A Stacking-based Efficient Method for Toxic Language Detection on Live Streaming Chat

Yuto Oikawa[1,2] **Yuki Nakayama**[2] **Koji Murakami**[2]
[1]Graduate School of Engineering, Kitami Institute of Technology
[2]Rakuten Institute of Technology, Rakuten Group Inc.

Abstract

In a live streaming chat on a video streaming service, it is crucial to filter out toxic comments with online processing to prevent users from reading comments in real-time. However, recent toxic language detection methods rely on deep learning methods, which can not be scalable considering inference speed. Also, these methods do not consider constraints of computational resources expected depending on a deployed system (e.g., no GPU resource). This paper presents an efficient method for toxic language detection that is aware of real-world scenarios. Our proposed architecture is based on partial stacking that feeds initial results with low confidence to meta-classifier. Experimental results show that our method achieves a much faster inference speed than BERT-based models with comparable performance.

1 Introduction

With the rapid growth of online social platforms, posting text comments to a content have become a familiar part of people's lives for growing human connection and advertising purposes. However, these comments can include toxic language, which can be harmful or offensive to others. Toxic comments lead to damages of the user experience, human well-being, and even product promotion. Particularly, toxic language is a very common problem in live feeds on video streaming services (e.g., YouTube) (Liebeskind et al., 2021). Toxic comments can appear more frequently in live broadcasting, since users tend to impulsively post comments in real-time with less introspection during live streaming (Gao et al., 2020). It is not possible to manually rule out toxic comments from a large number of comments continuously posted across multiple live feeds. We aim for an automatic detection system to address toxic comments, such as Figure 1.

To capture powerful latent features for detecting toxic language, recent existing methods rely on

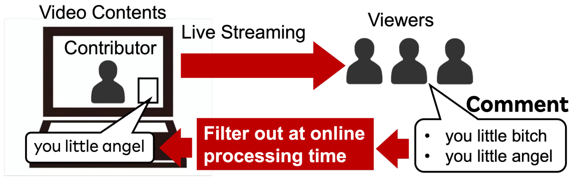


Figure 1: Diagram of Toxic Comment Detection System

deep learning techniques such as BERT. Although the techniques have performed well on the task, the following issues can be raised when assuming application to comments on live streaming.

1. Inference Speed: During live streaming, many posted comments can stream to all viewers in real-time. To promptly prevent both viewers and video contributors from reading toxic comments, online processing should be crucial. However, deep learning-based methods might not be scalable for online processing due to slow inference speed.

2. Computation Resource: A computational resource-friendly method is essential from an industry point of view. Even if the inference speed is satisfied with one or more GPUs at the PoC phase, there should be computational resource constraints, depending on the requirements of a deployed system when assuming real-world operations, such as API on a mobile application.

To make up for the above two issues, we propose an efficient method for toxic comment detection on live streaming chat. Our target includes offensive, insulting, and obscene expressions, similar to the expressions used by Leite et al. (2020). We handle Japanese comments since our method is meant to be deployed on a Japanese live streaming service. Our proposed architecture is a stacking-based two-layer classification model in which detection results with lower confidence scores in fastText classification are re-classified by LightGBM with five features. Thus, our method does not require a GPU environment. That enables a developer to facilitate deploying a system with low computation resources and enough reproducibility. Experimental results show that our method achieves up to 17 times faster inference speed than several BERT-based methods with a comparable F1 score under an online processing setting.

2 Related Work

Many methods for toxic comment detection have been proposed, mainly for Twitter (Leite et al., 2020; Fehn Unsvåg and Gambäck, 2018; An et al., 2021; ElSherief et al., 2021; Founta et al., 2019) and comments to online news articles (Jigsaw, 2019; Baldini et al., 2022). Traditional methods use lexical patterns based on offensive words (Hosseini et al., 2017; Gröndahl et al., 2018) and opinion words (Pouran Ben Veyseh et al., 2022). Since malicious viewers create various toxic words in diverse writing styles, this approach is not robust to variants of words in the lexicon. In addition, it is costly to regularly update the lexicon to keep a deployed toxic detection system accurate (Nejadgholi et al., 2022).

