Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 4309–4319
Marseille, 11–16 May 2020

Offensive Video Detection: Dataset and Baseline Results

Cleber de S. Alcântara, Diego Feijó, Viviane P. Moreira
Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS) – Porto Alegre – RS – Brazil
{cleber.alcantara,dvfeijo,viviane}@inf.ufrgs.br

Abstract

Web-users produce and publish high volumes of data of various types, such as text, images, and videos. The platforms try to restrain their users from publishing offensive content to keep a friendly and respectful environment and rely on moderators to filter the posts. However, this method is insufficient due to the high volume of publications. The identification of offensive material can be performed automatically using machine learning, which needs annotated datasets. Among the published datasets in this matter, the Portuguese language is underrepresented, and videos are little explored. We investigated the problem of offensive video detection by assembling and publishing a dataset of videos in Portuguese containing mostly textual features. We ran experiments using popular machine learning classifiers used in this domain and reported our findings, alongside multiple evaluation metrics. We found that using word embedding with Deep Learning classifiers achieved the best results on average. CNN architectures, Naive Bayes, and Random Forest ranked top among different experiments. Transfer Learning models outperformed Classic algorithms when processing video transcriptions, but scored lower using other feature sets. These findings can be used as a baseline for future works on this subject.

Keywords: offensive content, hate speech, dataset, video, Portuguese

1. Introduction

The wide adoption of social media platforms popularized the creation of user-generated content. Together with the democratization of content creation enabling users to express their ideas, came the dissemination of hate speech and other types of offensive material and behavior such as profanity, cyberbullying, and harassment. When users publish and disseminate offensive content, they are contributing to a hostile environment. This type of content might be harmful for users and discourage them from using the platforms. An unpleasant environment could also cause loss of revenue to the owners. Additionally, companies do not wish to be associated with this type of content, which could happen if their advertisements get displayed in an offensive video, for example. To tackle these problems, researchers from both companies and academia have proposed approaches aiming at detecting offensive content on different platforms (Schmidt and Wiegand, 2017).

On the Web, there is a variety of platforms that enable user content production in various formats, such as text, photo, audio, and video. So far, the text has been the most popular format used by people to do so, thanks to its input and storage simplicity, and the diffusion of comment sections supported by social networks. As a result, the vast majority of the existing works focused on identifying offensive content in text (social network posts, news comments, tweets, *etc.*). However, videos also play an essential role in the diffusion of content as they can reach a broad audience, including young children. Estimates say that 1 billion hours of videos are watched daily on YouTube alone[1]. To provide a safe environment for children and a healthy environment for users in general, detecting offensive videos becomes necessary.

Offensive content detection is usually addressed as a supervised learning task and, as such, demands training data. Whereas there is a growing number of datasets for textual content, datasets of videos are far less common. To address this gap, we assembled and made available `OffVidPT`, a dataset of videos annotated as to whether they present offensive content. We define as offensive, videos that express racism, sexism, homophobia, xenophobia, religious intolerance, or profane language. Also, we selected videos in Portuguese, which is an underrepresented language in terms of the availability of datasets.

The source of the videos used in our work was YouTube since it is the most widely used video-sharing platform on the Internet. YouTube has over two billion users, coming from more than 100 countries, with one billion hours watched daily[2]. The platform establishes policies regarding hateful content, harassment, and cyberbullying, and other sensitive topics[3]. To ensure the content being published in the platform complies with the policies and guidelines, YouTube has moderators working intensively to review videos flagged by users. However, YouTube also employs machine learning to analyze and flag videos for further review. However, due to the massive number of videos uploaded daily, it is hard to verify whether all videos comply with the established policies. Also, while YouTube is a vast platform, smaller platforms with less revenue might not be able to afford human labor to review videos published in their environment to protect their users. This scenario makes affordable and automated ways to detect offensive content desirable.

To provide baseline results, we experimented with a series of classification strategies and configurations. We tested a variety of classifiers, including Classic (Naive Bayes, Logistic Regression, SVM, C4.5, and Random Forest), Deep Learning (CNN and LSTM), and Transfer Learning (BERT and ALBERT) algorithms.

[1] `https://techjury.net/stats-about/youtube/`

[2] `https://www.youtube.com/intl/en/yt/about/press/`

[3] `https://www.youtube.com/intl/en/yt/about/policies/`

Our goal is to answer three research questions (RQ):

- RQ1: *Is it possible to accurately classify whether a video has offensive content just by analyzing its textual features?*
- RQ2: *Which features are the most helpful in detecting offensive content?*
- RQ3: *Which class of algorithms performs better at detecting offensive videos?*

The results of the experiments showed that textual features could be used for offensive video detection, but there is still room for improvement. Combining the predictions from the different sets of features and classifiers helped to improve the results in some cases. Yet, a more detailed analysis is necessary to investigate the impact of each feature in the ensemble. The results also showed that Deep Learning algorithms, especially CNN architectures, achieved the best performance in our domain. Also, n-gram provided better results than word embedding for Classic algorithms, but word embedding in combination to Deep Learning algorithm performed better. Furthermore, Transfer Learning models yielded accurate classification using just the video transcriptions, but they did not achieve the best result with other feature sets.

This work has two main contributions. The first one is the compilation of a dataset containing four textual and one statistical feature sets extracted from 400 videos in Portuguese from YouTube, which can be used for researchers to investigate the problem of offensive content detection. The second contribution is an analysis of offensive content detection using this dataset with Classic, Deep Learning, and Transfer Learning classifiers under different feature representations.

2. Related Work

Researchers have been engaged in detecting offensive content on the Web in the last few years (Anand et al., 2019; Pelle et al., 2018; Davidson et al., 2017). Such effort resulted in studies that aim to detect profanity (Fišer et al., 2017), harassment (Chatzakou et al., 2017; Kennedy et al., 2017) and cyberbullying (Chatzakou et al., 2017; Vigna et al., 2017). Hate Speech, which is a type of offensive content, was addressed in different ways. Some works dealt with Hate Speech detection more broadly (Davidson et al., 2017; Fišer et al., 2017; de Pelle and Moreira, 2017), but other researchers took a more fine-grained approach and worked on the identification of the subcategories of Hate Speech, such as religious intolerance (Pete and L., 2015), xenophobia (Bretschneider and Peters, 2017), and sexism/racism (Hasanuzzaman et al., 2017; Tulkens et al., 2016; Pete and L., 2015; Gambäck and Sikdar, 2017). As a result of the growing interest on the topic, dedicated workshops and evaluation campaigns were run, such as the 3rd Workshop on Workshop on Abusive Language Online[4], and HatEval[5] and OffensiveEval[6] for SemEval 2019.

