UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CLEBER DE SOUZA ALCÂNTARA

A Study on Offensive Video Detection

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof^{a}. Dr^{a}. Viviane Moreira

Porto Alegre
May 2020

CIP — CATALOGING-IN-PUBLICATION

Alcântara, Cleber de Souza

A Study on Offensive Video Detection / Cleber de Souza Alcântara. – Porto Alegre: PPGC da UFRGS, 2020.

84 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Viviane Moreira.

1. Offensive content. 2. Hate speech. 3. Dataset. 4. Classification. 5. Machine learning. 6. Youtube. 7. Video. I. Moreira, Viviane. II. Título.

“If I have seen farther than others,
it is because I stood on the shoulders of giants.”
— Sir Isaac Newton

ACKNOWLEDGMENTS

I would like to thank my master's advisor Prof^{a}. Dr^{a}. Viviane Moreira. Her guidance and experience were fundamental to indicate the right direction in my studies and find solutions to the problems encountered in my way. Her motivation and patience helped me go through difficult moments. I am sincerely grateful to her for giving me this opportunity and letting me learn so much through these years of study.

I also would like to thank my research group friends for supporting me, providing insights, help, and feedback on my research. I am also grateful to them for the good time through these years in and out of the laboratory. My research and time in Porto Alegre would not have been the same without them.

Not least, I would like to especially thank my mom Luciene and my dad Edvaldo, who never stopped supporting my decisions, motivating me to go through this research, and believing in me. I am also thankful for the support of my relatives, old friends, and new friends I made after moving to Porto Alegre.

Additionally, I would like to thank the Universidade Federal do Rio Grande do Sul and the Instituto de Informática for providing the environment and equipment to develop my research. I would like to thank CNPq/Brazil and CAPES as well, for financially supporting my research.

ABSTRACT

Web users around the world produce and publish high volumes of data of various types, such as text, images, and videos. To keep a friendly and respectful environment, the platforms in which this content is published usually restrain users from publishing offensive content 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 automatically performed using machine learning, but it needs an annotated dataset. Although there are datasets for offensive text detection available, there are no such datasets for videos. Also, most of the published datasets process English data, leaving Portuguese and other languages underrepresented. In this work, we investigate the problem of offensive video detection. We assemble, describe, and publish a dataset of videos in Portuguese. Also, we run experiments using popular machine learning classifiers used in offensive language detection and report our findings, alongside multiple evaluation metrics. In the results, we found that word embedding provided better results when used with Deep Learning classifiers, but n-gram performed better than word embedding for Classic algorithms. Random Forest and Naive Bayes classifiers presented the best performance across most of the features when compared to the other Classic algorithms. The W-CNN architecture employed in our study presented the best results for most of the feature sets using Deep Learning algorithms. For Transfer Learning models, BERT was the best classifier for most of the feature sets. Also, for the ensemble experiments, Naive Bayes, Random Forest, M-CNN, and M-LSTM achieved the best results for the experiments with all features and the ones using feature ablation. Using ensemble improved the results for some categories of algorithms and feature representation. Also, feature ablation experiments helped to identify the contribution of each feature in the ensemble results, improving the results in some cases. Overall, Deep Learning algorithms scored the best results, followed by Classic and Transfer Learning algorithms.

Keywords: Offensive content. hate speech. dataset. classification. machine learning. youtube. video.

Um Estudo sobre Detecção de Vídeo Ofensivo

RESUMO

Usuários da Web em todo o mundo produzem e publicam grandes volumes de dados de vários tipos, como texto, imagens e vídeos. Para manter um ambiente amigável e respeitoso, as plataformas nas quais esse conteúdo é publicado geralmente impedem os usuários de publicar conteúdo ofensivo e contam com moderadores para filtrar as postagens. No entanto, esse método é insuficiente devido ao alto volume de publicações. A identificação de conteúdo ofensivo pode ser realizada automaticamente usando aprendizado de máquina, mas precisa de um conjunto de dados anotado. Embora existam conjuntos de dados disponíveis para detecção de texto ofensivo, não existem conjuntos de dados para vídeos. Além disso, a maioria dos conjuntos de dados publicados processa dados em inglês, deixando português e outras linguagens com pouca representatividade. Neste trabalho, investigamos o problema da detecção de vídeo ofensivo. Nós montamos, descrevemos e publicamos um conjunto de dados de vídeos em português. Além disso, realizamos experimentos usando classificadores populares de aprendizado de máquina usados na detecção de linguagem ofensiva e relatamos nossas descobertas, juntamente com várias métricas de avaliação. Nos resultados, descobrimos que *word embedding* forneceram resultados melhores quando utilizado com *Deep Learning*, mas n-gram foi melhor do que *word embedding* para algoritmos Clássicos. Os classificadores *Random Forest* e *Naive Bayes* apresentaram o melhor desempenho na maioria dos atributos quando comparados aos outros classificadores Clássicos. A arquirtetura W-CNN utilizada no nosso estudo apresentou os melhores resultados para a maioria dos conjuntos de atributos utilizando *Deep Learning*. Para modelos de *Transfer Learning*, BERT foi o melhor classificador para a maioria dos conjuntos de atributos. Além disso, para os experimentos com ensemble, *Naive Bayes*, *Random Forest*, M-CNN and M-LSTM conseguiram os melhores resultados para experimentos com todos os atributos e aqueles utilizando remoção de atributos. Utilizar ensemble melhorou os resultados de alguns grupos de algoritmos e representações de atributos. Adicionalmente, experimentos de remoção de atributos ajudaram a identificar a contribuição de cada atributo nos resultados de ensembles, melhorando os resultados em alguns casos. Em geral, algoritmos de *Deep Learning* conseguiram os melhores resultados, seguidos por algoritmos Clássicos e de *Transfer Learning*.

Palavras-chave: conteúdo ofensivo, discurso de ódio, conjunto de dados, classificação, aprendizado de máquina, youtube, vídeo.

LIST OF ABBREVIATIONS AND ACRONYMS

ALBERT	A Lite BERT
API	Application Programming Interface
AUC	Area Under the ROC Curve
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Network
F1	Weighted F-measure
ISLRN	International Standard Language Resource Number
KPP	Kappa
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
OOV	Out-of-Vocabulary
POS	Part-of-Speech
PRE	Weighted Precision
REC	Weighted Recall
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
RQ	Research Question
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TPR	True Positive Rate
WAV	Waveform Audio File Format

LIST OF FIGURES

LIST OF TABLES

CONTENTS

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; WIEGAND, 2017).

On the Web, there is a variety of platforms that enable user content production in different formats, such as text, image, audio, and video. So far, 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 detecting 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.

Detecting offensive content 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. The language of the videos is Portuguese, which is underrepresented 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, com-

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

ing 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 24 hours a day, seven days a week reviewing 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.

In order to explore the solutions to offensive video detection, 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.

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 show that textual features can be used for offensive video detection, but there is still room for improvement. Combining the predictions from the different sets of features and classifiers helps to improve the results. Still, a finer analysis is necessary to investigate the impact of each feature in the ensemble. In some cases, the performance can be improved by removing one feature from the group. As an example, the ensemble built using the results of Classic algorithms using n-gram achieved the best `AUC` in our experiments by removing the descriptions feature set from its feature list. The results also show 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 algorithms yielded

[2]<https://www.youtube.com/intl/en/yt/about/press/>
[3]<https://www.youtube.com/intl/en/yt/about/policies/>

accurate classification using just the transcriptions of the videos, 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.

As a byproduct of this thesis, we had a paper accepted at the 12th Language Resources and Evaluation Conference (rated as Qualis A1) (ALCANTARA; FEIJO; MOREIRA, 2020). The paper is a summarized version of this document containing the main ideas and results. Additionally, our dataset was accepted and published under the International Standard Language Resource Number[4] (ISLRN) 529-322-484-169-1.

The next chapters present the details of our work. Chapter 2 covers the background concepts that served as the basis for this work, such as feature representations, classification algorithms, and evaluation metrics. Chapter 3 surveys related work to our study, describing approaches proposed by other researchers. Chapter 4 provides information on the dataset we built and used in our work. Chapter 5 introduces our methodology for offensive video detection. In Chapter 6, we present the experimental setting and discuss the results achieved by our experiments, including their limitations. Finally, Chapter 7 summarizes our goals and approach, highlights our best findings, and discusses future work.

[4]<http://www.islrn.org/>

2 BACKGROUND

In this chapter, we present concepts used across our work that are fundamental to understand it. These concepts include terminology for offensive content, feature representations, classification algorithms, and evaluation metrics.

2.1 Terminology

In our work, use the term *offensive content* to refer to any material that offends people, dependently or independently of their personal or shared characteristics. There are many works with variant definitions for quite similar problems.

An example of these variants used across some works is *profanity*, which can be understood as the type of language employed by people through profane, vulgar, malicious, or inappropriate terms. These terms sometimes might not be addressed to other people and become offensive, but they might make the environment where it is published inappropriate for some public (XIANG et al., 2012). Works such as Fišer, Erjavec and Ljubešić (2017), Xiang et al. (2012), Sood, Antin and Churchill (2012a), Sood, Antin and Churchill (2012b), Su et al. (2017), among others, address the identification of profanity. Similarly, the term *abusive language* is employed by Nobata et al. (2016) to refer to the language used by users that can be harmful to others.

Harassment can be understood as the use of content against other people with the intention of being offensive to them, but without recurrence (YIN et al., 2009). Ducharme (2017) uses the term *cyberharassment*, as it happens on the Internet. Users harassing others might make use of a profane vocabulary to do so, such as insults. Bretschneider, Wöhner and Peters (2014), Kennedy et al. (2017), Pelle and Moreira (2017), Chatzakou et al. (2017) are examples of works employed on detection of harassment in the Web. Similarly to harassment, *cyberbullying* is the employment of offensive language against people in the Web, but in this case, it tends to happen repeatedly against the victim (REYNOLDS; KONTOSTATHIS; EDWARDS, 2011). Detection of cyberbullying was studied by Chatzakou et al. (2017), Vigna et al. (2017), Ross et al. (2016), and many others.

Hate Speech is the type of offensive content employed against specific people, motivated by and targeting the victim's characteristics, such as religion, gender, and skin color. Some works dealt with Hate Speech detection more broadly (DAVIDSON et al., 2017; FIŠER; ERJAVEC; LJUBEŠIĆ, 2017; PELLE; MOREIRA, 2017), but other researchers took a more fine-grained approach and worked on the detection of the subcategories of Hate Speech, such as religious intolerance (PETE; L., 2015), xenophobia (BRETSCHNEIDER; PETERS, 2017), and sexism/racism (HASANUZZAMAN; DIAS; WAY, 2017; TULKENS et al., 2016; PETE; L., 2015; GAMBäCK; 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 Language Online[1], and HatEval[2] and OffensiveEval[3] for SemEval 2019.

2.2 Feature Representations

Natural language cannot be understood by computers in the same way humans understand it. Thus, humans had to come up with strategies to enable computers to understand and process natural language. This need gave origin to a field named Natural Language Processing (NLP) and has been very important to contribute to the progress in related areas such as machine learning. Computers can be trained to understand natural language by looking into large volumes of textual data to identify relations between its elements and extract information from the text. This ability is fundamental in text classification tasks, as the computer needs to extract features from textual documents to perform the prediction. Different representations have been used for text to enable better processing by classifiers, with the most prominent being n-gram and word embedding.

2.2.1 n-grams

On text processing, an n-gram is used to split textual information into sequences of n tokens. The text becomes a composition of multiple tokens, which might be present in different documents. This way, the presence of a token in the text can be used as an indicator to determine its relationship with the class to be predicted, like whether it is offensive or not. Two popular types of n-gram for text use *words* or *characters* as tokens. So, for example, if we take the sentence $The\ book\ is\ on\ the\ table$, we could create word n-grams of size three and have the tri-grams: $The\ book\ is$, $book\ is\ on$, $is\ on\ the$, and

[1]<https://www.aclweb.org/portal/content/3rd-workshop-abusive-language-online>
[2]<https://competitions.codalab.org/competitions/19935>
[3]<https://competitions.codalab.org/competitions/20011>

$on\ the\ table$. Alternatively, we could take the word $table$ and create character n-grams of size two and have the bi-grams: ta, ab, bl, and le.

2.2.2 Word Embeddings

Although useful to help computers understand the text by analyzing the presence of or absence of tokens in it, n-grams do not provide semantic information, which limits the advance of text processing. To solve this problem, researchers came up with the idea of embeddings: vector representations of text in a continuous space. These multidimensional vectors enable the computer to catch the semantic meaning and perform some operations like assessing the similarity between two words, for example.

To obtain these vectors, a large volume of textual data is necessary, because the more data for training, the more representative the vectors will be. We say this data is used to train the vectors to enable the adjustments of each of their indexes, also called weights. Those weights are what allows the computer to perform calculations with the vectors. However, the first techniques to create embeddings were not able to efficiently use the available hardware to process large volumes of data due to the computational cost necessary to do so. Later, researchers came up with new techniques and, aligned with the advance of the hardware industry, embeddings for NLP became popular. In the next paragraphs, we present Word2Vec (MIKOLOV et al., 2013) considered as the most influential embedding, and other techniques proposed later, such as FastText Bojanowski et al. (2017), Joulin et al. (2017), Wang2Vec (LING et al., 2015) and GloVe (PENNINGTON; SOCHER; MANNING, 2014).

Word2Vec. Mikolov et al. (2013) proposed new techniques for word vector training, contributing to improve the vectors and reduce the complexity of the training. Previous works could not perform well with large volumes of training data, which also ended up affecting the quality of their vectors. Mikolov et al. (2013) proposed two different architectures to approach the problem, both using neural networks (sets of algorithms used to perform learning tasks, similar to the human brain): Continuous Bag-of-Words (CBOW) and Skip-gram. They work similarly, but while the first architecture is used to predict words based on others, provided as context, the second one does the opposite, *i.e.,* it predicts the context words based on a given one. Table 2.1 shows some examples of relationships between pairs of terms presented by Word2Vec.

GloVe. GloVe stands for Global Vectors. It was proposed by Pennington, Socher

Table 2.1: Examples of semantic and syntactic word relationship provided by Word2Vec shown in Mikolov et al. (2013)

	Type of relationship	Word Pair 1		Word Pair 2	
Semantic	Common capital city	Athens	Greece	Oslo	Norway
	All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
	Currency	Angola	kwanza	Iran	rial
	City-in-state	Chicago	Illinois	Stockton	California
	Man-Woman	brother	sister	grandson	granddaughter
Syntactic	Adjective to adverb	apparent	apparently	rapid	rapidly
	Opposite	possibly	impossibly	ethical	unethical
	Comparative	great	greater	tough	tougher
	Superlative	easy	easiest	lucky	luckiest
	Present Participle	think	thinking	read	reading
	Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
	Past tense	walking	walked	swimming	swam
	Plural nouns	mouse	mice	dollar	dollars
	Plural verbs	work	works	speak	speaks

Source: Mikolov et al. (2013)

and Manning (2014) and combines methods based on predictions used in previous studies, like Word2Vec, with a novel approach based on statistical information related to word co-occurrences. This change enabled the model to obtain more semantic and syntactic information. The goal was to create more meaningful word vectors. Although related, this model does not provide CBOW and Skip-gram architecture variations.

Wang2Vec. Proposed by Ling et al. (2015), this model is a modification of Word2Vec created to additionally take into consideration the order of the words in the sentence. The changes were proposed in an attempt to gather more syntactic information about the words in the corpus, improving results on syntactic tasks. Changes were performed to both CBOW and Skip-gram to make them aware of the position of context words, originating the Continuous Window Model and Structured Skip-gram architectures, respectively. With those changes, the authors were able to improve state-of-the-art results for tasks such as part-of-speech tagging.

FastText. This model is more recent than the previous ones and was proposed by Bojanowski et al. (2017), Joulin et al. (2017). It attempts to create word embeddings that consider word morphology. To do so, they proposed an approach that relies on character n-gram. The first step is breaking the word in a bag of character n-grams. Then, embeddings are created for those n-grams. Finally, the word embedding is obtained by summing its character n-grams embeddings. This model is derived from the Skip-gram model used in Word2Vec.

Comparison of Word Embedding Methods. Hartmann et al. (2017) trained these word embeddings with multiple vector dimensions varying from 50 to 1000 using an extensive Portuguese corpus. They evaluated the performance of the embeddings intrinsically using syntactic and semantic analogies (word level) and extrinsically using Part-of-Speech (POS) tagging and sentence similarity tasks. For the intrinsic evaluation, FastText was the best embedding for syntactic analogies, which was expected due to its capacity to capture morphological features better. On the other hand, GloVe performed best for semantic similarities, probably thanks to its ability to better model semantic information. Wang2Vec ranked second in both tasks, which can probably be justified by its consideration of word order during training. For the extrinsic evaluation, Wang2Vec achieved the best results for POS tagging using 300 dimensions embeddings, and it is valid to say that usually, the bigger the number of dimensions, the better the result for all models . Word2Vec using CBOW was the best model for European Portuguese, while Wang2Vec Skip-Gram achieved the best results for Brazilian Portuguese, both using 1000 dimensions. By looking at their results, we can say that for all models, using 300 dimensions provides a significant enhancement when compared to lower dimension embeddings, although not too much improvement is achieved when compared to higher dimension embeddings.

2.3 Classification Algorithms

Data mining comprehends the extraction of useful knowledge from data, usually performed using large volumes of it (CHAKRABARTI et al., 2006), which has been possible thanks to the advances in technology in the last years. The different domains and types of problems being addressed enabled the definition of varying task classes in data mining. One of these classes is data classification, which attempts to learn the relationship between a set of feature variables and a target variable of interest (AGGARWAL, 2014). In other words, this means we use information about something like a tweet, a video, or an image, for example, to classify that entity using a target class, like offensiveness.

Data classification is usually divided into two phases. The first one is the training, in which we use data to train the computer to classify the data. The data used in this phase is called training data. The resulting product of this phase is a trained classification model. The quality of this model relies significantly on the quality and amount of data used for training. In the second phase, called testing, we use the trained model to classify a test instance. This classification can output two different types of results: a discrete value, like $offensive$ or $not\ offensive$, or a numerical score for each class, like 25% offensive and 75% not offensive. Numerical scores can be converted to discrete values according to predefined thresholds. For example, one could consider a video as offensive if the score is above 75% for the $offensive$ class.

The splitting of the data between training and testing can be done in different ways. A popular one is to use k-fold cross-validation. This method consists of splitting the data into k disjoint subsets, named *folds*, of the approximately same size. Then, the model is trained k times. For each time, $k - 1$ different folds are used to train the model. The remaining fold is used for testing. The final performance is given by averaging the performance obtained for each training (KOHAVI, 1995). Sometimes, the data is manually split into instance sets to train and test the model. As an example, this is used in classification competitions, such as SemEval-2018 Task 7 (GÁBOR et al., 2018) and SemEval-2019 Task 6 (ZAMPIERI et al., 2019), where only the training data is provided to the participants to train their models. The testing data is provided when it is time to evaluate the trained models and measure their performance.

Providing a labeled dataset to train and test classification models enables what we call supervised learning. Here, the models use the features provided for each instance to predict the target variable. On the other hand, the data is not labeled for unsupervised learning, requiring learning models to find patterns in the dataset using different strategies by looking for similarities among the instances. An example of unsupervised learning is data clustering. There is also semi-supervised learning, which is in between supervised and unsupervised learning. Here, models make use of both labeled and unlabelled data to classify the data (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).

