I2C at SemEval-2020 Task 12: Simple but Effective Approaches to Offensive Speech Detection in Twitter

Victoria Pachón Álvarez
Escuela Técnica Superior de Ingeniería
Universidad de Huelva (Spain)
vpachon@uhu.es

Jacinto Mata Vázquez
Escuela Técnica Superior de Ingeniería
Universidad de Huelva (Spain)
mata@uhu.es

José Manuel López Betanzos
Escuela Técnica Superior de Ingeniería
Universidad de Huelva (Spain)
josemanuel.betanzos@icloud.com

José Luis Arjona Fernández
Escuela Técnica Superior de Ingeniería
Universidad de Huelva (Spain)
jose.arjona@dti.uhu.es

Abstract

This paper describes the systems developed for I2C Group to participate on Subtasks A and B in English, and Subtask A in Turkish and Arabic in OffensEval (Task 12 of SemEval 2020). In our experiments we compare three architectures we have developed, two based on Transformer and the other based on classical machine learning algorithms. In this paper, the proposed architectures are described, and the results obtained by our systems are presented.

1 Introduction

Social media is, currently, one of the most popular mass media. Much of the world's population communicates and expresses their opinions through these media. Unfortunately, the use of offensive language is very frequent in social networks, having detrimental effects for its users. In (Salminen et al. 2020), the authors categorize the hate language as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats or toxicity. All of them have in common the use of offensive language, being a major threat to social media platforms.

In recent years, automatic detection of offensive language on social media, using machine learning algorithms, has aroused great interest among researchers. In (Schmidt and Wiegand 2017), authors present a survey on automatic hate speech detection using NLP, and a critical overview on how the automatic detection of hate speech in text has evolved over the last years is described in (Fortuna and Nunes 2018). The early works related to offensive language detection employing the use of features like bag of words, word and character n-grams in conjunction with classical machine learning classifiers (Nobata et al. 2016). NLP approaches have the drawback of being dependent on the language used in the text so, recently, other machine learning approaches such as neural networks or deep learning are being used (Pitsilis, Ramampiaro, and Langseth 2018).

Most of the researches to detect offensive language focus on the English language. However, there are not many studies in other languages. In this sense, the proposal of 2020 edition of OffensEval is very novel, also offering corpus labeled in Turkish, Arabic, Danish and Greek.

In this paper, we present our participation in OffensEval (task 12 of SemEval 2020) as I2C team. We focus our efforts on subtasks A and B for English, but we also have worked on Subtask A in Arabic and Turkish.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1968–1977
Barcelona, Spain (Online), December 12, 2020.

2 Data description and preprocess

We participated in Subtask A (Offensive language identification) in English, Turkish and Arabic, and in Subtask B (Automatic categorization of offense types) in English (Zampieri et al. 2020).

Dataset Subtask A in English. The data set consisted of 9,075,418 file records. Each of file record have the following format (Rosenthal et al. 2020):

ID | TWEET | AVG_CONF | CONF_STD

Where **ID** is the tweet identifier, **TWEET** is the text of the tweet, **AVG_CONF** is the average of the confidences predicted by several supervised models for this specific instance and **CONF_STD** is the confidence standard deviation for an instance and a corresponding class.

Since there was no criteria to set the threshold for selecting the tweets with offensive language, and based on our experience with other similar corpus, we decided that values of AVG_CONF bigger than 0.6 should be considered as offensive (1) and not offensive (0) on the opposite case. Using this split, the number of tweets with offensive content was 1,042,905 and not offensive 8,032,513 (11% of the tweets is offensive as opposed of 89% with not offensive content). Due to the large number of tweets we decided to reduce to 20,000 tweets chosen randomly. Different types of files, balanced and imbalanced (keeping the percentages to carry out the experiments described in Section 3), were created. Also, we created a version of each of these datasets with "*clean tweets*" where the "#" of each hashtag was removed and when the hashtags had words joined by "_", we divided it into simple words. The "URL" and "@user" was removed, but stop words stayed so as to not lose the sentence meaning. Finally, emojis were substituted by words that represents the same meaning, using the demojize tool[2].

