Toxic comment classification

Foundation of Deep Learning

Final Project Report

Team Name:

Zheng WAN&Wejing ZENG(in the first leaderboard)

Team Member:

Zheng Wan

Weijing Zeng

Leaderboard Score:

79.988

1. Problem statement

The objectives of our project are to use data analysis and deep learning techniques to address the following key issues:

1. **Creating an Unbiased Classifier:**
   The core of our challenge is to design a model that performs uniformly well across different demographic subpopulations. This involves ensuring that the model does not inherit any biases from the training data, particularly biases related to comments about certain demographic groups.
2. **Balanced Performance Across Subpopulations:**
   Our goal is to achieve high accuracy in toxicity classification not just on average but across all demographic identities. This requires a careful analysis of the training data and innovative modeling to ensure fair treatment of all groups.
3. **Evaluating Model Performance:**
   We will measure the model's performance using the 'Worst-group accuracy' metric, focusing on the model's ability to detect toxicity without relying on demographic identifiers. This approach is intended to ensure the model's sensitivity and specificity are balanced across different groups.

The primary goal of our classification task, along with the evaluation criteria, is to minimize the intergroup errors among different subgroups while maintaining the model's accuracy. For instance, the model should not misclassify a non-toxic comment as toxic merely because it involves a reference to the commenter being black. This underscores our commitment to reducing biases across demographic subgroups and ensuring that the classification is fair and accurate, devoid of any undue influence based on the subject matter of the comments.

2. Data description

We have five datasets available: two training sets, two validation sets, and one test set. Let's start with an overview of the training sets, as the validation sets are similar.

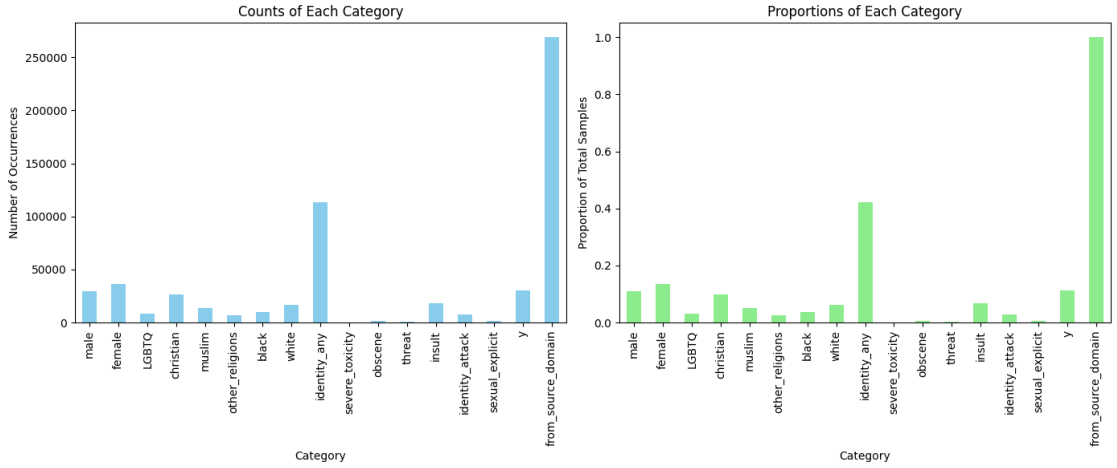
In the dataset ‘train_x’, there are 269,038 user-generated comments, which are the focus of our analysis. Our task is to classify these comments, marking them as toxic or non-toxic. A comment marked as 1 indicates it is toxic, while a 0 indicates it is non-toxic.

Also, in the dataset ‘train_y’, there are 269,038 rows of demographic Group Indicators. The first eight features (male, female, LGBTQ, christian, muslim, other_religions, black, white) represent different demographic groups. These features are crucial for ensuring that our model treats all demographic subpopulations fairly and does not exhibit bias towards any particular group. Additional features in train_y correspond to various aspects of toxicity.

