LING 573 Project Report: D3

Levon Haroutunian*, **Yi-Chien Lin***, **Bridget Tyree***, **David Yi***
Department of Linguistics
University of Washington
{levonh, yichlin, btyree, davidyi6}@uw.edu

Abstract

This project considers the task of offensive language detection. We use OLID, the data set from OffensEval 2019 (Zampieri et al., 2019b), to train an initial model to classify English tweets as offensive or non-offensive. In our next steps, we plan to adapt our model to classify multilingual tweets and address class imbalance.

1 Introduction

This project focuses on detecting offensive speech in social media. The detection of offensive or abusive language has been a rich area of research in NLP, which is overviewed in Waseem et al. (2017). Automated abuse detection can be a helpful tool for content moderation, though it should be noted that such a tool is only one component of a system that can adequately address and prevent toxic behavior (Jurgens et al., 2019).

For our initial model, we used static GloVe word embeddings (Pennington et al., 2014) trained on Twitter data[1] to form a representation for each tweet, and feed those representations into a logistic regression classifier. This model achieves a Macro F1 score of around 0.598.

For our second model, we use the same word embeddings, but feed them into a BiLSTM classifier. We also make significant changes to our preprocessing methods including handling negation and emojis. We experiment with many different combinations of potential improvements, and our current model achieves a Macro F1 score of roughly 0.732.

The rest of this paper is organized as follows. Section 2 details the task description, including both the primary task of detecting offensive language in English tweets and the secondary task of detecting offensive language in other languages. Sections 3 and 4 provide a system overview and describe our approach for both of our models. Section 5 provides the results from our second model, which are analyzed in Section 6. Our analysis includes planned directions for improvements to future iterations of this project. Finally, Section 7 provides a conclusion.

2 Task Description

The tasks in this project are based on OffensEval 2019[2] (Zampieri et al., 2019b) and OffensEval 2020[3] (Zampieri et al., 2020), both of which were SemEval shared tasks. Description regarding the primary task and adaptation task are presented respectively in the following subsections.

2.1 Primary Task

For our primary task, we tackle the subtask A of OffensEval 2019 – identifying offensive language in English (i.e. determining whether a given statement is offensive). More specifically, the affect type of this task is interpersonal stance and the target is sentiment. The identification of offensive language is obtained by implementing binary text classification on tweets.

We use the same training and test sets as the ones used in the subtask A of OffensEval 2019: the Offensive Language Identification Dataset

* Equal contribution, sorted in alphabetical order.

[1] https://nlp.stanford.edu/projects/glove/

[2] https://sites.google.com/site/offensevalsharedtask/offenseval2019

[3] https://sites.google.com/site/offensevalsharedtask/results-and-paper-submission

(OLID)[4] (Zampieri et al., 2019a). OLID is annotated tweets using crowdsourcing based on a three-layer annotation scheme. This annotation scheme enables developers to use the same dataset for different subtasks.

Our training data for the primary task contains 14,100 English tweets, in which 4,640 (33%) are annotated as offensive (OFF) and 9,460 (67%) are labeled as not offensive (NOT). The provided test set contains 240 OFF tweets and 620 NOT tweets. For evaluation, we compute the Macro F1 score, which is the official evaluation metric of OffensEval 2019 (Zampieri et al., 2019b).

2.2 Adaptation Task

For our adaptation task, we plan to extend our system to classify tweets in the four other languages included in OffensEval 2020: Arabic, Danish, Greek, and Turkish. These datasets are annotated and preprocessed using the same guidelines as OLID (Zampieri et al., 2020). These datasets have a range of sizes: there are 3,020 Danish tweets; 10,000 Arabic tweets; 10,287 Greek tweets; and 35,284 Turkish tweets (Zampieri et al., 2020) (See Table 4). The evaluation method is same as the one used for our primary task – calculating the Macro F1 scores of the classification results.

3 System Overview

This section describes our initial baseline (a logistic regression model) and our improved system (a BiLSTM).

3.1 Data Preprocessing

For our improved system, in order to reduce out of vocabulary (OOV) tokens, we implement preprocessing methods that tackle capitalization, punctuation, hashtags, emojis, and phrases containing negation. In addition, to address class imbalance, we implement resampling methods (i.e. under- and over-sampling) on our training dataset.