As mentioned, the data used in the classification algorithms should be labeled, so we have a target variable to train and test the classification models. This labeling step is a crucial phase in the construction of the data to be used in the experiments, also named dataset, because it affects the quality of training and, consequently, testing. However, manually labeling the data requires a lot of effort, since the amount of data needed is usually large. This manual task also enables the risk of the data being affected by the annotator's bias. In an attempt to mitigate this problem, researchers usually provide clear instructions to the annotators on how to perform the labeling. Another measure adopted during the labeling process is to have each instance labeled by more than just one person.

This way, the researchers can establish thresholds for the agreement among the annotators and define the final label for their dataset.

In the next sections, we present the classification algorithms we used in this work. We grouped the algorithms into three different categories to provide a better understanding of their characteristics: Classic, Deep Learning, and Transfer Learning.

2.3.1 Classic Models

In this section, we present widely adopted models in data classification, which we refer to as Classic models. These models use features defined beforehand to train and learn how the data should be represented and classified. We group these methods because they do not employ Deep Learning or Transfer Learning techniques, which are more recent and work differently.

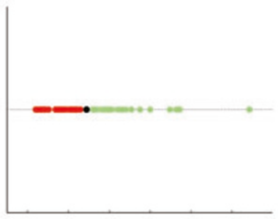
Naive Bayes. This is a probabilistic classification algorithm, very fundamental among the classification methods. Probabilistic models use statistical inference in the data to find the best target variable (class) of an instance (AGGARWAL, 2014). Naive Bayes is an algorithm based on Bayes' theorem, which is used to calculate conditional probabilities (JOYCE, 2019). This model is a generative model, which roughly means the likelihood to classify an instance as a particular target variable comes from the observation of the probabilities of both classes and features of the document (joint probability). Naive Bayes uses the Bayes formula with the assumption of independence of all feature variables given the value of the class of the instance to be classified. This assumption is why it is called naive, as this independence among the feature variables is a simplification. However, despite this naive assumption, this classifier performs surprisingly well in many applications (ZHANG, 2004).

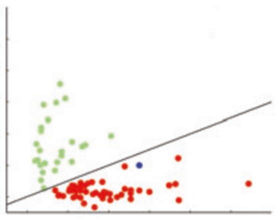
Logistic Regression. A logistic regression model, like Naive Bayes, is a probabilistic model. However, it is a discriminative model, which roughly means the probability to classify an instance as a particular class comes from the observation of its features (conditional probability). This model outputs a discrete value (usually binary) instead of a continuous outcome and fits the previously calculated probability into one of the classes by using likelihood. A logistic regression model is able to use discrete and continuous features to perform the prediction (AGGARWAL, 2014). This model is similar to a linear regression model. However, it uses a sigmoid function to make the prediction interval range from zero to one instead of an infinite continuous space, enabling the calculation and interpretation of the probabilities.

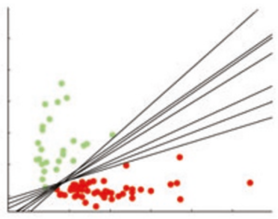
SVM. The Support Vector Machine (SVM) algorithm takes a multidimensional vector space and attempts to find a hyperplane that best separates the instances of the vector space in the target variables. Noble (2006) presents four concepts about SVM to help understand how it works. The first one is the *separating hyperplane*, which creates the sectors that will contain the vectors for each of the target variables, exemplified in Figure 2.1a and 2.1b. As there might be multiple possible separating hyperplanes for a dataset, the concept of the *maximum-margin hyperplane* is introduced. This concept states that the best hyperplane should be located in a position where the distance from each of its nearest vectors (margin) is the highest (maximum), as it maximizes the model's ability to classify new data correctly. As an example, Figure 2.1d illustrates the best hyperplane selected among possible ones displayed in Figure 2.1c. The third concept is the *soft margin* parameter. It defines how far away exception vectors might reside from the hyperplane without resulting in a misclassification, as they would be on the other side of the hyperplane. Finally, the last concept is the *kernel function*, which is a mathematical operation performed over the data to enable its projection in a space with a higher dimension. This operation is performed when it is not possible to define the hyperplane using the current vector space. Figure 2.1f shows an example of the application of a kernel function where each instance of the dataset in Figure 2.1e was squared.

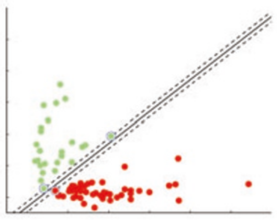
Decision Tree. A decision tree splits the data in a hierarchical structure so that each path through the nodes leads to a leaf node defining the predicted class of the instance. The data partition is done using a split criterion, which is a condition using one or more data features to determine the split logic. When just one feature is used, the split is named *univariate*. A *multivariate* split occurs if multiple features are combined to define the split condition. Both methods are exemplified in Figure 2.2a and 2.2b. The goal of decision trees is to recursively split the training data, as much as possible, to enable the best discrimination for the instance class. However, it is not possible to decide when to stop splitting the data to prevent the model from start overfitting. In other words, a decision tree could get too many levels and become excessively adapted to the training data, which could cause the model to perform very well for the training data but poorly for new data (testing data). Parameters are used to prune the tree to avoid this problem (AGGARWAL, 2014). Two decision tree methods were found to be widely employed: C4.5 (QUINLAN, 1993) and Random Forest (BREIMAN, 2001). The first one is the implementation of a single decision tree. Random Forest, on the other hand, is an ensemble method consist-

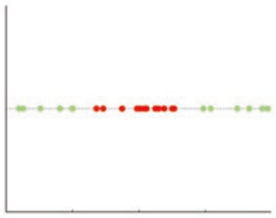
Figure 2.1: Examples of SVMs using data related to the prediction of two different types of leukemia. The blue dot in Figure 2.1b represents an unseen example to the model

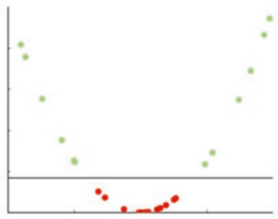

(a) An uni-dimensional hyperplane, represented by the black dot


(b) A bi-dimensional hyperplane, represented by the black line


(c) Some possible hyperplanes for current the vectors


(d) The maximum-margin hyperplane, selected from Figure 2.1c


(e) An uni-dimensional dataset on which a hyperplane can not be created


(f) The hyperplane obtained after applying a kernel function to Figure 2.1e

Source: Noble (2006)

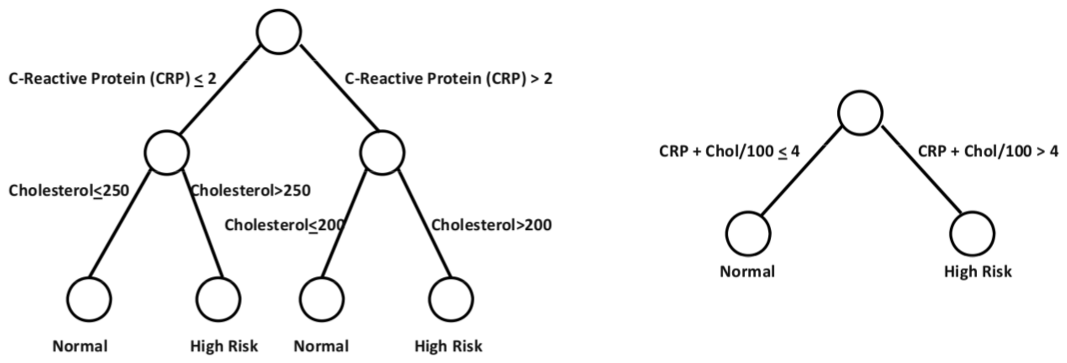
ing of a collection of random decision trees where the final prediction comes from the majority of the votes cast by each tree.

2.3.2 Deep Learning Models

Dee 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 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. The first layer is only responsible for inputting the data into the model without modification. Each hidden layer takes the output from the previous one as input, processes it, and outputs a set of features to be processed by the next layer. The last one, also known as the output layer, can be seen as a classification layer, as it is responsible for outputting the final prediction of the model.

The data flow among the units is usually unidirectional. However, depending on the architecture employed in the model, this flow might be bidirectional. Internally, each neuron receives the input data, calculates the net value, and then converts it to an output

Figure 2.2: Examples of decision trees used to predict the high risk of cardiovascular disease based on two features extracted from a data snapshot presented by Aggarwal (2014): C-Reactive Protein and Cholesterol



(a) Univariate Splits (b) Multivariate Splits

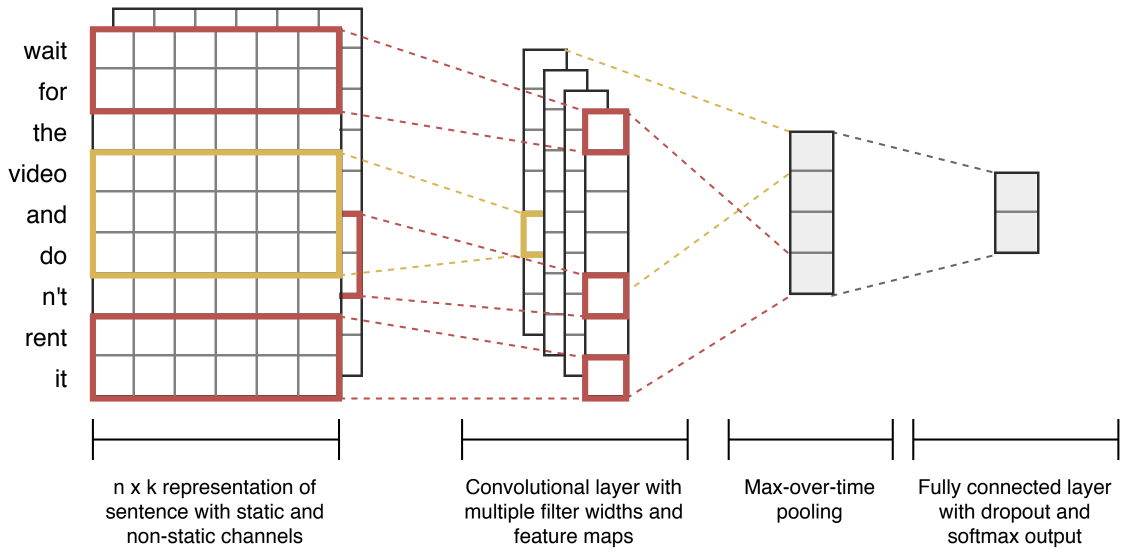
Source: Aggarwal (2014)

to be sent to the next inputs. The calculation of the net value is done using a net value function, which uses the unit parameters or weights. The weights are adjusted in the learning process. The calculation of the output is done using an activation function, which depends on the unit type (AGGARWAL, 2014).

While classic models do not require too much data to achieve substantial performance in some cases, Deep Learning models usually demand large volumes of data to be able to learn from them and perform well. However, processing large amounts of data, allied to running complex algorithms, was a problem in the past due to the limitation of the hardware power. Thankfully, technology has evolved in the last years, significantly increasing the processing and storage powers available and reducing the cost to consume them. Alongside with new techniques to handle large volumes of data, Deep Learning research and commercial solutions have appeared worldwide. Two types of deep neural networks have been widely used for text classification and will be described next: Convolutional and Recurrent Neural Networks.

CNN. Convolutional Neural Networks (CNNs) were created and applied initially in computer vision for image processing (Lecun et al., 1998). A CNN works by sliding (convolving) a specific sized window over an image and processing its pixels using filters, resulting in feature maps. These feature maps can have their dimensions reduced by a process called pooling to result in a single number to be processed by further layers in the network, or they can be used as inputs in a new convolutional layer and then be pooled. These convolutional and pooling layers can be combined many times. After the final pooling layer, many different architectures with diverse layers and classifiers can be employed to output the classification. Inspired by this architecture, researchers adapted CNNs to use them in NLP. Kim (2014) proposed a CNN model for sentence classification, depicted in Figure 2.3. Instead of convolving over pixels, the convolutional layer processes word embeddings extracted from the text using a trained embedding. The window size has a width equal to the embedding dimension and the height as three, four, and five. The network uses 100 filters for each window, then performs max-pooling in the feature maps, which means the maximum value in each feature map is selected. The max-pooled feature-vector is used in a fully connected layer to output the class prediction for the sentence.

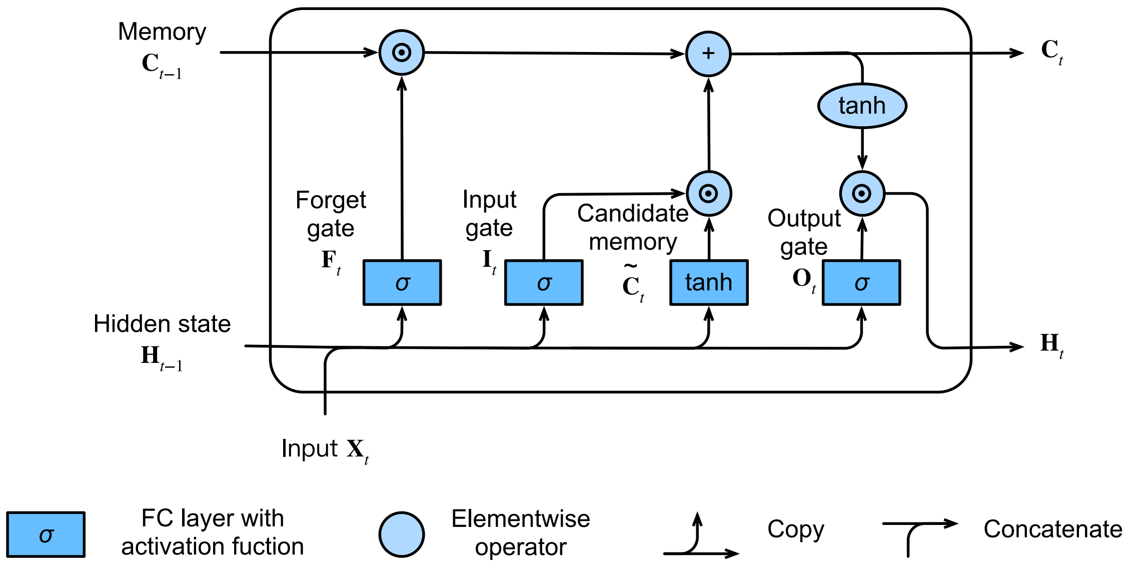
Figure 2.3: CNN architecture proposed by Kim (2014) for sentence classification


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Source: Kim (2014)

LSTM. The Long Short-Term Memory model was introduced by Hochreiter and Schmidhuber (1997). It is a type of Recurrent Neural Network (RNN), which is designed to handle data with sequential information, such as text. RNNs have state variables (hidden states) that enable storing prior knowledge and use it to calculate the output data. The hidden state from one iteration is used as one of the inputs for the next iteration. A unit in LSTM has three gates to control the input data: forget gate, input date, and output gate. The input data used by those gates are the previous hidden state and iteration input data. Additionally, a unit in LSTM has a candidate memory cell, which uses the same input as the gates and calculates the candidate memory. The function of the input gate is to determine how much new data will be added to the new memory. The forget gate is used to calculate how much data from the previous memory cell will be retained. The model combines the output from the forget gate, input gate, and candidate memory cell to produce the new memory. Finally, the output gate is used to generate the new hidden state of the unit. Thus, in the end, the neuron outputs the new memory and the new hidden state (LIN et al., 2019). Figure 2.4 illustrates this data flow in an LSTM unit.

Figure 2.4: The data flow in an LSTM unit to produce the new memory and hidden state



Source: Lin et al. (2019)

2.3.3 Transfer Learning Models

While the models presented so far require the training and testing data to be from the same domain, there are some domains in the real world for which it is not always possible to assemble a dataset big enough to create machine learning solutions for them. To approach this problem, researchers sought ways to reuse the knowledge gained in models trained for one domain into a different, though related, one. The solution found to achieve this result was called *Transfer Learning* (WEISS; KHOSHGOFTAAR; WANG, 2016). Thus, Transfer Learning is especially useful when little training data is available.

There are two types of Transfer Learning: feature-based and fine-tuning. The most notorious example of feature-based is using previously trained weights as the vector representation layer when building a task specific model. Usually, this embedding layer is kept frozen (or non-trainable) to avoid messing with the already trained weights. Fine-tuning is the process of building a generic model and then changing the output layers to address a specific task. In this case, the whole model is fine-tuned using a low learning rate for a few epochs to avoid damaging the previously learned representations. In the next paragraphs, we present two Transfer Learning models that have been used by many researchers to solve their NLP problems lately: BERT and ALBERT.

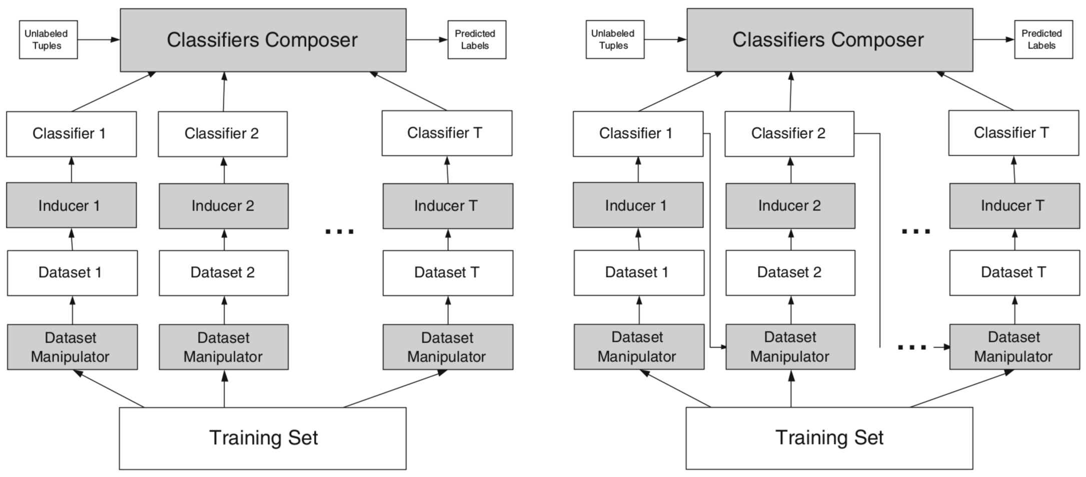
BERT. Bidirectional Encoder Representations from Transformers (BERT) (DEVLIN et al., 2019) is a framework designed to pre-train vector representations from plain text in an unsupervised manner. The pre-trained model can be fine-tuned with just one additional output, adapting to several natural language 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.

ALBERT. A Lite BERT (ALBERT) (LAN et al., 2019) 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.

2.3.4 Ensemble-based Model

The goal of this model, which is also referred to as meta-classifier or meta-learner, is to combine multiple classification models to create a classifier able to outperform each of the base classifiers (ROKACH, 2010). There are two different frameworks for ensemble classifiers, according to Rokach (2010): *independent*, where each classifier is built independently from the others, and *dependent*, where the output of a trained model is used as input into another classifier. Figure 2.5 shows both frameworks and their workflow. The Training Set is submitted to a Dataset Manipulator, which is responsible for taking the data to be used in that model, merge any existing data coming from other classifiers, and then feed the data to the Inducer. The Inducer represents the algorithm used to take the data and create the trained model (Classifier). The output of each Classifier is then combined by the Classifiers Composer block to predict the label of an unlabeled input.

Figure 2.5: Frameworks of ensemble models



(a) Independent framework (b) Dependent framework

Source: Rokach (2010)

2.4 Evaluation Metrics

In classification problems, there are many metrics used to verify the performance achieved by classification models. In the following subsections, we present metrics based on Japkowicz and Shah (2011) that we used to assess the results of the experiments in our work.