In recent years, many methods utilize deep learning-based methods such as BERT and LSTM, using sentiment information (Brassard-Gourdeau and Khoury, 2019; Zhou et al., 2021; Cao et al., 2020; Pouran Ben Veyseh et al., 2022), topic contents (Almerekhi et al., 2020; Bose et al., 2021), and context information such as text replies (Dahiya et al., 2021; Bhat et al., 2021) and attention-based context vectors (Chakrabarty et al., 2019). Baldini et al. (2022) explored how BERT-based models affect the relationship between performance and fairness for toxic comment detection. Here, fairness means equalized performance across various sensitive groups such as religion and race. However, none of the studies in this section explore performance that consider inference speed and computational resources for toxic comment detection for real-world applications.

Comment characteristics in video live streaming chats are fundamentally different from Twitter posts and news articles. There is no information on replies (i.e., parent-child relationship) in live streaming chats. Moreover, the comments are often short, which lacks context and topic information. According to Yousukkee and Wisitpongphan (2021), 63% of messages in live streaming chats on YouTube contained fewer words than the average word count of 8 with standard deviations of 7.

In video live streaming chat on Twitch[1], Gao et al. (2020) applied a fine-tuned RoBERTa model to toxic comment detection. We show the efficiency of our method by making a comparison with various BERT-based models.

Edge computing can be an applicable solution to address latency and scalability challenges for NLP services with deep learning (Chen and Ran, 2019; Han et al., 2020). There is a wide range of deployed candidates for edge computing architectures and deep learning models. The deployment should be carefully considered to accomplish system requirements. Thus, we leave the applicability of deep learning with edge-computing in our task for future research.

3 Data Collection

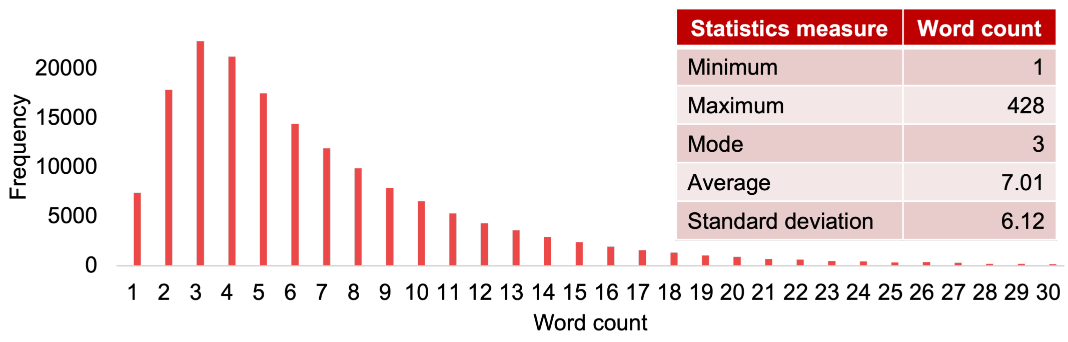
We create annotated data for toxic comment detection on the video live streaming domain. We use "NicoNico-Doga comment data" (DWANGO Co., 2021-12-22) provided by the National Institute of Informatics[2]. NicoNico-Doga is one of the largest-scale Japanese video streaming services. In the whole dataset, we used the file lists from 0000.zip to 0005.zip, and labeled them as toxic/non-toxic comments via human annotation. In total, 168,071 comments were annotated, which comprise 21,156 toxic comments and 146,915 non-toxic comments. We randomly divided the annotated dataset into a training set, development set, and test set at a ratio of 80%, 10%, and 10%, respectively. The statistics are shown in Table 1. To evaluate inter-annotator agreement, additional two Japanese annotators independently identified a toxic or non-toxic label to 2,122 comments from scratch. The inter-annotator agreement was $\kappa = 0.77$, which indicated substantial agreement. Figure 2 shows the distribution of comments divided by word count in our dataset. The distribution of our dataset is similar to one reported in Yousukkee and Wisitpongphan (2021) on YouTube live stream. In our dataset, 67% of comments contained fewer words than the average word count of 7 with standard deviations of 6.