Datasets were created from various sources such as news portals (de Pelle and Moreira, 2017; Nobata et al., 2016), Wikipedia (Wulczyn et al., 2017), Reddit (Kennedy et al., 2017), Facebook (Vigna et al., 2017) and Twitter (Davidson et al., 2017; Kennedy et al., 2017; Hasanuzzaman et al., 2017). YouTube was used as a data source too (Ducharme, 2017; Kandakatla, 2016). Similar to the other sources, which collected comments, articles, or posts, just textual content is usually collected from YouTube, not the videos themselves. Only a few works are devoted to analyzing videos. Gangwar et al. (2017) evaluated approaches for the detection of pornography in image and video using different datasets and Deep Learning models. Anand et al. (2019) proposed a framework to filter videos in English on YouTube with inappropriate content (insults, hate speech, promotion of extremism or terrorism) to prevent advertisements from using them. However, among the features approached, transcriptions are not used.

English is the most widely used language, which increases the availability of data in that language. Consequently, most of the studies on offensive language detection built or used English datasets. However, some researchers have been assembling and experimenting with datasets in other languages, such as German (Bretschneider and Peters, 2017), Italian (Vigna et al., 2017), and Portuguese (Fortuna et al., 2019; Pelle et al., 2018). Although most authors publish their datasets, others do not, which prevents the reproducibility of their work and comparison against other research.

The approaches used to address offensive content detection have evolved over the years from simple techniques (such as dirty-word lists) to the use of classic machine learning classifiers (such as Naive Bayes and SVM) (Davidson et al., 2017; de Pelle and Moreira, 2017), and Deep Learning (Vigna et al., 2017; Gambäck and Sikdar, 2017; Gao et al., 2017). Recently, Transfer Learning classifiers have also been employed (Basile et al., 2019a; Wu et al., 2019; Aggarwal et al., 2019).

Classifiers typically take features extracted from the text to perform classification. Many different features were used throughout the time in an attempt to improve performance, but the most common was n-gram of words and characters (Davidson et al., 2017; Vigna et al., 2017; Kennedy et al., 2017). Word embedding was adopted in many studies and achieved significant results (Vigna et al., 2017; Hasanuzzaman et al., 2017; Nobata et al., 2016). Some authors also used Sentiment Analysis to obtain sentiment polarity scores to be used as features. Others explored features extracted from the author of the text (Hasanuzzaman et al., 2017; Waseem and Hovy, 2016) (such as gender, location, age, and popularity).

In this work, we fill some of the identified gaps by assembling and sharing a dataset in Portuguese, a language that is typically underrepresented in terms of the availability of annotated training data for machine learning algorithms. Be-

[4] https://www.aclweb.org/portal/content/3rd-workshop-abusive-language-online

[5] https://competitions.codalab.org/competitions/19935

[6] https://competitions.codalab.org/competitions/20011

sides textual features such as title, description, and tags, we collected the video files and extracted the transcription from them to use as a feature. Additionally, we obtained multiple numerical and nominal features to study. Our goal was to provide a performance analysis of each feature set according to a wide range of algorithms and feature representations.

3. A Dataset for Offensive Video Detection

In this Section, we detail the process employed in the creation of the dataset for offensive video detection.

3.1. Data Collection

To retrieve video data from YouTube, we used its official API[7]. To search for potentially offensive videos, we used a list of dirty words provided by Pelle et al. (2018). Each seed word was searched individually, retrieving a set of video ids. We merged the video ids to avoid duplicates that might have appeared in more than one search and discarded the channel and user ids. This process resulted in a total of 101,759 video ids. Then, we retrieved detailed and structured data for each video through the same API.

We filtered the videos looking for the ones with default audio language attribute explicitly set to Portuguese, as one of our objectives is to build and provide a dataset of videos in this language. At the end of this filtering step, we ended up with a set of 5,180 videos.

To obtain the transcriptions, we used the Google Speech-to-Text[8] service, which makes use of machine learning to transcribe audio files automatically. This service has an option to filter profanity words and phrases, which we disabled to keep the transcription more loyal to the original audio and provide more realistic results during our experiments. We also used an unofficial YouTube API[9] to retrieve the subtitles for the videos. However, we found that most of the downloaded videos lack captions or have subtitles only in a language different from the one in the audio track. Due to this reason, we did not use the subtitles in our experiments.

3.2. Data Annotation

We chose to annotate a random sample of the videos which satisfied the following conditions: (i) do not include unclear speech, or no speech at all (just noise or sounds, without any spoken words), and (ii) be in Portuguese. The goal of (i) was to enable annotators to watch videos with better audio quality, and also yield higher quality transcriptions. The language filter (ii) was also necessary because the author of the video could have set the default audio language attribute incorrectly, or the video could have mixed languages, affecting the transcription quality to Portuguese. We also filtered out the videos with a duration longer than five minutes from the sample, which corresponded to approximately 43.3% of all Portuguese videos. The goal was to keep annotators engaged and save time in the annotation process, which is the bottleneck in dataset creation.

We developed a crowdsourcing web tool to enable volunteers to annotate the videos. Each video was annotated by three judges, like in other studies (de Pelle and Moreira, 2017; Davidson et al., 2017). Annotators were asked to watch the full video and tag offensive moments during the video. If nothing offensive was found, the annotator should explicitly specify the video was not offensive to carry on with the annotation process. Instructions were presented to the annotators alongside with the definition of what should be considered offensive in the videos and their definition according to the Online Oxford Dictionary[10]: racism, as *"Prejudice, discrimination, or antagonism directed against a person or people on the basis of their membership of a particular racial or ethnic group, typically one that is a minority or marginalized."*; sexism, as *"Prejudice, stereotyping, or discrimination, typically against women, on the basis of sex."*; homophobia, as *"Dislike of or prejudice against homosexual people."*; xenophobia, as *"Dislike of or prejudice against people from other countries."*; religious intolerance, as *"Unwillingness to accept views, beliefs, or behavior that differ from one's religion."*; and profane language, as *"Blasphemous or obscene language."*. These definitions were presented to guide the annotators in the process and prevent them from letting their personal beliefs or emotions affect their judgment, as they were not experts in the domain. These guidelines were presented to each annotator right before they started to evaluate the videos and were available at any time in the annotation page.

