A Unified Deep Learning Architecture for Abuse Detection

Antigoni-Maria Founta
Aristotle University
Thessaloniki, Greece
founanti@csd.auth.gr

Despoina Chatzakou
Aristotle University
Thessaloniki, Greece
deppych@csd.auth.gr

Nicolas Kourtellis
Telefonica Research
Barcelona, Spain
nicolas.kourtellis@telefonica.com

Jeremy Blackburn
University of Alabama
Alabama, USA
blackburn@uab.edu

Athena Vakali
Aristotle University
Thessaloniki, Greece
avakali@csd.auth.gr

Ilias Leontiadis
Telefonica Research
Barcelona, Spain
ilias.leontiadis@telefonica.com

ABSTRACT

Hate speech, offensive language, sexism, racism, and other types of abusive behavior have become a common phenomenon in many online social media platforms. In recent years, such diverse abusive behaviors have been manifesting with increased frequency and levels of intensity, e.g., [9]. Despite social media's efforts to combat online abusive behaviors [23, 33] this problem is still apparent. In fact, up to now, they have entered an arms race with the perpetrators, who constantly change tactics to evade the detection algorithms deployed by these platforms. Such algorithms, not disclosed to the public for obvious reasons, are typically custom-designed and tuned to detect only one specific type of abusive behavior, but usually miss other related behaviors. In the present paper, we study this complex problem by following a more holistic approach, which considers the various aspects of abusive behavior. We focus on Twitter, due to its popularity, and analyze user and textual properties from different angles of abusive posting behavior. We propose a deep learning architecture, which utilizes a wide variety of available metadata, and combines it with automatically-extracted hidden patterns within the text of the tweets, to detect multiple abusive behavioral norms which are highly inter-related. The proposed unified architecture is applied in a seamless and transparent fashion without the need for any change of the architecture but only training a model for each task (i.e., different types of abusive behavior). We test the proposed approach with multiple datasets addressing different abusive behaviors on Twitter. Our results demonstrate high performance across all datasets, with the AUC value to range from 92% to 98%.

KEYWORDS

Abusive Behavior, Hate Speech, Twitter, Deep Learning

ACM Reference Format:
Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2019. A Unified Deep Learning Architecture for Abuse Detection. In *11th ACM Conference on Web Science (WebSci '19), June 30-July 3, 2019, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292522.3326028

WebSci '19, June 30-July 3, 2019, Boston, MA, USA

ACM ISBN 978-1-4503-6202-3/19/06…$15.00
https://doi.org/10.1145/3292522.3326028

1 INTRODUCTION

Social media ubiquity has raised concerns about emerging problematic phenomena such as the intensity of abusive behavior. Unfortunately, this problem is difficult to deal with since it has many "faces" and exhibits complex interactions among social media users. Such multifaceted abusive behavior involves instances of hate speech, offensive, sexist and racist language, aggression, cyberbullying, harassment, and trolling [34, 41]. Each form of abusive behavior has its own characteristics, and manifests differently, depending on the social media objectives, the users participating in it, and the topic's sensitivity. Indeed, popular social media platforms like Twitter and Facebook are not immune to abusive behavior, even though they have devoted substantial resources to deal with it [32].

This type of behavior is harmful both socially, reducing the proclivity and trust of users to the particular online social media platform, as well as from a business perspective. For instance, concerns about racist and sexist attacks on Twitter seem to have impacted a potential sale of the company [18]. Even though social media platforms have increased their measures against abusive behaviors online [37], there is always the need to develop advanced mechanism to be one step ahead of the abusers.

Apart from the social media platforms themselves, the research community has also made attempts at detecting abusive behavior. For example, there have been various works attempting to detect hate speech [2, 12, 40, 42], cyberbullying [6, 11, 16], and abusive behavior in general [7, 10, 27]. Furthermore, various techniques have been applied to detect offensive language [26, 43], and even racism and sexism [20, 22, 24]. However, these solutions are typically custom built and tuned for a specific platform and *type* of abusive behavior, and not generalizable.

Additionally, abusive behavior cannot be assumed just by a "monolithic" consideration of the content (e.g., text of an individual post). Instead, in this work, we follow a more "holistic" approach to consider other facts that may carry important signals and predictors for this type of behavior. Such features can include users' prior posts, social network, popularity, account settings, and even the metadata of posts, to reveal a more global abusive behavior tendency. We study all such user activities as they can help capture different facets of the abusive behavior. We tackle the problem by designing a novel, *unified* deep learning architecture, able to digest and combine any available attributes, to detect abusive behavior. The deep learning approach allows us to capture subtle, hidden commonalities and differences among the various abusive behaviors within the same model, while being careful not to overfit on the available data.

