Automatic Extraction, Classification and Neutralization of Toxic Comments

Yakoob Khan, Luca C. L. Lit, Louis Murerwa, Aadil Islam
Department of Computer Science, Dartmouth College
{yakoob.khan.21, chun.lun.lit.21, louis.b.murerwa.21, aadil.islam.21}@dartmouth.edu

Abstract

In this work, we propose a three-part system that automatically extracts controversial online posts, classifies toxic comments into 6 different categories, and neutralizes the offensive language contained in such comments. Our *Extractor* module leverages the Twitter API to obtain recent posts from Twitter regarding anti-Asian rhetoric related to the Covid-19 pandemic. Our *Classifier* Module, trained using Jigsaw's Toxic Comment Classification dataset, is then used to filter posts that are likely to contain toxicity. We explore a variety of classical machine learning models and BERT to identify an accurate hate speech classifier. Finally, our *Neutralizer* module performs linguistic style transfer by removing the toxic words contained in the filtered posts while still preserving the underlying content. As each of the three modules are independent of each other, our proposed pipeline is generalizable to automatically extract data from other sources (e.g Reddit), classify and neutralize toxicity using other techniques by replacing the appropriate sub-module.

1. Introduction

The rise of social media platforms has transformed the way individuals communicate today. While offering numerous benefits, the anonymous nature of such mediums empowers bad actors to promulgate toxic comments that go against the principles of civilized discourse. Being able to automatically detect offensive posts is crucial to ensuring that social media remains healthy and inclusive for all. To assist in this effort, our work[1] proposes a three-part system that automatically extracts comments from an online platform, accurately classifies the posts to filter toxic comments, and finally neutralizes the offensive language. Given the nature of this work, we caution readers that the examples used in this work contain offensive language and reader discretion is advised.

2. Related Work

There has been extensive work in the study of hate speech detection in the literature. Numerous offensive language datasets (Wulczyn et. al., 2017) with various toxic taxonomies have been created to assist the training of robust computational models for detecting offensive language. Competitions on hate speech detection (Zampieri et. al, 2020), such as the Kaggle Toxic Comments Classification Challenge on which this work is based, have further attraction to this topic. While a lot of attention has been placed on the classification of toxic comments, neutralizing offensive language through text style transfer (Hu et. al, 2020) have been relatively under-studied.

3. Data Scraping

In this study, we made use of Twitter's Search API to capture tweets containing Anti-Asian rhetoric. To do so, we made use of the Tweepy Python package to access the API, with OAuth authentication enabled through a Twitter Developer account.

Next, to filter tweets with Anti-Asian connotations, we compiled a list of common phrases and terminologies that reflected racist undertones. It is as follows:

"China virus, kung flu, kungflu, chinx, chinese virus, Wuhan Flu, chinaman, china men, china man, chinamen, ching chong, chink, chinavirus, gook, coolie, yellowman"

Formulating a search query with the aforementioned expressions, tweets were scraped between 2021-03-09 (00:00:00), until 2021-03-16 (11:59:59). To reduce the runtime, we only extracted

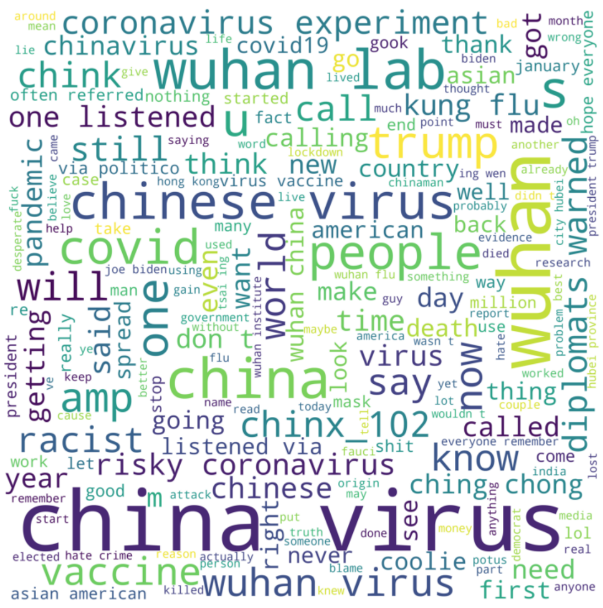
[1] All authors contributed equally to this work.

tweet attributes that were relevant to the scope of our project - for instance, the tweet text, the author, the date, etc. Retweets were also filtered out, so that only unique texts were returned. This yielded a total of 28,408 rows of data, each row corresponding to one tweet. All raw data was saved in the form of a .CSV file.

