Offensive Language Detection in Social Media Using Transformers and Importance of Pre-training

Beyzanur Saraclar*, Birol Kuyumcu, Selman Delil, and Cuneyt Aksakalli

Abstract—**Being exposed to offensive language on social media platforms is relatively higher because of anonymity and distant self-expression compared to real communication. Billions of contents are shared daily on these platforms, making it impossible to detect offensive posts with manual editorial processes. This situation arises the need for automatic detection of offensive language in social media posts to provide users' online safety. In this paper, we applied different Machine Learning (ML) models on over manually annotated 36,000 Turkish tweets to detect the use of offensive language messages automatically. According to the results, the most successful model for predicting offensive language is pre-trained transformer-based ELECTRA model with 0.8216 F-1 score. We also obtained the highest F-1 score with 0.8342 in this dataset up to now by combining transformer-based ELECTRA and BERT models in an ensemble model.**

Index Terms—**NLP, deep Learning, transformers, offensive language detection**

I. Introduction

With the development of new communication technologies, the use of social media has become an important part of daily life. Social media platforms provide users the opportunity to reach billions of people anonymously without any time and place restrictions [1]. While it provides quick and easy access to information, this situation has played an important role in bringing the excessive and uncontrolled use of social media. Also, the possibility of hiding the real identity in the digital environment causes the disappearance of adhering to rules of courtesy and respect in face-to-face communication. In addition, there is no real-time inspection mechanism in these platforms that is able to check the appropriateness of the used language. Currently, the only way to check whether the post has offensive language is based on feedback from the users after the post is shared. However, this method doesn't prevent users from being exposed to offensive language which negatively affects the mental and psychological health of social media users from child users to adults [2]. To tackle the mentioned issues, building an automatic system that will quickly inspect and remove such contents in social media posts before that reaches the users becomes an urgent task.

Manuscript received on August 10, 2021; revised March 22, 2022; accepted April 4, 2023.

Beyzanur Saraclar, Birol Kuyumcu, Selman Delil and Cuneyt Aksakalli are with the Sefamerve R&D Center Istanbul, Turkey.

*Correspondence: beyzanursaraclar@marun.edu.tr(B.S.)

We considered this issue as a text classification problem to determine whether a social media post has an offensive language. Text classification is a process of assigning a text to one or more predefined categories. It has a broad application such as sentiment analysis, spam detection, intent detection and topic labeling. As a subfield of natural language processing discipline, text classification has been studied in different application areas. Ketmaneechairat and Maliyaem [3] retrieved Twitter and Instagram posts related to natural disasters topics to obtain name entities from unstructured messages using different Machine Learning models. The authors compared the performances of Conditional Random Fields (CRF) and Long Short Term Memory (LSTM) and stated that CRF with optimization obtained best performance. Guo [4] analyzed the sentiment of the student from social media posts employing Text-Processing API software tools for sentiment analysis. In this study, Twitter data was used to detect the use of offensive language. By processing the data with different techniques, useful information can be extracted depending on research purpose.

Recently, deep learning methods have obtained successful results in text classification tasks. In traditional machine learning methods, it is necessary to apply feature engineering processes such as creating and selecting features from the raw text data. However, in deep learning, the model can extract features automatically during the training period. The development of the word embedding techniques is one of the main contributions in this area. With these techniques, the performance of classification tasks is improved by considering the semantic proximity of words without the need for pre-processing. In recent years, the use of pre-trained transformers-based models in different NLP applications has outperformed other methods. These models have been successful in many languages since they consider the context of the word in corresponding text.

