|  |
| --- |
| **Twitter Hate-Speech Detection** |
|  |
| **Rohit Sunku**  210008043  Data Science MSc  [rohit.sunku@city.ac.uk](mailto:rohit.sunku@city.ac.uk) |

Problem statement and Motivation

Hate Speech is commonly referred to as any form of verbal, written or behavioral communication that uses derogatory/ discriminatory language against a person/ group based on what they are based on religion, ethnicity and nationality. The primary goal of our project is to design models for sentiment classification, and then do a final comparison of all models in the end to determine the best one. We aim to address a real-world issue where hate-speech and derogatory language should be identified and removed from social media platforms in order to make them friendly and accessible to users of all ages. Hate-speech detectors can prevent hateful comments/ posts from being shared before they must be removed. Our motivation is to make social media platforms more inclusive and friendly for all users, regardless of their age, gender or ethnicity.

Our approach involves implementing a data pipeline structure, starting with data pre-processing, followed by analyzing, designing, and evaluating machine learning models. Initially, we will evaluate simple models such as Multinomial Naïve Bayes, Decision Tree Classifiers, Random Forest Classifiers, K-Nearest Neighbors classifier and Logistic Regression, before moving on to a more complex transformer model BERT. By the end of the project, we aim to determine the best-performing model based on performance metrics. The use of hate-speech detectors can prevent harmful comments and posts from being shared and help create a safer and more inclusive online community.

Research hypothesis

In this paper, we are going to explore how much better transformers are than simple machine learning models for classifying hate-speech as a sentiment analysis question. The hugging-face library has different implementations of BERT, “BERT-base-uncased”, “BERT-large-cased”. Larger BERT models have up to 340M parameters whereas smaller BERT models have 110M parameters. These state-in-the-art models have already be trained on a large number of datasets, so only a small amount of fine-tuning can be done before running them on our dataset. Due to the size and memory of these models, we will have to run these on a GPU, not our standard CPU. Cased models have been trained on cased input text whereas uncased models have all text in lower-case letters before training.

Research Hypothesis 1: Are simple machine learning models enough to classify hate-speech?

Simple machine learning models take less time to train and are easier to implement. Small start-up companies or simple social media websites only used in private companies may not need to use complicated models, if they have only a slightly higher accuracy and are computationally more expensive. This can help companies make better financial decisions.

Research Hypothesis 2: BERT will outperform all machine learning models.

The other hypothesis for this study is that fine-tuning and experimenting on BERT will improve the accuracy of classification compared to traditional machine learning models. By leveraging a pre-trained transformer model with contextual understanding of languages, we can identify subtle nuances in languages indicative of hate-speech, leading to better classification accuracy. We test this hypothesis by comparing the performance of our hugging-face transformer model to more traditional machine learning models and then evaluating their accuracy, precision and recall. If the results support this hypothesis, it could demonstrate the potential of BERT models being used in hate speech detection paving the way for future research within this area.

1. Related work and background

Sentiment analysis is a well-established, thoroughly researched project to natural language processing, and there has been significant prior work in this area. In this section, we provide a survey of prior work related to our sentiment analysis problem and cite our sources in the references section.

Paper [1] by Bo Pang et al. (2002) features one of the first introductions to using a machine learning approach to sentiment classification using a feature-based model. It measures SVMs and Naïve Bayes models for their accuracy, changing the features and whether there are binary frequencies or binary presences within the document, for the purpose of feature selection and machine learning modelling. Different feature sets include unigrams, unigrams + bigrams and then adjectives too. SVMs outperformed the other models, however the differences were not too big, with accuracies ranging from 72-83% overall. We take a slightly different approach, focusing on the models and measuring different performance metrics and a wider range of models, as opposed to using unigrams and bigrams. However, we will follow a similar approach with our result table layout.

Attention mechanisms have also been widely used in sentiment analysis. The Paper [2] by Yequen Wang et al. (2016) proposed an attention-based LSTM model for aspect level sentiment- classification.

