ALT at SemEval-2020 Task 12: Arabic and English Offensive Language Identification in Social Media

Sabit Hassan, Younes Samih, Hamdy Mubarak, Ahmed Abdelali
Qatar Computing Research Institute, Doha, Qatar
{sahassan2,ysamih,hmubarak,aabdelali}@hbku.edu.qa

Abstract

This paper describes the systems submitted by the Arabic Language Technology group (ALT) at SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media. We focus on sub-task A (Offensive Language Identification) for two languages: Arabic and English. Our efforts for both languages achieved more than 90% macro-averaged F1-score on the official test set. For Arabic, the best results were obtained by a system combination of Support Vector Machine, Deep Neural Network, and fine-tuned Bidirectional Encoder Representations from Transformers (BERT). For English, the best results were obtained by fine-tuning BERT.

1 Introduction

Social media has given a platform to users to express their thoughts and beliefs. Then, it is not surprising that in the sea of content generated by users, we often find offensive language that can contain hatred, racism or profanity. Such content may be inappropriate for others and need to be identified so that they can be regulated. At SemEval-2020, Task 12 (Zampieri et al., 2020), we are presented with a multilingual dataset featuring 5 different languages: Arabic, Danish, English, Greek, and Turkish. We focus on sub-task A for Arabic and English. The goal of this sub-task is offensive language identification in tweets.

For Arabic track, we experimented intensively with Support Vector Machines(SVMs), Deep Neural Networks (DNNs) and Multilingual Bidirectional Encoder Representations from Transformers (Multilingual-BERT). Our best results were obtained by a system combination based on majority voting by the aforementioned models. The system combination ranked 2nd (out of 53 teams) in the official rankings, with macro-averaged F1-score of 90.16%, behind by just 0.01% from the top ranked submission (90.17%).

For English track, we again experimented with SVMs, DNNs and Multilingual-BERT. Relatively low performance by SVMs and DNNs with the given training data prompted us to explore external datasets. The best results were obtained by fine-tuning Multilingual-BERT on combination of the given training data and Wikipedia Detox (Wulczyn et al., 2017). The system achieved macro-averaged F1-score of 91.14%, which is ranked 18 (out of 82 teams), trailing from the top ranked submission (92.04%) by less than 1%.

Alongside description of our approaches, we also perform error analysis of our best systems. We manually identify different categories (e.g., humor/sarcasm) where our systems made errors. This provides insight on how the data and the systems can be improved in future.

2 Related Work

Being representative of diverse linguistic phenomena among a great number of users, Arabic Twitter has become a major data source for researchers (Abdelali et al., 2020). Along with popular social media analysis tasks such as sentiment analysis (Elmadany et al., 2018) or dialect identification (Bouamor et al., 2019), offensive language identification for Arabic has gained notable interest among researchers. Mubarak et al. (2017) present a dataset of abusive tweets and deleted comments from Aljazeera. Hassan et al. (2020) use Support Vector Machine (SVM) and deep learning models for offensive language identification in tweets. Mubarak and Darwish (2019) compare performances of SVMs and fastText (Joulin et al., 2016). Albadi et al. (2018) propose recurrent neural networks trained with pre-trained embeddings to detect religious hate-speech in Arabic social media. Alakrot et al. (2018) propose use SVMs to detect offensive language in Youtube comments.

Compared to Arabic, offensive language identification in English has attracted significantly more attention among researchers. Task 6 at SemEval-2019 (Zampieri et al., 2019b) received submissions from 115 teams. The top ranking team (Liu et al., 2019) for offensive language detection (sub-task A) fine-tuned Bidirectional Encoder Representation from Transformer (BERT) (Devlin et al., 2019) for their submission. Some of other notable work on offensive language detection are by Davidson et al. (2017) and Djuric et al. (2015). Davidson et al. (2017) use logistic regression classifier with L2 regularization to detect hate speech and offensive language. Djuric et al. (2015) use distributed representation (Le and Mikolov, 2014) of online comments, which is then used as features for a logistic regression classifier to detect offensive language.

3 Task and Data Description

SemEval-2020 Task 12 consists of three sub-tasks: i) sub-task A: offensive language identification, ii) sub-task B: automatic categorization of offense types, iii) sub-task C: offense target identification. Sub-task-A is available for 5 languages: Arabic, Danish, English, Greek and Turkish. Other sub-tasks are available only for English. We focus on sub-task A for Arabic and English.