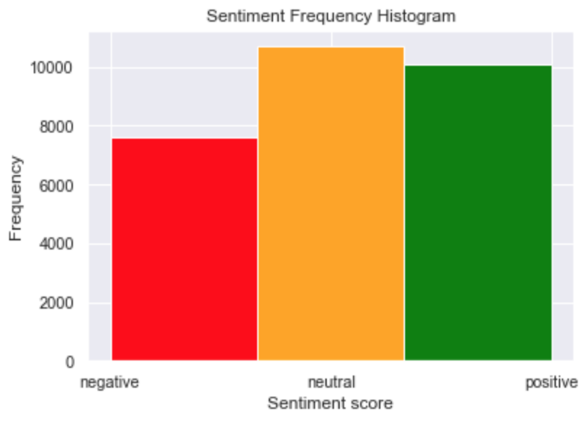
In addition, we also leveraged datasets from Kaggle's *Toxic Comment Classification Challenge* as training and testing data for our toxic comment detection models, described in sections 5 and 6. The data is in the form of comments, all of which are sourced from Wikipedia talk page comments. The classifications of toxicity are labeled by human raters detecting instances of derogatory language. In total, a total of 159,571 comments constituted the training dataset, and 153,164 comments for the testing set.

4. Exploratory Data Analysis

To learn more about our raw data, we generated a word cloud and sentiment distribution plot to illustrate high-level patterns in the tweet texts. The textual data was cleaned for stopwords and special characters.

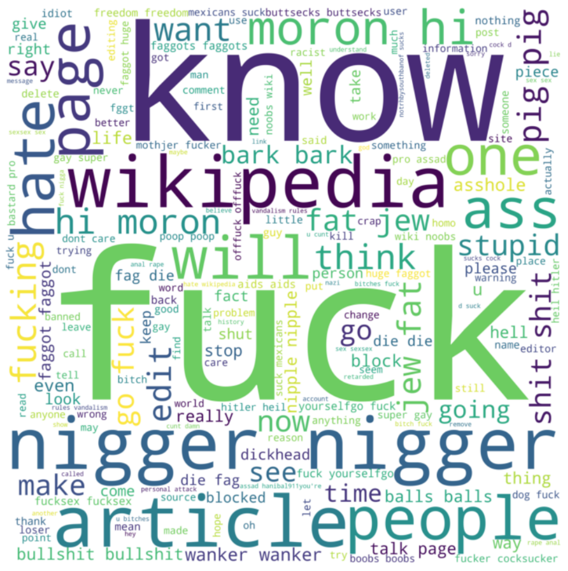


From initial observation, there is a high frequency of the usage of "china virus", "chinese virus", "wuhan lab", "coronavirus experiment", "ching chong" and "kung flu". One may recognize these as the common instances of derogatory Anti-Asian vocabulary.



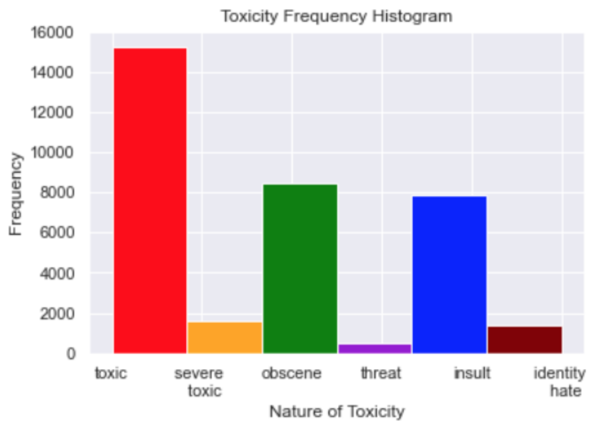
However, our sentiment distribution plots showed surprisingly lower proportions of negative sentiment tweets. The ratios of negative, neutral and positive were 26.82%, 37.65% and 35.53%, respectively. This may be attributed to the fact that recently popularized terminology including "kungflu" or "ching chong" may not yet be accounted for in the TextBlob sentiment polarity classifier.