The paper [3] by Sanh et al. 2019 introduced Distil-BERT, a smaller and faster version of BERT which achieved a comparable performance on several NLP tasks. While this model only has half the number of layers with token-type embeddings and poolers, it retains 97% of BERT’s performance and is significantly smaller while also being constantly faster. The models in these papers have been trained for up to 90 hours which is significantly higher than the amount of time taken to train our models. Another transformer-based model XL-Net was trained and tested in paper [7] yang et al. (2019) achieving state-of-the-art results on several NLP benchmarks. XL-Net was shown to capture more long-term dependencies than BERT and outperform variations of BERT and GPT models all being trained on the same data. We will follow a similar approach with training all our models on the same data; with all hugging-face pre-trained library models being trained on the same datasets too.

Paper [9] by Gong et al. (2022) uses Data Augmentation in NLP tasks to improve performance of classification models. We will use under sampling and oversampling, as there is a notable class imbalance within our dataset and evaluate how this affects our results. Wang and Manning (2012) in paper [8] found bigrams to be effective features for sentiment and topic classification, proposing a simple baseline model. Meanwhile, Tabinda et al. (2021) in paper [10] also used transformers for sentiment analysis on social media data and showed the effectiveness of their proposed model over other models. Their work highlights the importance of selecting appropriate techniques for sentiment analysis on different types of text data and suggests that transformer-based models may be particularly effective for social media data.

Davidson et al. (2015) features the use of a logistic regression model with L1 regularization to reduce the dimensionality of the data. A comparison of all machine learning models like our study have been tested (SVMs, naïve bayes, decision trees, random forests, and logistic regression models). Results show that 40% of hate-speech was misclassified however this was in a multiclassification problem between hateful, offensive and normal tweets.

Malmasi et al. (2021) tests a linear SVM model in paper [12] on different word-representation n-grams (surface n-grams, word skip-grams, word representation n-grams). This addresses the problem of how the model captures its input, like paper 1, comparing unigrams (up to 8-grams) with/ without brown clustering and how this affects classification.

1. Accomplishments

* Task 1: Data loading/ Pre-Processing
  + converting the string to lowercase
  + removing twitter usernames
  + remove non-alphanumeric characters.
  + Remove any URLs.
  + Remove the string “rt” (retweet)
  + Remove stop-words
* Task 2: Data Analysis
  + Word-Cloud
  + Data Im-Balance Pie Chart
  + Most Frequent/ Least Frequent Words
  + Average number of characters and word count for each label (0 or 1)
* Task 3: Sampling
  + Create under sampled dataset.
  + Create oversampled dataset.
* Task 4: Tokenization, Lemmatization and Vectorization
  + Tokenization – change words into tokens
  + Lemmatization – change words to their root (remove suffixes and prefixes)
  + Count-Vectorizer
* Task 5: Machine Learning Models (SVM will be a baseline as it has proven to be good from our references)
  + Baseline Model: Dummy Classifier
    - Accuracy
    - Precision
    - Recall
    - F1-Score
    - Confusion Matrix
    - AUC, ROC and P-R curves
* Task 6: Experiment with different pipelines
  + Pipeline 1: CountVectorizer + SVM
  + Pipeline 2: CountVectorizer + Logistic Regression
  + Pipeline 3: Tfid Vectorizer + XG - Boost
  + Pipeline 4: Tfid Vectorizer + KN-Neighbors
  + Pipeline 5: Count Vectorizer + Decision Tree
  + Pipeline 6: Tfid Vectorizer + Decision Tree
  + Pipeline 7: Count Vectorizer + Multi-Nomial-NB()
  + Pipeline 8: Count Vectorizer + Random Forest
* Task 7: Calculate all metrics, confusion matrices and classification reports for comparison
* Task 8: Build XLNet, BERT and GPT models
  + Data Preprocessing: convert data to tensors.
  + Convert tensors into tensor datasets.
  + Convert tensor datasets into data loaders.
  + Build Model architecture.
* Task 9: Use Skorch (if PyTorch pure implementation is taking too long) to analyze BERT model
* Task 10: Final comparison of all models on binary classification metrics
  + Do the results support the hypotheses?

Could not fully complete Task 9 and 10.