4 Proposed method

4.1 Overview

Our task is to classify posted comments as toxic or non-toxic. As mentioned in §1, we assumed that

[1]https://www.twitch.tv
[2]https://www.nii.ac.jp/dsc/idr/nico/nicocomm-apply.html



Statistics measure	Word count
Minimum	1
Maximum	428
Mode	3
Average	7.01
Standard deviation	6.12

Figure 2: Number of comments divided by word count in our dataset

	Toxic	Non-toxic	Total
Train	16,925	117,532	134,457
Dev	2,116	14,692	16,808
Test	2,115	14,691	16,806
Total	21,156	146,915	168,071

Table 1: Statistics of dataset

online processing should be crucial for comments posted continuously in live streaming, and computational resources are constrained depending on the specification of the deployed system. Thus, we design a model architecture under the following two limitations for a deployed system.

- Avoid using a GPU resource
- Avoid using a deep learning model

Figure 3 shows the architecture of our proposed method, which comprises two layers. We first use fastText classification model (Joulin et al., 2017) as described in §4.2 in order to prioritize fast inference speed. Our preliminary experiments with our dataset showed that the lower the prediction probability for the fastText classification, the lower the F1 score. Thus, there is much room for improvement in results of lower prediction probability as described in the triangle area in Figure 4. The results motivate us to utilize another classification model for the case where the fastText model has less confidence.

After getting classification results by the fastText, we use the stacking technique (Džeroski and Ženko, 2004). This technique is a simple but effective way that predictions of different classifiers are fed to a meta-level classifier to generate final results. For efficiency, the second classifier is applied only to the first results with lower confidence. We

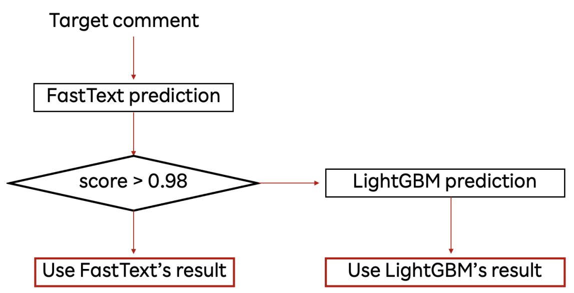


Figure 3: Model Architecture

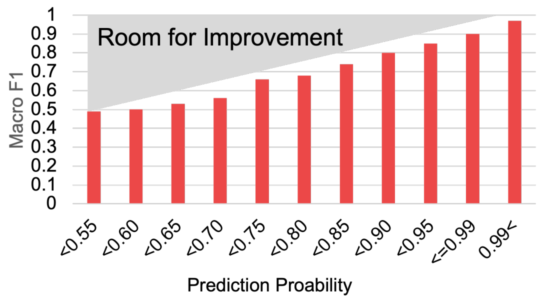


Figure 4: Macro F1-score by prediction probability for fastText classification model

will describe the meta classifier and used features in §4.3.

4.2 First layer: fastText classification model

The model is a simple neural network with only one layer. The bag-of-words representation is first fed into a lookup layer to obtain a word representation for every word. The model takes an average of word representations into a text representation, which is in turn fed to a linear classifier. We use the softmax function to compute the probability distribution over the two classes. We calculate confidence score of classification result by Equation (1)

$$\mathrm{Conf}(c) = \max\{\mathrm{P}(y = 1|c), \mathrm{P}(y = 0|c)\} \quad (1)$$

Input comments are fed to the meta-level classifier if the confidence score is equal to or less than the threshold θ. Otherwise, we use fastText prediction as output.

4.3 Second layer: Meta-level Classification

As meta classifier, we select a LightGBM (Ke et al., 2017), gradient boosting decision tree-based model. The advantage of using LightGBM is its efficiency and interpretability. Error analysis is indispensable to keep a deployed system performance accurate for future data. LightGBM enables us to facilitate detecting a cause of an error through tracing decision flow. For the model training, we propose five features from three perspectives.