2.4.1 Kappa

Referred to `KPP` in our work, this metric is the relative improvement of the current predictor on the random predictor. Witten et al. (2016) calculate this metric using Equation 2.1, where $SRAP$ represents the success rate of the actual predictor, and $SRRP$ represents the success rate of a random predictor.

$$KPP = \frac{SRAP - SRRP}{1 - SRRP} \tag{2.1}$$

2.4.2 True Positive Rate

This metric is the rate of positive instances which were classified correctly as such. In our work, we refer to this metric as `TPR`, and it is calculated as presented in

Equation 2.2.

$$TRP = \frac{true\,positive}{true\,positive + false\,negative} \tag{2.2}$$

2.4.3 Precision

Precision is the percentage of the classified instances that do belong to that class. The weighted precision is defined as the weighted average precision of the positive and negative classes using the number of cases in each class as weights. In our work, we report the weighted precision (`PRE`). The precision is calculated as presented in Equation 2.3.

$$PRE = \frac{true\,positive}{true\,positive + false\,positive} \tag{2.3}$$

2.4.4 Recall

The recall is the percentage of correctly classified instances among all instances from that class. Recall also has its variation named weighted recall, which is the weighted recall average for the positive and negative classes using the number of instances on each class as weights. As for precision, we report the weighted recall (`REC`) in our work. The recall is calculated as presented in Equation 2.3.

$$REC = \frac{true\,positive}{true\,positive + false\,negative} \tag{2.4}$$

2.4.5 F-measure

The F-measure, also known as F1, is the harmonic mean between precision and recall. The weighted harmonic mean between weighted precision and weighted recall is named weighted F-measure, which is the one we report in our work as `F1`. The F-measure is calculated as presented in Equation 2.5.

$$F1 = \frac{2 * PRE * REC}{PRE + REC} \tag{2.5}$$

2.4.6 Area Under the Receiver Operating Characteristics Curve

Also known as Area Under the ROC Curve, or just AUC, this is the relationship between true positives and false positives, representing how well the model can distinguish each instance between the classes. There are different methods to calculate this metric, and Equation 2.6 shows one of them, where T_{p} and T_{n} are the subsets of positive and negative instances in the test set T, respectively, and R_{i} is the rank of the ith instance in T_{p}.

$$AUC = \frac{\sum_{i=1}^{|T_{\mathrm{p}}|}(R_{\mathrm{i}} - i)}{|T_{\mathrm{p}}||T_{\mathrm{n}}|} \tag{2.6}$$

2.5 Summary

In this Chapter, we presented popular representations and classification models used for text classification, alongside with standard metrics used to measure their results. However, there are other ways of processing text, and other models employed in text classification in the literature, with new methods coming up every year. In the next Chapter, we provide an overview of related works in the matter of text classification, especially about the detection of offensive content, presenting data sources, methods, and models employed to do so.

3 RELATED WORK

Text classification has been studied for many years, with works in the area of offensive detection dating as earlier as 2004, such as Greevy and Smeaton (2004). Since then, a lot of effort have been applied to create new datasets and techniques to be used to detect offensive content and its different subcategories. This happened mainly in the last years when computational power increased, new methods were developed and advanced models emerged. In this Chapter, we provide an overview of related works in tasks of offensive content detection.

3.1 Datasets

The first studies on offensive content detection had almost no datasets available for experimentation. Due to that reason, most works had to assemble their datasets. Since English is the most widely used language on the Internet[1], most of the studies on offensive language detection have built or used English datasets, such as Magu, Joshi and Luo (2017), Ducharme (2017), Wulczyn, Thain and Dixon (2017), Kennedy et al. (2017), Nobata et al. (2016), for example. More recently, researchers have been building and experimenting with datasets in other languages as well, such as German (BRETSCHNEIDER; PETERS, 2017; ROSS et al., 2016), Italian (VIGNA et al., 2017), Slavic (FIŠER; ERJAVEC; LJUBEŠIĆ, 2017), Dutch (TULKENS et al., 2016; HEE et al., 2015), and Portuguese (FORTUNA et al., 2019; PELLE; MOREIRA, 2017). Although most authors publish their datasets, others do not, which prevents the reproducibility of their work and comparison against other research.

The preferred data sources for gathering data and building datasets are social platforms, thanks to their large volumes of data. Examples of these platforms include, but are not limited to, Reddit (KENNEDY et al., 2017; SALEEM et al., 2016), Facebook (VIGNA et al., 2017; TING et al., 2013), Instagram (HOSSEINMARDI et al., 2015; ZHONG et al., 2016), and Ask.fm (HEE et al., 2015; SAMGHABADI et al., 2017). Twitter deserves a special mention because it is the most used data source across the literature, probably due to its massive user base and its user constant activity over the years. It was used in works such as Fortuna et al. (2019), Davidson et al. (2017), Kennedy et al. (2017), Hasanuzzaman, Dias and Way (2017), Burnap and Williams (2014), and many others.

[1]<https://www.internetworldstats.com/stats7.htm>

Earlier works also used social platforms such as MySpace (DADVAR; JONG, 2012; YIN et al., 2009; NAHAR; LI; PANG, 2013), and Formspring (REYNOLDS; KONTOSTATHIS; EDWARDS, 2011; KONTOSTATHIS et al., 2013; BIGELOW; (KONTOSTATHIS); EDWARDS, 2016).

Moreover, thanks to social plugins and the implementation of comment sections, news portals became a space for user interaction. These platforms were found as a great source of data for analysis. Yahoo! news portals (NOBATA et al., 2016; SOOD; CHURCHILL; ANTIN, 2012; DJURIC et al., 2015) and the Brazilian news portal G1[2] (PELLE; MOREIRA, 2017), for example, were used as a source for collecting comments for offensive content detection. Some authors used comments from Wikipedia to build their datasets for offensive content detection (WULCZYN; THAIN; DIXON, 2017; PAVLOPOULOS; MALAKASIOTIS; ANDROUTSOPOULOS, 2017; SAMGHABADI et al., 2017).

YouTube was used as a data source by Anand et al. (2019) and earlier works such as Ernst et al. (2017), Ducharme (2017), Kandakatla (2016), Dadvar, Trieschnigg and Jong (2013), Chen et al. (2012), Xu and Zhu (2010). 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) evaluate approaches for the detection of pornography in image and video using different datasets and Deep Learning models. Anand et al. (2019) propose a framework to filter videos in English with inappropriate content (insults, hate speech, promotion of extremism or terrorism) to prevent advertisements from using them. However, among the features used, they do not take transcriptions as a feature. In our work, besides 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. Differently from the other studies, we targeted videos in Portuguese, and our goal was to provide an analysis of the performance of each feature set according to a wide range of algorithms and feature representations.

To annotate their datasets, authors have mostly used crowdsourcing platforms, such as Amazon Mechanical Turk[3] (BIGELOW; (KONTOSTATHIS); EDWARDS, 2016; ZHONG et al., 2016; KONTOSTATHIS et al., 2013), CrowdFlower[4] (BURNAP PETEAND WILLIAMS, 2016; DAVIDSON et al., 2017; SAMGHABADI et al., 2017; PETE; L., 2015; HOSSEINMARDI et al., 2015; BURNAP; WILLIAMS, 2014), and

[2]<https://g1.globo.com/>
[3]<https://www.mturk.com/>
[4]CrowdFlower became Figure Eight in 2018: <https://www.figure-eight.com/>

brad rapit[5] (HEE et al., 2015). However, other works conducted their labeling processes without using worldwide crowdsourcing platforms (VIGNA et al., 2017; KENNEDY et al., 2017; WASEEM; HOVY, 2016; ROSS et al., 2016; DJURIC et al., 2015; DADVAR; TRIESCHNIGG; JONG, 2013; KWOK; WANG, 2013). Some authors reported having developed their own annotations tools as well (PELLE; MOREIRA, 2017; WARNER; HIRSCHBERG, 2012).

3.2 Features

Data pre-processing is the first step in classifying offensive content in text. In this step, stop-words were usually removed (ALMEIDA et al., 2017; MAGU; JOSHI; LUO, 2017; SALEEM et al., 2016; DJURIC et al., 2015; BURNAP; WILLIAMS, 2014). Punctuation and special characters were removed as well (SALEEM et al., 2016; BURNAP; WILLIAMS, 2014), although some authors used the presence of exclamations and interrogations as features (NOBATA et al., 2016; CAPUA; NARDO; PETROSINO, 2016). Similarly, other text elements like emojis, hashtags, number, mentions, or URLs were removed by some authors (ALMEIDA et al., 2017; MAGU; JOSHI; LUO, 2017; SALEEM et al., 2016; DJURIC et al., 2015; BURNAP; WILLIAMS, 2014) and used as features by others (SAMGHABADI et al., 2017; DAVIDSON et al., 2017; NOBATA et al., 2016; CAPUA; NARDO; PETROSINO, 2016). Lowercasing the text (DAVIDSON et al., 2017; SALEEM et al., 2016; DJURIC et al., 2015) and applying stemmers on it (DAVIDSON et al., 2017; MAGU; JOSHI; LUO, 2017; BURNAP; WILLIAMS, 2014) are additional operations used in this pre-processing step by researchers.

Binary features indicating the presence or absence of profanity, or offensive words based on lists were used as well (PAPEGNIES et al., 2017; CAPUA; NARDO; PETROSINO, 2016; NOBATA et al., 2016; HOSSEINMARDI et al., 2015; KWOK; WANG, 2013; REYNOLDS; KONTOSTATHIS; EDWARDS, 2011). Some works also used data related to the users who published the contents to obtain features. These user-based features we used by Hasanuzzaman, Dias and Way (2017), Waseem and Hovy (2016), Dadvar, Trieschnigg and Jong (2013) and included age, gender, and location, for example. The user activity and profile statistics were collected in Chatzakou et al. (2017) to trace a profile of Twitter users. The goal was to identify similar users and potential offenders in the network.

[5]<https://brat.nlplab.org/>

In addition to the features mentioned so far, the vast majority of the related works use features based on tokenization and word or character n-gram, or both, as in Davidson et al. (2017), Vigna et al. (2017), Kennedy et al. (2017), Ducharme (2017), Papegnies et al. (2017), Nobata et al. (2016), Saleem et al. (2016), for example. For word n-gram, unigram, bigram, and trigram were the most used. On the other hand, character n-gram varied mostly from two from up to five characters. Some works used POS tags (NOBATA et al., 2016) and even extracted n-grams from them (VIGNA et al., 2017; DAVIDSON et al., 2017; SAMGHABADI et al., 2017; CAPUA; NARDO; PETROSINO, 2016). Embeddings were adopted in many studies and obtained significant results, with word embedding being the most popular one (VIGNA et al., 2017; HASANUZZAMAN; DIAS; WAY, 2017; PAVLOPOULOS; MALAKASIOTIS; ANDROUTSOPOULOS, 2017; SAMGHABADI et al., 2017; NOBATA et al., 2016). Paragraph embeddings (NOBATA et al., 2016) and document embeddings (SAMGHABADI et al., 2017) were used as well. Some authors also used Sentiment Analysis to obtain sentiment polarity scores to be used as features (VIGNA et al., 2017; SAMGHABADI et al., 2017; KENNEDY et al., 2017; PAPEGNIES et al., 2017; DAVIDSON et al., 2017; CAPUA; NARDO; PETROSINO, 2016; SALEEM et al., 2016).

3.3 Classifiers

The approaches used to address offensive content detection evolved over the years from simple techniques, such as solely using dirty-word lists, to more advanced ones using machine learning with multiple features. The most used classifier employed in the literature is SVM, being used in works such as Park and Fung (2017), Ducharme (2017), Magu, Joshi and Luo (2017), Vigna et al. (2017), Samghabadi et al. (2017), Davidson et al. (2017), Pelle and Moreira (2017), and many others.

Probabilistic models were widely used as well. The best example in related works is Naive Bayes (DAVIDSON et al., 2017; SALEEM et al., 2016; KWOK; WANG, 2013; TING et al., 2013). Another example is Logistic Regression (PARK; FUNG, 2017; WULCZYN; THAIN; DIXON, 2017; GAO; KUPPERSMITH; HUANG, 2017; SALEEM et al., 2016), which is sometimes used with some modifications (DAVIDSON et al., 2017). Decision tree models, such as C4.5 (TING et al., 2013; REYNOLDS; KONTOSTATHIS; EDWARDS, 2011), and random forests (CHATZAKOU et al., 2017; KENNEDY et al., 2017; DAVIDSON et al., 2017; PETE; L., 2015) were also employed.

With its popularization in the last few years in different areas, researchers started to use Deep Learning for NLP problems. LSTM models were used in Vigna et al. (2017), Gao, Kuppersmith and Huang (2017), for example. Pavlopoulos, Malakasiotis and Androutsopoulos (2017) employed both RNN and CNN models in their works. Park and Fung (2017) proposed three CNN models based on characters (CharCNN), words (WordCNN) and both of them (HybridCNN) to identify abusive language on Twitter. Gambäck and Sikdar (2017) use CNN to detect Hate Speech. CNN was also used in Pavlopoulos, Malakasiotis and Androutsopoulos (2017). Transfer Learning models have become more prevalent in the last couple of years, and some works on offensive content detection are starting to emerge (BASILE et al., 2019a; WU et al., 2019; AGGARWAL et al., 2019).

In addition to using single models, some authors employed ensemble classifiers in their experiments. Pelle, Alcântara and Moreira (2018) combined the output of three classifiers (HateWord2Vec, HateDoc2Vec, and SVM) in a meta-classifier to predict offensive comments in the Web. Burnap and Williams (2014) predict Hate Speech on Twitter using the output of three Classic classification models (random forest, SVM, and logistic regression) in an ensemble classifier. In both works, the use of a meta-classifier improved the results when compared to the combined models used separately.

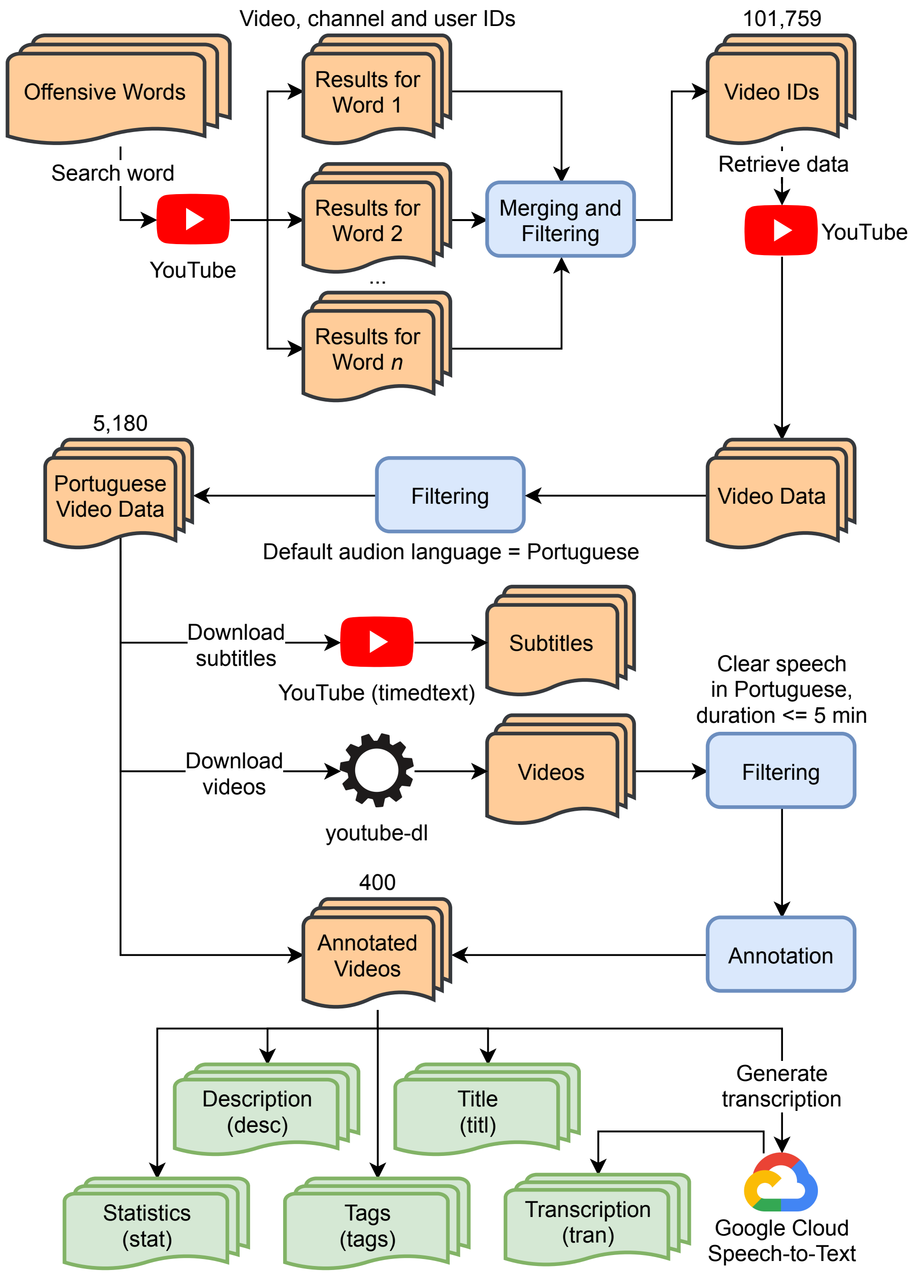
3.4 Summary

In our work, we followed a similar approach adopted by related works on offensive content detection but filled some of the gaps identified in this Chapter, such as processing videos in Portuguese. This language is typically underrepresented in terms of the availability of annotated training data for machine learning algorithms. Also, the related works presented here supported our choice about what features and classifiers to use. Our dataset is presented next, in Chapter 4. Also, we tested different classification models from the most seminal up to the most recent, using features selected based on their popularity and results use across related works presented in this Chapter. The experimental setup and results are presented in Chapters 5 and 6.

4 DATASET

In this Chapter, we detail the process employed in the creation of the dataset for offensive video detection and describe the dataset. The entire process is illustrated in Figure 4.1 and explained in the next sections.

Figure 4.1: Dataset creation process



Source: The Authors

4.1 Data Collection

To retrieve video data from YouTube, we used its official Application Programming Interface[1] (API). To search for potentially offensive videos, we used the `<search.list>` endpoint and provided seed words using a list of dirty/offensive words provided by Pelle, Alcântara and Moreira (2018), which contains terms in Portuguese and English. Each seed word was searched individually, retrieving a set of video, channel, and user ids. However, we kept only the video ids and discarded the channel and user ids retrieved. Next, the search results were merged to avoid duplicates of videos that might have appeared in more than one search. This process resulted in a total of 101,759 video ids. Then, we used the `<videos.list>` endpoint to retrieve detailed and structured[2] information for each video. Such information contains the features to be processed in this study.

Next, we filtered the videos looking for the ones with default audio language attribute explicitly set to Portuguese. Although we also retrieved videos in English, we decided to focus on processing Portuguese videos at first, as one of our objectives is to build and provide a dataset of videos in this language to contribute with the development of resources in Portuguese. At the end of this filtering step, we ended up with a set of 5,180 videos. Next, we downloaded these videos using *youtube-dl*[3], so they could have their transcriptions extracted later. A second reason for downloading the videos was to embed them directly in the annotation tool we developed instead of using YouTube video player. We adopted this measure to prevent the unavailability of videos in case they were removed from YouTube by the time they were annotated in this study. However, we did not share or publish the downloaded video files, restricting their use to this work.

We also used an unofficial API[4] provided by YouTube to retrieve the subtitles for the videos. However, although some videos have subtitles in more than one language, 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.