Dataset Subtask B in English. The dataset consisted of 188,974 file records (same Subtask A format) (Rosenthal et al. 2020). However, the meaning of AVG_CONF is different (low values meant the tweet does not show targeted insult). In this case, AVG_CONF represents tweets with targeted insult (if the value is higher than 0.6) or not (if this value es less than 0.6). The number of tweets with targeted insult is 17,825 and 171,148 without. It represents 9% and 91% respectively. Also, we decided to reduce to data sample to 20,000 file records chosen randomly in two files with data balanced and imbalanced with the original percentage. The rest of preprocessing is equals to those in Subtask A.

Dataset Subtask A in Turkish. The Turkish dataset for Subtask A consisted of 31,756 file records with the following format (Çöltekin 2020):

ID | INSTANCE | SUBA

Where **ID** is the tweet identifier, **INSTANCE** is the text of the tweet, **SUBA** could be two values: NOT (the tweet has no offensive language or profanity) or OFF (the tweet has offensive language or profanity).

We changed the values NOT and OFF by 0 and 1 respectively. The number of tweets labelled as NOT was 25,622, while the OFF as 6,134 (81% class NOT, 19% the class OFF). Because there are more tools for English language, we tried to translate each tweet to English, but results were not as good as we expected so we decided not translate to English and use each tweet in its original language. The rest of preprocessing is equal to those in Subtask A in English.

Dataset Subtask A in Arabic. The Arabic dataset consisted of 8,000 file records with the same format as dataset subtask in Turkish (Mubarak et al. 2020). We applied the same preprocessing to the subtask in Turkish. The class balance is the following, the tweet labelled as NOT is 6,411, while the tweet labelled as OFF is 1,589. The percentage is 80% to the class NOT and 20% to the class OFF.

2 https://pypi.org/project/emoji/

3 Methodology and experiments

For all the subtasks, the methodology behind our approach was oriented to find the best combination of algorithms and training dataset to elaborate our models.

We decided to test BERT (Devlin et al. 2018), RoBERTa (Liu et al. 2019), DistilBERT (Sanh et al. 2019), ALBERT (Lan et al. 2019), XLM-RoBERTa (Conneau et al. 2019b), Transformer XL (Dai et al. 2019), XLM (Lample and Conneau 2019), XLMNet (Yang et al. 2019) and XLM-RoBERTa (Conneau et al. 2019a) pretrained models from Transformers (Wolf et al. 2019), and some of the classical machine learning algorithms: Logistic Regression (H. Wang and F. Hao 2012), SVM (Chang and Lin 2011), XGBoost (Chen and Guestrin 2016), RandomForest (Breiman 2001) and K-Neighbours (J. Laaksonen and E. Oja 1996) for Subtask A and Subtask B in English and only BERT for Subtask A in Arabic and Turkish.

To obtain the best model for each subtask, 3 architectures were tested. In the first (#Architecture 1), each tweet was preprocessed (hashtags, urls, @users, emojis, ... were removed) using the "Bag of Word" model with TF-IDF for giving weight to the word. Stop words were also removed and all words were transformed to lowercase, and lemmatization was used to reduce the vocabulary of words. In this approach, the generated vector is passed as input to the machine learning method. The algorithms applied were Lineal Regression, SVM (RBF kernel), XGBoost, RandomForest and K-Neighbors. We added the hate words existence on each tweet, lexical characteristics and entities using the spaCy[3] tool. The number of times that any of these hate words appears in the tweet indicates the weight of the characteristic itself. Due to the large number of characteristics, a selection of them was made. To make this selection we used a method which eliminates all the characteristics except for the best N of them, where N was 20, 50 and 100. In our case, the chi^2 test was used as selection criterion for the best characteristics.

For this architecture we used, for all the experiments, balanced and imbalanced datasets with 20,000 random tweets (training 80%, test 20%). For the balanced dataset, 10,000 random tweets were selected from each of the classes in the original file, and for the imbalanced dataset, the same proportion as the original files had, that is 89% and 11% for Subtask A, and 91% and 9% for Subtask B.

The second tested architecture (#Architecture 2) is based on generating the features with a set of transformers (Wolf, Debut et al. 2019) and using them to feed the same machine learning methods mentioned above (Figure 1). We decided to test the models BERT, TransfomerXL, XLM, XLMNet, DistilBERT, RoBERTa y XLM-RoBERTa. We did not use pre-processing, in order to benefit from the representation's capabilities of transformers. We tokenized the tweets with a specific tokenizer provided by transformer for each model.