Furthermore, we also ran a preliminary analysis on the training data from Kaggle. Using the same methodology as earlier, it yielded:



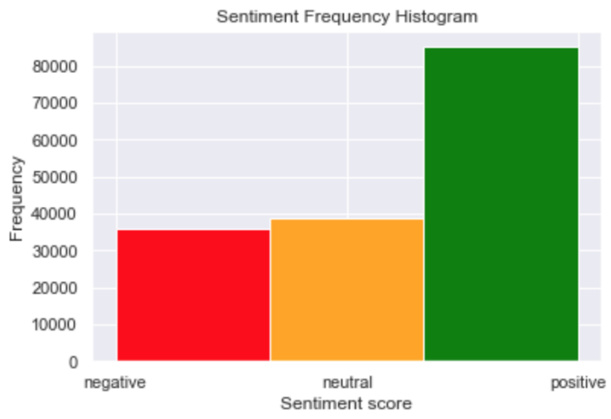
In particular, note that this word cloud was generated after a filtering step was applied to the dataset. Since neutral tweets that were not classified into any toxicity category were relatively uninformative, we decided to subset the data to only comments that were classified with at least 1 of the 6 toxic categories (toxic, severe_toxic, obscene, threat, insult, identity hate).

Furthermore, we also analyzed the frequencies of each nature of toxicity within the dataset, to detect whether there were class imbalance between the different training class categories:



It revealed there was a significantly higher proportion of training data for toxic tweets, followed by obscene, insulting, severe toxic, identity hate, and threat.

Finally, the sentiment distribution plot illustrated similar trends to our raw data, with a relatively lesser proportion of negative tweets in comparison to the neutral and positive sentiments:



5. Classical Machine Learning Models

Our first approach to classify our dataset was to employ the use of 3 classical machine learning classifiers: Naive Bayes, Logistic Regression and Random Forest. By default, these classifiers actively support classification for no more than two classes so they can only assign a single label to a piece of text. For example, a sample text like "louis went home" can be assigned one binary [0,1] label like "toxic". To achieve Multi-label classification, we employed the use of the "One vs Rest Classifier" which is a heuristic method for using binary classification algorithms on multi-label classification problems. To enhance the performance of the ML models, we employed data pre-processing techniques such as the conversion of popular short phrases like "what's" and "cuse" to "what is" and "excuse" which is their un-shortened form. We created two similar sets of training sets, one which was lowered and the other one which is unlowered and we also removed stopwords to save up valuable processing time and focus on important word keys. Lastly we employed the use of the CountVectorizer and the TF-IDF Vectorizer to fine tune our predictions.

6. Deep Learning-based Models

Alongside our aforementioned classical machine learning-based models, we also try an end-to-end, feature learning-based approach that bypasses having to tediously construct a feature set by hand.

Having seen LSTM-based approaches being used in past works to model, say, sentences for the task of complex word identification (Hartmann and dos Santos, 2018; De Hertog and Tack, 2018), we were tempted to implement such an approach. An issue with an LSTM is its ability to read tokens of an input sentence sequentially in a single direction. This inspired us to instead try a Transformer-based approach (Vaswani et al., 2017), architectures that process sentences as a whole (not word-by-word) through the use of *attention* mechanisms. Attention weights can be interpreted as learned relationships between words. BERT (Devlin et al., 2018) is one such model used for a variety of natural language understanding tasks. We harnessed HuggingFace's

Transformers library to fine-tune a pre-trained BERT base model.

Before fine-tuning our BERT model, we take our original training set and construct smaller training and development (dev) sets. We carefully create a 90/10 split using the `skmultilearn`[2] library to perform iterative stratification; this ensures each set contains a roughly fair share of samples across each class label. We then tokenize each sample using HuggingFace's BERT tokenizer; this generally takes care of pre-processing whitespace or punctuation. Next, we instantiate a BERT transformer with a sequence classification head on top (ie. a linear layer on top of the pooled output), and settle upon the AdamW algorithm as our optimizer. Our model's output layer yields predicted probabilities of each given sample being of each class toxicity label. We also use the output layer to extract attention maps across each of our model's 144 attention heads (ie. 12 heads per 12 columns) for each dev set sample. We will analyse this in Section 8.2. For training, we tried fine-tuning our BERT model for 5 epochs, finding that the model overfits beyond 1 epoch. Thus, we show the performance of our model on the test set after fine-tuning for 1 epoch.