3. Exploratory Data Analysis

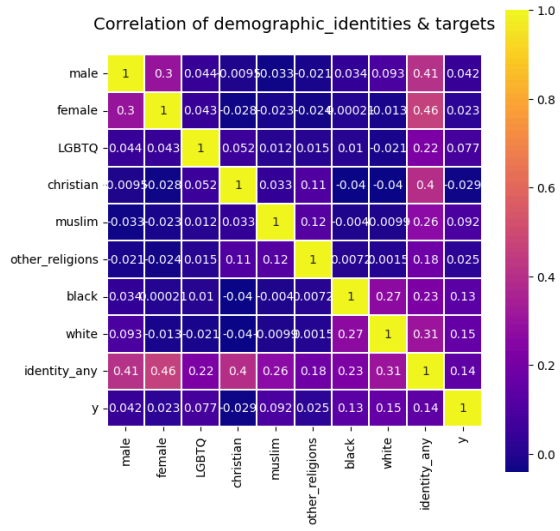
Composition analysis

Our Exploratory Data Analysis commenced with a deep dive into the dataset composition. To start with, we visualized the distribution of each feature in ‘train y’. Notably, while some groups like 'white' and 'identity any' are more prevalent in the dataset, categories such as 'black', 'LGBTQ', and 'other religions' are underrepresented in the comments, as highlighted by their minimal proportions. This discrepancy poses a challenge as it could potentially lead to a model bias, where the classifier's accuracy is lowest for these subgroups due to the lack of sufficient training data.

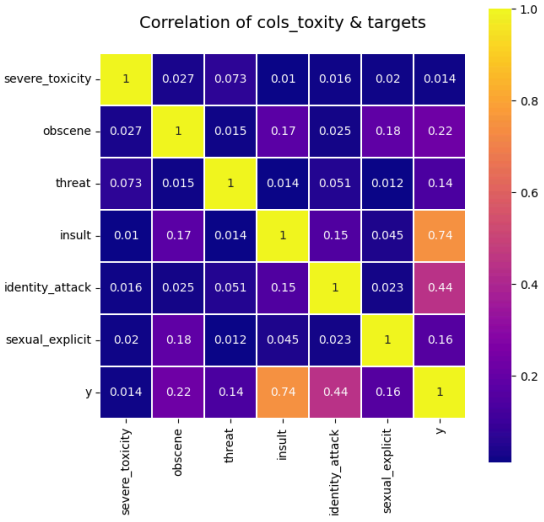


Correlation Analysis

The heatmap 1 illustrates the correlation coefficients between demographic features and the target variable. The coefficients indicate a generally weak correlation, suggesting that these demographic attributes do not have a strong direct relationship with the toxicity of a comment. This is a crucial observation, implying that the model should not heavily weigh these features when predicting toxicity to avoid bias.



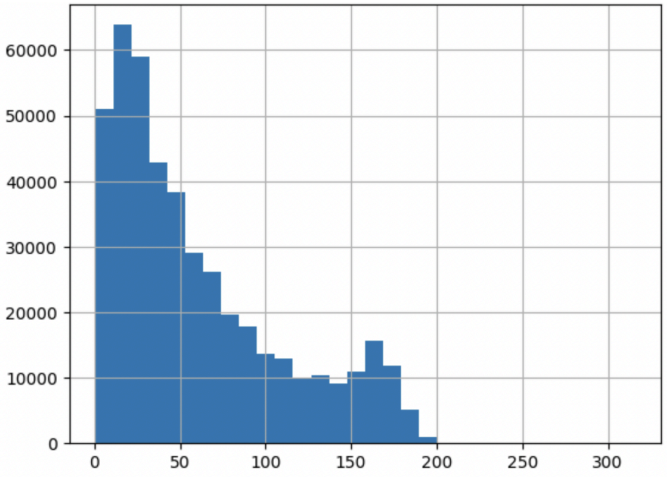
heatmap 1.



heatmap 2.