The third approach tested (#Architecture 3) is based only in transformers. We decided to test BERT, RoBERTa, DistilBERT, ALBERT and XLM-RoBERTa pretrained models from transformers. Again, we did not use pre-processing, in order to benefit from the representation's capabilities of transformers. We tokenized the tweets with a specific tokenizer provided by transformer for each model. For the finetuning of BERT and others, we used the default parameters of their respective repository but trained on 2 epochs and a maximum sequence-length of 128.

Because the original file was huge and imbalanced, for #Architecture 2 and #Architecture 3 we carried out two tests: one with a random data set with the same proportion of tweets of each class as the original file supplied by the organizers, and the other with a balanced data set (both datasets with 20,000 tweets) and a maximum sequence length of 128, epoch.

[3] https://spacy.io/

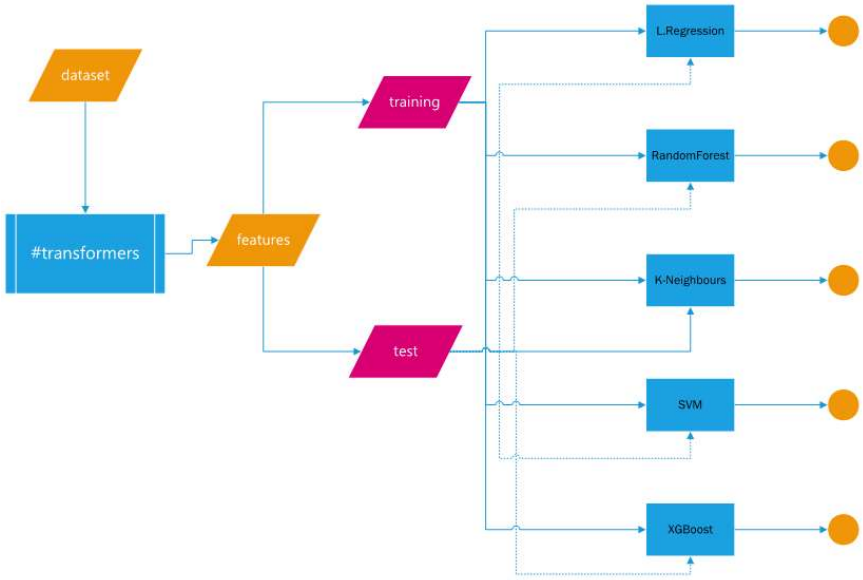

Figure 1. #Architecture 2. Use of BERT to embed all the tweets and feed machine learning models

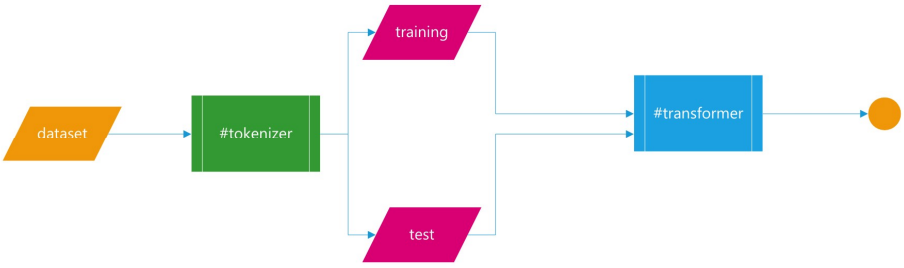

Figure 2. #Architecture 3. Use of transformers to obtain a model

4 Results

In this section, we describe the systems developed for the SemEval20 Subtask A in English, Turkish and Arabic and Subtask B in English.

Subtasks A and B in English. In first place, the corpus of tweets provided by the organizers was pre-processed. Using *#Architecture 1* described in the previous section, we tested the algorithms Logistic Regression, RandomForest, XGBoost, SVM and K-Neighbours. Results are shown in Tables 1, 2, 3 and 4 (best result marked in bold).