To obtain the transcriptions, we used the Google Speech-to-Text[5] service, which makes use of machine learning to transcribe audio files automatically. Since this is a paid

[1]<https://developers.google.com/youtube/v3/>
[2]<https://developers.google.com/youtube/v3/docs/videos#resource-representation>
[3]<http://ytdl-org.github.io/youtube-dl/>
[4]<https://www.youtube.com/api/timedtext>
[5]<https://cloud.google.com/speech-to-text/>

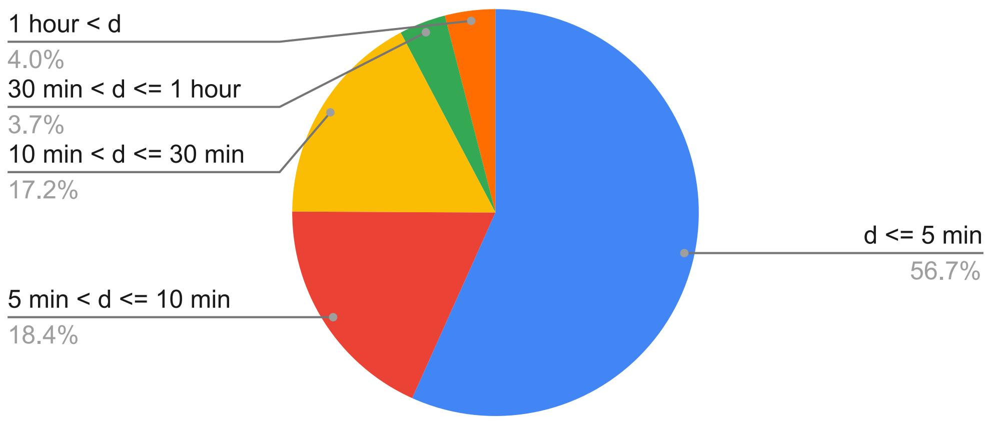
service, we only submitted for transcription the videos successfully annotated by three annotators. To provide the video to the transcription service, we had to convert the file to Waveform Audio File Format[6] (WAV) and also specify Portuguese as the target language for the transcription. Google Speech-to-Text has an option to filter profanity words and phrases named profanity filter, which we disabled to keep the transcription more loyal to the original audio and provide more realistic results during our experiments.

4.2 Annotation Process

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 user who uploaded the video may have set the default audio language attribute with a wrong value, 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 (see Figure 4.2). The goal was also to keep annotators engaged and save time in the annotation process, which is the bottleneck in dataset creation.

Figure 4.2: Distribution of videos in Portuguese according to their duration



Source: The Authors

We developed a web tool to enable volunteers to annotate the videos. Each video

[6]<http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html>

was analyzed by three annotators, as this is the number used by Pelle and Moreira (2017), Davidson et al. (2017), and other authors who annotated their datasets. The annotators were mostly from the academic environment, with varied age, gender, location and expertise in the subject. 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 be able to carry on with the annotation process. These instructions are presented in Figure 4.3, alongside with the definition of what should be considered offensive in the videos. This definition was designed to guide the annotators in the process and prevent them from letting their personal beliefs or emotions affect their judgment. These guidelines were presented to each annotator right before they started to evaluate the videos and were available at any time in the video annotation page, as shown in Figure 4.4. The annotation tool supported Portuguese and English for the interface. However, as our target language was Portuguese, we presented the annotation guidelines in Portuguese even if the interface language was set to English.

The tool was also used to show the general and user-specific annotation progress, as an attempt to engage them in the annotation process, as can be seen in Figures 4.5 and 4.6. We anonymized the annotators' names in Figures 4.5 to prevent exposing their personal information. Since we believe this tool could help other researchers creating their datasets, we made its source code available[7].

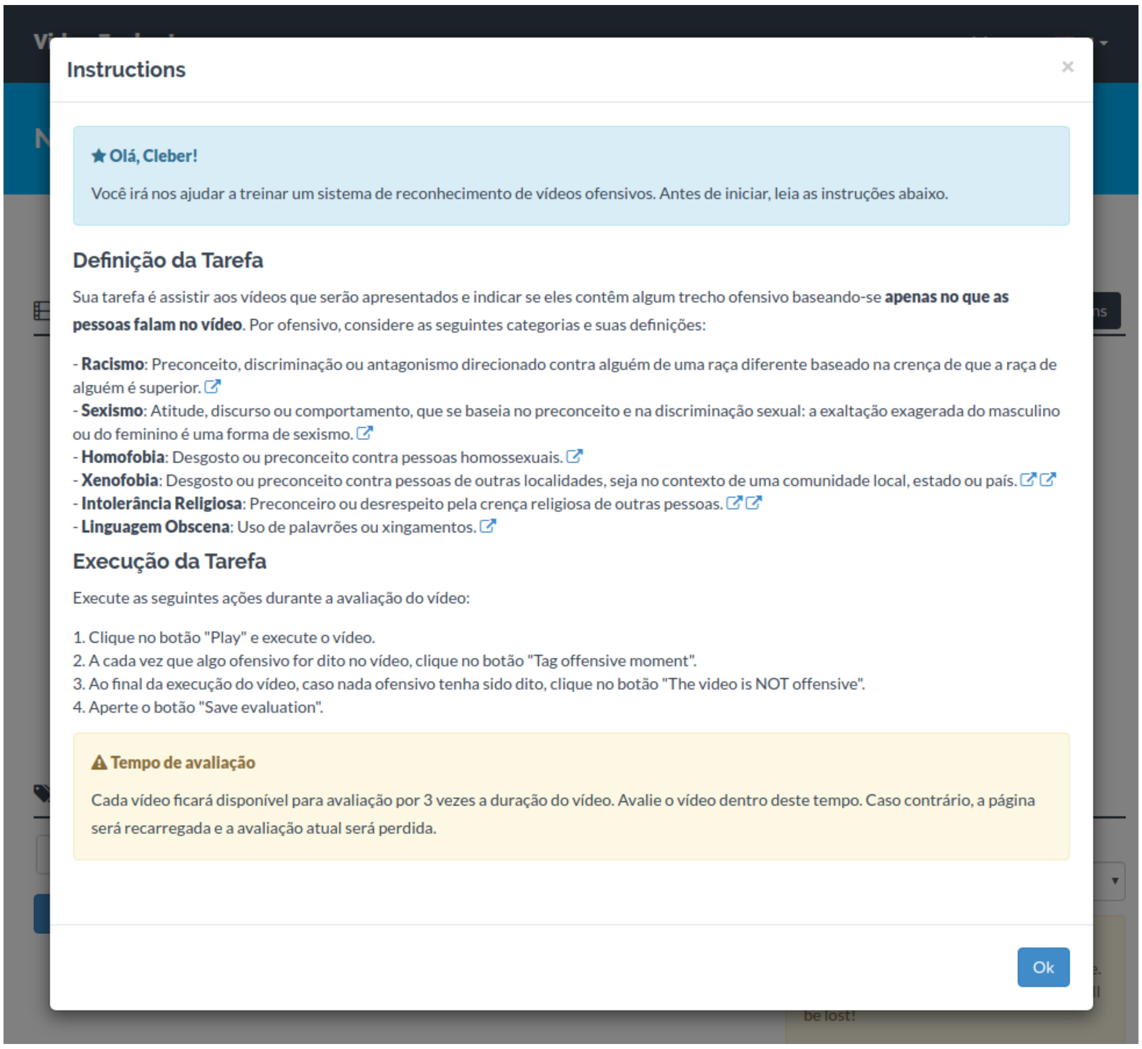
By the end of the process, we had 400 videos annotated. For classification purposes, we considered a video as offensive if it had *at least one moment tagged as offensive by at least two annotators*. According to the agreement among the annotators, we created two datasets:

- `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[8]. They were accepted and published under the ISLRN 529-322-484-169-1. However, due to YouTube API Services Developer Policies[9], the video contents cannot be published. Therefore, in addition to the video id and annotations (labels assigned by each annotator), we included the following sets of information (feature sets): description, tags, title, transcription, and statistic. Each of these feature sets

[7]<https://gitlab.com/cleber.93cd/video-hate-detector>
[8]<http://www.inf.ufrgs.br/~csalcantara/offensive-video-detection/datasets/>
[9]<https://developers.google.com/youtube/terms/developer-policies>

Figure 4.3: Guidelines provided to annotators at the beginning of the annotation process, also available at any time in the video annotation page


Instructions

Olá, Cleber!

Você irá nos ajudar a treinar um sistema de reconhecimento de vídeos ofensivos. Antes de iniciar, leia as instruções abaixo.

Definição da Tarefa

Sua tarefa é assistir aos vídeos que serão apresentados e indicar se eles contêm algum trecho ofensivo baseando-se **apenas no que as pessoas falam no vídeo**. Por ofensivo, considere as seguintes categorias e suas definições:

- **Racismo**: Preconceito, discriminação ou antagonismo direcionado contra alguém de uma raça diferente baseado na crença de que a raça de alguém é superior.
- **Sexismo**: Atitude, discurso ou comportamento, que se baseia no preconceito e na discriminação sexual: a exaltação exagerada do masculino ou do feminino é uma forma de sexismo.
- **Homofobia**: Desgosto ou preconceito contra pessoas homossexuais.
- **Xenofobia**: Desgosto ou preconceito contra pessoas de outras localidades, seja no contexto de uma comunidade local, estado ou país.
- **Intolerância Religiosa**: Preconceiro ou desrespeito pela crença religiosa de outras pessoas.
- **Linguagem Obscena**: Uso de palavrões ou xingamentos.

Execução da Tarefa

Execute as seguintes ações durante a avaliação do vídeo:

1. Clique no botão "Play" e execute o vídeo.
2. A cada vez que algo ofensivo for dito no vídeo, clique no botão "Tag offensive moment".
3. Ao final da execução do vídeo, caso nada ofensivo tenha sido dito, clique no botão "The video is NOT offensive".
4. Aperte o botão "Save evaluation".

Tempo de avaliação

Cada vídeo ficará disponível para avaliação por 3 vezes a duração do vídeo. Avalie o vídeo dentro deste tempo. Caso contrário, a página será recarregada e a avaliação atual será perdida.

Source: The Authors

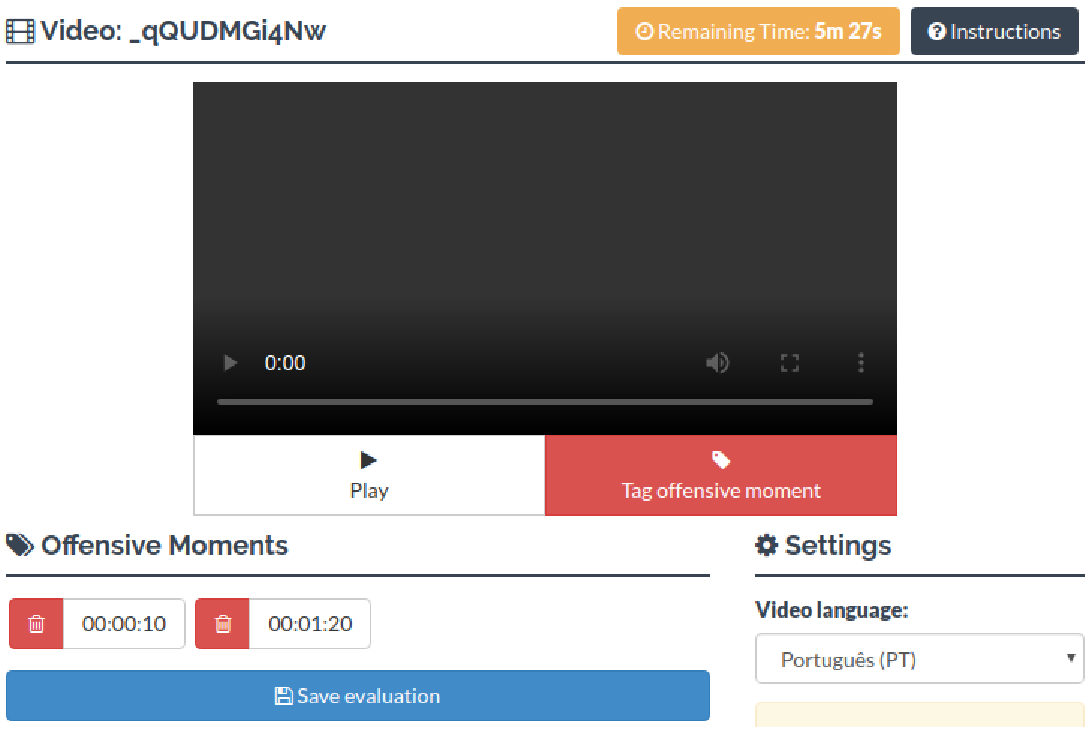
is described in detail in the next Section.

4.3 Feature Sets

We did not use the whole data retrieved from YouTube, as a lot of it was not relevant in our task. Instead, we extracted four sets of textual information and a collection of information containing statistics for each video. 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

Figure 4.4: Video annotation page with a video being evaluated and two moments tagged as offensive

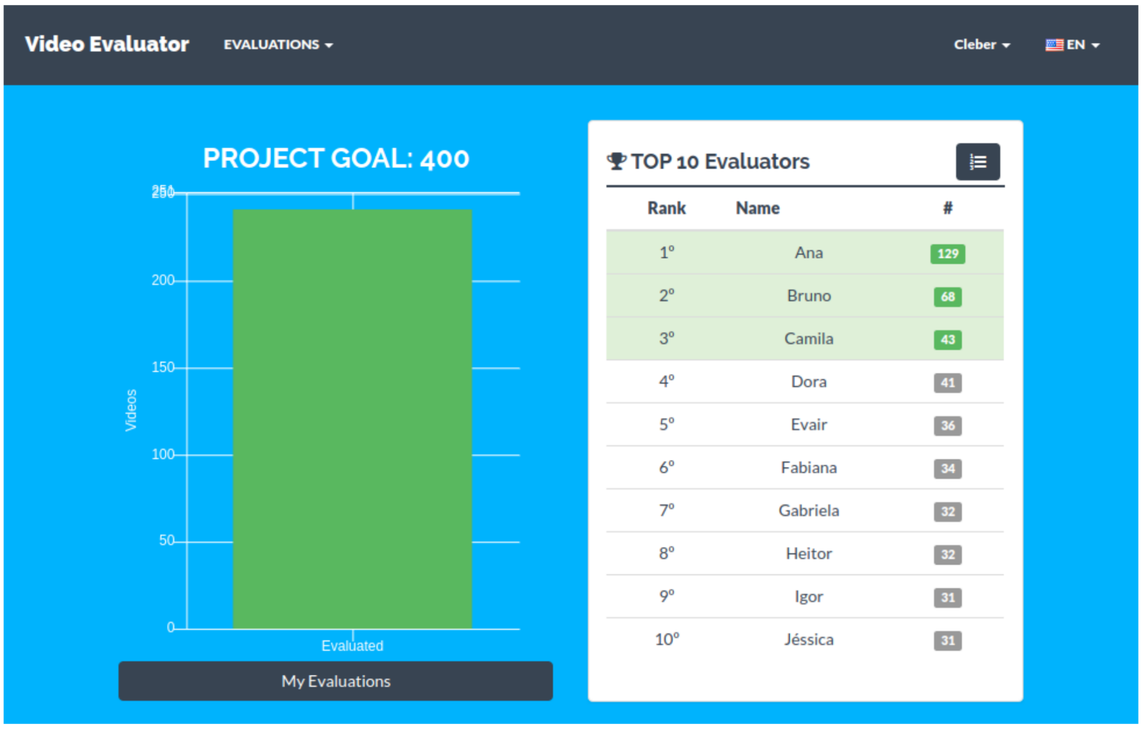


Source: The Authors

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 automatically generated 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 did not perform normalization for these counters. 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, Alcântara and Moreira (2018)), *video duration*

Figure 4.5: Home page of the annotation tool with annotator ranking and project goal



TOP 10 Evaluators

Rank	Name	#
1º	Ana	129
2º	Bruno	68
3º	Camila	43
4º	Dora	41
5º	Evair	36
6º	Fabiana	34
7º	Gabriela	32
8º	Heitor	32
9º	Igor	31
10º	Jéssica	31

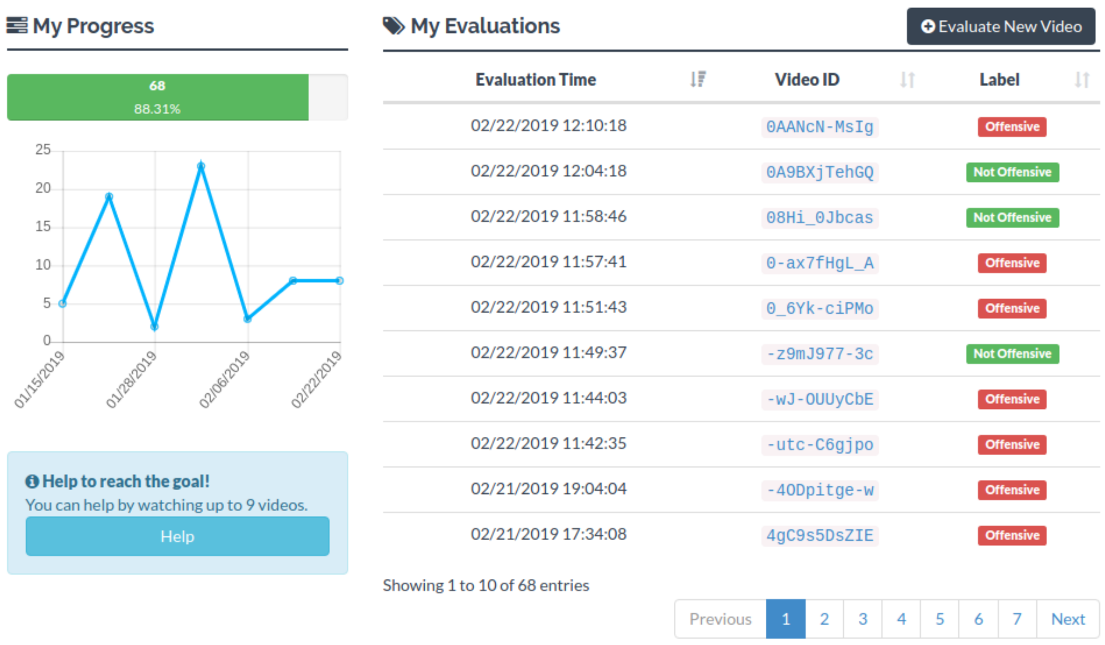
Source: The Authors

(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).

4.4 Dataset Statistics

To assess the degree of agreement among the annotators, we calculated the Fleiss Kappa (FLEISS, 1971; LANDIS; 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 that annotators of one instance to be the same as for the other instances, which is exactly the 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. Samghabadi et al. (2017) reported 0.45, Warner and Hirschberg (2012) found 0.63, and 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.

Figure 4.6: User personal page with their progress and annotations



Help to reach the goal!
You can help by watching up to 9 videos.
Help

My Evaluations

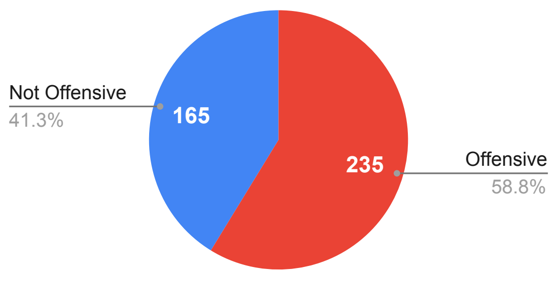
Evaluation Time	Video ID	Label
02/22/2019 12:10:18	0AANcN-MsIg	Offensive
02/22/2019 12:04:18	0A9BXjTehGQ	Not Offensive
02/22/2019 11:58:46	08Hi_0Jbcas	Not Offensive
02/22/2019 11:57:41	0-ax7fHgL_A	Offensive
02/22/2019 11:51:43	0_6Yk-ciPMo	Offensive
02/22/2019 11:49:37	-z9mJ977-3c	Not Offensive
02/22/2019 11:44:03	-wJ-OUUyCbE	Offensive
02/22/2019 11:42:35	-utc-C6gjpo	Offensive
02/21/2019 19:04:04	-4ODpitge-w	Offensive
02/21/2019 17:34:08	4gC9s5DsZIE	Offensive

Showing 1 to 10 of 68 entries

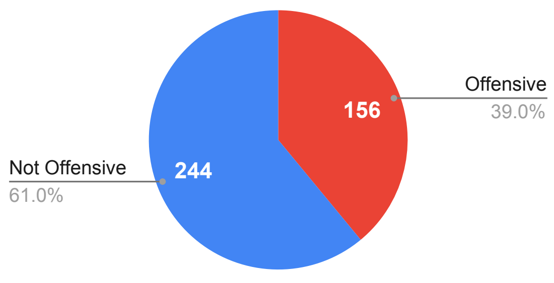
Source: The Authors

In `OffVidPT-2`, 235 videos (out of 400) were classified as offensive, which corresponds to 58.8% of the total. As for the `OffVidPT-3` dataset, there were 156 videos considered offensive, corresponding to 39.0% of the total. These distributions for the datasets are presented in Figure 4.7. 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.