7. Results

Model Performances on Test Set

Model	**Label: ROC-AUC**	**Mean AUC**
Naive Bayes	toxic: 0.89 severe_toxic: 0.82 obscene: 0.88 threat: 0.80 insult: 0.86 identity_hate: 0.83	0.85
Random Forest	toxic: 0.94 severe_toxic: 0.86 obscene: 0.95 threat: 0.82 insult: 0.94 identity_hate: 0.87	0.89
Logistic Regression	toxic: 0.96 severe_toxic: 0.98 obscene: 0.97 threat: 0.98 insult: 0.96 identity_hate: 0.98	0.97
BERT	toxic: 0.97 severe_toxic: 0.99 obscene: 0.98 threat: 0.99 insult: 0.98 identity_hate: 0.98	0.98

[2] `http://scikit.ml`

8. Analysis

In the following sections, we comment on the classification performance of the various models we experimented with for toxic comment detection.

8.1 Classical Machine Learning Models

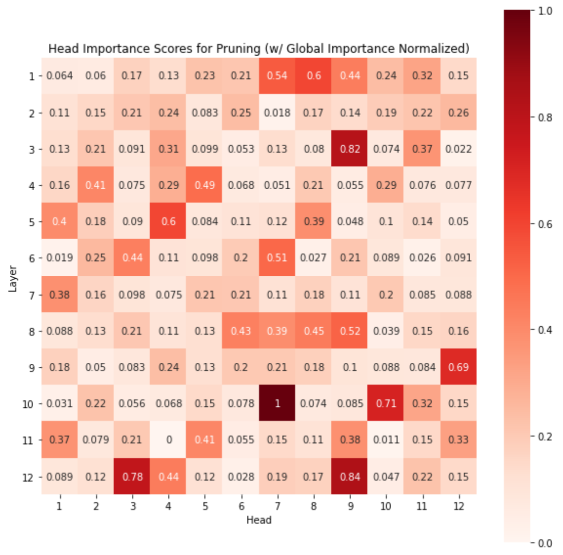
The first step toward discovering the best machine learning model for our toxic tweet dataset was to train all the three potential classic classifiers with multiple features and text pre-processing that would potentially maximize the model performances. We used a combination of 3 classifiers, 2 text pre-processing variations (lowered and unlowered) and 2 Vectorizers (CountVectorizer and TfidfVectorizer). These combinations allowed us to create a total of 12 classifier-feature-vectorizer models from which we picked our best model. The performances of our classic machine learning models varied greatly depending on the tuning mentioned above but the three best models for each classifier had an average AUC of 0.85 for Naive Bayes, 0.9 for Random Forest and 0.97 for Logistic Regression. The Logistic Regression classifier in combination with the TfidfVectorizer and the lowered text exceeded our expectations as it rivaled the performance of our best deep learning model, BERT, which had a mean AUC of 0.98.

Some of the limitations faced in using classical learning models for multi-label classification was that all of these models tend to underperform when there are multiple labels or non-linear boundaries. They take a lot of processing time and when coupled with multiple features they don't scale very well. Despite these limitations, our Logistic Regression model surpassed expectations and we were impressed by its ability to rival BERT.

8.2 BERT

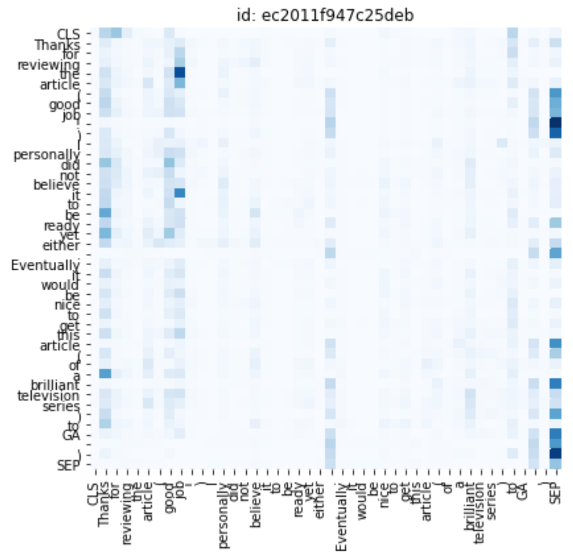
We seek to understand the potential token relationships learned by the BERT model that may be predictive of a given comment's toxicity. We study the attention maps produced for each sample in our dev set. For each dev set sample, we extract 144 attention maps, ie. one from each of BERT's attention heads. Each attention map is a 2D matrix, where an arbitrary cell (i, j) contains how much (a ratio from 0-1) the i^{th} token of a given comment *attends* to the j^{th} token of the comment (where $0 \leq i, j < n+2$). Note: each comment comprises a list of n tokens, but BERT prepends the [CLS] token to the *start* of the list for classification purposes, and [SEP] token to the end of the list as a sort of no-op, yielding a total of n+2 tokens.