The dataset for Arabic (Mubarak et al., 2020) contains 8,000 tweets for training. The tweets are labeled as Offensive (OFF) or Not Offensive (NOT). No separate development data is provided. The blind test set contains 2,000 tweets, made available via Codalab[1]. The task is to predict labels (OFF or NOT) for the tweets. The gold labels for the test set are made available after the evaluation phase.

The dataset for English (Rosenthal et al., 2020) contains ∼9M tweets for training. For each tweet, participants are provided with the following: i) average of the confidences scores by several supervised models for each tweet to belong to the Offensive class, and ii) standard deviation from the average confidence score for each tweet. Similar to Arabic, no development data is provided. The blind test set consists of 3,887 tweets (without any other information). The task is to detect whether these instances are Offensive (OFF) or Not Offensive (NOT). The gold labels for these instances (manually verified) are made available after the evaluation phase.

4 Arabic Offensive Language Identification

In this section, we describe our preprocessing steps, experiments, results and error analysis for Arabic.

4.1 Preprocessing

Our steps for preprocessing the Arabic tweets include: i) split hashtags along ”_”, ii) remove all punctuation and diacritics, iii) normalize Arabic characters to their standard form (e.g. replace أ with ا), and iv) remove all words that contain non-Arabic characters. In our early experiments, we explored impact of other preprocessing steps such as keeping emoji, and replacing emoji with representative words. These extra preprocessing did not affect the performance of systems significantly and were excluded.

4.2 Experiments and Results

Since we are not provided with a development set, we perform 5-fold cross-validation and then average the results to evaluate different individual classifiers (Table 1). Then, we create a system combination from individual classifiers to generate labels for the test set (Table 2). We report accuracy (Acc.), macro averaged precision (P), recall (R), F1-score (F1). We use macro-averaged F1-score (the official metric) for comparison in our discussion.

[1]www.codalab.org

fastText fastText (Joulin et al., 2016) is a light-weight deep-learning based system that learns word embedding representation of text and can perform text classification. Although it offers a fast and easy way to perform text classification, it can be outperformed by more sophisticated models. fastText primarily serves purpose of a baseline in our experiments. From Table 1, we can see that fastText (F1-score of **79.8%**) is outperformed by all other classifiers.

Multilingual-BERT Deep contextualized transformer models such as Bidirectional Encoder Representations from Transformers (Devlin et al., 2019) have been shown to be extremely effective in many NLP tasks. For our task. we fine-tune $\text{BERT}_{\text{base-multilingual}}$ (Mult-BERT) which is pretrained on Wikipedia text from 104 languages (including Arabic). The architecture of the model consists of 12 transformer blocks, hidden size of 768 and 12 self-attention heads. During fine-tuning, we add a softmax layer that projects class probabilities of the labels. We use PyTorch [2] implementation by HuggingFace [3] for the fine-tuning.

We can see from Table 1 that, with F1-score of **82.9%**, Mult-BERT is outperformed by all classifiers except fastText. This could be attributed to the fact that wikipedia text differs significantly from informal Twitter text (domain of the shared task data).

Convolutional-Long Short Term Memory (C-LSTM) We design a neural network that consists of two parts. First part of the network acts as a character-level feature extractor. The first layer is an embedding layer that projects input to character embeddings, which is then passed through a convolutional layer. After applying max-pooling over time, we obtain fixed length representation of words. In the second part, the character-level representation of words are merged with pretrained Mazajak word embeddings (Abu Farha and Magdy, 2019) and passed to the a Bidirectional LSTM layer. Then, a softmax layer is used to make the predictions. Having access to both character-level and word-level features may give the network an edge over just the individual features. With F1-score of **87.1%**, this hybrid network outperforms fastText and Mult-BERT in Table 1.

Support Vector Machines (SVMs) SVMs trained on word and character level features have been shown to perform well for text classification (Hassan et al., 2020). To construct the features in our experiments, we use term frequency-inverse document frequency (tf-idf) vectorizer to produce word and character n-gram vectors. We experimented with different ranges of character and word n-grams. We also experimented with combinations of Mazajak word embeddings (Abu Farha and Magdy, 2019), character n-gram and word n-gram vectors. For conciseness, we report only the most significant results. From Table 1 (C and W refer to character and word respectively), we can see that the combination of n-gram vectors and Mazajak embeddings yield the second-best classifier (F1-score of **88.6%**).