Figure 4.7: Class distribution of our datasets



(a) `OffVidPT-2`



(b) `OffVidPT-3`

Source: The Authors

4.5 Summary

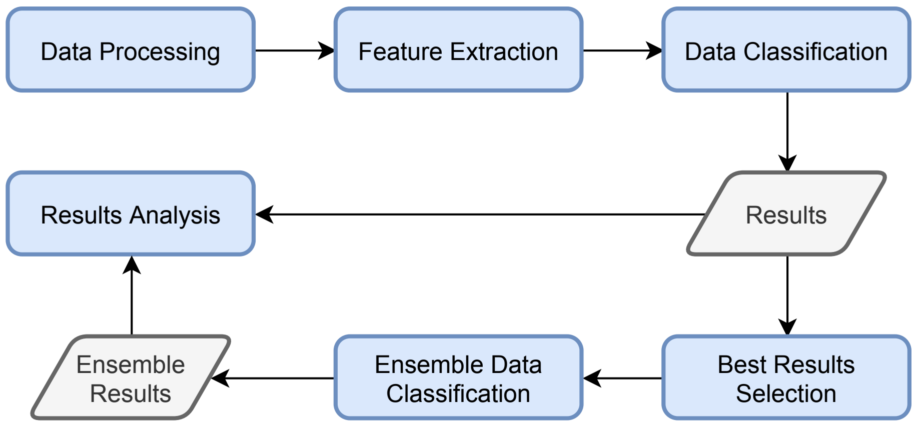
The process described in this Chapter shows how we collected and annotated our data. It also indicates which data we selected to use in our study, coming up with different feature sets and two datasets based on the annotators' agreement (`OffVidPT-2` and `OffVidPT-3`). In the next Chapter, we present the approach employed in this study for offensive video detection, which includes an overview of the procedures used to process, classify, and analyze our data, experiments, and results.

5 DETECTING OFFENSIVE VIDEOS

The goal of this work is to address the problem of offensive video detection. The hypothesis we wish to investigate is whether it is possible to accurately classify if a video is offensive just by analyzing its textual features. In this investigation, we apply a wide range of Classic, Deep Learning, and Transfer Learning algorithms.

The methodology adopted in this study is depicted in Figure 5.1. In summary, we first process the input data to extract features to submit in the classifiers to perform data classification. With the results in hand, we select the best classifiers and combine their predictions, which in turn are submitted as input to an ensemble classifier. Finally, a detailed analysis is performed addressing the offensive video detection problem. Our methodology, described in the next sections of this Chapter, is generic and could be applied to related and future works in this subject.

Figure 5.1: Overview of our methodology for offensive video identification



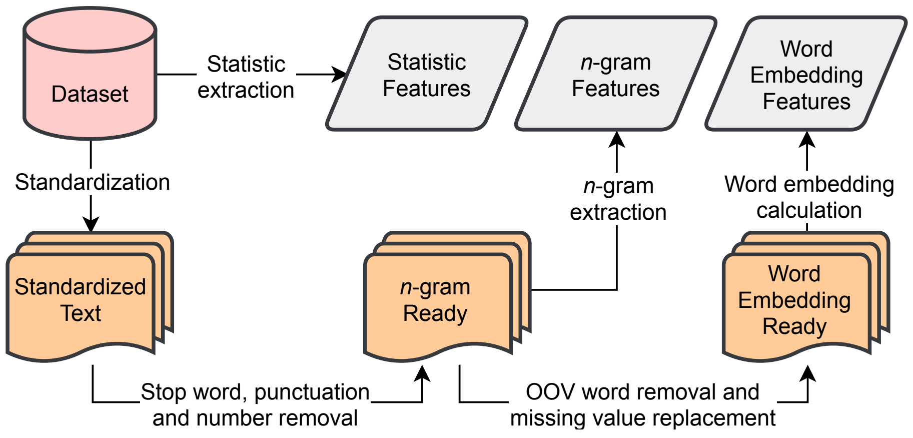
Source: The Authors

5.1 Data Processing and Feature Extraction

The first step in our approach is to process the input data to make them suitable for being handled by classification algorithms. This processing stage is fundamental to get more uniform and standardized data, which increases the performance of classification algorithms.

The different types of algorithms used in this work require different types of processing and feature extraction. The Transfer Learning algorithms deal with raw text, which typically undergoes transformations such as tokenization and discarding noisy characters. Deep learning methods take word embedding representations as input. The Classic algorithms can use textual features from word or character n-gram and word embedding. Additionally, Classic algorithms also processed statistical features. Figure 5.2 details the steps in the process of data processing and feature extraction applied to our dataset, where out-of-vocabulary is abbreviated to OOV.

Figure 5.2: Steps to process the data and extract features for use in the classifiers



Source: The Authors

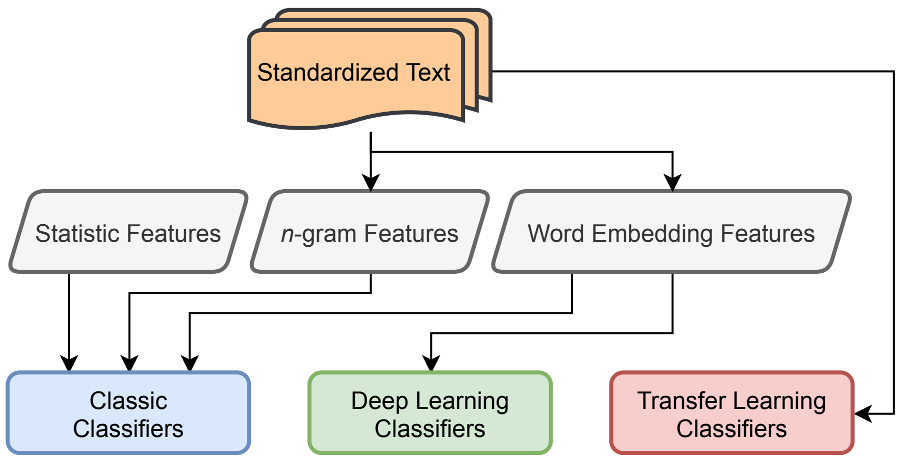
5.2 Data Classification

The features extracted in the previous step are used to train and test different machine learning algorithms for classification. Figure 5.3 presents the relation between features and classifiers, which are grouped by their categories.

The Classic algorithms employed in this study were covered in Section 2.3.1. For Deep Learning, we designed architectures based on two main algorithms: CNN and LSTM. Both networks use word embedding as input. Two different CNN architectures are employed, both inspired by Kim (2014). The first one, named W-CNN, is shown in Figure 5.4. It has multiple convolution layers, each one responsible for processing a different number of embeddings at a time. The max-pooling layer reduces the dimension of the output of the convolution layers and concatenates them to feed the fully connected layer that comes next. 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.

The second architecture, named M-CNN, is depicted in Figure 5.5. It uses a single

Figure 5.3: Features used to feed the classifiers from each category



Source: The Authors

embedding per document as input, which is manually calculated and described in Section 6.1.1. To process the embedding, we use a single convolution layer. The remaining layers of the network are the same as in the W-CNN. The difference between the two architectures is that while the W-CNN takes multiple embeddings as input (one per word in the instance), M-CNN takes a single embedding generated by an aggregation (*i.e.,* the average). The goal was to compare the two alternatives.

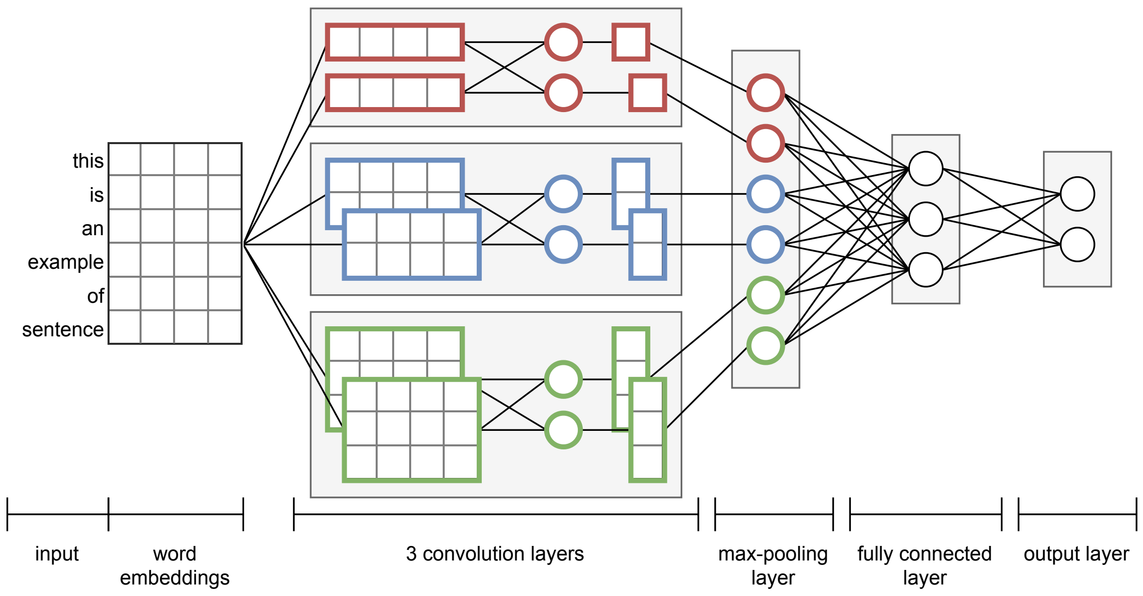
For our first LSTM implementation, named W-LSTM, the memory cells process each word embedding extracted from the sentences. At the end of the processing, the LSTM layer outputs the data used by the last layer in the network to calculate and output the predictions. An overview of this architecture is depicted in Figure 5.6.

Similarly to the CNN architecture, we additionally defined an alternative LSTM architecture to process a single embedding per document. We refer to this architecture as M-LSTM, and it is represented in Figure 5.7. The other layers were not changed.

5.3 Best Results Selection and Ensemble Data Classification

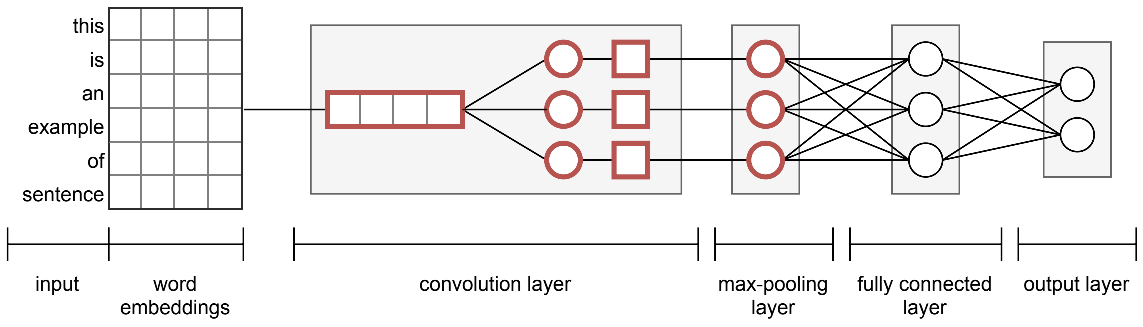
After running the experiments with all the classifiers and feature sets, we selected the best results for each algorithm category and feature set, reporting the feature representation and classifier that achieved the best result. The outputs of the selected classifiers are used as inputs in a meta-classifier, *i.e.,* in an ensemble. The entire process used in the creation of the ensemble is shown in Figure 5.8. Additionally, we performed feature ablation to understand the contribution of each member of the ensemble to the final result.

Figure 5.4: Architecture employed in the W-CNN model



Source: The Authors

Figure 5.5: Architecture employed in the M-CNN model



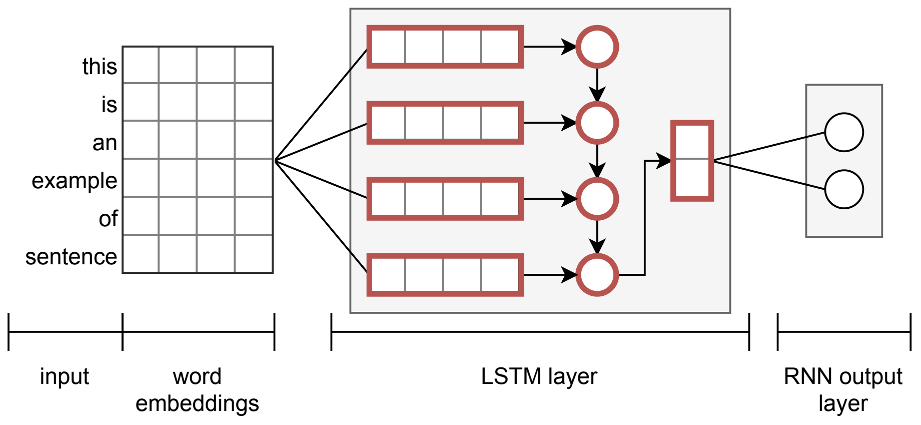
Source: The Authors

This procedure means we excluded the predictions of one feature set at a time to check the impact of each of them in the results.

5.4 Results Analysis

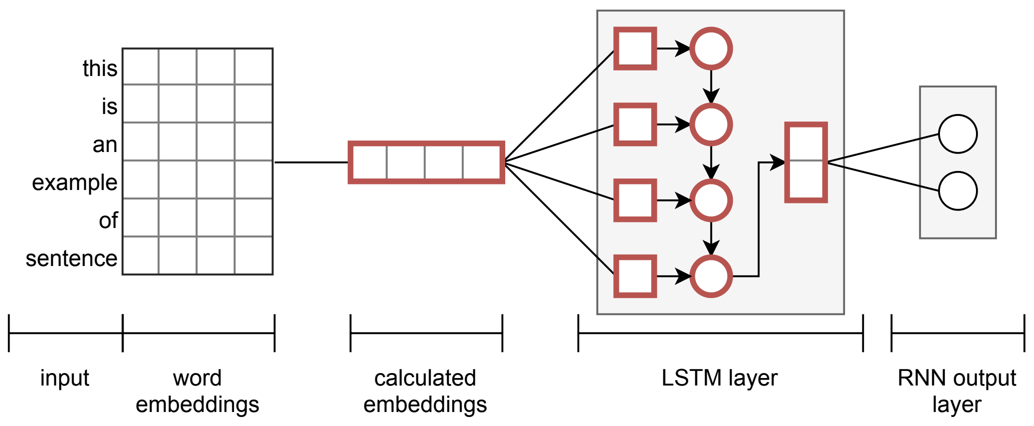
In this final stage, we analyzed the results of all classifiers, aiming at providing an evaluation of the ability to detect offensive videos, which features are more helpful at this task, and which algorithms perform better at detecting offensive content. We analyze the results by algorithm category, feature representation, and feature set. Results were scored

Figure 5.6: Architecture employed in the W-LSTM model



Source: The Authors

Figure 5.7: Architecture employed in the M-LSTM model



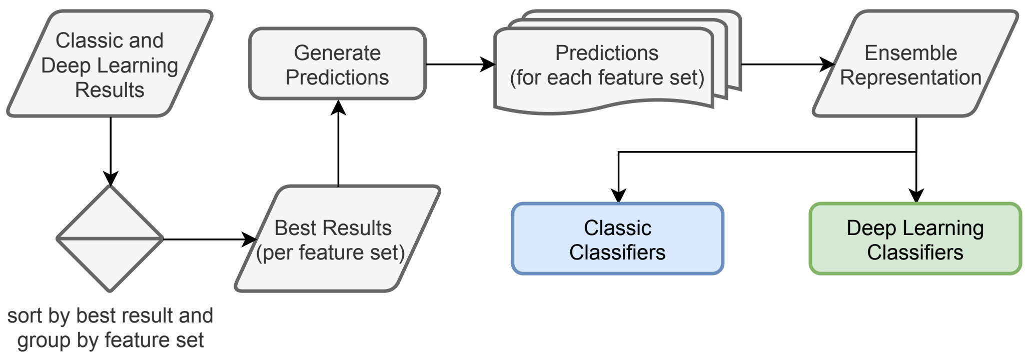
Source: The Authors

under different metrics to allow for a richer analysis.

5.5 Summary

In this Chapter, we presented the process we used in our study to provide information on offensive video identification by using different features and classifiers. We detailed each phase of this process to provide a comprehensive view of their function in the whole process. In the next Chapter, we describe in detail our experiments, including the configuration we applied to the algorithms, and present and discuss their results.

Figure 5.8: The process employed to process the results of the classifiers and create an ensemble classifier in our work


Classic and Deep Learning Results
Generate Predictions
Predictions (for each feature set)
Ensemble Representation
sort by best result and group by feature set
Best Results (per feature set)
Classic Classifiers
Deep Learning Classifiers

Source: The Authors

6 EXPERIMENTS

The experiments reported in this Chapter aim at answering the following research questions:

- 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 next sections describe the experimental setup and discuss the results.

6.1 Experimental Setup

In this Section, we present the materials used in the experiments, which includes the features extracted from the feature sets of the `OffVidPT` dataset, classifiers, metrics, and the framework adopted.

6.1.1 Features

We used all feature sets of both datasets introduced in Chapter 4 (`OffVidPT-2` and `OffVidPT-3`). The statistic feature set was already built in a format that would not require additional processing before the submission to the Classic classifiers. However, the textual feature sets needed to be pre-processed before the experiments, since the text was noisy and not standardized. All the textual feature sets were used without sectioning, including the transcriptions, which could have been sectioned according to the offensive moments tagged during the annotation process.

We present statistics about the textual feature sets at each step of the pre-processing phase, starting with the ones in Table 6.1 for the raw text. Those statistics include the number of instances with missing value in the feature set (NP), minimum (min), maximum (max), mean (avg), and standard deviation (sdv) of both words and characters in each instance.

We pre-processed the textual feature sets using the same script developed by Hartmann et al. (2017) to pre-process their corpus and train word embeddings. The code

Table 6.1: Statistics extracted from the raw data of the textual feature sets, before any pre-processing being applied

Feature Set	NP	Chars				Words			
		min	max	avg	sdv	min	max	avg	sdv
`desc`	11	0	4,938	762.7	873.8	0	826	108.2	144.5
`tags`	54	0	537	185.9	151.8	0	89	27.3	22.9
`titl`	0	4	100	50.6	23.8	1	22	9.5	4.9
`tran`	0	6	8,423	1,723.8	1,437.7	1	1,494	324.6	265.4

Source: The Authors

changed the text to lowercase, converted any URL to the `URL` token, converted any email address to the `EMAIL` token, converted numbers to the `0` token, disconnected punctuation from words, and standardized different quotes and hyphens. The original script 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. Table 6.2 shows statistics of the textual feature sets after this pre-processing was applied.