1. Approach and Methodology

We approach this question initially by analysing our dataset to see what observations made/ any noise can be and if there are any underlying issues in the data.

A blue and yellow pie chart

Description automatically generated

We can see a notable class imbalance with a very high percentage of “No Hate” tweets in Figure 1. This will hinder our model results which needs to be trained on a similar number of “hate” and “no hate” tweets to avoid bias.

After data-processing, we lemmatise our data. Tokenisation is not necessary here, however a hyperparameter we can control is with our vectorizers reading data as unigrams, bigrams or n-grams where nER. Nevertheless, we tokenise our data to make it easier to lemmatise, which brings reduces words to their root meaning (e.g. greatest to great). Lemmatisation focuses on where the meaning of the words is crucial to ensure the model learns the underlying semantics of the text. Moving on, after using sci-kit learns dummy classifier model, we train, fit and predict on several different pipelines, calculating their accuracy, precision, recall and f1-score. The top two performing models will be optimised before being evaluated critically (through a series of ROC, AUC, P-R curves).

We finally compare our models with the dummy model results before a conclusion of our best model. We then move on to our transformer stage, where we can compare our best two models to our transformers.

Machine Learning models and transformers can be trained to identify patterns of language commonly associated with hateful speech. These models can learn to recognise these patterns and classify new examples of hate speech with high accuracy. Our two best models can be deployed in real-word scenarios for detecting hate-speech on Twitter.

We expect my models to perform better than the baseline model. The main limitation in our approach is simply due to the nature of the dataset. Classifying hate-speech can be considered a multi-classification problem which will improve implementation efficiency in real life contexts. For example, twitter can ban users with extremely hateful/ misogynistic tweets while issuing warnings to those with “minor” derogatory language. We could improve our baseline model by choosing a more sophisticated model.

We managed to complete working implementations of 10 different NLP pipelines.

**Tfidf Vectorizer + SVM Pipeline**

The TFIDF vectorizer and SVM pipeline both work to classify hate speech. The lemmatised strings are vectorised (converted to numerical vectors) by calculating the frequency of each word in each tweet and weighting the frequency by inverse document frequency giving more weight to words that are rare within the corpus. These TF-IDF vectors are then used as input for the SVM model trained on labelled data learning a decision boundary between hate and non-hate speech by finding the hyperplane separating the two classes by the largest margin. These two components work in sync to classify labelled data.

**BERT model**

BERT works through a method called transfer learning, which involves leveraging knowledge learnt from a source task on a target task. We add a classification layer to our BERT model for our tweets and then update the weights of this layer by training it on labelled data. BERT is bidirectional, so it can capture meaning from both left and right contexts and uses multi-head attention and feed forward neural networks to encode the input sequence (from input tensors) into a set of hidden representations to capture the contextual meaning of words. Residual connections and layer normalisation alleviate the problem of vanishing gradients during training.

We use sci-kit learn to test our machine learning models out, and then the “hugging face” transformers library for our BERT. To build extra-architecture on BERT, XLNet models, and convert our data into tensors, we use Pytorch. We use Skorch to train our models and evaluate our transformers, as pure pytorch is time-consuming and takes a long-time to run on jupyter. Matplotlib is used for graphical representations, and pandas and numpy are used for dataset handling.

The models implemented are: SVM, KNN, Decision Trees, Random Forests, Multinomial Naïve Bayes, XG-Boost and Logistic Regression, DummyClassifier(Baseline) and BERT, XLNet.

For BERT, as well as optimisation for our SVMs, jupyter notebooks on the CPU did not have to processing power or speed to handle this. I attempted to address this by shifting some code into google colab and running it on a GPU, with the XLNet and BERT models connected to a “torch cuda” device.

1. Dataset

A close-up of words

Description automatically generated with low confidenceA close-up of words

Description automatically generated with low confidenceExamining the dataset can help us understand the data, including the structure and content of text data which we work with alongside the format, length and vocabulary used in the text.

We observe word clouds (white depicting positive and black depicting negative “hate speech” both feature word clouds created depicting the most common words used in the hate-speech dataset for each class. Models will try to identify these key words in order to classify a tweet as being hateful or not.