SVMs + ValenceList To reduce errors made by the SVM, we calculated valence score (Conover et al., 2011) for all Arabic words. Valence score helps determine the distinctness of a word relative to a given class. Valence score relative to offensive (OFF) class for any term *t* is computed as below:

$$\vartheta(t)_{OFF} = 2\frac{\frac{freq(t,OFF)}{N(OFF)}}{\frac{freq(t,OFF)}{N(OFF)} + \frac{freq(t,NOT\_OFF))}{N(NOT\_OFF)}} - 1 \tag{1}$$

We take the 160 most frequent words that have valence score of 1. We call it ValenceList. We look at all predictions made by the SVM. If we see that any instance is classified as Not Offensive by the SVM but contains any word from ValenceList, we consider the instance to be offensive. Combining SVM with this ValenceList yields the best individual result (F1-score of **88.9%**) out of all systems in Table 1.

System Combination Out of the classifiers discussed above, we take 3 significantly different but promising classifiers to create an ensemble: i) SVM & ValenceList, ii) CNN-LSTM, and iii) Mult-BERT. We train these classifiers on the full training data and perform majority voting to determine labels for the test set. We expect this system combination to cover weaknesses of the individual classifiers to an extent. Table 2 shows results of this system combination on the official test set. These results are computed

[2]https://pytorch.org/

[3]https://github.com/huggingface/transformers

No.	Model	Features	Acc.	P	R	F1
1	fastText		88.4	84.0	77.0	79.8
2	Mult-BERT		89.9	82.6	83.1	82.9
3	C-LSTM		92.4	86.8	87.3	87.1
4	SVM	C[2-7]	91.4	87.5	84.6	85.9
5	SVM	W[1-3]	90.3	87.53	80.3	83.2
6	SVM	C[2-7] + W[1-3]	91.6	89.5	82.8	85.5
7	SVM	C[1-5] + W[1-3] + Mazajak	92.9	89.3	87.9	88.6
8	7 & ValenceList		**93.0**	**88.9**	**89**	**88.9**

Table 1: Average cross-validation performance for Arabic

Model	Acc.	P	R	F1
SVM & ValenceList + C-LSTM + Mult-BERT	93.85	91.36	89.08	90.16

Table 2: Performance of system combination for Arabic on official test set

post-evaluation phase when test labels were made available. With F1-score of **90.16**, the model placed 2nd in the official ranking.

4.3 Error Analysis

We manually annotated 100 randomly sampled errors made by our best system (system combination from Table 2) on the test set for different categories of error (Figure 1).

False Positives: Misclassifying Not Offensive (NOT) as Offensive (OFF) We noticed that some offensive/mildly offensive data had gold labels of NOT. For example, يا فخامتك يا ثعبان (O your excellency, O snake) is labeled as NOT. Figure 1 shows that such noise in data annotation attributes to about 13% of the false positive errors. Existence of words such as كريه (distasteful) that can have negative or offensive meaning may lead to misclassifying as offensive although the sentence may not be offensive overall. About 25% of the errors that we examined had words with negative or offensive meaning. During our error analysis, we noticed an interesting weakness of our system: it was unable to detect humor/sarcasm effectively. يا دي خيا يا برشلوني يا عميل (O De Gea, agent of Barcelona) is tagged as offensive. We found 15% of the errors had humor or sarcasm rather than offensiveness. We could not identify any specific category for 47% of the errors.

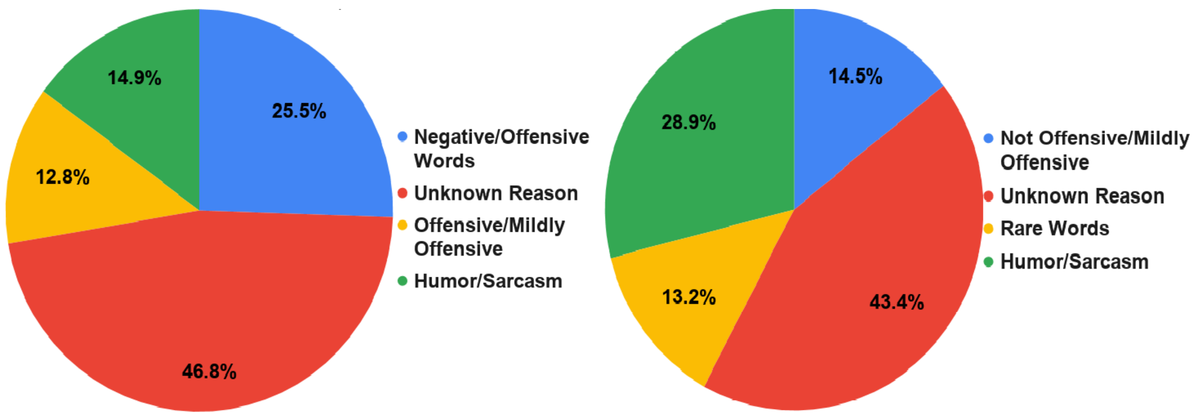


Figure 1: Errors in Arabic when true label is Not Offensive (left) and, Offensive (right)