Our method is a global and lightweight solution with respect to computational resources needed, with the capacity to consider the plethora of available (meta)data, to recognize various types of abusive behavior, and without too much feature engineering and model tuning. Even though the feature engineering for every behavior to be analyzed and detected can offer a better understanding about the data and domain at hand, it can be a very laborious process which has no guarantees that it will provide a significantly improved performance, if any. We show that the combination of all available metadata with the proposed training methodology can substantially outperform the state of the art over various datasets, each of which captures a different facet of abusive behavior: i) cyberbullying, ii) hateful, iii) offensive, and iv) sarcasm.

More concretely, we make the following contributions:

- We demonstrate the importance of combining all available (meta)data for detecting abusive behavior. We measure its importance by experimenting and producing superior results, with a deep-learning-based architecture able to combine the available input. To the best of our knowledge, we are the *first to demonstrate the power of such a unified architecture in detecting various facets of abuse* in online social networks.
- We show how naive training methodology fails to make optimal use of heterogeneous inputs. To address this, we implement a training technique that focuses separately on each input by alternating training between them. *This optimization substantially boosts detection capabilities*, as it allows the model to avoid considering only the most dominant features for each task.
- We show that our architecture is *portable* across different forms of abusive behavior, as opposed to previous works which use customized detectors for each type of abuse. We experiment on five datasets covering several forms of abuse, and find that our unified architecture works across all *without the need for any tuning or reconfiguring*. Our architecture can handle the imbalance of the various classes without special tuning, and especially for the minority classes. Our results shed light on how different feature types contribute to abuse detection, and provide evidence that text-only features alone are insufficient to reliably detect generic abusive behavior.
- We demonstrate that our methodology can be easily adapted for the detection of toxic behavior in domains such as online gaming, *without further tuning*.
- We provide our implementation to the community, which will be made available on github.com.

2 BACKGROUND AND RELATED WORK

Abuse detection is an increasingly trending topic over the past few years. Numerous studies have been published, trying to address this problem especially in social networks, and in various forms. Hate speech detection [2, 12, 25, 40, 42], cyberbullying identification [6, 11, 16], and the detection of abusive [7, 10, 27] or offensive language [8, 26, 43], are some of the facets of this problem. Some works try to detect more specific types of hateful behavior, such as racism [22, 24] or sexism [20]. However, as [41] points out, there are many similarities between these subtasks, and scholars tend to group them under "umbrella terms" - like [36] do for hate speech - or use them interchangeably. Yet, major advancements on these tasks are quite new and many of the related studies are preliminary.

Table 1: Comparison of our method with past works. TF: text, UF: user, CF: content, NF: network features.

	Features				ML method used		Platform	
Related Work	**TF**	**UF**	**CF**	**NF**	**Neural Nets**	**Classic**	**Twitter**	**Other**
[2],[21], [15],[28], [13],[44]	x				x		x	
[35]	x		x		x		x	
[30]	x	x			x		x	
[38]	x		x		x	x	x	x
[1]	x				x	x		x
[6]	x	x		x		x	x	
[42]	x	x				x	x	
[10], [31]	x		x			x	x	
[25]	x					x	x	
This work	x	x	x	x	x	x	x	x

Most of previous works use traditional machine learning classifiers, such as logistic regression [10, 42, 43] and support vector machines [40], or ensemble classifiers of such traditional methods [5]. Some studies experiment with deep learning on this task, especially after the major advancements of the last years. Due to the large amount of related research concerning these tasks, we only analyze works that are most relevant with ours in terms of domain and methodology, like in [2, 15, 28]. Table 1 compares our method to those that are most relevant to our problem setting. Under the features category four main types of features are listed, i.e., text-, user-, content-, and network-based. N-grams, term frequency, or word embeddings are commonly used text-based features. The user-based features contain information extracted from a user's profile (e.g., number of posts, accounts age, etc.), while the network-based are related to a user's friendship network (e.g., number of followers and friends). Finally, the content-based features are highly related to a user's behavioral patterns, as for instance the average number of the used hashtags and/or urls in his posts, the average words' and posts' length, or the expressed sentiments/emotions.

Twitter Data Sources. Authors in [2] focus on hate speech detection, and specifically attempt to detect racism and sexism based on various deep learning architectures. These architectures include Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs), and FastText [21], combined with numerous features like TF-IDF and Bag of Words (BoW) vectors. Their LSTM classifier with random embeddings results to significantly improved performance compared to baseline methods. In [44] a CNN with GRU network is used, combined with word embeddings, to detect hate speech on Twitter. The proposed method is tested on various datasets, in order to either discriminate among racism, sexism, and neither (or both), or between hate and non-hate tweets. Compared to existing methods, the authors achieved the highest F1-score.