Table 6.2: Statistics extracted from the standardized textual feature sets, the output of the pre-processing using customized scripts based on Hartmann et al. (2017)

Feature Set	NP	Chars				Words			
		min	max	avg	sdv	min	max	avg	sdv
`desc`	11	0	4,623	631.9	821.0	0	984	119.5	160.0
`tags`	54	0	602	199.6	163.2	0	139	41.4	34.3
`titl`	0	4	101	51.3	24.2	1	25	10.6	5.5
`tran`	0	6	8,425	1,723.6	1,437.6	1	1,500	324.6	265.6

Source: The Authors

The instances at this point were ready to be used in the Transfer Learning models, which can process plain text as they internally create the feature representations for the data. However, the instances required further pre-processing to be used in the Classic and Deep Learning models, which requires 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, also illustrated in Figure 5.2: n-gram and word embedding.

n-gram. To generate the features for this representation, 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)[1]. Table 6.3 shows the statistics for the data after passing through this processing. Then, we generated n-grams using two different types of tokens: words and characters. For word n-grams, we generated 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 6.4 counts the features created for each feature set and representation. The statistic feature set, which is not textual, contains eight features, described previously in Section 4.3.

Table 6.3: Statistics extracted from the pre-processed text used for n-gram feature extraction

Feature Set	NP	Chars				Words			
		min	max	avg	sdv	min	max	avg	sdv
`desc`	11	0	3,707	467.6	610.0	0	633	69.6	93.0
`tags`	54	0	465	157.2	129.2	0	79	23.4	19.3
`titl`	0	4	86	39.9	18.7	1	15	6.4	3.2
`tran`	0	6	5,937	1,211.0	1,004.2	1	809	181.8	144.1

Source: The Authors

Table 6.4: Number of features in each n-gram representation for the textual feature sets

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

Source: The Authors

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 represent 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 OOV words, *i.e.,* tokens not found in the trained embeddings were discarded. At this point, some instances of the tags feature set ended up with no value, additionally to the previously missing ones. To generate embeddings for all instances of the textual feature sets and enable the correct processing of our models, we replaced missing descrip-

[1]<http://www.nltk.org/>

tions and tags with transcriptions and titles, respectively. We chose this criterion because these feature sets instances have similar lengths. Table 6.5 shows the average and standard deviation of OOV words of each feature set, and the statistics about the words before and after the replacement of the missing values. Statistics on the number of characters are not reported this time because we generated embeddings only for words, and the number of characters, in this case, was irrelevant to the calculation. We generated two sets of word embeddings for each instance, which have their creation process depicted in Figure 6.1 and described as follows:

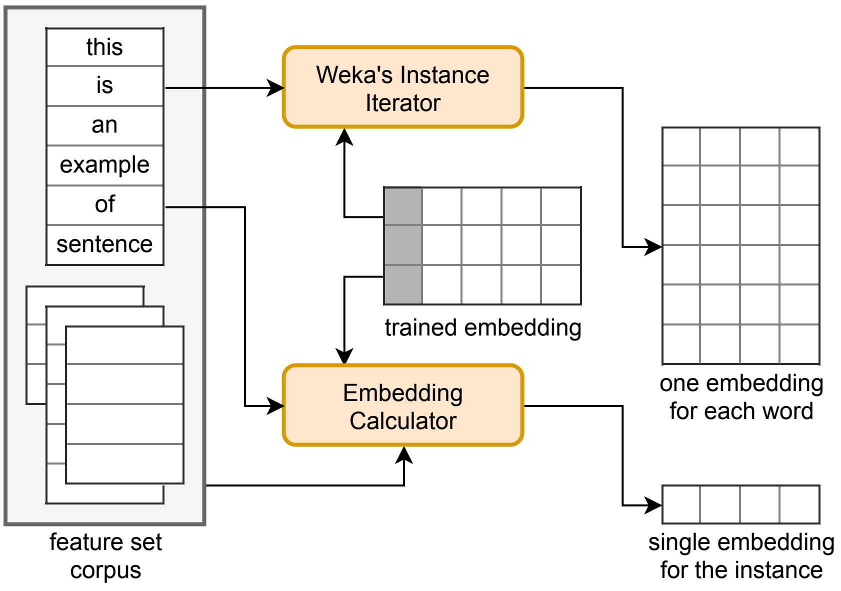
- **Single embedding**: This set was composed of a single embedding for each instance and had 300 dimensions – the same as the trained word embeddings. Our Embedding Calculator (Figure 6.1) performed the embedding calculation. For each instance, we took the weighted average of their unigram feature vectors. 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 the experimentation tool 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 of the textual feature sets.

Table 6.5: Statistics extracted from the pre-processed text used for word embedding generation

Feature Set	NP	OOV W.		Words before replacement				Words after replacement			
		avg	sdv	min	max	avg	sdv	min	max	avg	sdv
`desc`	11	2.4	4.7	0	588	67.2	89.7	1	588	70.5	91.1
`tags`	57	1.4	2.2	0	78	22.0	18.7	1	78	22.8	17.8
`titl`	0	0.4	0.7	1	14	6.0	3.1	1	14	6.0	3.1
`tran`	0	1.0	2.9	1	809	180.8	143.6	1	809	180.8	143.6

Source: The Authors

Figure 6.1: Generation process of embedding sets for use in the Classic and Deep Learning algorithms



Source: The Authors

6.1.2 Classifiers

We submitted our features to the Classic, Deep Learning, and Transfer Learning models covered in Section 2.3. To run these classifiers, we used two different tools (environments): Weka for the Classic and Deep Learning models, and Google Colab[2] for the Transfer Learning models. The implementations we used in this study for the Classic algorithms on Weka are presented in Table 6.6. Some of the classifiers have different names from the algorithms they implement.

Table 6.6: Weka classifiers used for the Classic algorithms

Classic Algorithm	**Implementation**
Naive Bayes	`<weka.classifiers.bayes.NaiveBayes>`
Logistic Regression	`<weka.classifiers.functions.SimpleLogistic>`
SVM	`<weka.classifiers.functions.SMO>`
C4.5	`<weka.classifiers.trees.J48>`
Random Forest	`<weka.classifiers.trees.RandomForest>`

Source: The Authors

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

[2]<https://colab.research.google.com/>

Weka's package manager to fill this gap. This package expands Weka's original set of classifiers with Deep Learning algorithms based on Deeplearning4j[3], enabling the user to define their architecture with the different algorithms and layers available.

For the Deep Learning classifiers, we used two main architectures: W-CNN and W-LSTM. Additionally, we defined two others based on the main ones: M-CNN and M-LSTM. While the main ones use the "Word Embedding Ready" text from Figure 5.2 as input, the M-CNN and M-LSTM architectures use the manually calculated embedding described in the Subsection 6.1.1 as input. To process it, the W-CNN and W-LSTM architectures convert each word in the instances to word embedding using a CnnTextEmbeddingInstanceIterator and RnnTextEmbeddingInstanceIterator, respectively. For each iterator, we provided the trained embedding, mini-batch size = 50 (choice based on Kim (2014)), truncation length = 810, and no stop word removal (Dl4jNull scheme), as we already removed stop words during pre-processing. The truncation length was chosen based on the longest sentence in our corpus, which comes from the transcriptions feature set. The default token pre-processor and tokenizer factory provided by Weka were used. For the M-CNN, we used a ConvolutionInstanceIterator, with same mini-batch size as the others, desired width = 300, desired height = 1, and desired number of channels = 1. Finally, for the M-LSTM, we used a DefaultInstanceIterator with a mini-batch size = 8.

Our W-CNN uses a similar architecture and parameter values of the network built by Kim (2014) and is illustrated in Figure 5.4. We used a Dl4jMlpClassifier, to which we added three convolution layers with the number of rows in the kernel (k) varying from one to three for each of them. We chose these numbers to keep the same range used to process word n-grams in our study. All the three convolution layers share the following other parameter values: number of columns in kernel = 300 (the same number of dimensions of the embedding), number of filters = 100, convolution mode = "SAME", stride = 1x300, no padding (0x0), and ReLU as the activation function. To merge the feature maps from the three convolution layers, we added a max-pooling layer (GlobalPoolingLayer with pooling type = "MAX") after them. Next, we added a fully connected layer with dropout = 0.5 to prevent over-fitting (DenseLayer followed by DropoutLayer on Weka), number of outputs = 100, ReLU as the activation function. Finally, we have an output layer with two outputs (one for each of our classes) and Softmax as the activation function. Additionally, we set L2 regularization factor = 0.0003 through the network configuration options of the classifier on Weka.

[3]<https://deeplearning4j.org/>

The M-CNN only differs from the W-CNN by the configuration of the convolution layers. Instead of three layers, there is a single convolution layer, as can be seen in Figure 5.5, with number of filters = 300, kernel = 300x1, stride = 1x1. The other parameters are the same as in the W-CNN. This configuration enabled the convolution layer to use the entire embedding of each instance to calculate the weights of the filters, providing the same output size as the W-CNN to the other layers in the network. We tried to set a different kernel size, but that performed very poorly.

The architecture we employed in our W-LSTM model is shown in Figure 5.6. We applied a similar configuration as in Gao, Kuppersmith and Huang (2017): one LSTM layer with hyperbolic tangent activation function, sigmoid function for the gate activation, and the number of outputs = 100. We used the `GravesLSTM` implementation provided by Weka for the LSTM layer, which implements the vanilla LSTM model presented in Greff et al. (2017). Next, we used a RnnOutputLayer for the output layer with binary cross-entropy loss function (LossBinaryXENT), sigmoid activation function, and the number of outputs = 2. This model, differently from the W-CNN, was built using a RnnSequenceClassifier, for which we also set the backpropagation through time = 50 for both backward and forward parameters. These parameters are used to reduce the complexity when updating weights in RNNs when the sequences under processing are generally long. The value of these parameters was set according to the recommendation on DL4J page[4]. Additionally, we used the network configuration option in the classifier to set dropout = 0.2, as in Gao, Kuppersmith and Huang (2017). The other parameters were left with their default values.

The last Deep Learning architecture, the M-LSTM, used the manually calculated embedding to learn how to identify offensive videos. Differently from the W-LSTM architecture defined previously, the M-LSTM used a Dl4jMlpClassifier, since the RnnSequenceClassifier could only handle text. That also means we did not have to set the backpropagation through time parameters, but we still used the network configuration option to set dropout = 0.2. We used the GravesLSTM implementation for the LSTM layer with the same configuration as in the W-LSTM model. Also, we used an OutputLayer with the same setup as the RnnOutputLayer in the W-LSTM. The architecture of this network is represented in Figure 5.7.

[4]<https://deeplearning4j.org/docs/latest/deeplearning4j-nn-recurrent#tbptt>

6.1.3 Evaluation Metrics

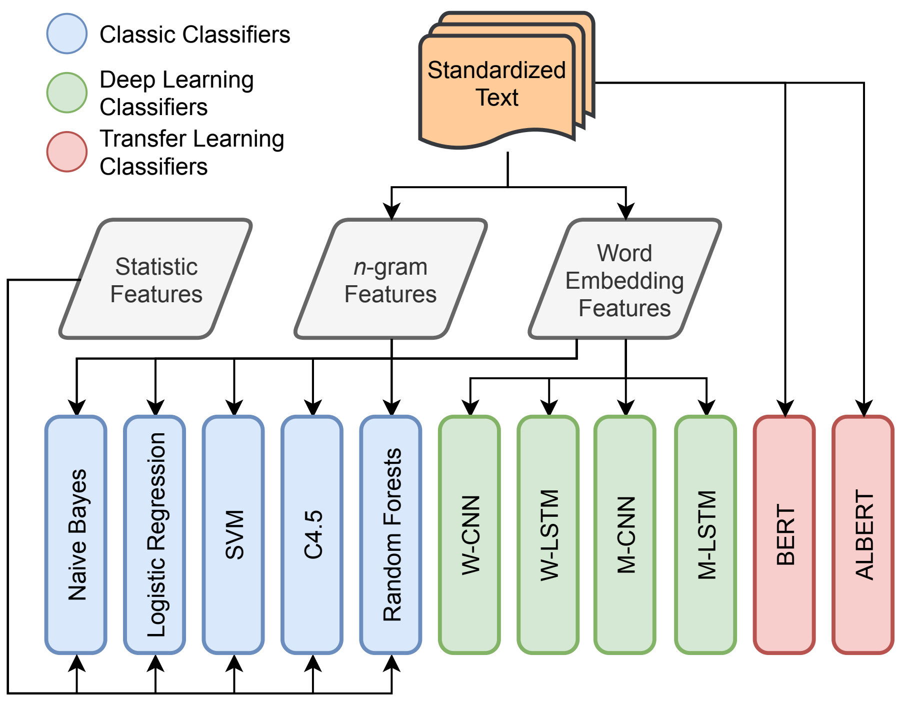
We calculated all the metrics described in Section 2.4, namely kappa (`KPP`), true positive rate (`TPR`), weighted precision (`PRE`), weighted recall (`REC`), weighted F-measure (`F1`), and area under the ROC curve (`AUC`). To keep in line with the existing research on offensive content detection (CHATZAKOU et al., 2017; PAVLOPOULOS; MALAKASIOTIS; ANDROUTSOPOULOS, 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. The number of instances gives the weights in the weighted metrics in the classes. Also, we used ten-fold cross-validation in our experiments. Therefore, the measures reported are the result of the average of the real values obtained for each fold. So, metrics that rely on others, such as `F1`, might not assume the same value that they would get using their equations.

6.1.4 Experimental Procedure

While the statistic feature set was submitted only to the Classic classifiers, the textual features were submitted to all the classifiers in the three categories covered previously. However, the Deep Learning and Transfer Learning classifiers did not use the n-gram features. This proceeding was performed for both `OffVidPT-2` and `OffVidPT-3`. The Transfer Learning classifiers used only the textual features sets after pre-processing. These models do not require removing stop word or punctuation, as they were already pre-trained to use them. The features, classifiers, and datasets combinations amounted to a total of 434 experimental runs, not considering the ensembles. The distribution of features and classifiers is shown in Figure 6.2. We used ten-fold cross-validation in our experiments and averaged the results of the iterations to get to a final result and selected the runs with the best score for `AUC` for each feature set.

Additionally, we created many independent ensemble classifiers (ROKACH, 2010) in an attempt to outperform the results of the other classifiers used in isolation. We used this type of ensemble to keep our study aligned with other works that also used the same technique (PELLE; ALCÂNTARA; MOREIRA, 2018; PETE; L., 2015). Our ensembles were created by combining the best result obtained for each feature set. In other words, there are five features in each ensemble, except for Deep Learning ensemble, since we did not use the statistic feature set in the Deep Learning algorithms. Each ensemble is

Figure 6.2: Architecture employed in the experiments for offensive video detection

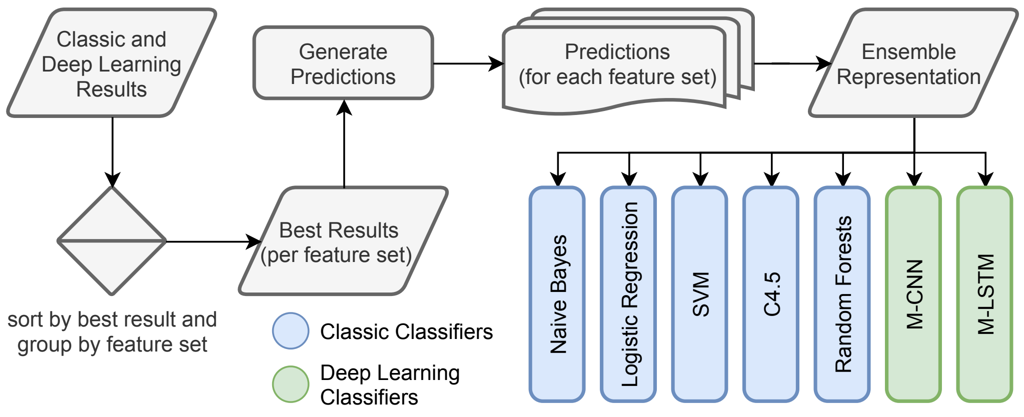


Source: The Authors

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 (`OffVidPT-2` or `OffVidPT-3`).

To create an ensemble, we took the classification results of its group, ranked them by the best `AUC`, and grouped them by feature set. Then, 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. Next, we combined these predictions in an ensemble representation, where each feature is the predictions for a specific feature set. Also, aiming at analyzing the impact of each feature set in the result of the ensemble classification, we created subsets out of the full ensemble representations by excluding one single feature at a time. This procedure, called feature ablation, resulted in a total of 17 ensemble representations, which we submitted to all the five Classic algorithms and two Deep Learning models (M-CNN and M-LSTM), adding 238 runs to our study, amounting to 672 in total. Thus, the input to the classifiers in the ensemble experiments is a combination of predictions generated by other classifiers. We changed the kernel size and instance iterator width of the W-CNN according to the ensemble representation size to enable the classifier to work. All the other configurations in this classifier and the others were kept the same. As in the other experiments, we used ten-fold cross-validation for the ensemble experiments and selected the best one using the arithmetic average of the folds results. Figure 6.3 shows this procedure of processing the classifier results to create the ensemble representations and submission of these representations to the classifiers.

Figure 6.3: Architecture employed in the experiments with the ensemble classifier for offensive video detection


Classic and Deep Learning Results
Generate Predictions
Predictions (for each feature set)
Ensemble Representation
sort by best result and group by feature set
Best Results (per feature set)
Classic Classifiers
Deep Learning Classifiers
Naive Bayes
Logistic Regression
SVM
C4.5
Random Forests
M-CNN
M-LSTM

Source: The Authors

6.2 Results

In this Section, we present the results achieved in our experiments and answer our research questions. As mentioned before, the results are given by the average of the scores obtained by each fold using ten-fold cross-validation. Table 6.7a presents the best results for the experiments using the feature sets for `OffVidPT-2`. The results are grouped by the type of learning algorithm, feature representation, and feature set. Also, the "all" feature set corresponds to the ensemble experiment where the features correspond to the predictions of all feature sets. Table 6.8a presents the results for the experiments with the Classic n-gram ensemble, *i.e.,* ensemble representation built using the best results of experiments using Classic algorithms and n-gram. Table 6.9a reports the best experiment results using the Classic word embedding ensemble instead. Table 6.10a shows the results for the Deep Learning word embedding ensemble, the last one for `OffVidPT-2`. The statistic feature set was included in the ensemble results of Tables 6.8a and 6.9a. Also, the best score achieved for each metric in each table is highlighted.

For the ensemble results tables, we report the best classifiers according to their

AUC. Some classifiers achieved the same AUC, and, for this reason, we present all of them, enabling a comparison according to the other metrics. Also, those tables show the result for the ensemble when the predictions from all feature sets were used (all), and the results for the case when we excluded one of the feature set predictions (all - <excluded feature set>).

The results are presented similarly for OffVidPT-3. Table 6.7b presents the best results grouped by algorithm category, feature representation, and feature set, including the "all" feature set for the ensemble experiments. Tables 6.8b, 6.9b, and 6.10b present the results for the Classic n-gram, Classic word embedding, and Deep Learning word embedding ensembles, respectively. These results show the best classifiers according to the achieved AUC.

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 model to identify such cases and thus compensate for smaller agreement in the annotations.

6.2.1 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 0.80 in AUC and 0.75 in F1 for Classic n-gram ensemble for OffVidPT-3 (Table 6.8b). 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.