Two Lexicon features: #Black words and #Gray words Unlike existing studies, we make a lexical feature considering certainty for toxic words. The more a comment includes toxic words, the more the text is likely to be toxic. However, whether or not a word is associated with being toxic depends on context. For example, the Japanese word "くそ (sh*t)" has two word meanings, which are "very" and a literal toxic word. If a comment uses the word with the former meaning, a system can incorrectly identify the comment as a toxic comment. To alleviate this problem, toxic words are divided into two categories in terms of certainty, called "black words" and "gray words" in this paper. Black words are words considered toxic regardless of the context. Gray words are words that are not considered toxic based on the context. Based on those ideas, we use the number of black words and the number of gray words in a comment as feature values. We manually created 1,338 black words and 1,614 gray words.

fastText Prediction We use the prediction label as a feature value. Our preliminary experiments show that fastText classification model tends to return a low confidence score when ground-truth is a "toxic" label, as illustrated in Figure 5. To utilize this empirical finding, we also use the confidence score $\in (0.5,1]$ computed in §4.2.

SVM Prediction As a third perspective, we use prediction results of Support Vector Machine (SVM) trained with TF-IDF weighting scheme. For each word w in a comment c, TF-IDF is calculated by the Equation (2).

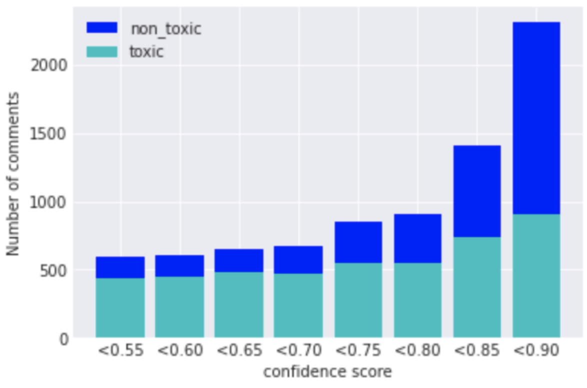


Figure 5: Ratio of toxic comments by confidence score

$$\text{TF-IDF}(w,c) = \frac{f_{w,c}}{\sum_{w' \in c} f_{w',c}} \cdot \log \frac{N}{\text{df}(w)+1} \tag{2}$$

where $f_{w,c}$ denotes frequency of w in c. $\text{df}(w)$ denotes the number of documents in which w appears. N is the total number of comments.

5 Experiments

5.1 Settings

We did experiments with the annotated dataset in §3 to show the effectiveness and the efficiency of our method. Model trainings were conducted on Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with a single processor. RAM size is 26GB. Cache memory consists of 32KB for L1d, 32KB for L1i, 1024KB for L2, and 28,160KB for L3. The model implementation is as follows:

fastText classification To get the word representations, we create a pre-trained model using the fastText module (Bojanowski et al., 2017) with all the comments in §3. For pre-processing, this data is tokenized by the Japanese morphological analyzer MeCab (Kudo, 2006). Additionally, a word was lemmatized and half-width characters were converted to full-width. Hyperparameters for the pre-training are as follows: The number of dimensions for word representation is 300. We used skip-gram to train word representation. The threshold θ in §4.2 was determined with development set ($\theta = 0.98$).

SVM We calibrated the prediction results using a calibrated ClassifierCV[3] provided by scikit-learn

[3]https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html

to remove the effect of bias in the unbalanced data.

LightGBM This model was optimized with optuna (Akiba et al., 2019).

As baseline models, we used each component of our method and existing pre-trained BERT-based models. We used the BERT-base[4] and BERT-large[5] models pre-trained from Japanese Wikipedia. Additionally, it is essential to compare with light BERT models for a fair evaluation of inference speed. Over the last couple of years, variants of BERT have been proposed to make the model size light and inference speed efficient. Specifically, we tried Distil-BERT[6] (Sanh et al., 2019), ALBERT[7] (Lan et al., 2019), and Poor-Man's BERT (Sajjad et al., 2020). Poor-Man's BERT is the method that removes some layers from an original BERT. Their experimental results showed that dropping the top layers works consistently well across different tasks when dropping 4 and 6 layers. Following those findings, we removed the top 4 layers in the BERT-base that we used.

