CYBERBULLYING DETECTION
DATA SCIENCE CAPSTONE PROJECT

Ayush Kumar
University of Georgia
Athens, GA
ayush.kumar1@uga.edu

Elise Karinshak
University of Georgia
Athens, GA
elise.karinshak@uga.edu

Lauren Wilkes
University of Georgia
Athens, GA
lauren.wilkes@uga.edu

ABSTRACT

Cyberbullying is a pressing challenge facing modern digital environments. This analysis builds upon previous literature to attempt to detect and classify Twitter content by type of cyberbullying. By focusing on data quality in preprocessing (specifically language detection), we find substantial improvements in model performance; controlling signal quality yielded single digit improvements in accuracy over previous literature in single models. Ultimately, this analysis contributes to methodology relating to Twitter data processing, and proposes future directions for research.

1 Introduction

Social media is transforming social interaction and digital connection. Social media has gained tremendous momentum over the last decade and is increasingly interwoven in our cultural and societal fabric. This trend was exacerbated by the COVID-19 pandemic; in the midst of isolation, people engaged with social media at all-time high levels. Accompanying this phenomenon, cyberbullying occurred at all-time high levels. To address this pressing issue, this report presents models to detect and classify harmful content. Models are trained and tested with a recently developed cyberbullying dataset [12].

2 Related Works

The dataset studied is the fine grain cyberbullying detection (FGCD) dataset introduced by Wang et. al in 2020 [12]. The FGCD dataset was constructed by utilizing online dynamic query expansion to reduce the inherent imbalance in traditional hate speech data-sets. Since the dataset was constructed in an unsupervised manner, we believe data cleaning and preprocessing to be of utmost importance to improve signal quality and reduce noise.

Wang only considered the classification problem with 5 levels leaving out the "not cyberbullying" tweets. Later work by Ahmed et. al [1] studied the 6 class problem with transformer architectures.

3 Preprocessing

Previous literature ignored or undervalued data cleaning and preprocessing. Both Wang [12] and Ahmed [1] removed special symbols. Ahmed et. al did no further preprocessing, and Wang also removed punctuation and the retweet flag "RT". We found substantial improvements in model performance by cleaning and processing the dataset. After preprocessing and cleaning 41,176 tweets remained. The cleaning process created a slight class imbalance.

	before	after
age	7,992	7,845
ethnicity	7,961	7,272
gender	7,973	7,218
not cyberbullying	7,945	5,996
other cyberbullying	7,845	4,959
religion	7,998	7,886

Table 1: Dataset Composition Before and After Cleaning

The "not cyberbullying" and "other cyberbullying" categories were the most affected by the cleaning. These categories were also the most confused in the literature [1]. This may point towards additional noise present in these categories before cleaning.

3.1 Dataset Description

The FGCD dataset [12] contains 47,692 total Tweets and a class label describing the type of cyberbullying (6 levels: "age", "ethnicity", "gender", "religion", "other cyberbullying", and "not cyberbullying"). The dataset contains 7992, 7961 7973, 7945, 7823, and 7998 tweets respectively.

3.1.1 Standard Preprocessing Techniques

First, standard preprocessing techniques were applied in the following sequence: converting text to lowercase, expansion of contractions, removal of special characters, removal of emojis, and standardizing encoding to ASCII. The FGCD dataset was encoded in UTF-8 and contained many HTML escape sequences. These sequences were unescaped and then all tweets were converted to ASCII with unknown characters ignored. Many unknown characters were sequences from other languages.

3.1.2 Twitter-specific Language

Twitter data also contains Twitter-specific language conventions. The proportions of certain Twitter-specific sequences (retweets, ats, hashtags) by class can be found in Table 2. There are substantial differences in occurrence by class. Therefore 'ats' and 'retweets' were replaced with generic tokens. Hashtag symbols, '#', were removed, but the language of the hashtag was left untouched. This decision is consistent with previous literature [12].

3.1.3 Language Detection

Natural language classification models rely on the key assumption that a corpus of words comes from a single shared language. Multilingual classification problems require adequate dataset sizes for each language and individualized models. After applying the standard techniques described in 3.2.1, tweet language was detected using a python port of langdetect [11]. Analysis of the initial dataset detected 32 languages. To reduce noise non-English observations were removed from the entire dataset. This reduced the overall dataset size from 47,692 tweets to 44,620 tweets. As can be seen in Figure 1, no other language has sufficient data for modeling.