Removing punctuation, symbols, links and stop-words are all important cleaning steps which will improve the accuracy of our models. Another reason is featuring engineering, which will help us select and engineer relevant features, which can capture underlying patterns and relationships in the text data. Lastly, from examining our dataset, we can carefully consider which models will be useful for application to this dataset. The dataset we work with contains three columns, tweets, labels and I.D. We perform feature-engineering and creation of new columns, number of characters in a tweet, number of words and then tokenized and lemmatized tweets too.

A screenshot of a computer

Description automatically generated with low confidence

This code snippet shows all the different columns. We use the lemmatized tweets in the “lemmas” column for training our models.

We notice significant class imbalance of 0 and 1, so we up sample the label “1” data to have the same number of values as those with 0 label (no hate). We will accommodate for the class imbalance by experimenting with oversampling and under sampling techniques on our data and observing how these have effect. The other issue is that the separate test csv file has got no labels on it, meaning we will have to split the training csv file dataset into train/test split for our machine learning models. We cannot test our data on the separate test .csv file since we cannot measure the accuracy/ f1\_score on this data. After final comparison of models, we will train our best model on the test dataset and its output can be used as true values.

Tweets also have lots of punctuation, symbols and stop-words in them which we remove before conducting sentiment analysis. Also, there is a significant amount of slang and abbreviations in these tweets. However, we do not investigate dealing with this within our project, however ways in which we can deal with slang language is by building a slang dictionary or using a context-aware model such as a neural language model or a transformer-based model.

Our twitter-hate-speech dataset is sourced from the Kaggle website and here is a report of the statistics:

Data Intake Report of Training Dataset

A picture containing text, font, number, screenshot

Description automatically generated

Our test dataset which we use in our final stage has 2 features. We also include statistics for under sampled and over sampled datasets.

Dataset Source: [Twitter hate speech | Kaggle](https://www.kaggle.com/datasets/vkrahul/twitter-hate-speech?select=train_E6oV3lV.csv)

Class Statistics

A picture containing text, font, screenshot, line

Description automatically generated

These are the statistics for the number of entries for each class and the percentage class imbalance for our original dataset of the minority class.

**6.1 Dataset Pre-Processing**

After loading the dataset, we begin by converting all the text to lowercase to ensure there are no errors in the model due to case. We then remove any URLs in the text corpus as this will introduce noise within the dataset. We proceed by removing twitter usernames (strings starting with "@") which are not relevant for classification and may not provide that useful information on the model. After this, we remove non-alpha-numeric characters ensuring focus only on words and are not distracted by special characters. Lastly, we remove stop-words which do not have much meaning in classification. These techniques are used in pre-processing and are aimed at improving the quality of the text data, reducing noise and increasing the accuracy of classification models.

The difficulties associated with pre-processing techniques include identifying and removing irrelevant information, such as URLs, special characters and Twitter usernames, while preserving important information that can improve model accuracy. Additionally, stop words removal can sometimes result in the loss of important words or phrases that are critical to understanding the sentiment of the text. It is therefore important to strike a balance between cleaning the data and retaining the relevant information.

1. Baselines

Sci-kit learn has a “dummy-classifier” model which we will train on our dataset first and evaluate the accuracy and prediction of.

When we initially test our baseline model without any vectorization, we get a very high accuracy score of 0.927, which is higher than some of our other models. Nevertheless, we do not consider the major class imbalance when splitting the data here. When we try with our up sampled and down sampled data for high accuracy then we get 0.495 and 0.493 respectively.

We will be using the up sampled dataset for the rest of our models since it has more data than our down sampled dataset.

Baseline models are useful for establishing a lower bound for model performance, as any model which performs worse than this is not able to predict the major class correctly and should be excluded. As we can see, sometimes a simple baseline model is better than a complicated machine learning algorithm for predicting noisy, imbalanced data, such as we notice here.