	F1-score		
Model	**#20**	**#50**	**#100**
Logistic Regression	0.80	0.81	**0.85**
Random Forest	0.78	0.79	0.82
XGB	0.76	0.76	0.76
SVM	**0.82**	**0.83**	**0.85**
K-Neighbours	0.76	0.76	0.77

Table 1. Results for Subtask A using TF-IDF Imbalanced Data

	F1-score		
Model	**#20**	**#50**	**#100**
Logistic Regression	**0.82**	**0.82**	0.83
Random Forest	0.80	0.81	**0.84**
XGB	0.77	0.78	0.78
SVM	0.79	**0.82**	0.83
K-Neighbours	0.77	0.77	0.78

Table 2. Results for Subtask A using TF-IDF Balanced Data

	F1-score		
Model	**#20**	**#50**	**#100**
Logistic Regression	0.74	0.74	**0.73**
Random Forest	**0.80**	**0.77**	0.71
XGB	**0.80**	0.70	0.61
SVM	0.71	0.70	0.66
K-Neighbours	0.79	0.73	0.67

Table 3. Results for Subtask B using TF-IDF Imbalanced Data

	F1-score		
Model	**#20**	**#50**	**#100**
Logistic Regression	0.70	0.74	0.76
Random Forest	0.80	**0.80**	**0.78**
XGB	0.80	**0.80**	0.77
SVM	0.70	0.73	0.77
K-Neighbours	**0.81**	0.79	0.75

Table 4. Results for Subtask B using TF-IDF Balanced Data

Before testing #Architecture 2, we decided to test how transformers behave with balanced and imbalanced datasets. In the same way as #Architecture 1, two datasets of 20,000 tweets each were built. For the balanced dataset, 10,000 tweets from each class were randomly selected, and for the imbalanced dataset, we selected 20,000 tweets with the same ratio of classes as the initial dataset.

The results are shown in Table 5 for Subtask A in English and Table 6 for Subtask B. For both subtasks, as it was expected, balanced dataset performed, on the whole, the best results.

	#Architecture 3		**#Architecture 2 Logistic Regression**	
	F_1-score		**F_1-score**	
Model (Pretrained Model)	**Balanced Dataset**	**Imbalanced Dataset**	**Balanced Dataset**	**Imbalanced Dataset**
BERT (bert-base-uncased)	0.96	0.92	0.84	0.74
XLM (xlm-mlm-tlm-xnli15-1024)	0.33	0.47	0.67	0.70
RoBERTa (roberta-base)	0.33	0.47	0.73	0.47
ALBERT (albert-base-v1)	0.91	0.89	-	-
DistilBERT (distilbert-base-cased)	0.95	0.92	0.83	0.72

Table 5. Sampling Results for Subtask A (English)

Model (Pretrained Model)	#Architecture 3		#Architecture 2 Logistic Regression	
	F_1-score		F_1-score	
	Balanced Dataset	Imbalanced Dataset	Balanced Dataset	Imbalanced Dataset
BERT (bert-base-uncased)	0.88	0.57	0.79	0.59
XLM (xlm-mlm-tlm-xnli15-1024)	0.33	0.47	0.69	0.70
RoBERTa (roberta-base)	0.34	0.65	0.64	0.48
ALBERT (albert-base-v1)	0.86	0.72	-	0.61
DistilBERT (distilbert-base-cased)	0.88	0.75	0.77	0.54

Table 6. Sampling Results for Subtask B (English)

Values of 0.33 and 0.34 were obtained when the model classified all the instances as belonging to only one class.

Using a balanced dataset, we decided to fine-tune the pretrained models showed in tables 5 and 6 for Subtasks A and B in English respectively. Best results were performed by *bert-base-uncased* and *distiltbert-base-cased* pretrained models. Our approach for Subtasks A and B in English is based on *bert-base-uncased* fine-tuned with a 20,000 random tweets dataset without cleaning phase.

Results for balanced training dataset for #Architecture2 are shown in Tables 7 and 8.

Model (Pretrained Model)	F1-score				
	Logistic Regression	Random Forest	XGB	SVM	K-Neighbours
BERT (bert-base-case)	0.79	0.66	0.72	0.72	0.60
BERT (bert-base-uncased)	**0.84**	0.72	0.77	0.77	0.62
XLM (xlm-mlm-tlm-xnli15-1024)	0.67	0.63	0.64	0.70	0.57
RoBERTa (roberta-base)	0.73	0.72	0.78	0.33	0.63
DistilBERT (distilbert-base-cased)	0.83	0.74	0.77	0.65	0.59
XLNet (xlnet-base-cased)	0.83	0.76	0.79	0.76	0.66
XLMRoBERTa (xlm-roberta-base)	0.58	0.59	0.63	0.33	0.54