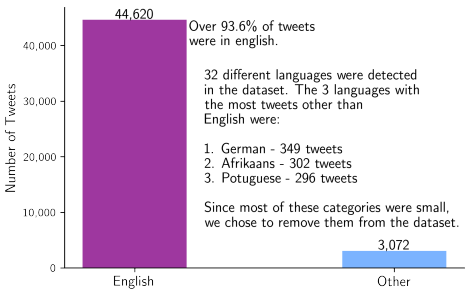

Figure 1: Language Distribution of Dataset

3.1.4 Removal of Duplicates and Long Tweets

After applying the above preprocessing techniques, empty and duplicate observations were removed.

Twitter has a hard limit of 280 characters per tweet. There were 685 tweets with more characters than this limit. We choose to remove these tweets because they were likely concatenations of multiple tweets, and manual labeling would've been required to properly classify them. We also removed many short tweets as they did not have strong enough signals for classification, or were simple retweets. This process removed around 3,000 empty or almost empty tweets.

4 Exploratory Analysis

Descriptive characteristics relating to the dataset dimensions and language are addressed in Section 3, as these characteristics informed the resulting preprocessing methods. By visualizing certain characteristics such as tweet length, closeness, and word frequency we can develop a better sense of the shape of the data.

4.1 Tweet Length

After preprocessing, the distributions of tweet lengths by class 2 display apparent trends. In 2017, Twitter increased the maximum tweet length from 140 characters to 280 characters. This is why most of the classes display bi-modal distributions with peaks just below 140 and 280 characters. Two interesting exceptions are the 'not cyberbullying' and 'other cyberbullying' classes which have no tweets over 140 characters indicating that these tweets were collected before 2017. These classes also show very similar distributions. In general tweet distribution does not vary much, and much of the distributional

	ats - '@'	hashtags - '#"	retweets - 'rt'
age	6.5%	9.3%	19.8%
ethnicity	61.1%	14.1%	13.1%
gender	57.1%	34.2%	17.9%
not cyberbullying	78%	41%	12.1%
other cyberbullying	86.1%	20.6%	12.3%
religion	51.9%	22.6%	37.9%

Table 2: Proportion of Twitter Specific Sequences by Class

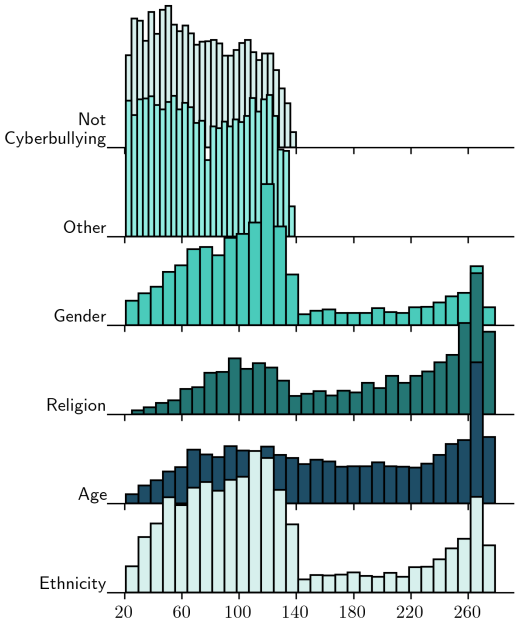

Figure 2: Length of Tweets by Class Label

variation can be explained by the proportion of tweets sampled before and after 2017.

4.2 Manifold Embedded Space

Natural language processing suffers from high dimensionality, which makes the data difficult to visualize. Dimensionality reduction techniques are often used visualize this data, but linear reduction techniques like latent semantic analysis (5.2.1) fail to capture the non-linearity of the data. We utilized uniform manifold approximation and projection (UMAP) [7] to embed the TF-IDF document term matrix in 2-dimensions for visualization.

The basic idea of UMAP [7] is that the data lies on a manifold in $\mathbb{R}^n$ and this manifold can be approximated using geodesic distances constructed from a nearest-neighbor graph.