The heatmap 2 shifts focus to the relationship between different types of toxicity and the target variable. Here, we observe stronger correlations, notably between 'insult' and 'y' (0.74), as well as 'identity_attack' and 'y' (0.44). These stronger correlations suggest that certain types of toxic behavior are more indicative of overall toxicity, which could be pivotal in training the model to identify toxic comments with greater accuracy.

We analyzed the word count of comments across the datasets. Most comments were found to be under 200 words, leading us to set 200 as the maximum sequence length for our model, ensuring we capture the majority of the data while maintaining computational efficiency.



Additionally, we conducted a thorough examination for missing values within the dataset and found that there were no missing values. This eliminates the need for imputation strategies or handling missing data procedures in our preprocessing pipeline, ensuring a streamlined and efficient preparation of the dataset for the subsequent modeling phases.

4. Preprocessing

In the preprocessing phase, we focused on optimizing the text input to align closely with pre-trained embedding models. We employed Crawl Embedding and GloVe Embedding two mature word embedding files.

Crawl Embedding is developed by the Facebook AI Research team, offering a robust representation of language by incorporating sub-word information. This is particularly useful for us to handle words not encountered during training, known as out-of-vocabulary (OOV) words. It achieves this by breaking words into character n-grams, enabling the model to create word representations using these smaller subword units. In our project, we leverage this feature to enhance the model's grasp of word similarities and as a fallback for words that may not exist in the pre-trained vocabulary.

In our code, we load these pre-trained embeddings using the load embeddings function, which reads a file containing the word vectors and constructs a dictionary mapping words to

their corresponding vectors. The build matrix function subsequently builds an embedding matrix that our neural network models will use. This matrix is indexed by words found in our dataset's vocabulary, ensuring that each word can be converted into a vector recognizable by the model.

On the other hand, GloVe Embeddings, which is created by researchers in Stanford, is more like a social network analyst for words. Unlike Crawl, GloVe constructs its word vectors by directly capturing global statistics of word occurrences in a corpus to deduce the relationships between words.

In our project, GloVe embeddings are particularly valuable for understanding the semantic and syntactic nuances of words based on their contextual relationships. By leveraging the load_embeddings function, we incorporate GloVe's comprehensive word vectors into our embedding matrix, enriching the model's ability to discern and predict the nuances in text data.

We aimed to match the words from our text prep work with the pre-trained word lists we had, like Crawl Embedding and GloVe. This ensures that the tokens generated during text preprocessing have corresponding vectors in the embedding matrix, which is key for our neural network to make sense of things.

As demonstrated in the preprocess function, our approach avoided chopping words down too much, ensuring maximum compatibility with the pre-trained vectors and minimal loss of informative content. By avoiding overly aggressive preprocessing steps such as stemming or lemmatization, we retain the original structure and meaning of the text. We also applied specific text normalization techniques such as handling contractions, isolating symbols that we have embedding vectors for, and fixing quotes. Such a reprocessing resulted in a significant increase in embedding coverage, ensuring that our models are trained on data that closely corresponds to the pre-trained embeddings' understanding of language.

Through these embeddings, along with a careful preprocessing pipeline, we optimize the coverage of our vocabulary, significantly enhancing the model's ability to understand and classify textual data accurately. Our efforts increased vocabulary coverage from 24.72% to 73.74% and text coverage from 89.44% to 99.61%, proving the effectiveness of our text processing approach.

5. Model preparation

Data Preprocessing and Sequence Handling

Before model preparation, we took several preprocessing steps for all the text data. To keep things uniform and avoid losing important details, we converted all our text data into sequences of the same length, specifically 200 words each (MAX_LEN = 200).

For processing these text sequences, we used a special tool in Keras called a tokenizer. We set this tokenizer to handle up to 120,000 different features or unique words (MAX_FEATURES = 120000). This limit is large enough to understand a wide range of words and phrases, ensuring our model doesn't miss out on understanding any key elements in the text.