3.2 Logistic Regression

As a baseline system, we implement a linear classification model that utilizes static pre-trained word embeddings. First, we tokenize the raw tweets from the OLID dataset into fixed-length token sequences. Then, for each tweet, we create a feature vector by concatenating the vector representation of each token in the sequence while preserving the order. Finally, we feed the concatenated feature vector into a linear classifier to predict whether a tweet is offensive (OFF) or not offensive (NOT).

3.3 BiLSTM

As our improved system, we implement a bidirectional LSTM that takes as input a sequence of pre-trained embeddings and outputs a score representing the probability that the sequence belongs to the positive (OFF) class. This model initializes the word feature vectors with static pre-trained word embeddings and has four LSTM layers with a hidden dimension size of 500. There is an additional fully connected layer that performs binary classification over the final hidden state.

3.4 System Runtime

Table 1 shows the approximate training and prediction runtimes for our D3 Condor script. The training loop includes any necessary preprocessing and the prediction loop includes metric evaluation.

4 Approach

This section presents detailed descriptions of the approaches used in our improved system. First we present the preprocessing methods we considered for our improved system. Then, we provide a description of the classification methods we used for our baseline system and our improved system.

4.1 Data Preprocessing

In our improved system, in order to reduce OOV, address class imbalance, and increase system performance, we implement basic preprocessing, negation handling, resampling, and emoji handling methods. Details of each method are as follows.

4.1.1 Basic Preprocessing

Because tweets are extremely noisy and often contain typos, irregular text, URLs, emojis, and tags, we found simple preprocessing methods to be extremely beneficial. The four "basic" preprocessing methods we applied to the raw

[4] https://sites.google.com/site/offensevalsharedtask/olid

Section	Approximate Runtime (in minutes)
Training Loop (n=11917, epochs=100)	222
Predictions (n=1325)	28
Total Runtime	250

Table 1: System Runtimes for the end-to-end biLSTM pipeline

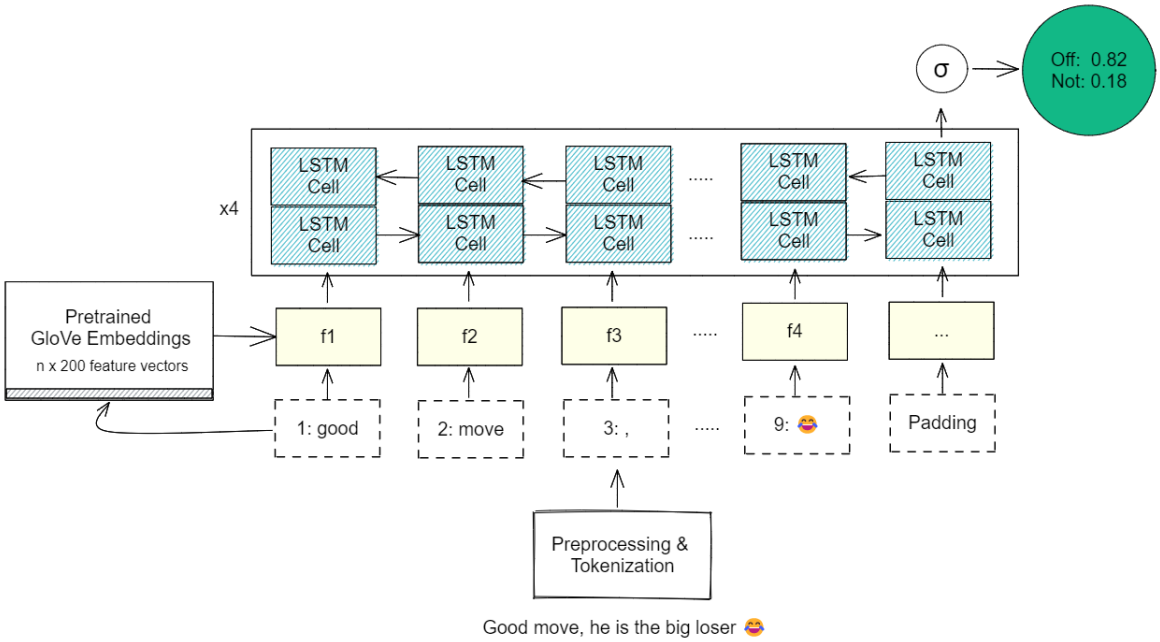

Figure 1: BiLSTM System Architecture

tweets include conversion to lowercase, splitting words from punctuation, removing apostrophes from contractions, and removing all hashtag symbols. These methods were chosen with the goal of minimizing the number of OOV tokens in any pretrained embeddings we used in our experiments. Hence, the following raw tweet would be converted in the following way:

