for floating-point numbers. One of the possible reasons can be not having proper tokenization. The model does not actually form a generic idea of named entities, although having good accuracy is a Named Entity Recognition problem. However, a little change in the names changed 85% of the predictions. With the probing tasks trained on BERT, this information can be verified.

World Knowledge

BERT is unable to do pragmatic influence e.g. "He caught a pass and scored a touchdown. He enjoyed nothing more than a _____". Here the expectation is to have the keyword "Football" but BERT predicts "Baseball". BERT generalizes better to unseen data. BERT can guess the affordance, whether something can happen but cannot really reason about it. For e.g. "People walk into houses, it knows that people can walk into houses but does not know if houses are bigger than humans".

Limitations

If a linguistic pattern is not observed, it does not mean it is not present, and also if found it does not tell us how it is used. Different probing tasks lead to different complementary or contradictory results which make a single test not foolproof. A given method might favor one model over another. This makes sense as the first layer of BERT receives as input a combination of tokens, segment, and positional embeddings. However, we see a decrease in the knowledge of linear word order around layer 4 in BERT base, we however see an increased knowledge of hierarchical sentence structure. Papers have shown the most success in reconstructing syntactic tree depth from the middle BERT layers. Middle layers are performing best overall in the whole network. Final layers are most task-specific, this implies final layers will be trained for specific tasks like Masked Language Modelling or Next Sentence Prediction. Thus middle layers are most transferable.

Training BERT

Training BERT is consistent with our old report that middle layers contain most of the syntactic knowledge and are more transferable. Changes in the number of heads and layers appear to perform different functions. (You et al. 2019) have shown that with a large training batch size, BERT's training time can be reduced significantly with no degradation in performance. (Gong et al. 2019) here showed that we can do a recursive training where a shallower version is trained first and trained parameters are copied to deeper layers. This results in 25% faster training without sacrificing performance.

Conclusion

The results that we get from the movie review dataset bolster the hypothesis we had that BERT actually is not affected by the negation. As we see that the Precision and Recall are high for the class with higher data samples. For example, the first experiment has a higher percentage of positive samples and thus has higher Precision and recall for the positive sample, for experiment 2 we have an equal number of positive and negative samples so almost equal scores. On the other end of the spectrum, we have higher negative samples which results in better scores for the negative samples. We get very good results on the Civil comments dataset in terms of worst group and average AUC scores, we see that the BERT trained model is able to perform quite well in terms of worst group AUC score as well i.e. getting the highest worst group score possible. We also can improve over such natural language models for worst group scores using the Distributed Robust Optimization algorithm. We see the results of this algorithm on the MultiNLI dataset and see an overall improvement in the task as compared to the conventional Empirical Risk Minimization algorithm.

References

[Borkan et al. 2019] Borkan, D.; Dixon, L.; Sorensen, J.; Thain, N.; and Vasserman, L. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, 491–500.

[Ettinger 2020] Ettinger, A. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics* 8:34–48.

[Gong et al. 2019] Gong, L.; He, D.; Li, Z.; Qin, T.; Wang, L.; and Liu, T. 2019. Efficient training of bert by progressively stacking. In *International Conference on Machine Learning*, 2337–2346. PMLR.

[Rogers, Kovaleva, and Rumshisky 2020] Rogers, A.; Kovaleva, O.; and Rumshisky, A. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics* 8:842–866.

[You et al. 2019] You, Y.; Li, J.; Reddi, S.; Hseu, J.; Kumar, S.; Bhojanapalli, S.; Song, X.; Demmel, J.; Keutzer, K.; and Hsieh, C.-J. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Appendix

BERT Architecture Components

Let's discuss the architecture of BERT in detail starting from attention.

- **Attention** : The idea of attention is that the context vector Z should have access to all parts of the input sequence of just the last one, a direct connection with each timestep.

$$\alpha(i,j) = \frac{exp(e_{ij})}{\Sigma_{k=1} exp(e_{ik})} \tag{1}$$

Model learns to swap the order of words. The attention is described as below :
Attention(y_{i-1}, h) : score between previous state of the decoder as y_{i-1} and hidden state h = (h_1, h_2, h_3). Then finally softmax (y_{i-1}, h).