Table 7. Results for Subtask A in English using #Architecture 2

Model (Pretrained Model)	F1-score				
	Logistic Regression	Random Forest	XGB	SVM	K-Neighbours
BERT (bert-base-case)	0.75	0.66	0.69	0.67	0.61
BERT (bert-base-uncased)	**0.79**	0.71	0.75	0.73	0.66
XLM (xlm-mlm-tlm-xnli15-1024)	0.69	0.64	0.66	0.70	0.60
RoBERTa (roberta-base)	0.64	0.70	0.75	0.34	0.64
DistilBERT (distilbert-base-cased)	0.77	0.70	0.75	0.68	0.65
XLNet (xlnet-base-cased)	0.68	0.64	0.68	0.65	0.56
XLMRoBERTa (xlm-roberta-base)	0.60	0.61	0.61	0.51	0.55

Table 8. Results for Subtask B in English using #Architecture 2

In Tables 9 and the results for Subtasks A and B in English using #Architecture 3 are shown. For these experiments, several pretrained models were tested for some of the models.

Model	Pretrained model	F_1-score
BERT	**bert-base-uncased**	**0.96**
	bert-large-uncased	0.33
	bert-base-cased	0.96
	bert-large-uncased-whole-word-masking	0.34
	bert-large-uncased-whole-word-masking-finetuned-squad	**0.96**
ALBERT	albert-base-v1	0.91
	albert-large-v1	0.34
	albert-xlarge-v1	0.34
	albert-xxlarge-v1	0.33
	albert-large-v2	**0.96**
	albert-xlarge-v2	0.33
	albert-xxlarge-v2	0.33
RoBERTa	distilroberta-base	0.95
	roberta-base	0.33
DistilBERT	distilbert-base-cased	0.95
XLM	xlm-mlm-tlm-xnli15-1024	0.33

Table 9. F1-score results (Subtask A - English)

Model	Pretrained model	F_1-score
ALBERT	albert-base-v1	0.86
	albert-large-v1	0.86
	albert-xlarge-v1	0.34
	albert-xxlarge-v1	0.33
	albert-large-v2	0.61
	albert-xlarge-v2	0.33
	albert-xxlarge-v2	0.33
BERT	**bert-base-uncased**	**0.88**
	bert-large-uncased	0.33
	bert-base-cased	0.87
	bert-large-uncased-whole-word-masking	0.34
	bert-large-uncased-whole-word-masking-finetuned-squad	0.87
RoBERTa	distilroberta-base	0.86
	roberta-base	0.34
DistilBERT	**distilbert-base-cased**	**0.88**
XLM	xlm-mlm-tlm-xnli15-1024	0.33

Table 10. F1-score results (Subtask B - English)

Subtask A in Turkish. Taken into account the results obtained with transformers for Subtask A in English, for this subtask, it was decided to test only BERT *multilingual-cased* pretrained model. File supplied by the organization was imbalanced and the number of instances of the minority class required a prior study regarding the need for sampling. Several sampling techniques were tested as it is shown in Table 11 and the most appropriate was Smote-Tomek approach. For these tests, train and test files were obtained randomly from the 85% and 15% of the original file. Our approach to Subtask A in Turkish was to fine tune *bert-multilingual* pretrained model with the full dataset previously sampled by Smote-Tomek.

Sampling Technique	F_1-score
No sampling	0.77
Under sampling majority class	0.75
K-neighbors	0.91
Oversampling minority class	0.90
Smote	**0.92**

Table 11. Results for Subtask A (Turkish)

Subtask A in Arabic. For this task, it was decided to test only BERT *multilingual-cased* pretrained model. Again, file supplied was imbalanced. Due to the number of files of the minority class, the original file was divided into train (85%) and test (15%) and several tests were carried out to find out which technique was best suited to the multilingual BERT pretrained model. As it can be seen in Table 12, Smote-Tomek approach reached the best results so we fine tune the multilingual BERT pretrained model with balanced Smote-Tomek dataset obtained from the full dataset supplied by the organization. We also tried to translate each tweet into English, but the results did not improve. Results are shown in Table 12.

Sampling Technique	F_1-score
No sampling	0.82
Under sampling majority class	0.74
K-neighbors	0.92
Oversampling minority class	0.92
Smote	**0.93**

Table 12. Results for Subtask A (Arabic)

Finally, Table 13 shows the results obtained in the competition for the I2C team (best result in the competition raking for each subtask is shown in brackets).