To get a sense of which attention heads are most 'important' for our multi-label classification task, we apply an approach devised by Michel et al. 2019 that quantifies the importance of each attention head as the sensitivity of the model to the attention head's being masked. This requires a single backward and forward pass of the model over the dev set, and accumulation of gradients of the loss function. This yields head importance scores as shown below, revealing that the majority of attention heads can even be pruned:



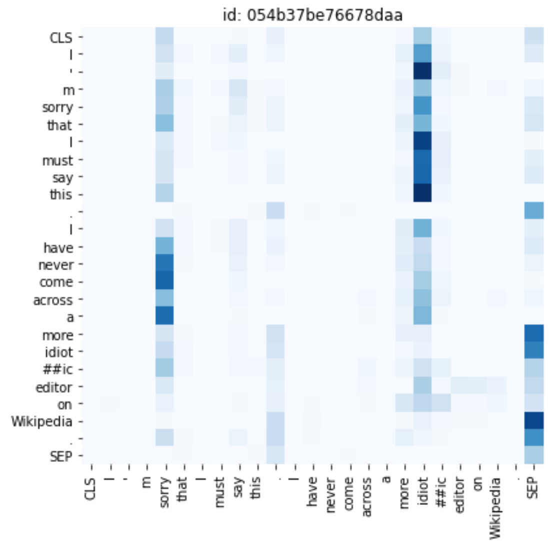
Due to time constraints, we limit our examination to just one of interesting heads: (10, 7). It appears to have the (global) maximum importance score across all attention heads. We may benefit from a closer look at the relationships learned by this particular head.

Below we show attention maps at head (10, 7) for two particular dev set examples, where it appears that attention is generally given to tokens that are most indicative of a comment's tone. This occasionally yields vertical stripe patterns over tokens that are critical to our perception of the given comment:



In the mapping shown above, notice how tokens/phrases given attention to include 'Thanks', 'good job', 'brilliant', as well as punctuation. It seems as though the positive praise given by the comment's author to the intended reader is being identified by the attention head. The apparent attention given to punctuation is harder to interpret as BERT may be more finicky with punctuation.

In contrast, the next mapping shown on the next page is for a comment exhibiting a relatively more complex tone to decipher; tokens that are heavily attended to include 'sorry', potentially capturing the author's initial apologetic tone, as well as 'idiot' (ie. the root of the word 'idiotic'), pertaining to the author's eventual harsh criticism of a supposed editor. This makes some intuitive sense, as both tokens play pivotal roles in embodying how the author's voice comes across; perhaps comments that sound more aggressive are more often toxic. We believe that such attention maps produced by the BERT model indicate the immense promise of deep learning-based approaches in conducting semantic analysis on text, even with very minimal fine-tuning.



9. Text Style Transfer

There are numerous techniques for neutralizing offensive language through text style transfer. As outlined in (Hu et. al, 2020), a possible approach is using parallel data between source and target styles to train a model to convert a sequence from one style to another. However, this approach is not scalable nor practical due to the lack of parallel data for most real-world tasks. Many creative approaches exist for dis-entangling the style and content latent representations without parallel corpus (e.g back-translation, adversarial learning, controllable generation, reinforcement learning etc.).

Based on our intuition that the offensive words significantly contribute to the toxicity of a comment, we chose to explore a rule-based solution where toxic tokens are identified and replaced with their respective antonyms. To identify the toxic tokens, we approached the task as a sequence labeling problem. We extended the work of Khan et. al 2021 and fine-tuned a BERT token classifier to identify toxic words and phrases in a given sequence. Upon identifying such tokens, we utilize WordNet to select a random antonym to replace each such offensive word in the original sentence.