For fine-tuning and testing for BERT-based methods, we used a single 32GB NVIDIA-V100 GPU. The batch size for fine-tuning was 16. We set batch size for inference to 1, since we assumed online processing at the inference phase. Another possible way of improving inference speed would be adjusting the maximum length of an input sequence. We explored the relationship between performance and inference speed on variation of the length (4, 8, 16, 32, and 64). We evaluated the average inference time over 10 trials.

5.2 Results

Table 2 shows classification results of our method obtained with the threshold optimized in terms of Macro F1 score. Our method achieved 0.942 in terms of average of Macro F1 across the two classes. Table 3 shows results and the effectiveness of each component. Looking at Table 3, one can see that our stacking-based method outperformed fastText single classification model by 3.3 points and the remaining ones as well. We observed that 24% of test samples proceeded to the meta-level classifier. In Table 3, our ablation study showed that our five features contributed to enhancing F1 score, especially black words and fastText prediction. Thus, we believe that each of our proposed features was independently effective for toxic language detection tasks, and that the improvement was even greater when used together.

	Precision	**Recall**	**Macro F1**
Non toxic	0.981	0.991	0.986
Toxic	0.931	0.867	0.898
Macro Avg.	0.956	0.929	0.942

Table 2: Classification performance

Method	**Macro F1**
fastText classification	0.919
Black words	0.905
Gray words	0.860
SVM	0.905
Our method	**0.942**
w/o SVM prediction	0.940
w/o graywords	0.938
w/o fastText probability	0.937
w/o blackwords	0.933
w/o fastText prediction	0.932

Table 3: Performance Comparison and Ablation Study

Table 4 shows the comparison between our method and BERT-based methods on various configurations. The average inference time for our method was 22.9 seconds for 16,807 test samples, with a standard deviation of 1.18. In the Table, we put QPS score (i.e., Throughput), the number of comments which can be processed per a second. Looking at the table, our method achieved much faster inference speed (734QPS$\pm$41) than any other BERT configurations. We found that the differences in inference speed between our method and BERT models were statistically significant at the 1% level, irrespective of the configurations by the two-tailed paired t-test for statistical testing.

BERT-large and ALBERT yielded slightly better performance than our method when the maximum sequence length was 64 (F1 = 0.948) and 32 (F1 = 0.943), respectively. However, these models sacrificed inference speed. For instance, the inference speed (47QPS$\pm$6) of BERT-large is 1.7 times slower than BERT-base (82QPS$\pm$4) and 15.6 times much slower than our method. On the other hand, if attaching great importance to inference speed,

[4]https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking

[5]https://huggingface.co/cl-tohoku/bert-large-japanese

[6]https://huggingface.co/bandainamco-mirai/distilbert-base-japanese

[7]https://huggingface.co/ALINEAR/albert-japanese-v2

	F1	QPS	max_len = 4		max_len = 8		max_len = 16		max_len = 32		max_len = 64	
			F1	QPS	F1	QPS	F1	QPS	F1	QPS	F1	QPS
BERT-base			0.775	89±3	0.882	86±6	0.931	84±5	0.936	84±3	0.937	82±4
BERT-large			0.783	50±2	0.891	48±2	0.939	48±2	0.943	47±2	0.948	47±2
Distil-BERT			0.773	145±6	0.878	144±4	0.923	134±9	0.924	134±9	0.928	131±7
ALBERT			0.699	81±5	0.866	84±4	0.928	79±4	0.943	78±3	0.938	79±2
Poor Man's BERT			0.774	114±7	0.882	115±9	0.930	112±4	0.936	112±6	0.935	109±6
Our method	0.942	734±41										

Table 4: Relationship between Macro-F1 and inference speed on various configurations

Distil-BERT achieved the highest inference speed (145QPS±6) of all the BERT-based settings when the maximum sequence length was 4, but F1-score dramatically went down to 0.773. It seems that Poor-Man's BERT and Distil-BERT had harmonized results of BERT settings when the maximum length was 32 or 64. However, all of the values did not reach those for our method. Thus, we believe that our method is intended for real-world deployment in terms of low-cost computational resources with the comparable F1 score.