In [15] authors also use deep learning models to address hate speech on Twitter. Specifically, they proceed with CNNs and feature embeddings, such as one-hot encoded character n-gram vectors and word embeddings. They outperform the baseline in terms of precision and F1 score, but not on recall. Similarly, [28] also uses CNNs with character- and word-level inputs for the same task. However, it investigates two different cases; performing the classification for all three labels (i.e., none, sexist, and racist) at once, or beginning with the detection of 'abusive language' and then further distinguishing

between sexist or racist. The results show that, in general, the two cases can have equally good performance. The deep learning model, though, does not seem to perform as well as traditional machine learning algorithms when it comes to the two-step approach. All previously mentioned works experiment with the dataset that was published in [42] and we also use it to compare the results.

Authors in [13] proceed with Italian text sources extracted from both Twitter and Facebook to detect hate speech. A neural network setup combined with word embeddings is used to show the effect of merging data from different sources. The results show small improvement compared to classifying texts separately. In [35] the focus is on identifying offensive language in German tweets. Two neural network setups are used, where text data (i.e., word embeddings) alone or combined with linguistic features (e.g., # of punctuation, sentiment, etc.) are used on top of an LSTM network. Overall, the combination of different features leads to a higher F1-score. Finally, authors in [30], based on an ensemble LSTM setup and a word-based frequency vectorization, focus on detecting racism, sexism, and neutral Twitter data. Best results are achieved when both text and behavioral (e.g., gender) features are used.

Other Social Media Platforms. Literature on the topic of hate detection on Twitter using deep learning has been sparse. However, there are some works addressing this problem in different online platforms. E.g., the study in [27] deals with Yahoo news comments which have been annotated as abusive or not, by trained Yahoo employees. More specifically, they employ several datasets from comments found on Yahoo! Finance and News. Most of them were annotated by employees of the company, one was crowdsourced using Amazon's Mechanical Turk, and one was provided from [12]. In [12], the authors also classify hate speech on Yahoo comments, using the continuous BOW neural language model to train word and comment representations into 'paragraph embeddings' (named *paragraph2vec*). They use logistic regression for classification.

Nobata et al. [27] compare directly their research with [12], due to their similarities. Except from working on the same task and domain, they also employ similar features, i.e., comment embeddings (named *comment2vec*). However, they treat these embeddings differently, without using deep learning based models. In addition to the embeddings, they construct a number of other features, all derived from the comment's text. For the classification task they use the Vowpal Wabbit's regression model,[1] and they outperform [12] by 0.10 in AUC with 82.6% F1-score. In [38] both traditional and deep neural networks (i.e., CNN, LSTM) are used to detect aggressive behavior on Facebook and Twitter posts, as well as on Hindi-English Code-Mixed texts. The best results are obtained with the CCN setup when combined with both text and content based features. Finally, authors in [1] succeed to detect aggression on Facebook posts with an LSTM classifier combined with word embeddings.

Contributions. This work studies in depth the application of deep learning on the detection of abuse in all its forms with a unified architecture. Departing from the previous works that use mostly textual or custom, task-specific, features, we design a neural network architecture that is able to digest all available input (both text and numerical metadata). Furthermore, training the proposed multi-input network is not straightforward. We introduce an interleaved approach that has only been adapted from image recommendation systems [19], and explain all technical details needed for its repeatability. As far as we know, we are the first to experiment with this architecture on classification of text. To sum up, our work is the first proposed unified technological solution to detect a diverse set of abusive behaviors on platforms like Twitter, while the results show significant improvement over various state-of-the-art methods.

[1] https://github.com/JohnLangford/vowpal_wabbit

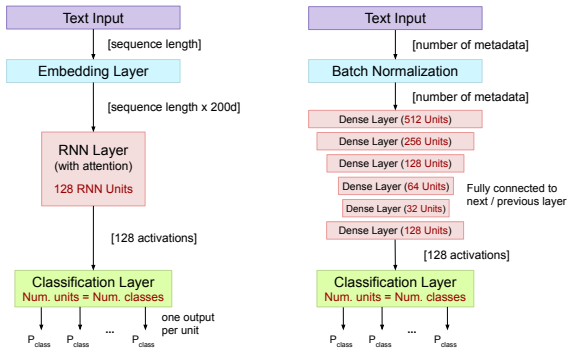

Figure 1: The individual classifiers that are the basis of the combined model. Left: the text-only classifier, Right: the metadata-only classifier.