6.2.2 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 0.76 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 with all feature sets and 5% when the description feature set was removed (Table 6.8b), the most significant improvement. The other ensembles of all feature sets had a slight decrease. 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 6.7a and 6.7b, 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 embeddings are better with Deep Learning algorithms, outperforming all results for their use in Classic algorithms for `OffVidPT-2` (Table 6.7a) and most of the results for their use with Classic algorithms for `OffVidPT-3` (except for precision for descriptions and `TPR`), as can be seen in Table 6.7b. GloVe was the best word embedding representation for both datasets for most textual feature sets. Wang2Vec ranked second, followed by FastText and Word2Vec. For n-gram representations, character n-gram and word unigram are 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, which was only submitted to the Classic algorithms, did not score close to the best results, but it still outperformed some results achieved using other feature sets (Table 6.7a and 6.7b). Nevertheless, this feature set contributed to improving the scores in the ensemble experiments by 1% for Classic n-gram ensemble and 2% for Classic word embedding ensemble (Tables 6.8a, 6.8b, 6.9a, and 6.9b).

When combined in ensemble representations, the feature sets did not increase the results in all the cases compared to their subgroup (feature set results used to create the ensemble). The simple combination of all feature sets provided a slight increase for `AUC` for Classic n-gram and Deep Learning word embedding ensembles for `OffVidPT-2`, and Classic n-gram ensemble for `OffVidPT-3`. However, Tables 6.8a and 6.8b show that removing the descriptions feature set from the Classic n-gram ensemble for both

OffVidPT-2 and OffVidPT-3 increased the AUC, maybe because the descriptions are long and provided by the user, increasing the chance to hide offensive content. The most significant improvement was observed for OffVidPT-3, where the AUC increased by 5%. As can be seen in Tables 6.9a and 6.9b, removing the tags feature helped to improve the AUC. However, this would still not outperform the AUC achieved by the title, and the tags feature sets alone in Classic word embedding experiments for OffVidPT-2 and OffVidPT-3, respectively.

6.2.3 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 with the Classic n-gram ensemble for OffVidPT-3: 0.80 for "all - desc" and 0.79 for "all - tags" ensemble representations (Table 6.8b). The Transfer Learning algorithms outperformed Classic algorithms when processing 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 other CNN classifier, W-CNN, achieved almost all of the best results when using the textual feature sets separately, as can be seen in Tables 6.7a and 6.7b. 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 CNN core idea is designed 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, the M-LSTM scored the best result for at least one of the feature ablation experiments, and, when it did not achieve the best AUC, it always scored close to the best. We observed this same behavior for M-CNN in the ensemble experiments, alongside with the Naive Bayes classifier.

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 6.7b) but did not outperform Random Forests and Naive Bayes for the other experiments with textual feature sets. On the other hand, the Logistic Regression algorithm was able to achieve some of the best `AUC` results in the ensemble experiments, but usually scoring behind in the other metrics. 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 `OffVidPT-3` (Table 6.7b), 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.

6.3 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, and no tuning was done, which could have contributed to better results.

Table 6.7: Best results by algorithm category, feature representation, and feature set

(a) `OffVidPT-2` dataset

	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**

(b) `OffVidPT-3` dataset

	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

Source: The Authors

Table 6.8: Feature ablation experiment for the Classic algorithms using n-grams

(a) `OffVidPT-2` dataset

Ensemble Repr.	**Best Classifier**	**KPP**	**TPR**	**AUC**	**PRE**	**REC**	**F1**
all	Naive Bayes	0.35	0.59	**0.75**	0.69	0.69	0.68
all - stat	Naive Bayes	0.36	0.66	0.74	0.70	0.69	0.69
all - stat	M-LSTM	**0.38**	**0.73**	0.74	**0.71**	0.69	0.69
all - desc	Naive Bayes	**0.38**	0.69	**0.75**	**0.71**	**0.70**	**0.70**
all - desc	M-CNN	**0.38**	0.70	**0.75**	**0.71**	0.69	0.69
all - tags	Naive Bayes	0.31	0.59	0.72	0.67	0.67	0.66
all - tags	M-CNN	0.32	0.60	0.72	0.68	0.67	0.67
all - tags	M-LSTM	0.33	0.62	0.72	0.68	0.68	0.67
all - titl	Naive Bayes	**0.38**	0.65	0.73	0.70	**0.70**	0.69
all - titl	M-LSMT	0.37	0.66	0.73	0.70	0.69	0.69
all - titl	Logistic Regression	0.33	0.57	0.73	0.68	0.68	0.67
all - tran	Naive Bayes	0.37	0.68	0.74	0.70	0.69	0.69

(b) `OffVidPT-3` dataset

Ensemble Repr.	**Best Classifier**	**KPP**	**TPR**	**AUC**	**PRE**	**REC**	**F1**
all	M-CNN	0.46	**0.85**	0.78	0.76	0.75	0.74
all - stat	M-CNN	0.43	0.82	0.77	0.75	0.74	0.73
all - desc	M-CNN	0.46	0.82	**0.80**	0.76	0.75	0.74
all - tags	M-CNN	**0.48**	0.84	0.79	**0.77**	**0.76**	**0.75**
all - titl	M-LSTM	0.43	0.77	0.77	0.74	0.73	0.73
all - titl	M-CNN	0.39	0.81	0.77	0.72	0.72	0.71
all - tran	Naive Bayes	0.33	0.82	0.70	0.69	0.69	0.68
all - tran	M-LSTM	0.29	0.78	0.70	0.67	0.67	0.66

Source: The Authors

Table 6.9: Feature ablation experiment for the Classic algorithms using word embeddings

(a) `OffVidPT-2` dataset

Ensemble Repr.	Best Classifier	KPP	TPR	AUC	PRE	REC	F1
all	Naive Bayes	0.29	0.52	0.70	0.66	0.66	0.66
all - stat	Naive Bayes	0.31	0.49	0.68	**0.68**	**0.68**	**0.67**
all - desc	Naive Bayes	0.30	0.50	0.67	0.67	0.67	0.66
all - desc	M-LSTM	0.21	0.48	0.67	0.63	0.62	0.62
all - tags	Naive Bayes	0.31	0.51	**0.71**	0.67	**0.68**	0.66
all - titl	M-CNN	0.24	0.44	0.70	0.65	0.65	0.63
all - titl	Naive Bayes	0.31	0.52	0.70	0.67	0.67	**0.67**
all - tran	Naive Bayes	**0.32**	0.53	0.69	**0.68**	**0.68**	**0.67**
all - tran	Logistic Regression	0.25	0.43	0.69	0.65	0.65	0.63
all - tran	M-LSTM	0.30	**0.54**	0.69	0.67	0.67	0.66

(b) `OffVidPT-3` dataset

Ensemble Repr.	Best Classifier	KPP	TPR	AUC	PRE	REC	F1
all	Naive Bayes	**0.35**	0.86	0.71	**0.72**	**0.71**	**0.69**
all	M-LSTM	0.34	0.80	0.71	0.70	0.70	**0.69**
all	Logistic Regression	0.28	0.87	0.71	0.70	0.68	0.66
all - stat	Naive Bayes	0.27	0.85	0.69	0.68	0.68	0.65
all - stat	M-LSTM	0.34	0.76	0.69	0.69	0.69	0.68
all - desc	Logistic Regression	0.30	0.86	0.70	0.70	0.69	0.67
all - desc	M-LSTM	0.30	0.75	0.70	0.68	0.67	0.66
all - desc	Naive Bayes	0.33	0.88	0.70	**0.72**	**0.71**	0.68
all - desc	M-CNN	0.29	0.86	0.70	0.69	0.69	0.67
all - tags	M-CNN	**0.35**	0.87	**0.72**	**0.72**	**0.71**	**0.69**
all - titl	M-LSTM	0.33	0.77	0.71	0.69	0.69	0.68
all - titl	Naive Bayes	0.31	**0.89**	0.71	0.71	0.70	0.67
all - titl	M-CNN	0.30	0.88	0.71	0.70	0.69	0.67
all - tran	M-LSTM	0.28	0.78	0.69	0.67	0.67	0.65
all - tran	Logistic Regression	0.30	**0.89**	0.69	0.71	0.70	0.67
all - tran	Naive Bayes	0.31	0.88	0.69	0.70	0.70	0.67
all - tran	M-CNN	0.31	0.87	0.69	0.70	0.70	0.67

Source: The Authors

Table 6.10: Feature ablation experiment for the Deep Learning algorithms using word embeddings

(a) `OffVidPT-2` dataset

Ensemble Repr.	Best Classifier	KPP	TPR	AUC	PRE	REC	F1
all	M-LSTM	**0.43**	**0.69**	**0.78**	**0.73**	**0.72**	**0.72**
all	M-CNN	0.40	0.65	**0.78**	0.72	0.71	0.71
all	Random Forest	0.37	0.53	**0.78**	0.71	0.71	0.69
all - desc	M-CNN	0.36	0.56	0.77	0.70	0.70	0.69
all - tags	M-LSTM	0.35	0.59	0.75	0.69	0.69	0.68
all - tags	M-CNN	0.33	0.61	0.75	0.68	0.67	0.67
all - titl	M-CNN	0.36	0.59	0.76	0.70	0.69	0.69
all - tran	M-LSTM	0.37	0.59	0.76	0.70	0.70	0.70
all - tran	Naive Bayes	0.39	0.59	0.76	0.71	0.71	0.70

(b) `OffVidPT-3` dataset

Ensemble Repr.	Best Classifier	KPP	TPR	AUC	PRE	REC	F1
all	M-CNN	0.31	0.84	**0.75**	0.69	0.69	0.68
all - desc	Naive Bayes	**0.36**	**0.88**	0.73	**0.72**	**0.72**	**0.70**
all - desc	M-LSTM	0.35	0.73	0.73	0.70	0.69	0.68
all - desc	Random Forest	0.31	0.83	0.73	0.69	0.69	0.67
all - tags	M-LSTM	0.33	0.73	0.71	0.70	0.68	0.67
all - tags	Naive Bayes	0.31	0.87	0.71	0.71	0.70	0.67
all - tags	Random Forest	0.26	0.83	0.71	0.68	0.67	0.64
all - titl	M-CNN	0.31	0.86	0.73	0.70	0.69	0.67
all - tran	M-CNN	0.35	0.84	0.74	0.71	0.70	0.69

Source: The Authors

7 CONCLUSION AND FUTURE WORK

In this work, we investigated the problem of detecting offensive videos. The dissemination of offensive content on the Web reaches many platforms, especially where users can create their content, such as social networks. Instead of keeping a friendly environment, some users use the platforms to publish offensive content, harass other people, and disseminate hate. The posts may be presented in different formats such as images, audio, and video, but the text is the most used one thanks to the diffusion of comment sections in many platforms and websites. Also, the offensive content can be classified in different categories according to the target audience of the attack, frequency of the attacks, and vocabulary employed.

Machine learning using supervised learning is the primary method used to detect offensive content on the Web. Past works used Classic algorithms, such as Naive Bayes, to approach the problem. Later, Deep Learning algorithms started to be widely used in the same task with complex and deep neural networks, such as CNN and LSTM, due to their improvement to earlier results. More recently, a new and promising category of algorithms called Transfer Learning was developed, and researchers started to apply them to the offensive content detection problem. Since the primary input of these algorithms is text, researchers used NLP techniques to turn textual information into meaningful data to the computer. Two methods, widely used across other works, are n-gram of words and characters, and word embedding.

We studied offensive video detection, as previous works limited their approach mostly to comments associated with videos. Our goal was to analyze textual features extracted from the video itself, such as transcription, and other features that are provided by the publisher but are still directly related to the video, such as title, description, and tags. More specifically, we wanted to know it is possible to accurately classify whether a video has offensive content just by analyzing its textual features, which features are the most helpful in detecting offensive content, and which class of algorithms performs better at detecting offensive videos. Moreover, we wanted to analyze how Classic, Deep Learning, and Transfer Learning algorithms would perform with these features. Additionally, we selected Portuguese as the content language to be studied, as most of the works target content in English, leaving Portuguese underrepresented in this type of problem.

We collected and annotated a dataset of 400 videos in Portuguese from YouTube, due to its popularity and content availability in Portuguese. We extracted features sets from the videos and used them in Classic (Naive Bayes, Logistic Regression, SVM, C4.5, and Random Forest), Deep Learning (CNN and LSTM), and Transfer Learning (BERT and ALBERT) algorithms. More specifically, we extracted one statistical feature set and four textual feature sets from the dataset (description, tags, title, and transcription). We also published these features to contribute to this research field by enabling other researchers to reproduce our experiments or evolve them. We extracted and used n-gram and word embedding from the textual feature sets to use as input in the Classic and Deep Learning algorithms. The Transfer Learning algorithms used raw but standardized text as they can process it to create their internal representations. Additionally, we created ensemble-based classifiers in an attempt to improve our results. The quality of the classifiers in our work was measured using the AUC, but we also reported other metrics for analysis.

The results of our experiments show a significant performance for the Deep Learning algorithms, especially for the CNN architectures using word embedding, outperforming the other categories. The Transfer Learning models achieved better AUC than Classic models for transcriptions. However, the best results for the other feature sets using Classic algorithms usually ranked better than the best ones for the same feature sets using Transfer Learning.

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. We also did not find a pattern for the performance of each of these 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 alone. Still, feature ablation experiments in the ensembles showed that some feature sets might play a negative role in the results.

Overall, our best result was 0.80 for AUC and 0.75 for F1, reached with the ensemble-based classifier using Classic algorithms and n-gram. 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.

We performed our study on offensive video detection with a set of classifiers and features. However, other possibilities can be explored in this matter. The dataset, for example, can be expanded with more annotated instances. We believe our scores can be increased by a higher number of cases in the dataset, since they would provide more data for the classifiers, enabling them to extract more information and possibly improve the classification quality. The data source can be expanded to other platforms other than YouTube, such as Facebook, Twitter, Instagram and even Netflix, for example. Also, the dataset can be extended to other languages to compare the results to the ones achieved for Portuguese.

Besides the dataset, the feature sets themselves can be expanded. One possibility could be the union of all textual feature sets to provide a single document for each video. Another possibility could be analyzing the comments videos. Although n-gram and word embedding are the most popular features generated from text, other features used in NLP can be extracted and used by the classifiers. Also, the statistic feature set can be expanded with additional information from the data initially retrieved during the dataset creation. New features can also be extracted from the video itself, like by processing the frames and thumbnails. Additionally, new combinations can be tested for the ensemble-based classification, not limited to subgroups of feature representations and classifiers in our work.

Since we did minimal parameter tuning, future work can also explore new combinations of values for those parameters in an attempt to improve the classification results. The algorithms employed in our study have many parameters that can be fine-tuned, but some techniques can help with this process, such as grid search. This task is time and resource consuming, but it can yield good improvement for the trained models.

Future work may also explore the employment of different classifiers and architectures. New deep neural networks can be created by combining different layers and algorithms to create robust classifiers. Other Transfer Learning classifiers, such as ULMFiT (HOWARD; RUDER, 2018), can also be employed in future work. Different Classic classifiers can be experimented and even combined with methods such as ensemble-based. These tests would provide performance baselines for other studies and comparisons analysis with the other classifiers in our study.

REFERENCES

AGGARWAL, C. C. **Data classification: algorithms and applications**. [S.l.]: CRC press, 2014.

AGGARWAL, P. et al. LTL-UDE at semeval-2019 task 6: BERT and two-vote classification for categorizing offensiveness. In: MAY, J. et al. (Ed.). **Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019**. [S.l.]: Association for Computational Linguistics, 2019. p. 678–682.

ALCANTARA, C. de S.; FEIJO, D. V.; MOREIRA, V. P. Offensive video detection: Dataset and baseline results. In: **12th International Conference on Language Resources and Evaluation**. Marselha: European Language Resources Association (ELRA), 2020. To appear.

ALMEIDA, T. G. et al. Detecting hate, offensive, and regular speech in short comments. In: **Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web**. New York, NY, USA: ACM, 2017. (WebMedia '17), p. 225–228.

ANAND, V. et al. Customized video filtering on youtube. **arXiv preprint arXiv:1911.04013**, 2019.

BASILE, V. et al. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In: MAY, J. et al. (Ed.). **Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019**. [S.l.]: Association for Computational Linguistics, 2019. p. 54–63.

BASILE, V. et al. 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**. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019. p. 54–63.

BIGELOW, J. L.; (KONTOSTATHIS), A. E.; EDWARDS, L. Detecting cyberbullying using latent semantic indexing. In: **Proceedings of the First International Workshop on Computational Methods for CyberSafety**. New York, NY, USA: ACM, 2016. (CyberSafety'16), p. 11–14.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics**, v. 5, p. 135–146, 2017.

BREIMAN, L. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, Oct 2001.

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

BRETSCHNEIDER, U.; WÖHNER, T.; PETERS, R. Detecting online harassment in social networks. In: **Proceedings of the International Conference on Information Systems - Building a Better World through Information Systems, ICIS 2014**. Auckland, New Zealand: Association for Information Systems, 2014. p. 1–14.

BURNAP, P.; WILLIAMS, M. Hate Speech , Machine Classification and Statistical Modelling of Information Flows on Twitter : Interpretation and Communication for Policy Decision Making. **Internet, Policy & Politics**, v. 6, p. 1–18, 2014.

BURNAP PETEAND WILLIAMS, M. L. Us and them: identifying cyber hate on twitter across multiple protected characteristics. **EPJ Data Science**, v. 5, n. 1, p. 11, Mar 2016.

CAPUA, M. D.; NARDO, E. D.; PETROSINO, A. Unsupervised cyber bullying detection in social networks. In: **2016 23rd International Conference on Pattern Recognition (ICPR)**. Cancun, Mexico: IEEE, 2016. p. 432–437.

CHAKRABARTI, S. et al. Data mining curriculum: A proposal (version 1.0). **Intensive Working Group of ACM SIGKDD Curriculum Committee**, v. 140, 2006.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. **Semi-supervised Learning**. [S.l.]: MIT Press, 2006. (Adaptive computation and machine learning).

CHATZAKOU, D. et al. Mean birds: Detecting aggression and bullying on twitter. In: **Proceedings of the 2017 ACM on Web Science Conference**. New York, NY, USA: ACM, 2017. (WebSci '17), p. 13–22.

CHEN, Y. et al. Detecting offensive language in social media to protect adolescent online safety. In: **Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust**. Washington, DC, USA: IEEE Computer Society, 2012. (SOCIALCOM-PASSAT '12), p. 71–80.

DADVAR, M.; JONG, F. de. Cyberbullying detection: A step toward a safer internet yard. In: **Proceedings of the 21st International Conference on World Wide Web**. New York, NY, USA: ACM, 2012. (WWW '12 Companion), p. 121–126.

DADVAR, M.; TRIESCHNIGG, D.; JONG, F. de. Expert knowledge for automatic detection of bullies in social networks. In: TU DELFT. **25th Benelux Conference on Artificial Intelligence, BNAIC 2013**. Delft, Netherlands: TU Delft, 2013. p. 57–64.

DAVIDSON, T. et al. Automated hate speech detection and the problem of offensive language. In: **Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017**. Montréal, Québec, Canada: AAAI Press, 2017. p. 512–515.

DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)**. [S.l.]: Association for Computational Linguistics, 2019. p. 4171–4186.

DJURIC, N. et al. Hate speech detection with comment embeddings. In: **Proceedings of the 24th International Conference on World Wide Web**. New York, NY, USA: ACM, 2015. (WWW '15 Companion), p. 29–30.

DUCHARME, D. N. **Machine Learning for the Automated Identification of Cyberbullying and Cyberharassment**. Thesis (PhD) — University of Rhode Island, 2017.

ERNST, J. et al. Hate beneath the counter speech? a qualitative content analysis of user comments on youtube related to counter speech videos. **Journal for Deradicalization**, n. 10, p. 1–49, 2017.

FIŠER, D.; ERJAVEC, T.; LJUBEŠIĆ, N. Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in slovene. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 46–51.