5.3 Error Analysis

To understand the difference between our method and the BERT-large, we analyzed error cases where the BERT-large made correct predictions whereas our method failed. We describe a cause of an error with the example comments in Table 5. In the Table, we input English translation from the original Japanese in parenthesis. On ethical grounds, some parts were replaced with "*".

The total number of our target errors was 195, divided into 61 false positives and 134 false negatives. In the false positives, 39 cases (64%) contained our gray word. Thus, we suspected that the gray word affects classification results for some patterns. We observed two cases in which the gray word can be noise.

recognized sub-word In the comment (a) in Table 5, the word "きめつ" (kimestu) is not a toxic word but just a title of Japanese animation. Our method mistakenly identified the sub-word "きめ (gross)" in the word as a gray word.

recognized without Word Sense Disambiguation In the comment (b), the word "はげ" is often used as the toxic word "bald". However, this word is sometimes also used as abbreviation of "はげしく (strongly)". In this context, the word had the latter meaning and thus should not be identified as a gray word.

In the false negatives, 96 errors (72%) were classified as a non-toxic comment for both SVM and fastText classification models despite the presence of gray words. Of the 96 errors, we identified that 39 cases (41%) would be due to either of the two causes.

Coarse-grained tokenization When the same word is written consecutively, such as the comment (c), the MeCab tokenizer did not split that phrase into a finer-grained word unit. We consider that our method could not capture a feature of being toxic due to a coarse-grained token with useless TF-IDF value and word representation.

lol (laugh out loud) slang expression SVM was trained so that the word "w (lol)" can contribute to being non-toxic, rather than toxic, since the word also appears in a non-toxic context. As a result, even though a gray word is included in a comment, the comment was not identified as a toxic comment if the expression is used many times in the comment (d).

Input	Gold	Ours
(a) だから、きめついらんよ (I told we don't need kimetsu)	N	T
(b) はげど！ (Strongly agreed!)	N	T
(c) エロエロエロエロ (EroticEroticErotic)	T	N
(d) タグまじかw*ねｗｗ復帰スンナｗ (tag is serious?lol fu**k off and die lol lol don't come back lol)	T	N

Table 5: Error cases (T: Toxic, N: Non-toxic)

6 Conclusion

Although many methods have been proposed to detect toxic comments on online social platforms, these methods have paid no attention to inference speed and constraints of computational resources for real-world applications. We presented a fast and computational resource-friendly method. Our method does not require GPU resources, which faclitate being adjustable with a requirement of a deployed system. We proposed a two-layer classification model that efficiently utilizes stacking techniques with five features. Experimental results showed that all of the proposed features were effective independently. Under the online processing setting, our method achieved a much faster inference speed than fine-tuned BERT-based methods, with the comparable F1 score. Our method is going to be deployed on our service soon. We leave the issues raised in error analysis for future research.

Acknowledgements

In this paper, we used "Nicovideo Comment etc. data" provided by DWANGO Co., Ltd. via IDR Dataset Service of National Institute of Informatics.

Ethical consideration

This study involves issues related to freedom of expression. Detecting and hiding inappropriate comments is an act that is closely associated with censorship. There is still room for detailed discussion on the extent to which it should be considered toxic.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework.

Hind Almerekhi, Supervised by Bernard J Jansen, and co-supervised by Haewoon Kwak. 2020. Investigating toxicity across multiple reddit communities, users, and moderators. In *Companion proceedings of the web conference 2020*, pages 294–298.

Jisun An, Haewoon Kwak, Claire Seungeun Lee, Bogang Jun, and Yong-Yeol Ahn. 2021. Predicting anti-Asian hateful users on Twitter during COVID-19. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4655–4666, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ioana Baldini, Dennis Wei, Karthikeyan Natesan Ramamurthy, Moninder Singh, and Mikhail Yurochkin. 2022. Your fairness may vary: Pretrained language model fairness in toxic text classification. In *Findings of the Association for Computational Linguistics*, pages 2245–2262, Dublin, Ireland. Association for Computational Linguistics.