No.	Model	Features	Training Data	Acc.	P	R	F1
1	fastText		OffEval 2020	81.6	80.9	79.3	79.9
2	SVM	C[3-7]	OffEval 2020	83.6	81.4	83.1	82
3	Mult-BERT		OffEval 2020 + Wiki-Detox	**90.8**	**87.6**	**84.6**	**89**

Table 3: Performance of models for English on OLID 2019

Model	Acc.	P	R	F1
Mult-BERT (fine-tuned on OffEval 2020 + Wiki-Detox)	92.67	90.06	92.47	91.14

Table 4: Performance of Multilingual-BERT for English on official test set

False Negatives: Misclassifying Offensive (OFF) as Not Offensive (NOT) We attribute about 14% of the false negative errors to noisy data: when the instance is not actually offensive or just mildly offensive but has true label of OFF. Once again, we found that humorous or sarcastic instances contributed to a significant portion of the errors (about 29%). About 13% of the error instances had words or phrases that do not appear often (e.g., يا ابن الغوريلا (O son of gorilla)). We could not identify specific reason for about 43% of the errors.

5 English Offensive Language Detection

Unlike Arabic, English does not have diacritics or different forms of characters. Thus, we do not perform the preprocessing from Section 4 and discuss only experiments, results and error analysis.

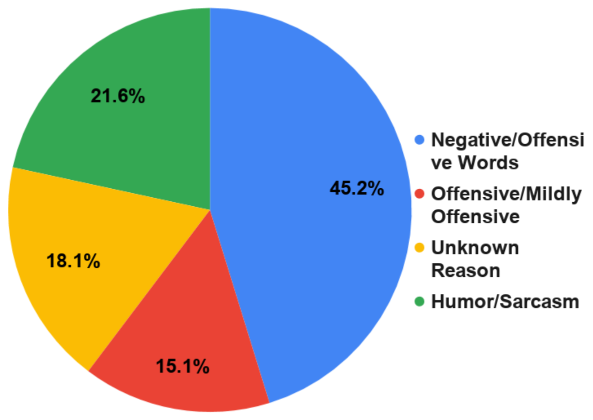
5.1 Experiments and Results

Since we are not provided with an official development set, we use OLID 2019 (Zampieri et al., 2019a) as development dataset in our experiments (Table 3). We report results of our best system on official test set in Table 4. We use the same metrics as Section 4.

fastText For the training set, we are provided with average confidence score (CONF) of several classifiers and standard deviation (STD) from CONF. Higher CONF means the instance is more likely to be offensive and higher STD means the classifiers are in higher disagreement on the instance. If an instance has CONF >= 0.5, we label it OFF and label it NOT if CONF <= 0.3, and discard rest of the instances. The threshold for CONF is skewed to ensure we have enough OFF labels (since it is underrepresented in the data). By discarding instances with high STD, we eliminate the more confusing instances. Similar to Arabic track, fastText (F1 score of 79.9) is outperformed by other classifiers.

Support Vector Machines (SVMs) Adhering to the reasoning described earlier, we experimented with several thresholds and chose the following due to better performance: label OFF if CONF >= 0.40 and STD <= 0.15; label NOT if CONF >= 0.40 and STD <= 0.15. We discard rest of the instances. We create character n-gram vectors in a way similar to Section 4. Other experiments from Section 4 did not improve performance and are omitted. With F1-score of **82%** on OLID 2019, SVM trained on character n-gram vector outperforms fastText.

Multilingual-BERT Due to low performance on the training data so far, we explored addition of external datasets. We choose Wikipedia Detox (Wulczyn et al., 2017) as it is a large dataset annotated for personal attacks, aggression and toxicity —components of offensiveness. First, we map labels of given training data with the same thresholds as the ones for SVMs described earlier. After mapping the labels of Wikipedia Detox to OFF or NOT and adding to the training data, we fine-tuned $\text{BERT}_{\text{base-multilingual}}$ (Mult-BERT) in the same way as Arabic track. This significantly improved performance on OLID 2019 (F1-score of **89%**) and achieved F1-score of **91.14%** on the test set (Table 4), less than 1% behind the top submission.