The tool was also used to show the general and user-specific annotation progress as an attempt to engage them in the annotation process. Since we believe this tool could help other researchers creating their datasets, we made its source code available[11].

By the end of the process, we had 400 videos annotated. For classification purposes, we considered a video as offensive if it had *at least one moment tagged as offensive by at least two annotators*. We chose this minimum agreement among the annotators to reduce the bias in the annotations, as they were crowdsourced, and the volunteers were not experts in the domain. Based on their agreement, we created two datasets:

- `OffVidPT-2`, in which at least two of the three annotators of each video agreed on the positive class (offensive); and
- `OffVidPT-3`, in which all three annotators agreed on the positive class.

We made these datasets available[12] (Alcântara et al., 2019). However, due to YouTube API Services Developer Policies[13], the video contents cannot be published. Therefore, in addition to the video id and labels assigned by each annotator, we included the following sets of information (feature sets), described in detail in the next Section: description, tags, title, transcription, and statistic.

[7] `https://developers.google.com/youtube/v3`
[8] `https://cloud.google.com/speech-to-text/`
[9] `https://www.youtube.com/api/timedtext`
[10] `https://www.lexico.com/`
[11] `https://gitlab.com/cleber.93cd/video-hate-detector`
[12] `http://www.inf.ufrgs.br/~csalcantara/offensive-video-detection/datasets/`
[13] `https://developers.google.com/youtube/terms/developer-policies`

3.3. Feature Sets

We extracted four sets of textual information and a collection of information containing statistics for each video from the data retrieved from YouTube. Each one of these sets is described below and referred to as a feature set in our work.

- **Description** (`desc`): this text is provided by the author of the video, with an average of 763 characters. It contains all sorts of characters and is not always present for every video.
- **Tags** (`tags`): this text is provided by the author of the video, usually short (∼186 characters), composed of words or sets of words separated by a dot (defining each video tag), and not always present.
- **Title** (`titl`): this text is provided by the author of the video, usually short (∼51 characters), and composed of all sorts of characters. Unlike the other feature sets, every video has a title.
- **Transcription** (`tran`): this text is obtained from the transcription of the video, which was generated automatically using Google Speech-to-Text. Transcriptions are usually long (∼1724 characters) and exist for every video.
- **Statistic** (`stat`): this is a snapshot of the statistics from the video at the moment of the dataset collection, which includes: *like counter*, *dislike counter*, *comment counter*, *view counter*, and *favorite counter*. We also included three additional features: *presence of offensive word* (binary feature representing whether the title or description had one of the offensive words provided in Pelle et al. (2018)), *video duration* (in seconds), and *video category* – a nominal information composed by the category identifier associated to the video, such as 10 (Music) and 25 (News and Politics).

3.4. Dataset Statistics

To assess the degree of agreement among the annotators, we calculated the Fleiss Kappa (Fleiss, 1971; Landis and Koch, 1977) in our dataset. This statistical score is used in the case where each instance was evaluated using discrete labels (nominal scale) by the same number of people. However, this score does not require annotators of one instance to be the same as for the others, which is the exact scenario of the annotation employed in our work. When we calculated the Fleiss Kappa for `OffVidPT-2`, we found a score of 0.512, which is considered a moderate agreement. Although not the greatest, this score is within the range found in dataset annotation of related works. Safi Samghabadi et al. (2017) reported 0.45, Warner and Hirschberg (2012) found 0.63, and de Pelle and Moreira (2017) achieved 0.71. Since `OffVidPT-3` is composed of instances for which all the annotators agreed, it did not make sense to calculate its Fleiss Kappa score.

In `OffVidPT-2`, 235 videos (out of 400) were classified as offensive, which corresponds to 58.8% of the total. As for `OffVidPT-3`, there were 156 videos considered offensive, corresponding to 39.0% of the total. Although not completely balanced, the distribution in the datasets shows a fair balance. It is typical for works involving data annotation to end up with unbalanced datasets with a more significant disproportion than ours. Chatzakou et al. (2017), for example, created a dataset with four classes for Twitter users and got the following proportion: 3.4% instances labeled as aggressors, 4.5% as bullies, 31.8% as spammers and 60.3% as none of them. Waseem and Hovy (2016) annotated a dataset of tweets for Hate Speech detection and obtained 11.7% of the instances labeled as racist, 20.0% as sexist, 68.3% as neither racist or sexist. Although some techniques can be applied to balance datasets, these works left their datasets unbalanced to provide a better match to the scenario found in the real world. Based on previous work and the distribution of our dataset, we decided to leave our datasets with their original balance.

4. Identifying Offensive Videos

The goal of this study is to address the problem of offensive video detection. In this Section, we describe our methodology, presenting the features, algorithms, and metrics used in our study.

4.1. Features

We used all feature sets introduced in Section 3.3 of both datasets (`OffVidPT-2` and `OffVidPT-3`). The format of the statistic feature set did not require additional processing before its use by the Classic classifiers. However, the textual feature sets needed to be pre-processed before the experiments, since the text was noisy and not standardized. We pre-processed them using the same script developed by Hartmann et al. (2017) to pre-process their corpus and train word embeddings. The original code discarded short sentences, but we changed it, so every sentence was kept. Additionally, due to the nature of our textual feature sets, we added new commands to the script to remove line breaks, symbols, and emojis. This processing turned the textual feature sets noise-free and standardized.

The instances at this point were ready to be used in the Transfer Learning models, as they internally create the feature representations for plain text. However, the instances required further processing to be used in the Classic and Deep Learning models, as they need the documents to have a standard format with the same dimension (size). We applied two different types of processing to come up with the following representations, our final features: n-gram and word embedding.