Meghana Moorthy Bhat, Saghar Hosseini, Ahmed Hassan Awadallah, Paul Bennett, and Weisheng Li. 2021. Say 'YES' to positivity: Detecting toxic language in workplace communications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2017–2029, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5.

Tulika Bose, Irina Illina, and Dominique Fohr. 2021. Generalisability of topic models in cross-corpora abusive language detection. In *Proceedings of the Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 51–56, Online. Association for Computational Linguistics.

Eloi Brassard-Gourdeau and Richard Khoury. 2019. Subversive toxicity detection using sentiment information. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 1–10.

Rui Cao, Roy Ka-Wei Lee, and Tuan-Anh Hoang. 2020. Deephate: Hate speech detection via multi-faceted text representations. In *12th ACM conference on web science*, pages 11–20.

Tuhin Chakrabarty, Kilol Gupta, and Smaranda Muresan. 2019. Pay "attention" to your context when classifying abusive language. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 70–79, Florence, Italy. Association for Computational Linguistics.

Jiasi Chen and Xukan Ran. 2019. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674.

Snehil Dahiya, Shalini Sharma, Dhruv Sahnan, Vasu Goel, Emilie Chouzenoux, Víctor Elvira, Angshul Majumdar, Anil Bandhakavi, and Tanmoy Chakraborty. 2021. Would your tweet invoke hate on the fly? forecasting hate intensity of reply threads on twitter. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2732–2742.

Ltd. DWANGO Co. 2021-12-22. Nicovideo comment etc. data. informatics research data repository, national institute of informatics (dataset).

Saso Džeroski and Bernard Ženko. 2004. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. 2021. Latent hatred: A benchmark for understanding implicit hate speech. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 345–363, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The effects of user features on Twitter hate speech detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 75–85, Brussels, Belgium. Association for Computational Linguistics.

Antigoni Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2019. A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM conference on web science*, pages 105–114.

Zhiwei Gao, Shuntaro Yada, Shoko Wakamiya, and Eiji Aramaki. 2020. Offensive language detection on video live streaming chat. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1936–1940, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All you need is "love": Evading hate-speech detection. *CoRR*, abs/1808.09115.

Yiwen Han, Xiaofei Wang, Victor C. M. Leung, Dusit Tao Niyato, Xueqiang Yan, and Xu Chen. 2020. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22:869–904.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Kaggle Jigsaw. 2019. Jigsaw unintended bias in toxicity classification. [Online:Acessed: 2022-07-18].

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. jp*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

João Augusto Leite, Diego Silva, Kalina Bontcheva, and Carolina Scarton. 2020. Toxic language detection in social media for Brazilian Portuguese: New dataset and multilingual analysis. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 914–924, Suzhou, China. Association for Computational Linguistics.

Chaya Liebeskind, Shmuel Liebeskind, and Shoam Yechezkely. 2021. An analysis of interaction and engagement in youtube live streaming chat. In *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, pages 272–279.

Isar Nejadgholi, Kathleen Fraser, and Svetlana Kiritchenko. 2022. Improving generalizability in implicitly abusive language detection with concept activation vectors. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5517–5529, Dublin, Ireland. Association for Computational Linguistics.

Amir Pouran Ben Veyseh, Ning Xu, Quan Tran, Varun Manjunatha, Franck Dernoncourt, and Thien Nguyen. 2022. Transfer learning and prediction consistency for detecting offensive spans of text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1630–1637, Dublin, Ireland. Association for Computational Linguistics.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. On the effect of dropping layers of pre-trained transformer models.

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

Sawita Yousukkee and Nawaporn Wisitpongphan. 2021. Analysis of spammers' behavior on a live streaming chat. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 10:139–150.

Xianbing Zhou, Yang Yong, Xiaochao Fan, Ge Ren, Yunfeng Song, Yufeng Diao, Liang Yang, and Hongfei Lin. 2021. Hate speech detection based on sentiment knowledge sharing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7158–7166, Online. Association for Computational Linguistics.