While the dimensions resulting from the reduction do not have a precise interpretable meaning, they can provide a sense of closeness. Figure 3 shows that 'gender', 'ethnicity', 'religion', and 'age' are easily separable from the dataset, but 'not cyberbullying' and 'other cyberbullying' are distributed very similarly and difficult to visually differentiate. This interpretation of closeness is echoed by model performance both in the literature and testing. The 4 specific cyberbullying classes usually have exceptional performance, but 'not cyberbullying' and 'other cyberbullying' are often confused and score much lower.

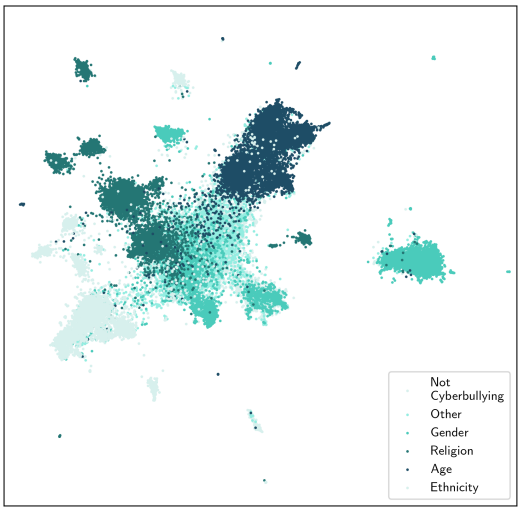

Figure 3: Manifold Embedding Space of Dataset

4.3 Most Common Words

After applying preprocessing techniques, the following word-clouds 4 represent the most frequently occurring terms in each class of the dataset.

5 Encoding Models

Language modeling attempts to quantify information present in a text relative to the entire document. In this context, the document is considered to be all tweets in the FGCD dataset. Encoding refers to the method used to vectorize

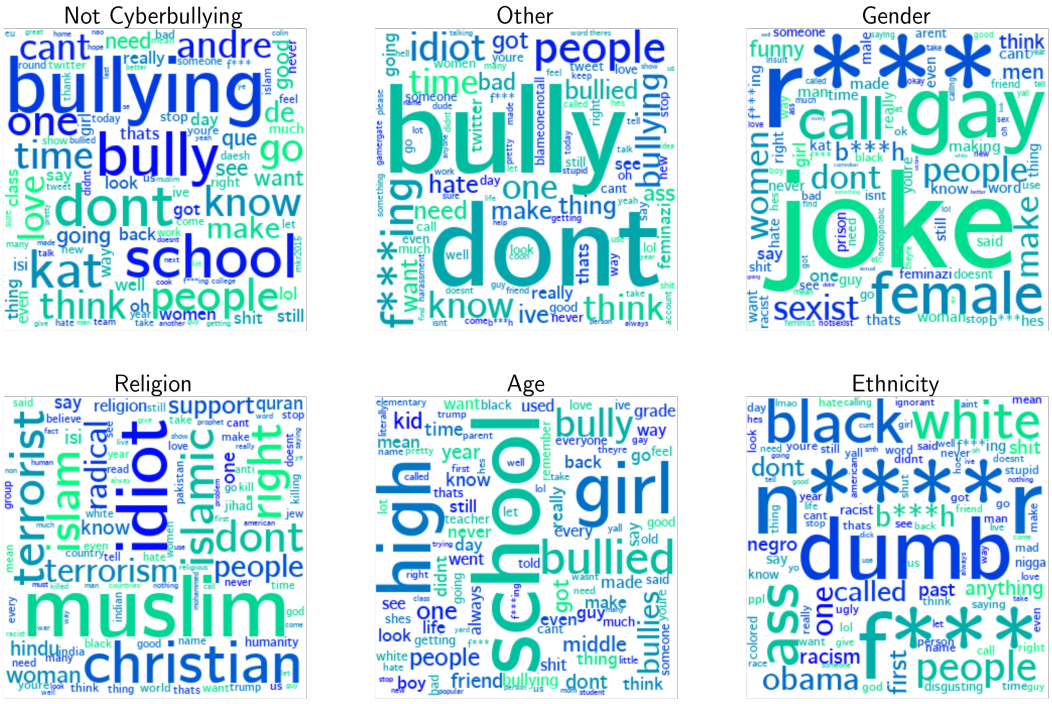

Figure 4: Word Clouds of Most Common Terms by Class

text and can be done through traditional statistical methods, or with deep learning.