We built a word embedding matrix combining Crawl Embedding and GloVe Embedding. We made sure every piece of text our model reads is of the same length so it can learn and predict consistently.

After these strategies, we transformed all text data into serialized sequences of uniform length, while also constructing a word embedding matrix. This setup was essential to enable subsequent model processing.

Model Configuration and Architecture

We chose the Keras framework for its flexibility and ease of use, and selected a Bidirectional Long Short-Term Memory Network (Bi-LSTM) as our core model architecture. This type of network is excellent for working with text data as it processes words not just in isolation but in the context of surrounding words. This helps in understanding the overall meaning and sentiment of sentences better.

We deployed two distinct models (NUM_MODELS = 2) to enhance the reliability of our results. By comparing the outputs from these models, we could select the most accurate predictions. Each model processes batches of 512 data points (BATCH_SIZE = 512), balancing computational efficiency and the depth of learning from our extensive dataset.

The LSTM layer is set with 128 units (LSTM_UNITS = 128), an optimal number to capture complex patterns within the text data. Additionally, the model's complexity is further amplified by dense hidden layers, which are four times the LSTM units in size (DENSE_HIDDEN_UNITS = 4 * LSTM_UNITS). This design enhances the model's ability to learn and process the nuanced features in the data.

Integration of Advanced Pooling Techniques

We integrated Global Max Pooling and Global Average Pooling layers. This design choice allows the model to aggregate and interpret key signals from all time steps in the sequences, leading to a more comprehensive understanding of the text.

We utilized Global Max Pooling to identify and emphasize the most significant elements within each section of the text. This approach is akin to pinpointing the most prominent features in a vast landscape, ensuring that key signals are not overlooked.

Simultaneously, we incorporated Global Average Pooling to provide a holistic view of the text. This layer averages out the entire text, offering a balanced perspective that encompasses all aspects, not just the standout features.

Multi-Task Learning Framework

The model's output is split into two segments: the primary task focuses on identifying toxic comments, and the other task centers on recognizing other attributes like severe toxicity, obscenity, and threats. This way, the model doesn't just say if a comment is bad, but also what kind of bad it is. This helps our model be more accurate and really get what's going on in different types of toxic comments. This multi-task learning framework significantly enhances the model's performance on the primary task.

Customized Loss Function

We designed a custom loss function integrated with binary cross-entropy loss. This approach guides simultaneous training on both the primary task of identifying toxic comments and the auxiliary task of recognizing various toxic attributes like severity, obscenity, and threats.

A critical aspect of our loss function design was its emphasis on demographic fairness. We meticulously incorporated weights for different identity groups, such as gender and religion, into our loss function. This strategy was instrumental in mitigating biases against specific subgroups. Our objective was to enhance the overall accuracy of our model's classification capabilities while ensuring equitable treatment across diverse demographic groups. This attention to detail in the loss function is a cornerstone of our project, ensuring that our model's performance is not just accurate, but also fair and balanced across various societal segments.

Our model preparation involved a meticulous process of text data preprocessing, ensuring uniformity and consistency in the input. We utilized the Keras framework to build a Bi-LSTM Network adept at understanding complex text patterns. The model architecture, enhanced with Global Max Pooling and Global Average Pooling layers, was designed for comprehensive text analysis. Our multi-task learning framework addressed not only the primary task of identifying toxic comments but also auxiliary attributes of toxicity, improving overall accuracy. Crucially, we integrated a custom loss function with a focus on demographic fairness, aiming to reduce biases and ensure equitable treatment across various groups. This thorough approach underscores our commitment to developing a model that is both effective in its performance and responsible in its societal implications. Overall, our model is aimed to perform well in classifying and understanding diverse text, prioritizing both accuracy and fairness across demographic groups.

6. Evaluation and Results

During the training phase, our goal was twofold: to achieve accurate toxicity detection in comments and to ensure fair treatment across various demographic groups.This strategy helped us balance the need for thorough learning from our large dataset against the constraints of computational resources.