1. Results, error analysis

Results Table

A picture containing text, screenshot, number, calendar

Description automatically generated

Our results table shows a comparison of the different machine-learning models used. Note that model pipelines labelled 1 feature a “count vectorizer” whereas 2 indicates the “tfidf vectorizer” before being fitted on the model. We observe the model hierarchy: (1. SVM, 2. Random Forest, 3. Logistic Regression, 4. Decision Tree, 5. Multinomial naïve bayes, 6. XG-Boost, 7. KNNs). We also see a colour scheme suggesting the performance of the model with (Green: Amazing Performance: 0.980 – 1.000, Yellow: Decent Performance: 0.940 – 0.980, Orange: Adequate Performance: 0.900 – 0.940, Red: Poor Performance: 0.750 – 0.900, Rouge: Bad Performance: <0.750).

We find the most accurate pipeline to be the “SVM + Tfidf Vectorizer” followed by “Random Forest + Tfidf Vectorizer”. KNNs were the least favorable method with a poor accuracy of 0.777. All our pipelines passed our baseline model statistics by a significant margin.

**ROC/ AUC Curve Analysis**

A picture containing text, line, screenshot, plot

Description automatically generated

A picture containing text, line, plot, diagram

Description automatically generatedA picture containing text, line, screenshot, plot

Description automatically generatedROC curves are a useful tool in assessment of diagnostic test performances and offer a comparison on the true positive rate and false positive rate. We can see from the figures labelled below that our SVM pipeline has a higher true positive rate per false positive rate with the curve having a slightly steeper gradient than our Random Forest model with a “hug” at the top left corner of the plot being closer to the corner. Both models feature a high TPR and low FPR at all thresholds. Figure 8 shows the ROC curve for our baseline model displaying a proportional TPR and FPR rate. Our SVM pipeline significantly outperforms our dummy classifier model as a binary classifier with an AUC of 0.5 suggesting our baseline is no better than random guessing. Our pipeline models both have AUC scores approximated at 0.99, which is very close to a perfect classifier.

A picture containing text, screenshot, line, diagram

Description automatically generated**Precision-Recall Curve Analysis**

A picture containing text, screenshot, line, diagram

Description automatically generatedPrecision-Recall curves display the tradeoff between Precision and Recall values at different thresholds. Both Random Forest and SVM models have a high value straight line close to 1, indicating that precision and recall values are the same throughout and are at high values. This suggests high performance in both models.

A graph with a red line

Description automatically generated with low confidenceOverall, from our P-R, ROC and AUC curves, we can say that our final SVM model is significantly better than our baseline (by 0.49 in AUC (0.99: 050)) and therefore should be considered.

**Optimization**

We run a GridSearchCV on our pipeline, and found the following hyperparameters:

A picture containing text, font, white, line

Description automatically generated

Our best SVM + TfidfVectorizer had the optimal hyperparameters to be:



Our optimized pipeline had a final accuracy of 0.997.

**Annotated Error Analysis**

In this subsection, we are going to analyze the misclassified inputs of our final SVM model.

A screenshot of a computer

Description automatically generated with medium confidence

We observe that in Sample 2539, a null value has been found perhaps after data cleaning and pre-processing, this tweet may have just had special characters and stop words. Also sample 1689, is incorrectly classified as it is not offensive but a controversial opinion. In a professional environment, there would be no scrutiny for a comment like this being made, however the model picks up words like “black” and “white” which have been used in other racist hate tweets before so it is classified as “hate speech” (1). Some of the other tweets have been predicted correctly and are considered misclassified like sample 1123 and sample 1264, suggesting problems with the dataset labelling. We notice the main issue with our model is that it identifies specific words in hate-speech due to its simplicity in comparison with transformers, as it only is analyzing “unigrams”. While it has the potential to analyze “bigrams”, the nature of XLNet can analyze relationships between words and understand contexts better.

1. Lessons learned and conclusions.

**Evaluations/ Conclusions/ Limitations**

Our results do not support our second hypothesis statement, with SVMs outperforming BERT and XLNet. This is due to the small size of our dataset.