FLEISS, J. L. Measuring nominal scale agreement among many raters. **Psychological bulletin**, American Psychological Association, v. 76, n. 5, p. 378, 1971.

FORTUNA, P. et al. A hierarchically-labeled portuguese hate speech dataset. In: **Proceedings of the Third Workshop on Abusive Language Online**. [S.l.: s.n.], 2019. p. 94–104.

GÁBOR, K. et al. SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 679–688.

GAMBäCK, B.; SIKDAR, U. K. Using convolutional neural networks to classify hate-speech. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 85–90.

Gangwar, A. et al. 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)**. [S.l.: s.n.], 2017. p. 37–42.

GAO, L.; KUPPERSMITH, A.; HUANG, R. 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)**. Taipei, Taiwan: Asian Federation of Natural Language Processing, 2017. p. 774–782.

GREEVY, E.; SMEATON, A. F. Classifying racist texts using a support vector machine. In: **Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: ACM, 2004. (SIGIR '04), p. 468–469.

Greff, K. et al. Lstm: A search space odyssey. **IEEE Transactions on Neural Networks and Learning Systems**, v. 28, n. 10, p. 2222–2232, Oct 2017.

HARTMANN, N. et al. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In: PAETZOLD, G. H.; PINHEIRO, V. (Ed.). **Proceedings of the 11th Brazilian Symposium in Information and Human Language Technology, STIL 2017, Uberlândia, Brazil, October 2-5, 2017**. [S.l.]: Sociedade Brasileira de Computação, 2017. p. 122–131.

HASANUZZAMAN, M.; DIAS, G.; WAY, A. Demographic word embeddings for racism detection on twitter. In: **Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Taipei, Taiwan: Asian Federation of Natural Language Processing, 2017. p. 926–936.

HEE, C. V. et al. Detection and fine-grained classification of cyberbullying events. In: **Proceedings of the International Conference Recent Advances in Natural Language Processing**. Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, 2015. p. 672–680.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.

HOSSEINMARDI, H. et al. Analyzing labeled cyberbullying incidents on the instagram social network. In: LIU, T.-Y.; SCOLLON, C. N.; ZHU, W. (Ed.). **Social Informatics**. Cham: Springer International Publishing, 2015. p. 49–66.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. In: GUREVYCH, I.; MIYAO, Y. (Ed.). **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers**. [S.l.]: Association for Computational Linguistics, 2018. p. 328–339.

JAPKOWICZ, N.; SHAH, M. (Ed.). **Evaluating Learning Algorithms: A Classification Perspective**. [S.l.]: Cambridge University Press, 2011.

JOULIN, A. et al. Bag of tricks for efficient text classification. In: **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers**. Valencia, Spain: Association for Computational Linguistics, 2017. p. 427–431.

JOYCE, J. Bayes' theorem. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Spring 2019. [S.l.]: Metaphysics Research Lab, Stanford University, 2019.

KANDAKATLA, R. **Identifying Offensive Videos on YouTube**. Dissertation (Master) — Wright State University, 2016.

KENNEDY, G. et al. Technology solutions to combat online harassment. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 73–77.

KIM, Y. Convolutional neural networks for sentence classification. In: MOSCHITTI, A.; PANG, B.; DAELEMANS, W. (Ed.). **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**. [S.l.]: ACL, 2014. p. 1746–1751.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: **Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes**. [S.l.]: Morgan Kaufmann, 1995. p. 1137–1145.

KONTOSTATHIS, A. et al. Detecting cyberbullying: Query terms and techniques. In: **Proceedings of the 5th Annual ACM Web Science Conference**. New York, NY, USA: ACM, 2013. (WebSci '13), p. 195–204.

KWOK, I.; WANG, Y. Locate the hate: Detecting tweets against blacks. In: **Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2013. (AAAI'13), p. 1621–1622.

LAN, Z. et al. ALBERT: A lite BERT for self-supervised learning of language representations. **CoRR**, abs/1909.11942, 2019.

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. **Biometrics**, [Wiley, International Biometric Society], v. 33, n. 1, p. 159–174, 1977. ISSN 0006341X, 15410420. Available from Internet: <http://www.jstor.org/stable/2529310>.

LANG, S. et al. Wekadeeplearning4j: A deep learning package for weka based on deeplearning4j. **Knowledge-Based Systems**, Elsevier, v. 178, p. 48 – 50, 2019.

Lecun, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, Nov 1998.

LIN, H. et al. Dive into deep learning for natural language processing. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts**. Hong Kong, China: Association for Computational Linguistics, 2019.

LING, W. et al. Two/too simple adaptations of Word2Vec for syntax problems. In: **Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. Denver, Colorado: Association for Computational Linguistics, 2015. p. 1299–1304.

MAGU, R.; JOSHI, K.; LUO, J. Detecting the hate code on social media. In: **Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017**. Montréal, Québec, Canada: AAAI Press, 2017. p. 608–611.

MAY, J. et al. (Ed.). **Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019**. [S.l.]: Association for Computational Linguistics, 2019.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: BENGIO, Y.; LECUN, Y. (Ed.). **1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings**. [S.l.: s.n.], 2013.

NAHAR, V.; LI, X.; PANG, C. An effective approach for cyberbullying detection. **Communications in Information Science and Management Engineering**, World Academic Publishing LTD, v. 3, n. 5, p. 238–247, 2013.

NOBATA, C. et al. Abusive language detection in online user content. In: **Proceedings of the 25th International Conference on World Wide Web**. Republic and Canton of

Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016. (WWW ’16), p. 145–153.

NOBLE, W. S. What is a support vector machine? **Nature biotechnology**, Nature Publishing Group, v. 24, n. 12, p. 1565, 2006.

PAPEGNIES, E. et al. Detection of abusive messages in an on-line community. In: **COnférence en Recherche d’Informations et Applications - CORIA 2017, 14th French Information Retrieval Conference**. Marseille, France: ARIA, 2017. p. 153–168.

PARK, J. H.; FUNG, P. One-step and two-step classification for abusive language detection on twitter. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 41–45.

PAVLOPOULOS, J.; MALAKASIOTIS, P.; ANDROUTSOPOULOS, I. Deep learning for user comment moderation. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 25–35.

PELLE, R.; ALCÂNTARA, C.; MOREIRA, V. P. A classifier ensemble for offensive text detection. In: **Proceedings of the 24th Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: ACM, 2018. (WebMedia ’18), p. 237–243.

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

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543.

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

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. [S.l.]: Morgan Kaufmann, 1993.

REYNOLDS, K.; KONTOSTATHIS, A.; EDWARDS, L. Using machine learning to detect cyberbullying. In: **Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops - Volume 02**. Washington, DC, USA: IEEE Computer Society, 2011. (ICMLA ’11), p. 241–244.

ROKACH, L. Ensemble-based classifiers. **Artificial Intelligence Review**, v. 33, n. 1, p. 1–39, Feb 2010.

ROSS, B. et al. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. In: **Proceedings of the 3rd Workshop on Natural Language Processing for Computer-Mediated Communication (NLP4CMC III)**. Bochum, Germany: Bochumer linguistische Arbeitsberichte, 2016. p. 6–9.

SALEEM, H. M. et al. A web of hate: Tackling hateful speech in online social spaces. In: **Proceedings of the LREC 2016 Workshop on Text Analytics for Cybersecurity and Online Safety (TA-COS 2016)**. Portorož, Slovenia: European Language Resources Association (ELRA), 2016. p. 1–10.

SAMGHABADI, N. S. et al. Detecting nastiness in social media. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 63–72.

SCHMIDT, A.; WIEGAND, M. A survey on hate speech detection using natural language processing. In: **Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media**. Valencia, Spain: Association for Computational Linguistics, 2017. p. 1–10.

SOOD, S.; ANTIN, J.; CHURCHILL, E. Profanity use in online communities. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. New York, NY, USA: ACM, 2012. (CHI '12), p. 1481–1490.

SOOD, S. O.; ANTIN, J.; CHURCHILL, E. F. Using crowdsourcing to improve profanity detection. In: **Wisdom of the Crowd, Papers from the 2012 AAAI Spring Symposium**. Palo Alto, California, USA: AAAI, 2012. p. 69–74.

SOOD, S. O.; CHURCHILL, E. F.; ANTIN, J. Automatic identification of personal insults on social news sites. **JASIST**, v. 63, n. 2, p. 270–285, 2012.

SU, H.-P. et al. Rephrasing profanity in chinese text. In: **Proceedings of the First Workshop on Abusive Language Online**. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 18–24.

TING, I.-H. et al. An approach for hate groups detection in facebook. In: UDEN, L. et al. (Ed.). **The 3rd International Workshop on Intelligent Data Analysis and Management**. Dordrecht: Springer Netherlands, 2013. p. 101–106.

TULKENS, S. et al. The automated detection of racist discourse in dutch social media. **Computational Linguistics in the Netherlands Journal**, v. 6, p. 3–20, 12/2016 2016.

VIGNA, F. D. et al. Hate me, hate me not: Hate speech detection on facebook. In: **Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)**. Venice, Italy: CEUR-WS.org, 2017. p. 86–95.

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

WASEEM, Z.; HOVY, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: **Proceedings of the NAACL Student Research Workshop**. San Diego, California: Association for Computational Linguistics, 2016. p. 88–93.

WEISS, K. R.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. **J. Big Data**, v. 3, p. 9, 2016.

WITTEN, I. H. et al. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2016.

WU, Z. et al. BNU-HKBU UIC NLP team 2 at semeval-2019 task 6: Detecting offensive language using BERT model. In: MAY, J. et al. (Ed.). **Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019**. [S.l.]: Association for Computational Linguistics, 2019. p. 551–555.

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

XIANG, G. et al. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In: **Proceedings of the 21st ACM International Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2012. (CIKM '12), p. 1980–1984.

XU, Z.; ZHU, S. Filtering offensive language in online communities using grammatical relations. In: **Proceedings of the Seventh Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, CEAS 2010**. Redmond, Washington, USA: CEAS, 2010. p. 20–29.

YIN, D. et al. Detection of harassment on web 2.0. In: **Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009**. Madrid, Spain: ACM, 2009. v. 2, p. 1–7.

ZAMPIERI, M. et al. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In: **Proceedings of the 13th International Workshop on Semantic Evaluation**. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019. p. 75–86.

ZHANG, H. The optimality of naive bayes. In: BARR, V.; MARKOV, Z. (Ed.). **Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA**. [S.l.]: AAAI Press, 2004. p. 562–567.

ZHONG, H. et al. Content-driven detection of cyberbullying on the instagram social network. In: **Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence**. New York, New York, USA: AAAI Press, 2016. (IJCAI'16), p. 3952–3958.

APPENDIX A — RESUMO EXPANDIDO EM PORTUGUÊS: UM ESTUDO SOBRE DETECÇÃO DE VÍDEO OFENSIVO

Usuários da Web em todo o mundo produzem e publicam grandes volumes de dados de vários tipos, como texto, imagens e vídeos. Para manter um ambiente amigável e respeitoso, as plataformas nas quais esse conteúdo é publicado geralmente impedem os usuários de publicar conteúdo ofensivo e contam com moderadores para filtrar as postagens. No entanto, esse método é insuficiente devido ao alto volume de publicações. A identificação de conteúdo ofensivo pode ser realizada automaticamente usando aprendizado de máquina, mas precisa de um conjunto de dados anotado. Embora existam conjuntos de dados disponíveis para detecção de texto ofensivo, não existem conjuntos de dados para vídeos. Além disso, a maioria dos conjuntos de dados publicados processa dados em inglês, deixando português e outras linguagens com pouca representatividade. Neste trabalho, investigamos o problema da detecção de vídeo ofensivo. Nós montamos, descrevemos e publicamos um conjunto de dados de vídeos em português. Além disso, realizamos experimentos usando classificadores populares de aprendizado de máquina usados na detecção de linguagem ofensiva e relatamos nossas descobertas, juntamente com várias métricas de avaliação. Os resultados dos nossos experimentos mostram um desempenho significante para algoritmos de *Depp Learning*, especialmente para arquiteturas de CNN utilizando *word embedding*. Os modelos de Transfer Learning atingiram melhores AUC do que os modelos Clássicos para transcrições. Entretanto, os melhores resultados para outros conjuntos de atributos utilizando algoritmos Clássicos geralmente foram melhores que os melhores para os mesmos conjuntos de atributos utilizando *Transfer Learning*.

Os melhores resultados obtidos nos nossos experimentos foram 0,80 para AUC e 0,75 para F1 para o ensemble Clássico com n-grama para `OffVidPT-3`. Em tarefas com objetivo similar (classificação binária de conteúdo ofensivo em texto), pesquisadores reportaram resultados similares. Embora os resultados que reportamos não possam ser comparados diretamente com esses dessas tarefas, os resultados delas nos dão um indicativo da qualidade de classificação esperada em um domínio semelhante com o mesmo número de classes. Nesse sentido, nossos resultados estão dentro da faixa dos resultados alcançados nessas tarefas. Essa descoberta pode indicar que, enquanto ainda há espaço para melhoramentos, atributos textuais podem ser utilizados para detecção de vídeo ofensivo.

Em geral, ao observar os melhores resultados para cada conjunto de atributos, nós encontramos valores de `AUC` variando entre 0,70 e 0,76 na maioria dos casos. Quando ensembles são utilizados para combinar as predições de todos os conjuntos de atributos, nós percebemos uma pequena melhora para os resultados dos ensembles Clássicos usando n-grama e *Deep Learning* utilizando *word embedding* para `OffVidPT-2`. Por outro lado, o ensemble Clássico com *word embedding* apresentou uma leve redução no desempenho. Para `OffVidPT-3`, a `AUC` melhorou 3$\%$ para o ensemble Clássico usando n-grama com todos os conjuntos de atributos e 5$\%$ quando o atributo descrição foi removido, a melhora mais significativa. Os outros ensembles de todos os conjuntos de atributos tiveram uma pequena baixa no desempenho. Entretanto, quando analisamos as outras métricas, os ensembles geralmente aumentam seus resultados, o que pode ser desejável em algumas situações.

Observando as representações de n-grama e de *word embedding*, observamos que n-grama apresenta um desempenho um pouco melhor que *word embedding* para todos os conjuntos de atributos e métricas, exceto para o conjunto de atributos de descrição (desconsiderando `AUC` para `OffVidPT-2`). No entanto, *word embedding* é melhor com os algoritmos de *Deep Learning*, superando todos os resultados para uso nos algoritmos Clássicos para `OffVidPT-2` e a maioria dos resultados para uso com algoritmos Clássic para `OffVidPT-3`. GloVe foi a melhor representação de *word embedding* nos dois conjuntos de dados para a maioria dos conjuntos de atributos textuais. Wang2Vec ficou em segundo lugar, seguido por FastText e Word2Vec. Para representações de n-grama, n-grama de caractere e unigrama de palavra foram os mais úteis para a classificação, enquanto n-grama de palavra (`wngram`) não obteve o melhor resultado para nenhum conjunto de atributos textuais.

O conjunto de atributos estatísticos, que foi submetido apenas aos algoritmos Clássicos, não obteve resultados próximos aos melhores resultados, mas ainda assim superou alguns resultados alcançados usando outros conjuntos de atributos. No entanto, esse conjunto de atributos contribuiu para melhorar o desempenho nos experimentos de ensemble em 1$\%$ para o ensemble Clássico utilizando n-grama e 2$\%$ para o ensemble Clássico utilizando *word embedding*.

Quando combinados em representações de ensemble, os conjuntos de atributos não melhoraram os resultados em todos os casos em comparação com seu subgrupo (resultados do conjunto de atributos usados para criar o ensemble). A combinação simples de todos os conjuntos de recursos proporcionou um leve aumento para `AUC` para ensemble Clássico utilizando n-grama e ensemble de *Deep Learning* utilizando *word embedding* para `OffVidPT-2`, e ensemble Clássico usando n-grama para `OffVidPT-3`. No entanto, os resultados mostraram que a remoção das descrições do ensembnle Clássico utilizando n-grama para `OffVidPT-2` e `OffVidPT-3` aumentou a `AUC`, talvez porque as descrições sejam longas e fornecidas pelo usuário, aumentando a chance ocultar conteúdo ofensivo. A melhoria mais significativa foi observada para `OffVidPT-3`, onde a `AUC` aumentou 5%. Os resultados também mostraram que a remoção das tags ajudou a melhorar a `AUC`. No entanto, isso ainda não superou a `AUC` obtida pelo títulos e tags sozinhos nos experimentos de *word embedding* utilizados em algoritmos Clássicos para `OffVidPT-2` e `OffVidPT-3`, respectivamente.

No geral, os modelos de *Deep Learning* e alguns ensembles tiveram os melhores resultados, com os dois mais altos `AUC` alcançados usando o classificador M-CNN com o ensemble Clássico utilizando n-grama para `OffVidPT-3`: 0,80 para "all - desc" e 0,79 para a representação de ensemble "all - tags". Os algoritmos de *Transfer Learning* superaram os algoritmos Clássicos ao processar as transcrições. No entanto, os algoritmos Clássicos pareceram ser capazes de aprender melhor com os conjuntos de atributos de descrição, tags e título do que os algoritmos de *Transfer Learning*. O outro classificador CNN, W-CNN, obteve quase todos os melhores resultados ao usar os conjuntos de atributos de texto separadamente. A única exceção é que o M-LSTM teve um desempenho melhor que o W-CNN ao processar descrições para `OffVidPT-2`. Esse desempenho mostra que, embora a idéia central de CNN tenha sido projetada para processamento de imagem, ela pode superar outros algoritmos Clássicos e de *Deep Learning* quando usada para Processamento de Linguagem Natural (PNL).

O classificador W-LSTM não obteve o melhor resultado para nenhum dos experimentos. O M-LSTM, por outro lado, obteve boa pontuação nos experimentos de ensemble. Para cada representação de ensemble, o M-LSTM obteve o melhor resultado em pelo menos um dos experimentos de ablação de atributos e, quando não alcançou a melhor `AUC`, sempre esteve próximo do melhor resultado. Observamos esse mesmo comportamento para o M-CNN nos experimentos de ensemble, juntamente com o classificador Naive Bayes.

O classificador Random Forest superou o Naive Bayes na maioria dos experimentos com algoritmos Clássicos. Para o conjunto de atributos estatísticos, por exemplo, o Random Forest alcançou os melhores resultados para os dois conjuntos de dados. No entanto, embora Naive Bayes tenha obtido o segundo lugar nos experimentos de conjuntos de atributos textuais, ele produziu muitos dos melhores resultados nos experimentos de ensemble, superando Random Forest nesse caso. O classificador de Regressão Logística foi o melhor para transcrições usando `cngram` para `OffVidPT-3`, mas não superou Random Florests e Naive Bayes para os outros experimentos com conjuntos de atributos textuais. Por outro lado, o algoritmo de Regressão Logística foi capaz de obter alguns dos melhores resultados de `AUC` nos experimentos de ensemble, mas geralmente ficando atrás nas outras métricas. Os algoritmos C4.5 e SVM, por outro lado, não foram tão bons quanto os outros em nossos experimentos.

Ao comparar os classificadores de *Transfer Learning*, o BERT superou a maioria dos resultados alcançados pelo ALBERT. Ambos os classificadores alcançaram a mesma `AUC` e `F1` para transcrições em `OffVidPT-3`, mas o ALBERT obteve melhor precisão e kappa. Em comparação com os classificadores Clássic e de *Deep Learning*, os modelos de *Transfer Learning* obtiveram os melhores resultados de precisão, recall e `F1` para `OffVidPT-2`. Para `OffVidPT-3`, por outro lado, BERT e ALBERT não foram capazes de obter nenhum dos melhores resultados, obtendo uma pontuação baixa em precisão, *recall* e `F1`. A única exceção, nesse caso, é para `AUC`, onde BERT e ALBERT obtiveram resultados próximos dos melhores.