4.1.1. n-grams

To generate the features for n-gram, we performed punctuation, number, and stop word removal, aiming at a more uniform set of tokens. For the stop words, we used the Portuguese list provided by the Natural Language Toolkit (NLTK)[14]. Then, we generated n-grams using two different types of tokens: words and characters. For word n-grams, we created two representations: word n-grams with only one word (`unigram`), and word n-grams with n set from one to three (`wngram`). For character n-grams, we generated one representation with n varying from two to five characters (`cngram`). Table 1 counts the features created

[14] `http://www.nltk.org/`

for each textual feature set and representation. The statistic feature set, which is not textual, contains eight features, described previously in Section 3.3.

Feature Set	unigram	wngram	cngram
desc	7,558	47,785	76,062
tags	3,324	17,047	41,055
titl	1,421	5,072	19,022
tran	10,554	117,259	87,839

Table 1: Number of features in each n-gram representation for the textual feature sets

4.1.2. Word Embeddings

We used the trained word embeddings for Portuguese published by Hartmann et al. (2017), namely `Word2Vec`, `FastText`, `Wang2Vec`, and `GloVe`. All the embeddings were trained using the CBOW variant, except `GloVe`, which does not have this setting. We used the embeddings with 300 dimensions, as they presented a good balance between quality and efficiency. The same input used for n-gram extraction was used to generate the word embeddings. The only difference was the removal of out-of-vocabulary (OOV) words, *i.e.,* we discarded tokens not found in the trained embeddings. To generate embeddings for all instances of the textual feature sets and enable the correct processing of our models, we replaced missing descriptions with transcriptions and missing tags with titles from each video that had them missing. We chose this criterion because these feature sets instances have similar lengths. We generated two sets of word embeddings for each instance, depicted in Figure 1 and described as follows:

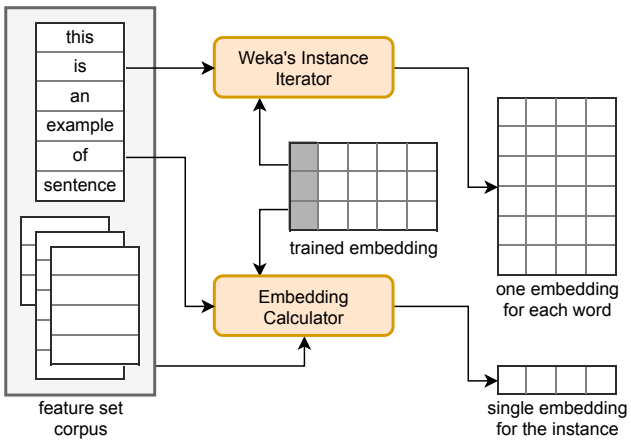
- **Single embedding**: This set was composed of a single embedding for each instance and had 300 dimensions – the same as the trained word embeddings. Our Embedding Calculator (Figure 1) performed the embedding calculation. For each instance, we took the weighted average of their unigram feature vectors. The weights were given by the *term frequency-inverse document frequency* (TF-IDF) for each word in the feature set corpus of the instance. This embedding set was used by all Classic algorithms and some Deep Learning classifiers (M-CNN and M-LSTM).

- **One embedding for each word**: This set was composed of multiple word embeddings for each instance. The generation of this set was performed by an instance iterator provided by Weka (Witten et al., 2016), which did a lookup in the trained word embedding to provide the embedding for each word in the instance. Thus, the number of words gives the size of each embedding set in each document. This variation was used by some Deep Learning algorithms (W-CNN and W-LSTM) to process all tokens in each instance.

4.2. Classification Algorithms

Data classification attempts to learn the relationship between a set of feature variables and a target variable of interest (Aggarwal, 2014). In our study, we selected the most used algorithms in related works for data classification. We grouped them into three different categories to provide a better understanding of their characteristics: Classic, Deep Learning, and Transfer Learning algorithms.

Figure 1: Generation process of embedding sets for use in the Classic and Deep Learning algorithms



4.2.1. Classic Algorithms

These algorithms are adopted widely in data classification. They use features defined beforehand to train and learn how the data should be represented and classified. We grouped these methods because they do not employ Deep Learning or Transfer Learning techniques, which are more recent and work differently.

In our experiments, we used five Classic algorithms. The first one is Naive Bayes, a probabilistic classification algorithm based on Bayes' theorem and is very fundamental among the classification methods. The second algorithm is Logistic Regression, which is also a probabilistic model and uses a sigmoid function to make the prediction interval range from zero to one instead of infinite continuous space, enabling the calculation and interpretation of the probabilities. The third algorithm is the Support Vector Machine (SVM). It takes a multidimensional vector space and attempts to find a hyperplane that best separates the instances of the vector space in the target variables. The fourth and fifth algorithms (C4.5 and Random Forest) are implementations of decision trees, which work by splitting the data in a hierarchical structure so that each path through the nodes leads to a leaf node defining the predicted class of the instance. C4.5 is the implementation of a single decision tree, and Random Forest is an ensemble method consisting of a collection of random decision trees where the final prediction comes from the majority of the votes cast by each tree.

4.2.2. Deep Learning Algorithms

Deep Learning models make use of neural networks with many layers and units (neurons) to explore the data and extract features to be used in the learning process. In other words, while a Classic method works with the features provided beforehand only, a Deep Learning model uses all of them to generate many more features itself, which is done

by using internal (hidden) layers of the model. Two types of deep neural networks have been widely used for text classification and are described next: Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

CNNs were created and applied initially in computer vision for image processing (Lecun et al., 1998). Later, Kim (2014) proposed a CNN model for sentence classification. Instead of convolving over pixels, the convolutional layer processes word embeddings extracted from the text using a trained embedding. In our study, we defined two CNN architectures, both inspired by Kim (2014). The first one, which we named as W-CNN, has multiple convolution layers, each one responsible for processing a different number of embeddings at a time per document. However, differently from Kim (2014), we used the number of rows in the kernel varying from one to three to keep the same range used for word n-gram in our study. Next, the max-pooling layer reduces the dimension of the output of the convolution layers and concatenates them to feed the fully connected layer that follows. This layer processes the data by applying a dropout to prevent overfitting and sends the output to the final layer, which calculates and outputs the predictions. We named the second CNN as M-CNN, and it uses the single embedding covered in Section 4.1.2 as input. To process the embedding, we use a single convolution layer, which does not slide over the embedding space but uses the entire vector to train 300 filters instead. The remaining network layers are the same as in the W-CNN, as the goal was to compare the two alternatives.