Previous studies have been conducted in different languages on detecting hateful speech and cyberbullying which are the sub-branches of offensive language [5]. Also, there has been a deep interest in shared tasks such as [6, 7] on this topic. In recent years, the number of studies on the Turkish text on these subjects has been increasing. Özel *et al.* [8] used the Turkish messages they retrieve from Instagram and Twitter as a dataset in their study to detect cyberbullying. Bag of words method was applied to generate vectors for each tweet. Various Machine Learning (ML) methods (C4.5, NB, SVM, and KNN) were used in this study. According to the results of the research, Naïve Bayes (NB) is the most successful model with a 79% accuracy rate in terms of F-measure. Also, including both words and emoticons in text messages made improvement for the performance of cyberbully detection. Bozyigit *et al.* [9] created the Turkish Cyberbullying dataset containing 3000 tweets. The authors stated that Supporting Vector Machines (SVM) with this dataset obtained better results than other machine learning models. They also achieved top success with an F1 score of 91% using different Neural Networks (NN) on the same dataset [10]. Before performing the models, information gain feature ranking was applied to decrease the feature space dimension. Ön and Yeniterzi [11] achieved 93.2% F1 score which is the highest success on Turkish Cyberbullying dataset by using Convolutional Neural Networks (CNN). To vectorize the tweets, two different word embedding methods were experimented. The authors applied Word2Vec, FastText and randomly initialized word vectors. Bozyiğit *et al.* [12] suggested that in addition to textual features, including social media features such as number of times a tweet was shared by other users (retweets), number of users who liked a tweet (favorites), increase prediction success of machine learning models. The authors prepared a dataset consisting of 5000 labeled contents with many social media features. Various ML models were employed (SVM, LR, KNN, NBM, AdaBoost, RF) to detect cyberbullying. AdaBoost outperformed other algorithms with 89% accuracy rate in terms of F-measure. Çöltekin [5] created a dataset consisting of over 36,000 Turkish tweets named offensive-turkish dataset by retrieving the posts from Twitter. The author applied Support Vector Machine (SVM) model on this dataset using a bag of words as a feature and the model reached an F1 score of 77.3%.

In previous studies, small-scale datasets have been used for offensive language detection. However, to train deep neural networks for text classification field, large-scale labeled dataset is required. Since these networks have a huge number of parameters and training on small datasets will result in overfitting. In cases where the data set is not big enough to train the network, it is efficient to utilize transfer learning technique. With this technique, pre-trained with a huge dataset model is used by fine-tuning it with a relatively smaller dataset. At this point, pre-trained Transformers based models provide powerful language representation for the NLP tasks.

In this study we applied recently developed Transformers technologies to the Turkish language in this field. We also conducted experiments involving traditional machine learning models and compared their performances according to the macro F-1 score. For this purpose, Logistic Regression, Deep Neural Networks (DNN), Gated Recurrent Unit (GRU), FastText, Bidirectional Encoder Representation from Transformers (BERT) and Pre-training Text Encoders as Discriminators Rather Than Generators (ELECTRA) models were applied on a dataset consisting of over 36,000 Turkish tweets prepared by Çöltekin [5]. This dataset also was used in Semeval-2020 task 12 [7] and we achieved F-1 score above the benchmark results in this task.

The rest of this paper is organized as follows. Section II presents the used methods in detail to detect offensive language. Section III gives information about the experiment stages. Section IV shows the experimental results of the methods. In Section V, we conclude the study.

II. Methods

A. Logistic Regression

We used Logistic Regression (LR) as a baseline model. LR is based on the Sigmoid function whose output takes a value between 0 and 1 that can be interpreted as a probability. Each input value is multiplied by its weights or coefficient values to get the predicted value. The model learns the weight values with the loss function from the training data. In this way, it can generate correct results by assigning the most optimum weight values to the features/inputs over the training period.

B. Deep Neural Network

Deep neural network (DNN) has a layered architecture consisting of neurons in which each neuron is interconnected. The network has a feed-forward mechanism in which the neurons transmit a signal to other neurons according to the input received. Error is obtained by comparing the predicted output value with the correct value. Depending on the error, the weights of each neuron are updated with back-propagation. In this study, a 3-layer structured network consisting of embedding, global average pooling, and classifier layers was used.

C. Gated Recurrent Unit

In a feed-forward network, there is no concept of time or order, the only input it is interested in is the current instance at the time. In recurrent neural networks (RNN), the output is not only based on the current input but also depends on previous hidden states. In addition to the input at time t in the RNN, the hidden layer results from the time t-1 are the input of the hidden layer at time t. The decision made for the input at time t-1 also affects the decision to be made at time t. In other words, inputs produce output by combining current and previous information to preserve the sequential information in these networks.

Gated Recurrent Unit (GRU) [13] is a variant of RNNs with gate vectors (update gate, reset gate) that decide how much of the past information in a neural network should be transferred to the output. By training these vectors, the model can keep long-term information without forgetting or removing information that is not related to the content. The reset gate determines relevant past information to store while the update gate decides how much of the past information will be retained.