	Hello	I	Love	You
Hello	0.8	1	0.05	0.05
I	0.1	0.6	0.2	0.1
Love	0.05	0.2	0.65	0.1
You	0.2	0.1	0.1	0.6

Figure 4: Probability score matrix

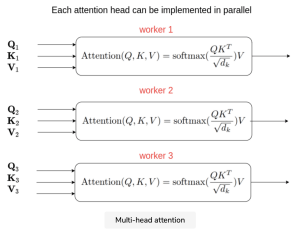

Figure 5: Multi-head attention

- **Self-Attention** : Attention mechanism to find correlations between different words(token) of the input indicating the syntactic and contextual structure of the sentence. For example, Here in the figure 4, we see the trained self-attention layer will associate the words "I" "You" with a higher weight than the word "hello". In practice, the transformer model uses three different representations of the embedding matrix Queries, Keys, and Values. We have three sets of weights, we define :
  Q = X Wq
  K = X Wk
  V = X Wv
  We then define Attention as following:

$$Attention(Q, K, V) = Softmax\frac{QK^T}{\sqrt{d_k}}V \quad (2)$$

- **Multi-headed Self-attention** : Multiple-headed attention as shown in figure 5 is an expansion of self-attention where we run through the attention mechanism several times output is called. Each iteration of self-attention gives rise to different heads(Map into different low dimensional spaces). Outputs of all heads in the same layer are combined and run through a fully connected layer. With multi-headed attention, the model has independent paths to understand the input.

$$Multihead(Q, K, V) = concat(head_1, head_2, ..., head_n)W_0 \quad (3)$$

$$Head_i = Attention(QW_{iq}, KW_{ik}, VW_{iv}) \quad (4)$$

  Heads are independent of each other, so computation can be done in parallel. The idea is to "allow us to use different parts of the sequence differently each time". Basically, each head will attend to different segments of the input so it captures better different contextual information by correlating words in unique names. Multi-headed attention enables the model to jointly attend to information from different positions. With single head, averaging inhibits this, single attention is prone to bad initialization.

Movie Review Dataset : Background

The IMDb dataset is a binary sentiment analysis dataset consisting of 50,000 reviews from the Internet Movie Database (IMDb) labeled as positive or negative. The dataset contains an even number of positive and negative reviews. Only highly polarizing reviews are considered. A negative review has a score 4 out of 10, and a positive review has a score 7 out of 10. No more than 30 reviews are included per movie. Models are evaluated based on accuracy. More information is available: here.

Civil Comments Dataset

Background

At the end of 2017 the Civil Comments platform shut down and chose to make their 2M public comments from their platform available in a lasting open archive so that researchers could understand and improve civility in online conversations for years to come. Jigsaw sponsored this effort and extended annotation of this data by human raters for various toxic conversational attributes. In the data supplied for this competition, the text of the individual comment is found in the comment_text column. Each comment in Train 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, test set examples with $target \geq 0.5$ will be considered to be in the positive class (toxic). The data also has several additional toxicity subtype attributes. Models do not need to predict these attributes for the competition, they are included as an additional avenue for research. Subtype attributes are:

- severe_toxicity
- Obscene
- Threat
- Insult
- Identity_attack
- sexual_explicit

Additionally, a subset of comments has been labeled with a variety of identity attributes, representing the identities that are mentioned in the comment. The columns corresponding to identity attributes are listed below. Only identities with more than 500 examples in the test set (combined public and private) will be included in the evaluation calculation. These identities are shown below.

- Male
- Female
- Homosexual_gay_or_lesbian
- Christian
- Jewish

- Muslim
- Black
- White
- psychiatric_or_mental_illness

Note that the data contains different comments that can have the exact same text. Different comments that have the same text may have been labeled with different targets or subgroups.

Examples Here are a few examples of comments and their associated toxicity and identity labels. Label values range from 0.0 - 1.0 represented the fraction of raters who believed the label fit the comment.

Comment: I'm a white woman in my late 60's and believe me, they are not too crazy about me either!!

- Toxicity Labels: All 0.0
- Identity Mention Labels: female: 1.0, white: 1.0 (all others 0.0)

Comment: Why would you assume that the nurses in this story were women?

- Toxicity Labels: All 0.0
- Identity Mention Labels: female: 0.8 (all others 0.0)

Comment: Continue to stand strong LGBT community. Yes, indeed, you'll overcome and you have.

- Toxicity Labels: All 0.0
- Identity Mention Labels: homosexual_gay_or_lesbian: 0.8, bisexual: 0.6, transgender: 0.3 (all others 0.0)

In addition to the labels described above, the dataset also provides metadata from Jigsaw's annotation: toxicity_annotator_count and identity_annotator_count, and metadata from Civil Comments: created_date, publication_id, parent_id, article_id, rating, funny, wow, sad, likes, disagree. Civil Comments' label rating is the civility rating Civil Comments users gave the comment.

Labelling Schema

To obtain the toxicity labels, each comment was shown to up to 10 annotators. Annotators were asked to: "Rate the toxicity of this comment".

- Very Toxic (a very hateful, aggressive, or disrespectful comment that is very likely to make you leave a discussion or give up on sharing your perspective)
- Toxic (a rude, disrespectful, or unreasonable comment that is somewhat likely to make you leave a discussion or give up on sharing your perspective)
- Hard to say
- Not Toxic

These ratings were then aggregated with the target value representing the fraction of annotations that annotations fell within the former two categories. To collect the identity labels, annotators were asked to indicate all identities that were mentioned in the comment. An example question that was asked as part of this annotation effort was: "What genders are mentioned in the comment?"