RNNs are designed to handle data with sequential information, such as text, by using state variables to store prior knowledge and use it to calculate the output data. Long Short-Term Memory model is a type of RNN and was introduced by Hochreiter and Schmidhuber (1997). Similarly to the CNN approach, we developed two LSTM architectures, which were inspired by Gao et al. (2017). For our first implementation, named W-LSTM, the memory cells process each word embedding extracted from the sentences in an LSTM layer. The output of this processing is used by the last network layer to calculate and output the predictions. The second LSTM architecture, which we refer to as M-LSTM, uses a single embedding per document as input, like the M-CNN. The other layers remained unchanged. For both architectures, we used the `GravesLSTM` implementation provided by Weka for the LSTM layer, which implements the vanilla LSTM model presented in Greff et al. (2017).

4.2.3. Transfer Learning

This machine learning category works by reusing the knowledge gained in models trained for one domain into a different, though related, one (Weiss et al., 2016). Transfer Learning is especially useful when little training data is available. Among recent models published in this category, we selected BERT (Devlin et al., 2018) and ALBERT (Lan et al., 2019) to run our experiments. Instead of using available English or Multilingual trained versions of these models, we trained BERT and ALBERT ourselves using the Wikipedia corpus in Portuguese.

Bidirectional Encoder Representations from Transformers (BERT) is a framework designed to pre-train vector representations from plain text in an unsupervised manner. The pre-trained model can be fine-tuned with just one additional output, adapting to several natural language tasks. This model is trained with two objectives: Masked Language Model and Next Sentence Prediction. In the first one, some random tokens are masked, and the model is trained to predict them. In the second one, the model must predict if one sentence follows the other.

A Lite BERT (ALBERT) is an improved version of the BERT architecture that was able to reduce its size significantly. While the standard BERT Base model has 110M parameters, the standard ALBERT base model has only 12M. This reduction was made possible by using cross-layer parameter sharing and factorized embedding parametrization. With these changes, the model can be trained significantly faster, enabling even larger models to be created.

4.3. Evaluation Metrics

We calculated the following metrics to evaluate our results:

- **Kappa** (`KPP`): the relative improvement of the current predictor on the random predictor.
- **True Positive Rate** (`TPR`): the rate of positive instances that were classified correctly as such.
- **Weighted Precision** (`PRE`): the weighted average precision of the positive and negative classes using the number of cases in each class as weights, where the precision is the percentage of the classified instances that do belong to that class.
- **Weighted Recall** (`REC`): the weighted recall average for the positive and negative classes using the number of instances on each class as weights, where the recall is the percentage of correctly classified instances among all instances from that class.
- **Weighted F1** (`F1`): the weighted harmonic mean between `PRE` and `REC`.
- **Area Under the ROC (Receiver Operating Characteristics) Curve** (`AUC`): the relationship between true positives and false positives, representing how well the model can distinguish each instance between the classes.

To keep in line with the existing research on offensive content detection (Chatzakou et al., 2017; Pavlopoulos et al., 2017; Nobata et al., 2016), we decided to use `AUC` to elect the best result achieved by the combination of algorithm and feature representation. Still, other evaluation metrics can be informative. Also, the weights in the weighted metrics are given by the number of instances in the classes.

4.4. Experimental Procedure

We used two different tools (environments) in our study: Weka for the Classic and Deep Learning models, and Google Colab[15] for the Transfer Learning models.

[15] `https://colab.research.google.com/`

Naive Bayes, Logistic Regression, SVM, C4.5, and Random Forest are implemented in the following classifiers on Weka, respectively: `NaiveBayes`, `SimpleLogistic`, `SMO`, `J48`, and `RandomForest`. Weka does not have the Deep Learning algorithms built-in. However, there is a package named WekaDeeplearning4j (Lang et al., 2019) that can be installed through Weka's package manager to fill this gap. This package expands Weka's original set of classifiers with Deep Learning algorithms based on Deeplearning4j[16], enabling the user to define their architecture with the different algorithms and layers available.

While the statistic feature set was submitted only to the Classic classifiers, the textual features were submitted to all the classifiers covered in Section 4.2. Classic algorithms used both n-gram and word embedding, Deep Learning algorithms used only word embedding, and Transfer Learning classifiers used only the textual features sets after pre-processing. This proceeding was performed for both `OffVidPT-2` and `OffVidPT-3`. The combination of these features, classifiers, and datasets amounted to a total of 434 experimental runs, ensembles excluded.

Additionally, we created independent ensemble classifiers (Rokach, 2010) in an attempt to outperform the results of the other classifiers used in isolation, keeping our study aligned with other works with the same approach (Pelle et al., 2018; Pete and L., 2015). Our ensembles were created by combining the best result obtained for each feature set and algorithm category according to the `AUC`. We used the classifier and feature representation of the best result to generate the predictions of the instances in the training data for each feature set. These predictions were combined to create the ensemble representation. Thus, there are five features in each ensemble for Classic algorithms and four for Deep Learning. Each ensemble is identified by the association of a feature representation and a classifier category: Classic n-gram ensemble, Classic word embedding ensemble, and Deep Learning word embedding ensemble. These three ensembles were created for each dataset and submitted to all the five Classic algorithms and two Deep Learning models (M-CNN and M-LSTM), adding 42 experiment runs to our study, amounting to 476 in total.

We used tenfold cross-validation in our experiments and averaged the results of the iterations to get to a final score and selected the runs with the best score for `AUC` for each feature set. Therefore, as the measures reported are the result of the average of the real values obtained for each fold, metrics that rely on others (*e.g.,* `F1`) might not assume the same value that they would get using their formulas.

5. Results and Discussion

In this Section, we present the results achieved in our experiments and answer our research questions. Tables 2 and 3 present the best results for the experiments using the feature sets for `OffVidPT-2` and `OffVidPT-3` datasets, respectively, grouped by the type of learning algorithm. They also include the best results for the ensemble representations. The best result for each metric is highlighted.