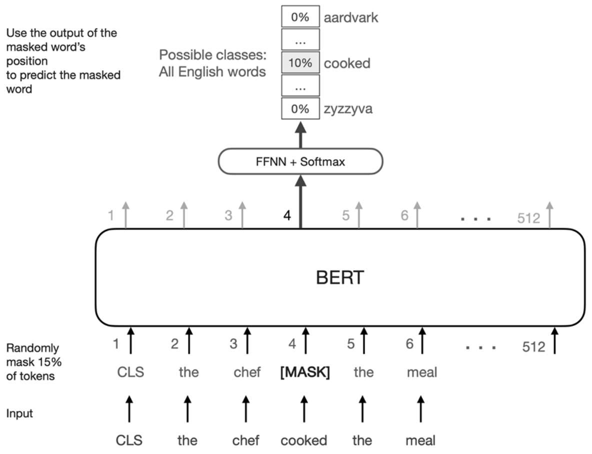
D. FastText

FastText [14] is a library to efficiently represent the words with vectors and classify the texts both unsupervised and supervised learning. Word vectors are generated based on the co-occurred words to keep semantic and syntactic information of the words. Two different algorithms are provided for computing the vector representations: Skip-gram and CBOW (continuous-bag-of-words). The Skip-gram model predicts the words that occur with the target word. On the contrary, the CBOW model predicts the target word using its nearby words. While training the word vectors, FastText uses n-gram of characters that compose the word. The value of n can range from one to the length of the word. Thus, the vector of a word that has not been seen before in the model is learned from the n-gram combinations of the word. This feature is the main contribution of FastText which differs the model from other word embedding methods such as word2vec [15] or GloVe [16]. Besides, FastText provides text classification using word vectors as features. According to [17], the model classifies the texts as accurate as deep learning classifiers, and it is faster for training and evaluation.

E. Transformers

1) BERT

Transformers architecture is a new and simple network model based on attention mechanism which provides focusing on desired parts of the input. It has been applied on sequence-to-sequence tasks, which is able to cope with the long-term dependencies despite having feed-forward networks. Unlike directional models that read the text input sequentially (left to right or right to left), BERT [18] reads the entire sequence at the same time. This feature allows the model to learn the context of a word based on its entire surroundings (left and right of the word). BERT, is a masked language modeling (MLM), breaks the 15% of input by replacing some terms with [MASK] and then trains a model to reconstruct the original terms. Fig. 1 illustrates BERT language modeling.

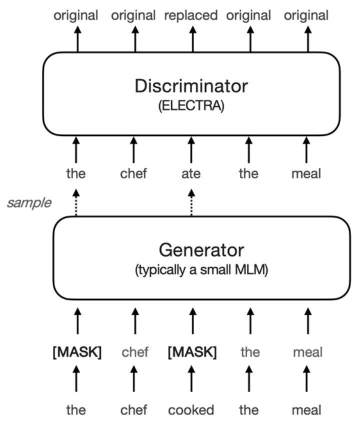

Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0% aardvark
...
10% cooked
...
0% zyzzyva
FFNN + Softmax
1 2 3 4 5 6 ... 512
BERT
Randomly mask 15% of tokens
1 2 3 4 5 6 ... 512
CLS the chef [MASK] the meal
Input
CLS the chef cooked the meal
Fig. 1. BERT language modeling.

2) ELECTRA

ELECTRA [19] consists of two components, the generator and the discriminator. Unlike estimating the original state of the terms modified by masking applied in the BERT model, ELECTRA predicts with the discriminator model whether each term in the input with some terms modified has been changed by a generator. Thus, the task was made more efficient by defining it to all input terms instead of just focusing on the masked subset. Fig. 2 shows the setup for ELECTRA pre-training.

III. Experiment Stages

In this study, we used Twitter data which includes 36,232 posts. Text processing is the first and essential step for NLP tasks and it affects the network performance. The data has already been pre-processed by Çöltekin [5] and we utilized lower casing technique for pre-processing. To implement Machine Learning models, text data needs to be encoded by converting it to numeric value or vector. We tokenized and vectorized the tweets using different techniques (TF-IDF, SentencePiece, Byte-Pair Encoding and WordPiece). After vectorization of the data, we experimented models mentioned in the Section II.


original original replaced original original
Discriminator (ELECTRA)
the chef ate the meal
sample
Generator (typically a small MLM)
[MASK] chef [MASK] the meal
the chef cooked the meal
Fig. 2. Setup for ELECTRA pre-training.