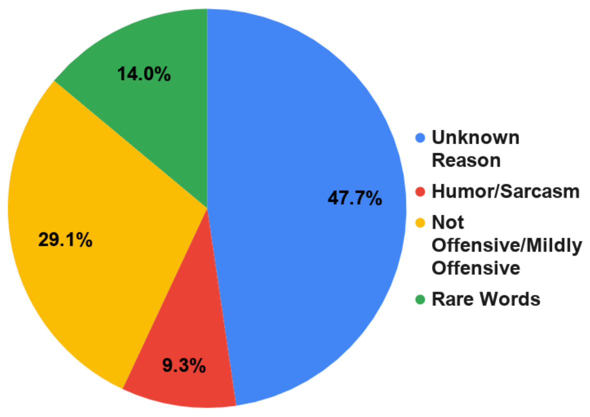


Figure 2: Errors in English when true label is (left) Not Offensive and, (right) Offensive

5.2 Error Analysis

Similar to Arabic track, we randomly sampled 100 errors and annotated them manually for different categories of errors. Figure 2 shows distribution of the errors on the test set by our best system (Mult-BERT from Table 4).

False Positives: Misclassifying Not Offensive (NOT) as Offensive (OFF) Similar to Arabic track, some offensive/mild offensive data had gold labels of NOT. For example, "It's the white supremacy stupid 2020." is labeled as NOT. Such noise in data annotation contributes to about 15% errors. A friendly banter such as "serges_ego you suck at sm4sh huehuehuehuehue" is considered OFF by the system. Such humor/sarcasm contributes to about 22% errors. We found that about 45% of the errors contained words related to offensiveness such as "disgusting". We could not identify any specific category for 18% errors.

False Negatives: Misclassifying Offensive (OFF) as Not Offensive (NOT) We found that about 9% errors can be attributed to humor/sarcasm and about 29% can be attributed to noisy data. We witnessed presence of words such as "narcissist" that have offensive meaning but are not commonly used in about 14% errors. We could not categorize 48% of the errors.

6 Conclusion

In this paper we presented highly effective systems for offensive language identification in Arabic and English tweets. Our best result for Arabic was achieved by a system combination of SVM, DNN and Multilingual BERT. For English, on the other hand, Multilingual-BERT outperformed other classifiers when it was trained on additional external data. The scarcity of resource for Arabic makes such approach difficult, suggesting that there is a need for data augmentation in the future to obtain better results. With our error analysis, we identified several issues (e.g., misclassifications due to humor/sarcasm) that can be addressed in the future to improve performance.

References

Ahmed Abdelali, Hamdy Mubarak, Younes Samih, Sabit Hassan, and Kareem Darwish. 2020. Arabic dialect identification in the wild. *ArXiv*, abs/2005.06557.

Ibrahim Abu Farha and Walid Magdy. 2019. Mazajak: An online Arabic sentiment analyser. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 192–198, Florence, Italy, August. Association for Computational Linguistics.

Azalden Alakrot, Liam Murray, and Nikola S. Nikolov. 2018. Towards accurate detection of offensive language in online communication in arabic. *Procedia Computer Science*, 142:315 – 320. Arabic Computational Linguistics.

N. Albadi, M. Kurdi, and S. Mishra. 2018. Are they our brothers? analysis and detection of religious hate speech in the arabic twittersphere. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 69–76, Aug.

Houda Bouamor, Sabit Hassan, and Nizar Habash. 2019. The MADAR shared task on Arabic fine-grained dialect identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 199–207, Florence, Italy, August. Association for Computational Linguistics.

Michael Conover, Jacob Ratkiewicz, Matthew R Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. 2011. Political polarization on twitter. *ICWSM*, 133:89–96.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 29–30. International World Wide Web Conferences Steering Committee.

AbdelRahim Elmadany, Hamdy Mubarak, and Walid Magdy. 2018. Arsas : An arabic speech-act and sentiment corpus of tweets.

Sabit Hassan, Younes Samih, Hamdy Mubarak, Ahmed Abdelali, Ammar Rashed, and Shammur Absar Chowdhury. 2020. ALT submission for OSACT shared task on offensive language detection. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 61–65, Marseille, France, May. European Language Resource Association.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Ping Liu, Wen Li, and Liang Zou. 2019. Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *SemEval@NAACL-HLT*.

Hamdy Mubarak and Kareem Darwish. 2019. Arabic offensive language classification on twitter. In *International Conference on Social Informatics*, pages 269–276. Springer.

Hamdy Mubarak, Kareem Darwish, and Walid Magdy. 2017. Abusive language detection on Arabic social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 52–56, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Hamdy Mubarak, Ammar Rashed, Kareem Darwish, Younes Samih, and Ahmed Abdelali. 2020. Arabic offensive language on twitter: Analysis and experiments. *arXiv preprint arXiv:2004.02192*.

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

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, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

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