1. By the way, I don't agree with your argument. #livid

2. by the way , i dont agree with your argument . livid

4.1.2 Negation Handling

Replacing negation is shown to be beneficial in improving classification accuracy (Jianqiang and Xiaolin, 2017). In our system, we adopt the negation handling method from Babanejad et al. (2020) which replaces negations and negated words with the antonyms of the negated words. More specifically, we focus on negated words that are adjectives and adverbs. For simplicity, a combination of negation and negated word is referred as ***negated phrase*** hereafter. For example, given a sentence *He is* ***not good***, we replace the negated phrase *not good* with the antonym of the negated word *good*. Therefore, the sentence *He is* ***not good*** would be converted into *He is* ***bad***.

To convert negation, we first identify negation words and the negated words with the spaCy library (Honnibal and Montani, 2017). After identifying the negated word, the word is disambiguated using the lesk algorithm (Lesk, 1986) in the nltk library (Bird et al., 2009). We then obtain the antonym of the disambiguated word from the WordNet corpus (Miller, 1995). If an antonym can be found for the negated word, we replace the negated phrase (e.g. *not good*) with the antonym (e.g. *bad*). On the other hand, if no antonym can be found for the negated word, the negated phrase in the sentence is left unchanged.

4.1.3 Emoji Handling

Since emojis can be helpful in sentiment analysis task (Liu et al., 2021), we decided to in-

clude emoji information in our raw data by converting the emojis into their corresponding text descriptions using spacymoji, a spaCy extension (Honnibal and Montani, 2017). For example, the sentence *Today is a nice day .* 😄 would be converted into *Today is a nice day . grinning face with smiling eyes*, where *grinning face with smiling eyes* is the text description of the emoji 😄.

4.1.4 Resampling

A class-imbalanced dataset can bias models that are trained on it. In our system, we implemented two resampling strategies – *oversampling* and *undersampling*. Oversampling refers to increasing sample size of minority class by duplicating samples in it. On the other hand, undersampling refers to decreasing the sample size of the majority class by selecting a subset of samples in it. The formula used for oversampling is $Nr_{min} = \alpha_o \times N_{maj}$, where Nr_{min} refers to the sample size of minority class after oversampling, α_o refers to the ratio for oversampling, and N_{maj} refers to the sample size of the majority class. For undersampling, we use the formula $Nr_{maj} = \alpha_u \times N_{min}$, where Nr_{maj} refers to the sample size of majority class after undersampling, α_u refers to the ratio for undersampling, and N_{min} refers to the sample size of the minority class. We ran experiments with $\alpha_o = [0.4, 0.6, 0.8, 1.0]$ for oversampling and $\alpha_u = [2.0, 1.5, 1.0]$ for undersampling.

4.2 Semantic Parsing

In an attempt to add semantic data to our classifier, we experimented with a post-processing technique using a tool called Retrofitting (Faruqui et al., 2015)[5]. We used this tool to retrofit the GloVe embeddings to include semantic data from PPDB (Ganitkevitch et al., 2013). Unfortunately, using this tool decreased our Macro F1 score on the BiLSTM to 0.68, so we did not include this in the final model. We did, however, add the code to run this and a README on how to run it into a folder in the repository titled "experiments" [6]. Surprisingly, though this tool decreased our Macro F1 score for the BiLSTM model, it slightly improved our Logistic Regression model's F1 score by roughly 0.015.

4.3 Classification Models

In this section, we present the classification models we used for our baseline and current systems. For our baseline system, we used a logistic regression classifier. For our current system, we decided to replace the linear classification model with a Bidirectional LSTM, which is a variable-length, sequence-based model. Detailed descriptions are as follows.