We assess the following vectorization techniques: the bag-of-words (BOW), term-frequency inverse document frequency (TF-IDF), and XLNet [13]. BOW and TF-IDF are traditional techniques, and XLNet is a deep learning method based on transformers. For each encoding method, we assess performance with logistic regression and SVM models to determine which method achieves the best results.

5.1 Bag of Words

The bag-of-words model begins by assuming that all sentences in a document share a common dictionary. For the FGCD dataset, we consider the concatenation of all tweets to be the document, and each individual tweet as a an individual unit. We define a dictionary of k-tokens a vector $(d_1, d_2, ...d_k)$ as all the unique terms in a set of of size t training examples $(x_1, x_2, ..., x_t)$. For each training example, x_i a vector of size $k+1$ is created and assigned to zero. For each token in x_i the BOW vector, D_i is incremented for the corresponding term. This BOW embedding creates the document-term matrix D. We choose to create a dictionary from the terms present in the training set due to twitter-specific language, slang terms, and slurs. We can only do this because the dataset is large enough to create it's own dictionary. A custom dictionary also allows for reduced dimensional compared to a large pre-defined dictionary.

$$\begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,k} & d_{1,k+1} \\ d_{2,1} & d_{2,2} & \dots & d_{2,k} & d_{2,k+1} \\ \vdots & & \ddots & & \vdots \\ d_{t,1} & d_{t,2} & \dots & d_{t,k} & d_{t,k+1} \end{bmatrix}$$

The $k+1^{th}$ term represents unknown tokens. Since we construct our dictionary from all tokens present in the training set, this term in unused during training. During validation and testing, unknown terms may be present which were not a part of our training dictionary. We append these unknown terms as our $k+1^{th}$ term. Most of the terms will be binary $(0, 1)$, and some BOW embeddings force this constraint. The document-term matrix will be very sparse.

5.2 Term Frequency-Inverse Document Frequency

One of the shortcomings of the BOW model is that all terms are weighted equally. This gives common words within the document equal weighting as less common words. Oftentimes these less common words can be specific language, slurs, or otherwise important for classification tasks. The solution to some of

these short-comings is using the TF-IDF, which is the product of the term-frequency and inverse document frequency.

There are many way to define term-frequency and inverse-document frequency for a term, t and document d. We utilized the following definitions to calculate the TF-IDF embeddings from our document-term matrix.

$$\text{tfidf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t)$$
$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Where the numerator $f_{t,d}$ is raw count of a term in a given tweet. The denominator simply represents the total number of terms in a given tweet.

$$idf(t) = ln \frac{n}{df(t)} + 1$$

Where n is the total number of tweets in the dataset, and $df(t)$ is the proportion of tweets in the dataset that contain the given term, t.

5.2.1 Latent Semantic Analysis

BOW and TF-IDF embeddings result in very high dimensionality. Latent Semantic Analysis (LSA) reduces the dimensionality of a document-term matrix by utilizing singular value decomposition (SVD) and projecting into a lower dimension space similar to principle component analysis (PCA).

$$D = U\Sigma V^T$$

D is the document-term matrix, and the right side is it's singular value decomposition. This decomposition can be used to project the data into k dimensional space where $k << t$ where t represents the number of dictionary terms in the document-term matrix.

$$D_k = U_k \Sigma_k V_k^T$$

In testing, LSA incurred a substantial drop in performance. We decided to utilize regularization in model training instead of dimensionality reduction to prevent overfitting to circumvent this performance cost.

5.2.2 Bag of N-grams & It's TF-IDF

Bag of N-grams (BON) is a modification of BOW that allows for multi-term encoding to allow for greater context. For example, `apple pencil` and `red apple` are two distinct n-grams that may lose their context in the simpler single term BOW model. BON will result in a larger, but similar document-term matrix as the BOW method, and we can perform TF-IDF vectorization on this matrix as well. We tested a bag-of-bigrams and a bag-of-trigrams model as well, but high dimensionality resulted in overfitting and instability. Using LSA to reduce dimensionality yielded worse results than the simple BOW model and its LSA.