A. Dataset

In this study, we used offensive-turkish dataset [5]. The dataset was created by randomly sampling tweets from the Twitter stream over a period of 18 months. To detect tweets with offensive language, strategies such as searching for Twitter messages containing specified words or following the posts of users who have offensive language style were not used. It has been observed that the ratio of posts with offensive language to randomly collected messages is generally around 10%.

This ensures that the resulting data is less biased and better represents the use of offensive language on this platform. Often each tweet was tagged by a volunteer. Agreement between labelers was also reported in the study. According to this, 92.3% agreement (Cohen's κ = 0.761) was reached from 4820 tweets. While 31,756 samples in the dataset were reserved for training, 3,528 tweets were used for testing. The distribution of labels in the training and test dataset is similar. For both datasets, 20% of the samples have offensive language.

The dataset has been pre-processed in order to fit the models. All user mentions in posts were anonymized with @USER. For all our models, we have converted the letters to lowercase in the tweet messages.

B. Vectorization of Text Data

Term frequency-inverse document frequency (TF-IDF) [20] score of each word was used as a feature to train the Logistic Regression model. The term frequency (TF) is the number of times a word appears in the given text. The inverse document frequency (IDF) is dividing the total number of documents by the number of documents that contain a word on the log base e. If the IDF value of the word is closer to 0, it means that the word is common and appears frequently in many documents. Multiplying these two values results in the TF-IDF score of a word in a document. As a result, if the word has a high TF-IDF score, the relevance of this word is high with those corresponding documents.

For the DNN and GRU models, inputs were vectorized by SentencePiece [21] tokenization. SentencePiece provides unsupervised text tokenization and de-tokenization without text pre/post-processing and language dependency. With this tokenizer, the most frequent and diverse subwords can be captured in fixed vocabulary size. In order to build up word vocabulary consisting of subword components, SentencePiece uses two main word segmentation algorithms Byte-Pair Encoding (BPE) [22] and unigram language model [23]. BPE forms new segmentation based on the next highest frequency symbol pair. In contrast to BPE, the unigram language model is not based on the merge rule. It removes the symbol which causes the lowest loss increase, from the vocabulary. SentencePiece treats the text as a sequence of unicode characters and white space is one of these characters. In this way, it also preserves white spaces with the meta (_) symbol which enables the de-tokenization without losing any information from a text.

For the Transformers, BERT tokenizer which is based on WordPiece [24] tokenizer, is implemented to the input texts. WordPiece is the subword segmentation algorithm and it is very similar to the BPE tokenizer. WordPiece is based on choosing the symbol combinations that maximize the likelihood of training data while BPE forms the new segmentation based on next highest frequency symbol pair.

C. Hyperparameter Tuning and Training of the Models

We used Logistic Regression (LR) as a baseline to compare methods with each other. TF-IDF was applied for the vectorization process of the Twitter post dataset to prepare input for the LR model. We selected the range of word n-gram as 1-3. We performed hyperparameter tuning for regularization and iteration number to obtain optimal parameters for LR. The best performing parameter combination obtained when the regularization parameters (C) is 10 and the number of iterations is 200. In order to perform TF-IDF and construct the models we utilized Sklearn library.

For the Deep Neural Network (DNN) and Gated Recurrent Unit (GRU) models, we used word embedding layer to obtain an efficient representation of textual data. In word embedding methods, words with similar meaning have a similar encoding. Word vectors are initialized with random weights and weight parameters are learned during training by the models while the models learn the network parameters. The size of the embedding vector in our study was selected as 128. The models were optimized with the Adam [25] optimizer. In order to prevent overfitting, we used dropout regularization with a rate of 0.5 on both models. Sigmoid activation function was used in the classification layer. During the training, we monitored the validation loss metric which is Dice-coefficient loss. The models were trained with 150 epochs. However, if no improvement is noted for 5 epochs, training is stopped to prevent overfitting. In order to develop the models, we utilized Keras library.

We applied FastText text classification model by changing the default parameters. We set the learning rate to 0.1, n-gram value as 3 and maximum character n-gram as 5 in our text classification model for training. The model was trained with 12 epochs.