4.3.1 Logistic Regression

In our baseline system, we featurize tweets and use logistic regression for classification. For tweet featurization, we convert raw tweets into fixed-length token sequences using the text tokenizer from Keras[7] and obtain word vector representations from the pre-trained 200-dimensional Twitter embeddings from Glove (Pennington et al., 2014) but additionally leverage custom trained 200-dimensional emoji2vec (Eisner et al., 2016) embeddings for any OOV emojis. To address the varying scale and magnitude of the two embeddings, we normalize each vector in the combined embedding matrix. Then, we form a feature vector for each tweet by horizontally stacking the embedding vector representation for each token in the tweet to form a 10000-dimensional vector. For the final representation of each tweet, we choose to concatenate the token vectors with the goal of preserving sequential properties from the original tweet.

After tweet featurization, we classify the tweets with a logistic regression classifier using the scikit-learn library (Pedregosa et al., 2011) with L2 regularization. To avoid our model being biased by the class imbalance data, we set the `class_weight` parameter to be `balanced`. Lastly, we select the limited-memory BFGS as the algorithm for the optimization problem.

4.3.2 BiLSTM

Our Bidirectional LSTM model, which we built using the PyTorch library[8], is composed of four LSTM layers with a hidden embedding size of 500 and an additional fully-connected layer that performs the final classification. We use

[5] https://github.com/mfaruqui/retrofitting

[6] https://github.com/kvah/ling-573-group-repo/tree/main/experiments

[7] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

[8] https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

a learning rate of 6×10^{-5} and dropout of 0.5 on the initial embeddings. We use the Adam optimizer with an L2 penalty of 1×10^{-4} and train for 100 epochs, selecting the best model based on loss on the development set. These parameters were selected via manual hyperparameter tuning.

To tokenize each tweet, we use a similar scheme as the method described for the Logistic Regression model. The main contrast in the feature vectors, however, is that the model's embedding layers are *initialized* with the static GloVe embeddings but receive additional updates during training. We preserve only the words that appear in the training set, representing any out-of-vocabulary tokens in the development set as zero vectors.

5 Results

For the evaluation, since the OLID dataset does not contain a distinguished development dataset, we designated 10% of our training data to be our development dataset. Table 2 presents the F1-Macro of baseline models provided in Zampieri et al. (2019b) (**SVM, BiLSTM, CNN**) as well as the top scoring model from the task (**BERT-based-uncased default params**), our earlier model (**D2 MODEL**), our current system (**OUR MODEL**), and several models from our experiments that achieved notable results (denoted in **bold**). In addition, Table 2 also includes baseline Macro-F1 on all offensive tweets (**ALL OFF**) and all non-offensive tweets (**ALL NOT**) (Zampieri et al., 2019a). As shown in Table 2, our best model has a Macro-F1 score of 0.732 which is slightly lower than the baseline CNN, BiLSTM, and BERT-base-uncased default params models, but higher than other baseline models and our previous model.

Table 3 shows a confusion matrix of the results of running our model on the development dataset. Our model has a higher accuracy classifying non-offensive tweets than offensive ones. It correctly predicts offensive tweets roughly 68% of the time, and correctly predicts non-offensive tweets roughly 80% of the time. We attempted to fix this class imbalance by implementing oversampling and undersampling methods. Surprisingly, none of these methods improved our Macro F1 score. For future experiments, we hope to try other methods that handle class imbalance, such as SMOTE.

6 Discussion

In this section, we analyze the results described above. We first perform some error analysis, which guides our plans for next steps.

6.1 Error Analysis

Though our BiLSTM model performs considerably better than our initial model, we still have areas in which we could improve. We continue to have more issues identifying positive examples of hate speech. We had hoped that using resampling techniques would help this class imbalance, but unfortunately that was not the case. Looking through examples of OFF tweets that our model incorrectly identified as NOT tweets, it seems that our model can generally correctly identify most tweets with curse words, but tends to miss curse words which are less common or less commonly thought of as swear words. For example, our model misidentifies the following two tweets as non-offensive:

- @USER iNDEED her @ is danikaharrod go follow her because she is the cutest and shitposts from tdp sometimes jhdsfhjsg
- @USER ....and boobs . Do NOT forget the tatas . ❤️

It does, however, correctly identify this tweet as offensive:

- @USER SJSHSJ THATS MY JOB BITCH

Our system also tends to misidentify tweets that include more uncommon or misspelled slurs:

- @USER But the Frogs said it wasnt a Muzzie terriorist attack .