5.3 XLNet

XLNet [13] is an improved transformer architecture that iterates on BERT by leveraging auto-regressive pre-training. BERT [5] used bi-directional transformers trained using masked data, which corrupted the input and reduced the information available. BERT was state-of-the-art at release, but XLNet improves on it by using auto-regressive context within a transformer architecture. BERT's bidirectional training brings many strengths in terms of context dependency, but loses some of the benefits of auto-regressive factorization. In contrast, auto-regressive models can only model unidirectionally and lose some contextual bearing. XLNet seeks to combine the best of both worlds with a permutation language modeling task. For a given sequence of words, $(w_1, w_2, \cdots, w_n)$ the parameters, θ will see every possible combination of word orders. For the set of all possible permutations, Z_t, of a length, T the objective is modeled by the following equation.

$$\max_{\theta} \mathbb{E}_{z \sim Z_T} \left[\sum_{t=1}^{T} \log p_\theta(w_{Z_t} | w_{z<t}) \right]$$

XLNet does not directly permute the order of the text to preserve semantic meaning, but rather relies on the attention mask from transformers to achieve the permutation. The attention patterns create permutations while allowing for parameters to see bi-directional contexts and train on them. This unique architecture allows XLNet to overcome many of the shortcomings of BERT and other derived architectures.

The key advantage of neural architectures in language modeling is context-dependency. Both BOW and TF-IDF have no way of encoding where in a sentence a term occurs, or what words and phrases come before or after it. XLNet allows for both auto-regressive modeling of a sentence, and bi-directional context marking a significant improvement over traditional statistical methods. More details about model fine-tuning, implementation, hyperparamterization, and classification can be found in section 6.5.

6 Classification Models

The classification models selected for analysis are: logistic regression, support vector machine (SVM), and gradient boosted trees implemented with XGBoost[4]. XLNet [13] was also utilized as a classification model.

6.1 Hyperparameter Tuning for Support Vector Machines & Gradient Boosting

Some methods like Logistic Regression are mostly ambivalent to hyperparameter initialization, but others including SVM's and gradient boosted trees are highly sensitive. Proper searching of the hyperparameter space is key to optimal results. We utilized a tree-structured parzen estimator approach (TPE) [3] implemented with optuna [2]. Grid searches are computationally expensive, and may spend an inappropriate time exploring hyperparameter subspaces with poor results. The TPE approach uses bayes rule to optimize an objective, y, given a certain hyperparamterization x.

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

The conditional distrubtion of the hyperparamters based on the previous objective function is modeled by two distrbutions, $l(x)$ and $g(x)$. These distributions are differentiated by a threshold value $y*$, of our objective function. Since we want to maximize the objective function, which in our case is model accuracy, we want to draw more from $g(x)$. This intuition is the key driver of the TPE approach for hyperparameter optimization. We found stable results in little as 20 iterations, a considerable improvement over a grid search.

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y* \\ g(x) & \text{if } y \geq y* \end{cases}$$

6.2 Multinomial Logistic Regression

Logistic regressions traditionally model binary outcomes using a single probability. Extending the logistic regression into the multi-class setting is done in one of two ways; one-vs-rest (OVR) or mutinomial logistic (MLR). In a one-vs-rest setting for S class classification, $s - 1$ binary logistic models are created to predict the probability for each class. The highest probability determines the class label. Each of these regressions will suffer from some degree of error, and these errors compound creating error-propagation which lowers the overall quality of the model.

The multinomial logistic regression overcomes this issue by maximizing the likelihood over the entire multi-class distribution. A key assumption of the MLR is the independence of irrelevant alternatives (IIA). In essence, the presence of an additional class will not change the relative distribution of the other classes. If class label A is preferred to class label B, the presence of class label C will not change the relative preference of A over B.

For class labels, $\{1, \ldots, k\}$, $k - 1$ probability measures can be created with the form

$$\Pr[Y_i = j] = \Pr[Y_i = K]e^{\beta_j X_i}; \quad j \in \{1, \ldots, k-1\}$$

Where j is the class label. The assumption of independence of irrelevant alternatives is needed because we are fitting multiple regressions over the same data. The sum of class probabilities must be equal to 1 in the MLR setting.

We choose the multinomial setting because we believe IIA to be sufficiently satisfied based on the embedding map from Figure 3, as suggested by the general separability of classes. There may be some potential violations resulting from the similarity of distributions from the "other cyberbullying" and "not cyberbullying" classes. In testing, the MLR outperformed the OVR in every instance. MLR and OVR were implemented with scikit-learn [9] methods.