For the Transformers models we used pre-trained BERT (BERTurk) and ELECTRA models for the Turkish language by fine-tuning with our dataset. BERTurk was trained with a very huge amount of dataset which has a size of 35GB and it has 44,04,976,662 tokens. It was trained on a TPU v3-8 for 2M steps by using Google's TensorFlow Research Cloud (TFRC) infrastructure. The pre-trained ELECTRA model uses the same dataset as BERTurk. The models were optimized with the Adam [25] optimizer. Cross Entropy Loss was used as a loss function. The models were trained with 10 epochs. We used dropout regularization with a rate of 0.3 and 0.00001 learning rate. As an activation function, we used Rectified Linear Unit (ReLU).

IV. Results

We reported precision, recall and F1-score results of the models in Table I. We considered F-1 macro averaged score as the main metric when comparing model performances. According to the results, it is seen that the F1 scores of LR, FastText, DNN and GRU are close to each other. Interestingly, it is seen that the LR model we chose as the base model in this study obtained better results than the aforementioned models. The reason of this success can be occurred that the TF-IDF vectorization has a better representation of the dataset. On the other hand, it has been observed that BERT and ELECTRA models achieved higher F-1 scores compared to other models. Among all models, ELECTRA has the highest F-1 score.

TABLE I: Precision, Recall and F1-Score Results of the Models

Models	Precision	Recall	F1 score
ELECTRA	0.82	**0.82**	**0.8216**
BERT	**0.83**	0.78	0.8053
LR	0.76	0.76	0.7619
FastText	0.75	0.74	0.7535
DNN	0.75	0.74	0.7450
GRU	0.76	0.73	0.7446

TABLE II: Ensemble Model Result and Top 10 Models that Ranked According to Macro Averaged F1 In the Semeval-2020 Task 12 [7] Competition for Turkish Language Task A

Models	F1 score
Ensemble Model (ELECTRA models+BERT)	**0.8342**
Galileo [26]	0.8258
ELECTRA	0.8216
SU-NLP [27]	0.8167
KUISAIL [28]	0.8141
KS@LTH [29]	0.8101
NLPDove [30]	0.7967
TysonYU	0.7933
RGCL	0.7859
Rouges [31]	0.7815
GruPaTo [32]	0.7790
MindCoders	0.7789

To obtain more accurate result, we built an ensemble model that uses variations of ELECTRA models and BERT. We employed three different ELECTRA models using different training parameters. We utilized the average output estimation of four models as an output. With this ensemble model, we achieved 0.8342 F1-score which is the highest score that we have reached.

We also compared our results with the successful models from Semeval-2020 task 12 [7] competition in Table II. Our ensemble model has obtained the highest success among the 10 most successful models in task A (whether the language used is offensive or not) performed for the Turkish language of the competition for the offensive-turkish dataset [5]. Also, F-1 score of ELECTRA took third place in the table where it was applied separately. It has been observed that successful results in the competition are also Transformers-based. We conclude that the contribution of the use of large-scale pre-trained transformers is a big part of the success of transformer models.

V. Conclusion

In this paper, different architectures have been applied to detect the use of offensive language in Twitter messages as a text classification problem. According to the results, transformer-based deep networks outperformed other methods. Among all models, ELECTRA has achieved the highest success.

We also provided performance improvement with the ensemble model by combining the BERT and ELECTRA variations. On the other hand, the use of Logistic Regression with TF-IDF has been relatively more successful than FastText and Deep Neural Network applications.

Considering all the experimental results, we've obtained that the pre-trained transformed based models perform much better. The main factor of this success is that these models have embodied the representation of the language structure by pre-training with a large-scale corpus. Other methods have no pre-training that will enable them to learn the structure of the language since models can only be trained on the relevant dataset.

The disadvantage of the proposed approach is associated with the overall training cost. Training transformers networks requires a large amount of computation time and dataset and consumes a significant amount of power when pre-trained models are not preferred to use. Therefore, we plan to apply different tokenization and vectorization algorithms to increase the performance of other methods in future work.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Beyzanur Saraclar and Birol Kuyumcu developed methodologies for offensive language prediction and analyzed the data. All authors evaluate the experimental results. Beyzanur Saraclar and Selman Delil wrote the paper. All authors had approved the final version.

References

[1] U. Dogan and T. S. Çolak, "Self-concealment, social network sites usage, social appearance anxiety, loneliness of high school students: a model testing," *Journal of Education and Training Studies*, vol. 4, no. 6, pp. 176–183, 2016.

[2] J. Juvonen and E. F Gross, "Extending the school grounds? Bullying experiences in cyberspace," *Journal of School Health*, vol. 78, no. 9, pp. 496–505, 2008.