It also seems to often misidentify tweets that specifically refer to Antifa as a terrorist group as offensive:

- @USER @USER Antifa is a domestic terrorist group URL

Task B of SemEval 2019 (Zampieri et al., 2019b) concerns the classification of an offensive tweet as a targeted insult (TIN) or untargeted offensive language (UNT), i.e. profanity,

Method	Macro F1 Score
ALL OFF	0.220
ALL NOT	0.420
D2 Model: GloVe + Logistic Regression	0.598
Pre-processing (basic) + GloVe/emoji2vec + Logistic Regression	0.667
SVM	0.690
Pre-processing (basic) + GloVe + BiLSTM	0.730
D3 Model: Pre-processing (basic+negation) + GloVe + BiLSTM	**0.732**
BiLSTM	0.750
CNN	0.800
BERT-base-uncased default params	0.829

Table 2: Table showing our model's performance against the baseline models and models created by other teams for this task provided in (Zampieri et al., 2019b). Models from our experiments are denoted in **bold**.

Confusion Matrix		Predicted Label	
		negative	positive
True Label	negative	753	124
	positive	185	262

Table 3: Confusion Matrix based on the results from the D3 model

Language	Training	Test
English (OLID)	13,240	860
Arabic	8,000	2,000
Danish	2,961	329
Greek	8,743	1,544
Turkish	31,756	3,538

Table 4: Number of utterances by language in the OffensEval 2020 datasets (Zampieri et al., 2020) compared to OLID (Zampieri et al., 2019a).

so we were able to use the data from Task B to test this hypothesis directly. In the development set, the initial model achieved a recall of about 0.41 for untargeted offensive tweets and about 0.43 for targeted offensive tweets. Our new model now achieves a recall of about 0.57 for untargeted tweets offensive tweets and 0.59 for targeted offensive tweets. In future iterations, it may be helpful to ensure that our model can correctly identify less-common offensive terms.

6.2 Next Steps

For our next steps, we plan to adapt our system to classify offensive tweets in one or more of the languages included in the data provided for OffensEval 2020 (Zampieri et al., 2020): Arabic, Danish, Greek, and Turkish.

This comes with a number of challenges. First and foremost, our current pipeline includes a number of English-specific resources, including the pre-trained GloVe embeddings. To adapt our system, we plan to investigate 1) static embeddings for these additional languages, and 2) multilingual contextual embeddings, such as those generated by mBERT[9] (Devlin et al., 2019) or XLM-RoBERTa (Conneau et al., 2020). We will likely need to experiment with both monolingual and multilingual models to determine the optimal approach.

Secondly, there is a large imbalance in amount of available data across different languages, as shown in Table 4. At the extreme ends, Danish has the fewest training utterances at less than 3,000, and Turkish has the most at nearly 32,000. While the methods we tried here to handle imbalanced classes (undersampling and oversampling) did not prove to be beneficial for classification, they may be helpful for handling data that is imbalanced by language. Additionally, we are hopeful that a more sophisticated methods such as SMOTE will improve performance in classifying minority-class instances in future iterations.

7 Conclusion

This paper presents a model for offensive language detection in English. This model, which uses static word embeddings and a BiLSTM

[9] https://github.com/google-research/bert/blob/master/multilingual.md

classifier, attains a 0.732 Macro F1 score. We hope to improve upon this result by trying different oversampling methods to better solve our class imbalance problem. We also plan to adapt our model to classify multilingual tweets.

References

Nastaran Babanejad, Ameeta Agrawal, Aijun An, and Manos Papagelis. 2020. A comprehensive analysis of preprocessing for word representation learning in affective tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 5799–5810.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 48–54, Austin, TX, USA. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Zhao Jianqiang and Gui Xiaolin. 2017. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5:2870–2879.

David Jurgens, Libby Hemphill, and Eshwar Chandrasekharan. 2019. A just and comprehensive strategy for using NLP to address online abuse. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3658–3666, Florence, Italy. Association for Computational Linguistics.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.

Chuchu Liu, Fan Fang, Xu Lin, Tie Cai, Xu Tan, Jianguo Liu, and Xin Lu. 2021. Improving sentiment analysis accuracy with emoji embedding. *Journal of Safety Science and Resilience*, 2(4):246–252.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

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.

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). *arXiv preprint arXiv:2006.07235*.