We will only focus on one dataset, and this is a limitation within our project as a wider scope of data will perhaps vary the results a lot more and cause some differences. Only training the data on a few classifier tweets with some duplicate data involved after up sampling is not enough to build a reliable model. Our outcomes support our hypothesis 1 that machine learning models are enough to classify hate speech, however in the grand scheme of things this still needs further testing on more tweeting data. Also, if we adopt a multiclassification method like we do within our sources, the XLNet and BERT models may have outperformed our simple machine learning models.

Issues we ran into involved optimizing our SVM model. This again is simply due to running of the CPU/ GPU, where we would ideally use the cloud and run optimization using batch parallelization on different virtual machines to achieve optimal results. Another way of improving our results would be to try more complicated pipelines, or experience with WordNet lemmatizes too. The other issue we faced was the testing data had no labels, so we had to split and use the training dataset instead. Further research should have been done into combining more datasets.

Tf-idf vectorization proved to be a better vectorizer than the count vectorizer for this question due to its nature being better at handling shorter texts and the ability to handle high frequency words with lower importance in the corpus.

**Lessons Learnt**

Machine learning models are surprisingly very accurate on small, balanced datasets and are enough. We can use these methods as they are more cost-effective and take less time train.

**Future Work**

Experimenting with different attention transformers and PyTorch/ TensorFlow neural networks is another area we can investigate. Future work on this project will include combining more datasets and then evaluating it on a higher range of models. We could also look at whether breaking down twitter hate speech by different word n-grams affects the accuracy and which “n” is the most efficient. We could also explore variations of the model, including aLBERT as well as XLNet and GPT models.

**Word Count Table**

|  |  |
| --- | --- |
| **Section** | **Word Count** |
| Introduction | 231 |
| Research Hypthoses | 306 |
| Literature Review | 609 |
| Accomplishments | 294 |
| Approach and Methodology | 724 |
| Dataset | 686 |
| Baselines | 168 |
| Results, error analysis | 595 |
| Lessons learned and conclusions | 387 |
| **Total** | **4000** |

References

[1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. [Thumbs up? Sentiment Classification using Machine Learning Techniques](https://aclanthology.org/W02-1011). In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

[2] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. [Attention-based LSTM for Aspect-level Sentiment Classification](https://aclanthology.org/D16-1058). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

[3] Jean-Benoit Delbrouck, Noé Tits, Mathilde Brousmiche, and Stéphane Dupont. 2020. [A Transformer-based joint-encoding for Emotion Recognition and Sentiment Analysis](https://aclanthology.org/2020.challengehml-1.1). In *Second Grand-Challenge and Workshop on Multimodal Language (Challenge-HML)*, pages 1–7, Seattle, USA. Association for Computational Linguistics.

[4] L. Wang and H. -D. Chiang, "Group-Based Line Switching for Enhancing Contingency-Constrained Static Voltage Stability," in IEEE Transactions on Power Systems, vol. 35, no. 2, pp. 1489-1498, March 2020, doi: 10.1109/TPWRS.2019.2944205.

[5] Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. [Early Exiting BERT for Efficient Document Ranking](https://aclanthology.org/2020.sustainlp-1.11). In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88, Online. Association for Computational Linguistics.

[6] Sanh, V., Debut, L., Chaumond, J. and Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[7] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R. and Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

[8] Sida Wang and Christopher Manning. 2012. [Baselines and Bigrams: Simple, Good Sentiment and Topic Classification](https://aclanthology.org/P12-2018). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea. Association for Computational Linguistics.

[9] Gong X, Ying W, Zhong S and Gong S (2022) Text Sentiment Analysis Based on Transformer and Augmentation. *Front. Psychol*. 13:906061. doi: 10.3389/fpsyg.2022.906061

[10] Kobab Tabinda, Sayyida, Asghar Sohail, Naz Shehneela (2021) Transformer-based deep learning models for the sentiment analysis of social media data.

[11] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. Proceedings of the 11th International AAAI Conference on Web and Social Media, 512-515.

[12] Malmasi, S., & Zampieri, M. (2018). Challenges in Discriminating Profanity from Hate Speech. Proceedings of the Second Workshop on Abusive Language Online, 14-18.