3 DEEP LEARNING ARCHITECTURES

Our overarching goal is to introduce a novel classifier built on cutting-edge technology such as deep learning, that can detect nuanced forms of abusive behavior. There is a host of literature that tackles this problem and uses a variety of approaches to do so. A key takeaway from the majority of this work is that building a model based on text is outperformed by those that additionally take domain specific features into account. Unfortunately, this is a cumbersome process, with slightly different problems and data sources requiring specially constructed models using different architectures. Ideally, we would prefer to have a *single* model/architecture that incorporates *domain specific* metadata, as well as text content and is performant on a large number of abusive content detection tasks.

To that end, we present a unified classification model for abusive behavior. Our approach is treating i) raw text, and ii) domain specific metadata, separately at first, and later combining them into a single model. Domain-specific data here means features that can be computed on the platform under study, such as popularity of users in the network, reposting counts, etc., depending on what functionalities are available.

In the remainder of this section, we provide details on how our final network is built from its component parts, paying particular attention to the specifics of how to train such a multi-input model. First, we present the two individual classifiers that we later fuse: the text and the metadata classifier. Next, we discuss technical details related to the classifier building, lessons learned and tradeoffs in the design and implementation of the classifier. In particular, we discuss how we can combine the various classifiers, either as an ensemble or a single network. Finally, we present the different ways of effectively training such a multi-path network.

3.1 Text Classification Network

This classifier only considers the raw text as input. There are several choices for the class of neural network to base our classifier on.

We use Recurrent Neural Networks (RNN) since they have proven successful at understanding sequences of words and interpreting their meaning. We experimented with both character- and word-level RNNs. The latter is the most performant across all our datasets.

Text preprocessing: Before feeding any text to the network, we need to transform each sample to a sequence of words. As neural networks are trained in mini-batches, every sample in a batch must have the same sequence length (number of words). Tweets containing more words than the sequence length are trimmed, whereas tweets with fewer words are left-padded with zeros (the model learns they carry no information). Ideally, we want to setup a sequence length that is large enough to contain most text from the samples, but avoids outliers as they waste resources (feeding zeros in the network), making the training of the network slower. Therefore, we take the 95th percentile of length of tweets (with respect to the number of words) in the input corpus as the optimal sequence length. For tweets, this results in sequences of 30 words (in effect, 5% of tweets that contain more than 30 words are truncated). We additionally remove any words that appear only once in the corpus, as they are most likely typos and can result in over-fitting. Once preprocessed, the input text is fed to the network for learning.

Word embedding layer: The first layer of the network performs a word embedding, which maps each word to a high-dimensional vector (typically 25-300 dimensions). Word embedding has proved to be a highly effective technique for text classification tasks, while also reduces the number of training samples required to reach good performance. We settled on using pre-trained word embeddings from GloVe [29], which is constructed on more than 2 billion tweets. We choose the highest dimension embeddings (200) available, as these produce the best results across all abusive behaviors investigated. If a word is not found in the GloVe dataset, we initialize it as random vector. The following layers will just treat it as an individual word and potentially learn its significance in this context.

Recurrent layer: The next layer is an RNN with 128 units (neurons): we tried different sizes, and this gave best results for all datasets. As mentioned previously, RNNs learn sequences of words by updating an internal state. After experimenting with several choices for the RNN architecture (Gated Recurrent Unit or GRUs, Long Short-Term Memory or LSTMs, and Bidirectional RNNs), we find that due to the rather small sequences of length in social media (typically less than 100 words per post, just 30 for Twitter), simple GRUs are performing as well as more complex units. To avoid over-fitting we use a recurrent dropout with $p = 0.5$ (i.e., individual neurons were available for activation with probability 0.5), as it empirically provided the best results across all studied behaviors. Finally, an attention layer [3] can be added as it provides a mechanism for the RNN to "focus" on individual parts of the text that contain information related to the task. Attention is particularly useful to tackle texts that contain longer sequences of words (e.g., forum posts). Empirically, we find this only helps for texts that exceed 100 words and, thus, disable it for any classification task that involves tweets.

Classification layer: Finally, we use a fully connected output layer (a.k.a. Dense layer) with one neuron per class we want to predict, and a softmax activation function to normalize output values between 0 and 1. The output of each neuron at this stage represents the probability of the sample belonging to each respective class. Note that this is the layer that is sliced off when we fuse the text and metadata models into the final combined classifier.