A key aspect of our training was the dynamic adjustment of the learning rate. We used a learning rate scheduler for this, which was crucial in fine-tuning our models. This approach allowed us to adaptively adjust the learning pace, enhancing the model's ability to generalize its learning to new data and avoiding overfitting.

After training, we focused on evaluating how well the model performed across different demographic groups. We used a specially designed function, group_accuracies, to calculate the accuracy within each demographic subgroup. This was vital to ensure that our model didn't just perform well overall but was also fair and unbiased, treating all groups equitably.

The results from the model training and evaluation phase were illuminating, offering a detailed view of the model's performance across different demographic categories. Our assessment focused on how accurately the model classified comments within each group, revealing significant variations in performance.

The accuracies for each group were as follows:

Male: 89.255%	Female: 90.3751%	LGBTQ: 80.3706%	Christian: 92.8791%
Muslim: 80.6635%	Black: 75.3632%	White: 78.7583%	Other Religions: 87.2211%

These figures demonstrated a higher degree of accuracy in categories such as 'Christian' and 'Female', suggesting that the model was more effective in identifying toxicity accurately in these groups. However, it became evident that there were disparities in performance, particularly in the 'LGBTQ' and 'Black' categories, where the accuracies were notably lower.

This disparity in accuracy across demographic groups underscored the need for further refinement of our model. It highlighted the challenge of ensuring balanced and equitable performance across all groups, a crucial aspect in the development of fair and unbiased AI systems. The lower accuracies in certain groups indicated potential biases in the model, which could stem from various factors, such as the representation of these groups in the training data or the model's sensitivity to certain linguistic patterns.

7. Reflection

Reflecting on our project, we acknowledged the challenge of reducing biases and improving accuracy for the worst-performing groups. Our observations led us to implement specific strategies aimed at achieving a more balanced model.

1. **Resampling Weights**
   To mitigate biases, we recalibrated the weights in our model. This resampling approach aimed to give underrepresented groups more influence during the training process, making the model more sensitive to these groups and thus reducing bias.
2. **Custom Loss Function**
   We devised a custom loss function that incorporates the importance of reducing subgroup biases. This function was specifically tailored to penalize misclassifications in underrepresented groups more heavily, pushing the model towards more equitable performance across all demographics.
3. **Data Augmentation**
   We explored data augmentation methods,which is ML-SMOTE, to artificially enhance our dataset. By generating synthetic examples, particularly for underrepresented groups, we aimed to provide the model with a more balanced view of the different subgroups. However, even though we observe an increase in subgroup accuracy, we decided to remove this method as the interpretation remained unclear.
4. **Merging Training and Validation Sets**
   We merged the training and validation datasets. This increased the volume of data available for training, providing the model with a broader range of examples to learn from.

While we got 79.988 in the leaderboard, there is always space for improvement. Future work could focus on more advanced techniques for balancing datasets or developing even more sophisticated loss functions that further penalize biases. Additionally, exploring other forms of data augmentation or even employing different model architectures or pretrained models like bert could provide new insights and advancements. Due to time and resource constraints, these were not explored in our current project but remain promising avenues for future research and development.

References:

1. YyzHarry. (2023). Change is Hard: A Closer Look at Subpopulation Shift. GitHub. https://github.com/YyzHarry/SubpopBench/blob/main/subpopbench/dataset/datasets.py
2. Swastik-25. (2020). ML Smote. GitHub. https://github.com/Swastik-25/Imbalanced-Data-with-SMOTE-Techniques/blob/main/Imbalanced_Data.ipynb
3. Hugging Face. (2020). Bert pretrained models. https://huggingface.co/unitary/toxic-bert?text=what+the+fuck
4. Kaggle User. (2021). LSTM related tutorials. Kaggle Notebook. https://www.kaggle.com/code/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru/notebook
5. Kaggle User. (2023). Some tips on preprocessing for GloVe. Kaggle Competition Discussion. https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification/discussion/92192