Subtask	F_1-score
English A	0.8919 (0.9222)
English B	0.6011 (0.7461)
Turkish	0.7735 (0.8257)
Arabic	0.8055 (0.9017)

Table 13. Best Results

5 Conclusions

In this paper, we have presented I2C's contribution to SemEval20 for Subtasks A in English, Arabic and Turkish and Subtask B in English. In order to achieve optimal results in the challenge suggested by the organization, three architectures were developed and tested: the first one was based on classical machine learning methods for text classification tasks, the second used a mixed architecture for features generation to embed all the tweets and feed machine learning classical models, and in the last one we fine-tuned a set of pretrained models from transformers to classify the tweets. Among the three architectures, the one based on transformers only obtained the best results, in particular, by BERT using the *bert-base-uncased* pretrained model for the two tasks in English, and BERT using the *bert-base-multilingual* pretrained model for tasks in Arabic and Turkish. We consider that more research should be put into multilingual solutions. Our experiments were done using the default parameters so there is room for improvement with many adjustments that we plan to consider for future works.

References

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5-32.

Chang, Chih-Chung and Chih-Jen Lin. 2011. "LIBSVM: A Library for Support Vector Machines." *ACM Transactions on Intelligent Systems and Technology (TIST)* 2 (3): 1-27.

Chen, Tianqi and Carlos Guestrin. 2016. *XGBoost: A Scalable Tree Boosting System*. Vol. 13-17- ACM.

Çöltekin, Çağrı. 2020. "A Corpus of Turkish Offensive Language on Social Media."ELRA OffensEval 2019 Report, .

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. "Unsupervised Cross-Lingual Representation Learning at Scale." *arXiv Preprint arXiv:1911.02116*.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. "Unsupervised Cross-Lingual Representation Learning at Scale." .

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. "Transformer-Xl: Attentive Language Models Beyond a Fixed-Length Context." *arXiv Preprint arXiv:1901.02860*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." .

Fortuna, Paula and Sérgio Nunes. 2018. "A Survey on Automatic Detection of Hate Speech in Text." *ACM Computing Surveys (CSUR)* 51 (4): 1-30.

H. Wang and F. Hao. 2012. *An Efficient Linear Regression Classifier*. 2012 IEEE International Conference on Signal Processing, Computing and Control.

J. Laaksonen and E. Oja. 1996. *Classification with Learning K-Nearest Neighbors*. Proceedings of International Conference on Neural Networks (ICNN'96). Vol. 3.

Lample, Guillaume and Alexis Conneau. 2019. "Cross-Lingual Language Model Pretraining." *arXiv Preprint arXiv:1901.07291*.

Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations." .

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." .

Mubarak, Hamdy, Ammar Rashed, Kareem Darwish, Younes Samih, and Ahmed Abdelali. 2020. "Arabic Offensive Language on Twitter: Analysis and Experiments." *arXiv Preprint arXiv:2004.02192*.

Nobata, C., J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. 2016. *Abusive Language Detection in Online User Content* International World Wide Web Conferences Steering Committee.

Pitsilis, Georgios, Heri Ramampiaro, and Helge Langseth. 2018. "Effective Hate-Speech Detection in Twitter Data using Recurrent Neural Networks." *Applied Intelligence; the International Journal of Research on Intelligent Systems for Real Life Complex Problems* 48 (12): 4730-4742.

Rosenthal, Sara, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. "A Large-Scale Semi-Supervised Dataset for Offensive Language Identification.".

Salminen, J., M. Hopf, S. A. Chowdhury, S. -G Jung, H. Almerekhi, and B. J. Jansen. 2020. "Developing an Online Hate Classifier for Multiple Social Media Platforms." *Human-Centric Computing and Information Sciences* 10 (1).

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter." .

Schmidt, Anna and Michael Wiegand. 2017. "A Survey on Hate Speech Detection using Natural Language Processing."Association for Computational Linguistics, apr.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, et al. 2019. "HuggingFace's Transformers: State-of-the-Art Natural Language Processing." .

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. "Xlnet: Generalized Autoregressive Pretraining for Language Understanding.".

Zampieri, Marcos, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çöltekin, Çağrı. 2020. "SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020).".