Intuitively, we were expecting the results to be higher in `OffVidPT-3`, as it had a full annotator agreement. However, our results showed that scores were very similar. The larger number of instances in the positive class presented in `OffVidPT-2` seemed to provide more evidence for the learning models to identify such cases and thus compensate for smaller agreement in the annotations.

Is it possible to accurately classify whether a video has offensive content just by analyzing its textual features?

The best scores achieved in our experiments were 0.78 in `AUC` in three experiments: Deep Learning word embedding ensemble for `OffVidPT-2`, and Classic n-gram ensemble and Deep Learning using the tags feature set alone for `OffVidPT-3`. For `F1`, the Classic n-gram ensemble performed best, achieving 0.74 (Table 3). In an analogous binary classification of offensive content on texts, the organizers of OffensEval-2019 (subtask A) reported the best scores in Hate Speech detection to be around 0.83 in F1 (Zampieri et al., 2019). On a similar task, the best results on HatEval (subtask A) (Basile et al., 2019b) were considerably lower for English (0.65) and slightly better in Spanish (0.76). Although the results we report here cannot be compared directly to any of those SemEval tasks, their scores give us an indication of the expected classification quality on a similar domain with the same number of classes. In this sense, our results are within the range achieved in HatEval. This finding may indicate that, while there is still room for improvement, textual features can be used for offensive video detection.

Which set of features is the most helpful in detecting offensive content?

Overall, looking at the best results for each individual set of features, we find `AUC` scores ranging between 0.70 and 0.77 in most cases. When ensembles are used to combine the predictions of all feature sets, we notice a slight improvement for Classic n-gram and Deep Learning word embedding ensembles results for `OffVidPT-2`. On the other hand, the Classic word embedding ensemble result showed a slight decrease. For the `OffVidPT-3`, the `AUC` improved by 3% for the Classic n-gram ensemble and had a small reduction for the other ensembles. However, when we analyze the kappa, precision, recall, and F1 metrics, the ensembles generally increase their scores, which might be desirable in some situations.

Looking at feature representation in Tables 2 and 3, we observed that n-gram performs slightly better than word embedding for all feature sets and metrics, except for the descriptions feature set (`AUC` not included in `OffVidPT-2`). However, word embedding was better with Deep Learning algorithms, outperforming all results for its use in Classic algorithms for `OffVidPT-2` and most of the results for their use with Classic algorithms for `OffVidPT-3`. GloVe was the best word embedding representation for both datasets for most textual feature sets, followed by Wang2Vec, FastText, and Word2Vec. For n-gram representations, character n-gram and word unigram were the most helpful for the classification, while word n-gram (`wngram`) did not achieve the best result for any textual feature set.

The statistic feature set did not score close to the best re-

[16] `https://deeplearning4j.org/`

	Feature		Best		KPP	TPR	AUC	PRE	REC	F1
	Repres.	**Set**	**Repres.**	**Classifier**						
Classic	-	stat	-	R. Forest	0.26	0.52	0.68	0.64	0.65	0.64
	n-gram	desc	cngram	R. Forest	0.19	0.36	0.72	0.63	0.63	0.60
		tags	unigram	N. Bayes	0.33	**0.82**	0.74	0.70	0.65	0.65
		titl	cngram	N. Bayes	0.29	0.74	0.74	0.67	0.64	0.64
		tran	unigram	R. Forest	0.31	0.61	0.73	0.67	0.67	0.66
		all	ensemble	N. Bayes	0.35	0.59	0.75	0.69	0.69	0.68
	word embedding	desc	Wang2Vec	R. Forest	0.26	0.45	0.71	0.65	0.66	0.64
		tags	Wang2Vec	R. Forest	0.23	0.40	0.71	0.64	0.65	0.62
		titl	Wang2Vec	R. Forest	0.19	0.38	0.72	0.62	0.63	0.61
		tran	FastText	R. Forest	0.20	0.37	0.67	0.62	0.63	0.61
		all	ensemble	N. Bayes	0.29	0.52	0.70	0.66	0.66	0.66
Deep L.	word embedding	desc	GloVe	M-LSTM	0.35	0.60	0.74	0.69	0.69	0.68
		tags	GloVe	W-CNN	0.37	0.55	0.77	0.71	0.71	0.70
		titl	Wang2Vec	W-CNN	0.31	0.60	0.75	0.67	0.66	0.66
		tran	Wang2Vec	W-CNN	0.31	0.49	0.77	0.68	0.68	0.67
		all	ensemble	M-LSTM	**0.43**	0.69	**0.78**	**0.73**	0.72	0.72
Transfer L.		desc		BERT	0.24	0.76	0.70	0.67	0.76	0.71
		tags		BERT	0.23	0.80	0.69	0.67	**0.80**	**0.73**
		titl		ALBERT	0.34	0.73	0.74	**0.73**	0.73	**0.73**
		tran		BERT	0.32	0.77	0.76	0.71	0.77	**0.73**

Table 2: Best results by algorithm category, feature representation, and feature set for OffVidPT-2