In evaluating the quality of the text style transfer, a few observations were made. First, this rule-based word substitution technique showed strength in preserving the content of the original comment. This is due to keeping the majority of the original sentence structure and content intact since only toxic word substitutions were made. Word substitutions using antonyms also ensured that transfer strength from offensive to neutral style remained high. However, we learnt that WordNet antonyms dictionary look-up is very under-developed compared to its synonym's dictionary. This led to numerous toxic tokens having no corresponding antonyms. While one could use a placeholder value (e.g ***) to denote the presence of an offensive word or even remove the token entirely without a word replacement, one notes that this significantly impacts the linguistic fluency of the generated text.

Below, we highlight selected neutralized sentences from the the scraped tweets that our classification model predicted to contain at least one hate speech category:

Original Sentence	***Neutralized Sentence***
I honestly don't give a fuck.	I honestly don't give a ***.
Fuck you Wuhan. Fuck you very much.	*** *** Wuhan. *** *** very much. .
@AOC Demand reparations from China for the China virus. Cowards.	@AOC Demand reparations from China for the China virus. ***.
@Breaking911 That God damn China virus fuckin their lives up.	@Breaking911 That God *** China virus *** their lives up.

While we made these manual observations, further work could make use of quantitative metrics to evaluate the performance of the rule-base style transfer technique. Content preservation can be measured using Self-BLEU, cosine similarity etc. while transfer strength is traditionally measured by using a TextCNN to make binary classifications to predict the style class label of the generated text. Due to limited time, we could not explore this line of work and investigate an encoder-decoder neural architectural approach that is typical in modern text style transfer solutions.

10. Conclusion and Future Work

In this work, we used the *Extractor* module to scrape recent tweets related to the rise of anti-Asian sentiment related to the COVID-19 pandemic. We showed how the *Classifier* module can be trained using the Toxic Comments Classification Challenge dataset to create a robust, fine-grained toxic comments detector. Finally, we demonstrated how the *Neutralizer* module uses a simple rule-based word substitution approach that can be surprisingly effective in text style transfer to disentangle content and style. The independence of each module enables our system to be generalizable across data sources and computational models.

Nevertheless, this work has a number of limitations. The Kaggle leaderboard for the Toxic Comments Classification Challenge shows that top scoring submissions relied on hand-crafted ensembles of different machine learning models. We note that such approaches improve the Mean-AUC metric by approximately 0.5%, so our standalone BERT model performs competitively in this regard. As noted earlier, the simple rule-based word substitution approach suffers significantly with regards to linguistic fluency, so more sophisticated encoder-decoder neural architectures could be explored alongside quantitative metrics for comparison purposes. We hope this paper inspires more creative approaches to address these limitations in future avenues of work so that discourse on social media platforms remains healthy and inclusive for all.

Acknowledgements

We thank Professor Soroush Vosoughi and the Teaching Assistant team for the wonderful course on Data Mining and Knowledge Discovery (Winter 2021).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Nathan Hartmann and Leandro Borges dos Santos. 2018. NILC at CWI 2018: Exploring feature engineering and feature learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340, New Orleans, Louisiana. Association for Computational Linguistics.

Dirk De Hertog and Anaïs Tack. 2018. Deep learning architecture for complex word identification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 328–334, New Orleans, Louisiana. Association for Computational Linguistics.

Zhiqiang Hu, Roy Ka-Wei Lee, and Charu C. Aggarwal. 2020. Text Style Transfer: A Review and Experimental Evaluation. https://arxiv.org/pdf/2010.12742.pdf

Yakoob Khan, Weicheng Ma, and Soroush Vosoughi. 2021. Lone Pine at SemEval-2021: Fine-Grained Detection of Hate Speech Using BERToxic.

Paul Michel and Omer Levy and Graham Neubig, . "Are Sixteen Heads Really Better than One?". CoRR abs/1905.10650. (2019).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In Proceedings of the 26th International Conference on World Wide Web, WWW ’17, page 1391–1399, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Cagri Coltekin. 2020. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pages 1425– 1447, Barcelona (online). International Committee for Computational Linguistics.