- Male
- Female
- Other gender
- Transgender
- No gender mentioned

Again, these were aggregated into fractional values representing the fraction of raters who said the identity was mentioned in the comment. The distributions of labels and subgroups between Train and Test can be assumed to be similar, but not exact. *Note: Some comments were seen by many more than 10 annotators (up to thousands), due to sampling and strategies used to enforce rater accuracy.

File Descriptions

- train.csv - the training set, which includes toxicity labels and subgroups
- test.csv - the test set, which does not include toxicity labels or subgroups
- sample_submission.csv - a sample submission file in the correct format

The following files were added post-competition close, to use for additional research.

- test_public_expanded.csv - The public leaderboard test set, including toxicity labels and subgroups. The competition target was a binarized version of the toxicity column, which can be easily reconstructed using $a \geq 0.5$ threshold.
- test_private_expanded.csv - The private leaderboard test set, including toxicity labels and subgroups. The competition target was a binarized version of the toxicity column, which can be easily reconstructed using $a \geq 0.5$ threshold.
- toxicity_individual_annotations.csv - The individual rater decisions for toxicity questions. Columns are:
  - **id** - The comment id. Corresponds to id field in train.csv, test_public_labeled.csv, or test_private_labeled.csv.
  - **worker** - The id of the individual annotator. These worker ids are shared between toxicity_individual_annotations.csv and identity_individual_annotations.csv.
  - **toxic** - 1 if the worker said the comment was toxic, 0 otherwise.
  - **severe_toxic** - 1 if the worker said the comment was severely toxic, 0 otherwise. Note that any comment that was considered severely toxic was also considered toxic.
  - **identity_attack, insult, obscene, sexual_explicit, threat** - Toxicity subtype attributes. 1 if the worker said the comment exhibited each of these traits, 0 otherwise.
- identity_individual_annoations.csv - The individual rater decisions for identity questions. Columns are:
  - **id** - The comment id. Corresponds to id field in train.csv, test_public_labeled.csv, or test_private_labeled.csv.

* **worker** - The id of the individual annotator. These worker ids are shared between toxicity_individual_annotations.csv and toxicity_individual_annotations.csv.
* **disability, gender, race_or_ethnicity, religion, sexual_orientation** - The list of identities within this category that the rater noticed in the comment. Formatted space-separated strings.

Evaluation Criteria

- **Overall AUC** : This is the ROC-AUC for the full evaluation set.
- **Bias AUCs** : To measure unintended bias, we again calculate the ROC-AUC, this time on three specific subsets of the test set for each identity, each capturing a different aspect of unintended bias. You can learn more about these metrics in Conversation AI's recent paper (Borkan et al. 2019).
- **Subgroup AUC** : Here, we 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, we 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, we 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.
- **Generalized mean of Bias AUCs** : To combine the per-identity Bias AUCs into one overall measure, we calculate their generalized mean as defined below:

$$M_p(m_s) = (\frac{1}{N}\Sigma_{s=1}^{N} m_s^p)^{1/p} \tag{5}$$

M_p = the pth power-mean function
ms = the bias metric m calculated for subgroup
N = number of identity subgroups
For this competition, we use a p value of -5 to encourage competitors to improve the model for the identity subgroups with the lowest model performance.

- **Final Metric** : We combine the overall AUC with the generalized mean of the Bias AUCs to calculate the final model score:

$$score = w_0 AUC_{overall} + \Sigma_{a=1}^{A} w_a M_p(m_{s,a}) \tag{6}$$

where: A = number of submetrics, m = bias metric for identity subgroup s using submetric a wa = a weighting for the relative importance of each submetric; all four w values set to 0.25

Distributionally Robust NN applications in Natural Language

Distributionally Robust Optimization helps us to learn models that instead minimize the worst case training loss over a set of pre-defined groups. Naively applying group DRO to an overparameterized network fails. These models perfectly fit the training data, models with vanishing average training loss have vanishing worst case training. Regularization is an important step for worst group generalization in the over-parameterized regime, even if it is not needed for average generalization. Finally, we introduce a stochastic optimization algorithm with convergence guarantees to efficiently train group DRO models. Standard ML models apply the Empirical Risk Minimization algorithm which tries to solve the following objective:

$$\theta_{ERM} = argmin_{\theta \in \Theta} E_{(x,y)\sim\hat{p}}[l(\theta;(x,y)] \tag{7}$$

$\hat{p}$: Empirical distribution

However, such models do not always result in the best models, because this leads to models with bad performance in the worst group case. Therefore, we jump to the DRO algorithm which improves upon the worst performance accuracy.