	Feature		Best		KPP	TPR	AUC	PRE	REC	F1
	Repres.	**Set**	**Repres.**	**Classifier**						
Classic	–	stat	–	R. Forest	0.30	0.77	0.71	0.67	0.67	0.67
	n-gram	desc	unigram	N. Bayes	0.24	0.79	0.67	0.65	0.65	0.64
		tags	cngram	R. Forest	0.26	**0.93**	0.73	0.70	0.69	0.64
		titl	unigram	N. Bayes	0.35	0.82	0.75	0.70	0.70	0.69
		tran	cngram	L. Regre.	0.38	0.87	0.72	0.72	0.72	0.70
		all	ensemble	M-CNN	**0.46**	0.85	**0.78**	**0.76**	**0.75**	**0.74**
	word embedding	desc	Wang2Vec	R. Forest	0.25	0.92	0.71	0.70	0.68	0.64
		tags	GloVe	R. Forest	0.20	0.91	0.73	0.68	0.66	0.61
		titl	FastText	R. Forest	0.23	0.88	0.69	0.67	0.67	0.63
		tran	GloVe	R. Forest	0.23	0.91	0.69	0.69	0.67	0.63
		all	ensemble	N. Bayes	0.35	0.86	0.71	0.72	0.71	0.69
Deep L.	word embedding	desc	GloVe	W-CNN	0.29	0.85	0.74	0.69	0.69	0.66
		tags	Word2Vec	W-CNN	0.37	0.84	**0.78**	0.71	0.71	0.70
		titl	GloVe	W-CNN	0.34	0.85	0.75	0.70	0.70	0.69
		tran	GloVe	W-CNN	0.31	0.88	0.75	0.70	0.70	0.67
		all	ensemble	M-CNN	0.31	0.84	0.75	0.69	0.69	0.68
Transfer L.		desc		BERT	0.30	0.40	0.71	0.63	0.50	0.54
		tags		BERT	0.34	0.49	0.71	0.70	0.49	0.56
		titl		BERT	0.32	0.52	0.70	0.62	0.55	0.57
		tran		ALBERT	0.37	0.52	0.76	0.67	0.52	0.58
		tran		BERT	0.35	0.57	0.76	0.63	0.57	0.58

Table 3: Best results by algorithm category, feature representation, and feature set for OffVidPT-3

sults, but it still outperformed some results achieved by other feature sets, mainly in Table 3. The results for this feature set for OffVidPT-3 were slightly better than the ones for OffVidPT-2.

The ensemble representations improved the results only in some cases. The combination of all feature sets provided a slight increase of AUC for Classic n-gram and Deep Learning word embedding ensembles for OffVidPT-2 (1%),

and Classic n-gram ensemble for OffVidPT−3 (3%). For the other ensembles, the AUC decreased slightly.

Which class of algorithms performs better at detecting offensive videos?

Overall, Deep Learning models and some ensemble learning had the best results, with the two highest AUC achieved using the M-CNN classifier OffVidPT−3: 0.78 for Classic n-gram ensemble and Deep Learning using tags (Table 3). This score was also reached by the M-LSTM classifier when processing the Deep Learning word embedding ensemble for OffVidPT−2. The Transfer Learning algorithms outperformed Classic algorithms when handling the transcription feature set. However, the Classic algorithms seemed to be able to learn better from the description, tags, and title feature sets than the Transfer Learning algorithms. The W-CNN classifier achieved almost all of the best results when using the textual feature sets separately, as can be seen in Tables 2 and 3. The only exception is that M-LSTM performed better than W-CNN when processing the descriptions feature set for OffVidPT−2. This performance shows that, although the core idea of CNN is to be applied for image processing, it can outperform other Classic and Deep Learning algorithms when used for NLP.

The W-LSTM classifier did not score the best result for any of the experiments. The M-LSTM, on the other hand, scored well for the ensemble experiments. For every ensemble representation, although we can not report all the results, the M-LSTM model scored among the best ones. We observed this same behavior for M-CNN in the ensemble experiments, alongside with Naive Bayes.

The Random Forest classifier outperformed Naive Bayes in most of the experiments with Classic algorithms. For the statistic feature set, for example, Random Forest achieved the best results for both datasets. However, although Naive Bayes scored second in the textual feature sets experiments, it produced many of the best results in the ensemble experiments, outperforming Random Forest in this case. The Logistic Regression classifier was the best for transcriptions using cngram features for OffVidPT−3 (Table 3) but did not outperform Random Forests and Naive Bayes for the other experiments. The C4.5 and SVM algorithms, on the other hand, did not score as good as the other ones in our experiments.

When comparing the Transfer Learning classifiers, BERT outperformed most of the results achieved by ALBERT. Both classifiers achieved the same AUC and F1 for transcriptions in the OffVidPT−3 (Table 3), but ALBERT achieved better precision and kappa. In comparison to the Classic and Deep Learning classifiers, the Transfer Learning models scored some of the best precision, recall, and F1 results for OffVidPT−2. For the OffVidPT−3, on the other hand, BERT and ALBERT were not able to achieve any of the best results, scoring poorly for precision, recall, and F1. The only exception, in this case, is for AUC, where BERT and ALBERT scored close to the best results.

Limitations. Since the creation of datasets for classification is a supervised task, we relied on human and their bias. Providing guidelines and definitions for annotators is helpful and extremely important. However, people still might judge instances using their beliefs or feelings, affecting the quality of their annotations and the dataset in general. Learning algorithms, especially Deep Learning models, expect a large volume of data to be able to extract features and improve their performance. The number of instances in our datasets probably prevented achieving better results. Also, algorithms used in our experiments had their parameters set to their default values, except in a few cases to get a working classifier. Performing fine-tuning could have contributed to better results.

6. Conclusion

In this work, we investigated the problem of detecting offensive videos. Our goal was to analyze how Classic (Naive Bayes, Logistic Regression, SVM, C4.5, and Random Forest), Deep Learning (CNN and LSTM), and Transfer Learning (BERT and ALBERT) algorithms would perform using different representations of mostly textual features extracted from videos. Additionally, we created ensemble-based classifiers in an attempt to improve our results, which were ranked by their AUC but also had other metrics reported for analysis.

We collected and annotated a dataset of 400 videos in Portuguese from YouTube, which we published. The data includes one statistical feature set and four textual feature sets (description, tags, title, and transcription). For the textual ones, we generated n-gram and word embedding so they could be submitted to the Classic and Deep Learning models. The Transfer Learning models used the pre-processed plain text. The predictions of the best results were used as input to the ensemble-based classifiers.

Overall, our best result was 0.78 for AUC, achieved in three cases, and 0.74 for F1, obtained by the M-CNN model using the Classic n-gram ensemble. These results are in the range of results achieved in competitions of offensive content identification with binary classification. This finding means that, although there is still room for improvement, textual features can be used to identify offensive content on the Web.

When compared against word embedding, n-gram achieved better results with Classic algorithms using character n-gram and word unigram. However, word embedding demonstrated to be more helpful in Deep Learning than in Classic algorithms. Additionally, we found GloVe and Wang2Vec to be the best-trained embeddings to most of our textual feature sets. Our results show that the helpfulness of each feature set varies according to the algorithm used. The ensemble-based experiments added little improvement to the best results using feature sets in isolation.