6.3 Support Vector Machine

Support Vector Machines (SVM) are a classification approach which applies a hyperplane to separate classes of observations in the feature space. We use scikit-learn's [9] implementation of SVC for multi-class classification. The algorithm seeks to optimally separate observations, allowing for a few misclassifications (incorporating a penalty when observations are misclassified or within the margin; maximal margin classification is a special case setting this slack constraint = 0). For both bag-of-words and TF-IDF, a linear kernel was used, and the constraints were set to 0.134 and 0.938 respectively (determined with optuna).

6.4 Gradient Boosted Trees

Decision trees partition the predictor space into regions, mirroring human-decision-making and providing increased interpretability. As an extension to decision trees, gradient boosting can be applied to improve performance. Gradient

boosting is a sequential learning approach fitting a large amount of small trees, fitting subsequent trees on the residuals of previous trees. In gradient boosting, tuning parameters are: the number of trees (B), the shrinkage parameter (λ) to control learning rate, and the number of splits in each tree (d) to control for depth (or complexity) of the sequential models.

In our analysis, we apply XGBoost [4], a gradient tree boosting system which has gained immense popularity in machine learning due to its efficient and high-performing design. Hyperparameter optimization was performed with optuna, and the hyperparameterizations in Table 3 were utilized for training.

	BOW	TF-IDF
Learning Rate - η	0.3957	0.2987
Max Depth	8	8
Steps	150	150
L1-regularization - λ	1.086e-5	1.002e-6
L2-regularization - α	1.836e-2	1.7602e-4

Table 3: Hyperparameters for XGBoost

6.5 XLNet

XLNet was implemented using the Transformers library developed by HuggingFace, and due to GPU constraints, we used XLNet Base which has 12 layers, 768 hidden units, 12 attention heads, and 110M parameters. Our model was trained on kaggle notebooks using the cuda GPU. In this implementation, we used XLNet base and fine-tuned the pre-trained model on our data. Before putting the text into the model, the text was tokenized into sub-words using HuggingFace XLNetTokenizer. Then, we converted the features, masks and label arrays to pytorch [8] tensors and created the data loaders for the test and validation sets, using a batch size of 32. With 250 input vectors, XLNet ouptuts 250 output vectors of size 768 and mean pooling was used to produce a single layer of size 768. We connect this layer to the fully connected layer to predict the 6 text classifications. The model was optimized using AdamW optimizer, with a learning rate of 2e-5 and a weight decay of 0.01. The test inputs were then predicted as the label with the highest prediction probability.

7 Results

For each classification model, accuracy and the macro F1-score were recorded to assess effectiveness. The macro F1-score reflects precision (accuracy of all positive predictions) and recall (proportion of positives which are accurately identified) in one metric. Accuracy is defined as the correct classification rate. For a testing set of size n, with c correct classifications, accuracy is simply c/n. The macro F1-score can be expressed as:

$$\text{F1}_{\text{macro}} = \frac{1}{|S|} \sum_{s_i \in S} \frac{\text{TP}}{\text{TP} + 1/2(\text{FP+FN})}$$

Where S is the set of all class labels, TP_i is the number of true positives for a given class, FP_i is the number of false positives for a given class, and FN_i is the number of false positives for a given class. For all given classes a simple arithmetic mean is taken to generate the macro F1-score. We choose to use the macro score over the weighted average because the dataset had minimal class imbalance.

Table 4 is a comparison the classification results for each model.

	f1-score	accuracy
BOW+Logistic Regression	0.8313	0.8527
BOW+SVM	0.8345	0.8540
BOW+XGBoost	0.8454	0.8645
TF-IDF+Logistic Regression	0.8242	0.8473
TF-IDF+SVM	0.8361	0.8575
TF-IDF+XGBoost	0.8360	0.8559
XLNet	**0.8549**	**0.8763**