$$\theta_{DRO} = argmin_{\theta \in \Theta}[R(\theta) = max_{g \in G} E[x,y][l(\theta : (x,y))] \tag{8}$$

In a group DRO setting, groups are made of a combination of spurious features and labels. Group DRO learns models with good worst-group training loss across groups. The most popular applications of the Group DRO algorithms are following :

- Object recognition with correlated backgrounds:
  E.g. Waterbird dataset where we have multiple birds of two kinds: water and land birds. These birds have two sorts of backgrounds: land backgrounds and water backgrounds. Here the model can rely on the background instead of learning to recognize the actual object. We combine the waterbirds dataset with an image background from the places dataset. We label each bird = [waterbird, landbird]. Background = [Water background, land background].
- Object recognition with correlated demographics:
  Here models can learn spurious relations between the label and demographic information like gender and ethnicity. Example : celebA face dataset using hair color as the target and gender as the spurious attribute. It is highly correlated that the male celeb has black color and the female celeb has blonde color.
- Natural Language Inference:
  E.g. MultiNLI dataset. Here we determine if a given hypothesis is entailed by, neutral with, or contradicts a given

premise. We find spurious correlations between contradictions and the presence of negative words. Here we have three target labels : [entailed, neutral, contradictory] and presence of negative features [negation, no negation] is the spurious feature, so a simple model will learn that with negation in the sentence the prediction will be contradictory else not.

The general modes based on ERM and DRO have the poor worst group accuracy in the overparameterized regime. ERM and DRO achieve near-perfect training accuracy and vanishing training loss even in the presence of default regularization. They suffer from low worst-group accuracy because of the variations in the generalization gaps across groups. However, we see that the DRO improves the worst group accuracy under appropriate regularization. A simple regularization like l2 prevents ERM and DRO models from achieving perfect training accuracy. It substantially reduces the generalization gap for each group. ERM performs even poorer in the worst group accuracy however the DRO attains the worst group training test accuracy(84.6% 86.7%). Other regularization techniques like early stopping prevent overfitting and control the generalization gap for each group even in the worst-case group.

DRO Algorithm

The Distributionally Robust Optimization (DRO) algorithm on group DRO models works efficiently with convergence guarantees. Earlier algorithms on group DRO either used batch optimization algorithms which do not scale to large datasets or stochastic optimization algorithms without convergence guarantees. Objective function for DRO algorithm looks as follows:

$$min_{\theta \in \Theta} sup_{q \in \delta m} \Sigma_{q=1}^{m} q_g E_{x,y} \sim P_g[L(\theta; (x, y))] \quad (9)$$

The above equation is the optimization problem we intend to solve for minimizing the worst group loss. For our algorithm, we keep a distribution q over groups with high masses on high loss groups and update on each example. We interleave SGD on theta(symbol) and exponentiated gradient ascent on q. Here the key difference from the existing group DRO algorithms is that the q is updated using gradients instead of the group with the worst average loss at each iteration, this is important for stability and obtaining convergence guarantees. Run time complexity is the same as of the SGD algorithm.

Algorithm :

- **Initialization** :
  - Choose a group g at random from m groups
  $$G \sim Uniform(1, ..., m) \quad (10)$$
  - Sample x, y from the group g
  $$x, y \sim P_g \quad (11)$$
  - Update weights for group g
  $$q' = q_{t-1} \quad (12)$$
  set old weights to current one
  $$q'_g = q'_g exp(\eta_q L(\theta^{t-1} : (x, y))) \quad (13)$$
  - Renormalize q by dividing with the sum of q of all groups
  $$q^t = \frac{q'}{\Sigma q_g} \quad (14)$$
  - Update Theta
  $$\theta^t = \theta^{t-1} - \eta_\theta q(t)_g \nabla L(\theta_{t-1} : (x, y)) \quad (15)$$

Discussion

ERM and DRO achieve near perfect training accuracy and vanishing training loss even in the presence of default regularization. They suffer from low worst group accuracy and this is not because of poor worst group training performance but because of the variations in the generalization gaps across groups. We find no improvement in the worst group test accuracies since the models already achieve vanishing worst group losses on the training data. However with good regularization, it prevents ERM and DRO models from achieving perfect training accuracy. Substantially reduces the generalization gap for each group. ERM and DRO still achieve high average test accuracies. ERM performs even poorer in the worst group accuracy(now 21.3% and 37.8% respectively). DRO attains better worst group training accuracy(97.5% , 93.4%) better the worst group test accuracy. Regularization in general prevents overfitting, thus it controls the generalization gap for each group, even on the worst-case group. Good worst group test accuracy becomes a question of good worst group training accuracy. Even with regularization, generalization gap can vary significantly across groups for example, smallest group has a train-test accuracy gap of 15.4% compared to 1.0% of the largest group.