Future work can explore the extraction of new features and a combination of the existing ones in our data. The dataset itself can be expanded to include more instances and other sources. Also, different classifiers and architectures can be tested. As we did minimal parameter tuning, new combinations of values for the parameters of the classifier can also be explored in an attempt to improve our results.

Acknowledgments. This work was partially supported by CNPq/Brazil and by CAPES Finance Code 001.

7. Bibliographical References

Aggarwal, P., Horsmann, T., Wojatzki, M., and Zesch, T. (2019). LTL-UDE at semeval-2019 task 6: BERT and two-vote classification for categorizing offensiveness. In May et al. (May et al., 2019), pages 678–682.

Aggarwal, C. C. (2014). *Data classification: algorithms and applications*. CRC press.

Anand, V., Shukla, R., Gupta, A., and Kumar, A. (2019). Customized video filtering on youtube.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Pardo, F. M. R., Rosso, P., and Sanguinetti, M. (2019a). Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In May et al. (May et al., 2019), pages 54–63.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F. M., Rosso, P., and Sanguinetti, M. (2019b). SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

Bretschneider, U. and Peters, R. (2017). Detecting offensive statements towards foreigners in social media. In *50th Hawaii International Conference on System Sciences, HICSS 2017*, pages 2213–2222, Hilton Waikoloa Village, Hawaii, USA. AIS Electronic Library (AISeL).

Chatzakou, D., Kourtellis, N., Blackburn, J., De Cristofaro, E., Stringhini, G., and Vakali, A. (2017). Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 13–22, New York, NY, USA. ACM.

Davidson, T., Warmsley, D., Macy, M. W., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017*, pages 512–515, Montréal, Québec, Canada. AAAI Press.

de Pelle, R. P. and Moreira, V. P. (2017). Offensive comments in the brazilian web: a dataset and baseline results. In *Brazilian Workshop on Social Network Analysis and Mining (BraSNAM) in conjunction witth Congresso da Sociedade Brasileira de Computação-CSBC*, pages 510–519, São Paulo, SP, Brazil. Sociedade Brasileira de Computação.

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

Ducharme, D. N. (2017). *Machine Learning for the Automated Identification of Cyberbullying and Cyberharassment*. Ph.D. thesis, University of Rhode Island.

Fišer, D., Erjavec, T., and Ljubešić, N. (2017). Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in slovene. In *Proceedings of the First Workshop on Abusive Language Online*, pages 46–51, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Fortuna, P., da Silva, J. R., Wanner, L., Nunes, S., et al. (2019). A hierarchically-labeled portuguese hate speech dataset. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 94–104.

Gambäck, B. and Sikdar, U. K. (2017). Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Gangwar, A., Fidalgo, E., Alegre, E., and González-Castro, V. (2017). Pornography and child sexual abuse detection in image and video: A comparative evaluation. In *8th International Conference on Imaging for Crime Detection and Prevention (ICDP 2017)*, pages 37–42, Dec.

Gao, L., Kuppersmith, A., and Huang, R. (2017). Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 774–782, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct.

Hartmann, N., Fonseca, E. R., Shulby, C., Treviso, M. V., Rodrigues, J. S., and Aluísio, S. M. (2017). Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In Gustavo Henrique Paetzold et al., editors, *Proceedings of the 11th Brazilian Symposium in Information and Human Language Technology, STIL 2017, Uberlândia, Brazil, October 2-5, 2017*, pages 122–131. Sociedade Brasileira de Computação.

Hasanuzzaman, M., Dias, G., and Way, A. (2017). Demographic word embeddings for racism detection on twitter. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 926–936, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Kandakatla, R. (2016). Identifying offensive videos on youtube. Master's thesis, Wright State University.

Kennedy, G., McCollough, A., Dixon, E., Bastidas, A., Ryan, J., Loo, C., and Sahay, S. (2017). Technology solutions to combat online harassment. In *Proceedings of the First Workshop on Abusive Language Online*, pages 73–77, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In Alessandro Moschitti, et al., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Lang, S., Bravo-Marquez, F., Beckham, C., Hall, M., and Frank, E. (2019). Wekadeeplearning4j: A deep learning package for weka based on deeplearning4j. *Knowledge-Based Systems*, 178:48 – 50.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov.

Jonathan May, et al., editors. (2019). *Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*. Association for Computational Linguistics.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. (2016). Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Pavlopoulos, J., Malakasiotis, P., and Androutsopoulos, I. (2017). Deep learning for user comment moderation. *CoRR*, abs/1705.09993:1–11.

Pelle, R., Alcântara, C., and Moreira, V. P. (2018). A classifier ensemble for offensive text detection. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, WebMedia '18, pages 237–243, New York, NY, USA. ACM.

Pete, B. and L., W. M. (2015). Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39, Feb.

Safi Samghabadi, N., Maharjan, S., Sprague, A., Diaz-Sprague, R., and Solorio, T. (2017). Detecting nastiness in social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 63–72, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Schmidt, A. and Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain, April. Association for Computational Linguistics.

Tulkens, S., Hilte, L., Lodewyckx, E., Verhoeven, B., and Daelemans, W. (2016). The automated detection of racist discourse in dutch social media. *Computational Linguistics in the Netherlands Journal*, 6:3–20, 12/2016.

Vigna, F. D., Cimino, A., Dell'Orletta, F., Petrocchi, M., and Tesconi, M. (2017). Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, pages 86–95, Venice, Italy. CEUR-WS.org.

Warner, W. and Hirschberg, J. (2012). Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Waseem, Z. and Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June. Association for Computational Linguistics.

Weiss, K. R., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *J. Big Data*, 3:9.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wu, Z., Zheng, H., Wang, J., Su, W., and Fong, J. (2019). BNU-HKBU UIC NLP team 2 at semeval-2019 task 6: Detecting offensive language using BERT model. In May et al. (May et al., 2019), pages 551–555.

Wulczyn, E., Thain, N., and Dixon, L. (2017). Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1391–1399, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019). 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.

8. Language Resource References

Alcântara, Cleber de S. and Diego Feijó and Viviane P. Moreira. (2019). *Offensive Videos in Portuguese*. 1.0, ISLRN 529-322-484-169-1.