Table 4: Classification Results

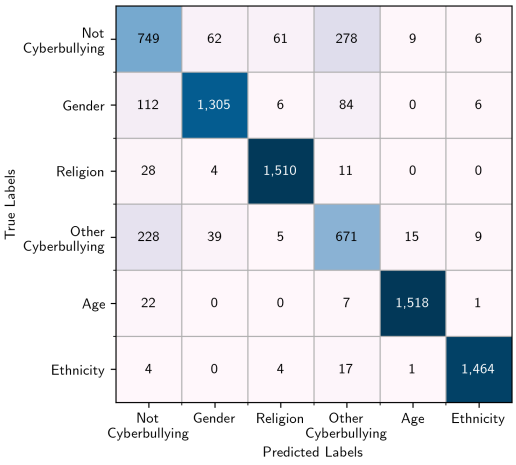

Figure 5: Confusion Matrix for XLNet[13]

8 Discussion

The results in Table 4 represent a substantial improvement over previous literature in the 6 class classification problem. Ahmed et.

al. [1] top performing single model was XLNet with a 84.19% accuracy score. Ahmed achieved 90.76% accuracy with an ensemble of 4 transformer architectures, RoBERTa[6], DistilBERT[10], XLNet [13], and BERT [5]. Our worst models outperformed any best single model, but did not match the performance of the transformer ensemble. The BOW models almost always performed better than TF-IDF models.

Our improvements in performance primarily come from extensive focus on data quality and preprocessing. We ensured language compatibility by filtering out non-English tweets. We improved signal strength by removing duplicate tweets and enforcing a minimum character requirement. These data quality improvements helped to lower the dimensionality of our embeddings, and greatly boosted model performance by 3 to 4 accuracy points in all instances.

While XLNet[13] performed the best out of all our models, it only marginally outperformed the BOW with Gradient Boosting. Since tweets are much shorter than many other natural language tasks, there may be less importance based on context, and terminology may play a greater role. Longer texts display much higher dimensionality and context-dependency is needed word-to-word, sentence-to-sentence, and paragraph-to-paragraph. The 280-character limit of tweets eliminate most but not all of this context dependency. While the transformer architectures are marginal improvements on traditional techniques most of the gains in model performance came from controlling data quality.

9 Conclusion and Future Directions

These findings illustrate the importance of data quality and signal strength in modeling. Ultimately, we conclude that higher quality and quantity of data is needed for substantial improvements in the classification task. There remains a high degree of confusion between the "not cyberbullying" and "other cyberbullying" classes; future efforts can seek to collect data from post-2017 Twitter (an apparent limitation in the current dataset).

Wang et. al [12] mentioned possible misclassification resulting from inaccurate class labels due to the automated nature in which the dataset was constructed. The classes applied are not mutually exclusive (for example, cyberbullying content could be classified as related to gender and age). Further improvements in the task may come from increasing the reliability of class labels.

We also recommend developing a stronger spell checker, as preliminary testing with a spell checker based on Levenshtein Distance did not result in gains in performance. However, spelling is often varied or incorrect on Twitter, due to its casual nature, the prevalence of acronyms, and varied spelling for emphasis (ex. repetition of a letter); we recommend further analysis of methods to account for these platform-specific nuances in language usage. These issues can also be overcome with a larger quantity of high-quality data allowing for models to more accurately learn the semantic meaning of platform specific language. Another source of high-quality data may be data augmentation.

Beyond the scope of this dataset, the challenge of cyberbullying is not unique to Twitter. We recommend future studies collect and analyze cyberbullying content from other social platforms, such as YouTube, Instagram, and Facebook, to achieve a more holistic understanding of malicious content online.

Furthermore, such analysis of cyberbullying content has significant societal implications across languages and cultures, and social media platforms facilitate multilingual exchanges. Thus, we recommend that future works collect multilingual data and extend cyberbullying detection to account for multilingual content.

Finally, while this analysis is focused on cyberbullying content, other types of harmful content also exist online. Similar classification methodology can be extended for the detection of such content.

References

[1] Tasnim Ahmed, Mohsinul Kabir, Shahriar Ivan, Hasan Mahmud, and Kamrul Hasan. Am i being bullied on social media? an ensemble approach to categorize cyberbullying. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2442–2453, 2021.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[4] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[7] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[11] Nakatani Shuyo. Language detection library for java, 2010.

[12] Jason Wang, Kaiqun Fu, and Chang-Tien Lu. Sosnet: A graph convolutional network approach to fine-grained cyberbullying detection. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1699–1708. IEEE, 2020.

[13] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. 2019.