[3] H. Ketmaneechairat and M. Maliyaem, "Natural language processing for disaster management using conditional random fields," *Journal of Advances in Information Technology*, pp. 97–102, 2020.

[4] D. Guo, "Tracking student sentiment from social media," *Journal of Advances in Information Technology*, vol. 6, no. 2, 2015.

[5] Ç. Çöltekin, "A corpus of Turkish offensive language on social media," in *Proc. the 12th Language Resources and Evaluation Conference*, Marseille, France, 2020, pp. 6174–6184.

[6] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra *et al.*, "Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)," *arXiv preprint arXiv:1903.08983*, 2019.

[7] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova *et al.*, "Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020)," *arXiv preprint arXiv:2006.07235,* 2020.

[8] S. A. Özel, E. Saraç, S. Akdemir, and H. Aksu, "Detection of cyberbullying on social media messages in Turkish," in *Proc. 2017 International Conference on Computer Science and Engineering (UBMK),* pp. 366–370.

[9] A Bozyigit, S Utku, and E. Nasibov, "*Sanal Zorbalık İçeren Sosyal Medya Mesajlarının Tespiti*, pp. 366–370, 2018.

[10] A. Bozyiğit, S. Utku, and E. Nasiboğlu, "Cyberbullying detection by using artificial neural network models," in *Proc. 2019 4th International Conference on Computer Science and Engineering (UBMK),* IEEE, 2019, pp. 520–524.

[11] E. P. Ön and R. Yeniterzi. "Cyberbullying detection using deep learning and word embedding analysis," in *Proc. 2020 28th Signal Processing and Communications Applications Conference (SIU)*, 2020, pp. 1–4.

[12] A. Bozyiğit, S. Utku, and E. Nasibov, "Cyberbullying detection: Utilizing social media features," *Expert Systems with Applications* vol. 179, 2021.

[13] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing EMNLP*, 2014, pp. 1724–1734.

[14] B. Piotr, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics,* vol. 5, pp. 135-146, 2017.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[18] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] K. Clark, M. T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[20] T. Tokunaga and I. Makoto, "Text categorization based on weighted inverse document frequency," *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ*), 1994.

[21] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226,* 2018).

[22] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine translation of rare words with subword units," in *Proc. the 54th Annual Meeting of the Association for Computational Linguistics,* Berlin, Germany, 2015, vol. 1, pp. 1715–1725.

[23] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[24] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014).

[26] S. Wang, J. Liu, X. Ouyang, and Y. Sun, "Galileo at SemEval-2020 Task 12: Multi-lingual Learning for Offensive Language Identification Using Pre-trained Language Models," in *Proc. the Fourteenth Workshop on Semantic Evaluation. International Committee for Computational Linguistics,* Barcelona (online), 2020, pp. 1448–1455.

[27] A. Ozdemir and R. Yeniterzi, "SU-NLP at SemEval-2020 Task 12: Offensive language identification in Turkish Tweets," in *Proc. the Fourteenth Workshop on Semantic Evaluation*, 2020*,* pp. 2171–2176.

[28] A. Safaya, M. Abdullatif, and D. Yuret, "Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media," in *Proc. the Fourteenth Workshop on Semantic Evaluation*, 2020*,* pp. 2054–2059.

[29] K. Socha, "KS@ LTH at SemEval-2020 Task 12: Fine-tuning multi- and monolingual transformer models for offensive language detection," in *Proc. the Fourteenth Workshop on Semantic Evaluation*, 2020, pp. 2045–2053.

[30] H. Ahn, J. Sun, C. Y. Park, and J. Seo, "NLPDove at SemEval-2020 Task 12: Improving Offensive Language Detection with Cross-lingual Transfer," *arXiv preprint arXiv:2008.01354*, 2020.

[31] T Dadu and K. Pant, "Team Rouges at SemEval-2020 Task 12: Cross-lingual inductive transfer to detect offensive language," in *Proc. the Fourteenth Workshop on Semantic* Evaluation, 2020, pp. 2183–2189.

[32] D. Colla, T. Caselli, V. Basile, J. Mitrović, and M. Granitzer. "Grupato at semeval-2020 task 12: Retraining mbert on socialmedia and fine-tuned offensive language models," in *Proc. the Fourteenth Workshop on Semantic Evaluation*, 2020*,* pp. 1546–1554.