3.2 Metadata Network

The metadata network considers non-sequential data. For example, on Twitter, it might evaluate the number of followers, the location, account age, total number of (posted/favorited/liked) tweets, etc., of a user (see Section 4 for a detailed list).

Metadata preprocessing: Before feeding the data into the neural network, we need to transform any categorical data into numerical, either via enumeration or one-hot encoding, depending on the particulars of the input. Then, each sample is thus represented as a vector of numerical features.

Batch normalization layer: Neural network layers work best when the input data have zero mean and unit variance, as it enables faster learning and higher overall accuracy. Thus, we pass the data through a Batch Normalization layer that takes care of this transformation at each batch.

Dense layers: We use a simple network of several fully connected (dense) layers to learn the metadata. We design our network so that a bottleneck is formed. Such a bottleneck has been shown to result in automatic construction of high-level features [17, 39]. In our implementation, we experimented with multiple architectures and we ended up using 5 layers of size 512, 245, 128, 64, 32, which provide good results across all studied behaviors. On top of this layer, we add an additional (6th) layer which ensures that this network has the same dimensionality as the text-only network; this ends up enhancing performance when we fuse the two networks. Finally, we use *tanh* as activation function, since it works better with standardized numerical data.

Classification layer: As with the test only network, we use one neuron per class with softmax activation.

3.3 Combining the Two Classification Paths

The two classifiers presented above can handle individually either the raw text or the metadata. To build a multi-input classifier we need to combine these two paths. There are two possible ways to perform such a task: i) just use the output of the two classifiers (probabilities of belonging to a given class) as input of a new classifier, or ii) combine the two classifiers on the previous layer that represents the automatically constructed features (Figure 2).

Instead of combining an ensemble of pre-trained classifiers, neural networks allow us to create arbitrary combinations of layers and construct complex architectures that resemble graphs. So, instead of training and then combining two separate classifiers, we can design from the beginning a single architecture that combines both paths before their inputs are squashed into classification probabilities (Figure 2). Therefore, we concatenate the text and metadata networks at their penultimate layer: i) the text path where sequences of raw text are input and 128 activations are produced (one for each RNN unit) and ii) the metadata path where each input produces 128 activations. We can think of this architecture as merging together 128 automatically constructed features from each input and then attempting the final classification task based on this vector.

Contrary to traditional machine learning, this architecture allows us to mix a diverse set of data (sequences of text and discrete metadata) without having to explicitly construct the text features

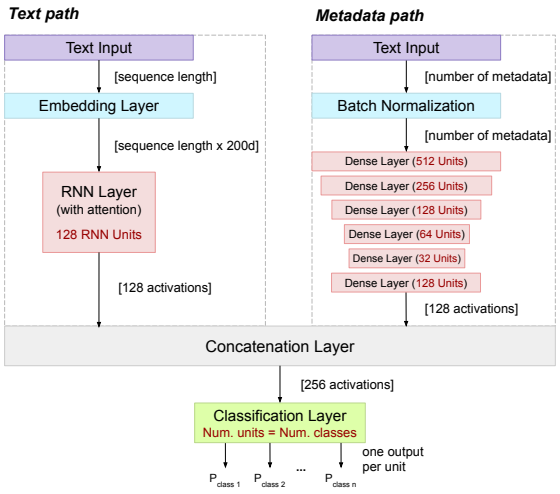

Figure 2: The combined classifier. The output of the two individual paths are concatenated and a classification layer is added over the merged data.

(e.g., TF-IDF vectors). Furthermore, we utilize the power of word embedding that have been pre-trained on much larger datasets.

3.4 Training the Combined Network

While the combined architecture is straightforward, just training the whole network at once is not the most optimal way. In fact, there are several ways to train the combined network: below, we list some of the possible ways and on the Evaluation Section we compare their performance.

Training the entire network at once (Naive Training): The simplest approach is to train the entire network at once; i.e., to treat it as a single classification network with two inputs. However, the performance we achieve from this training technique is suboptimal: the two paths have different convergence rates (i.e., one of the paths might converge faster, and thus dominate any subsequent training epochs). Furthermore, standard backpropagation across the whole network can also induce unpredictable interactions as we allow the weights to be modified at both paths simultaneously.

Transfer learning: We can avoid this problem by *pre-training* the two paths separately and only afterwards join them together to build the architecture of Figure 2. This involves a number of steps:

(1) We pre-train separately the text and the metadata classifiers.
(2) We *remove* the classification layer of each classifier, effectively exposing the activations of their penultimate layer. We treat these as the features that the two pre-trained networks have constructed based on their training.
(3) We freeze the weights of both networks, so no further retraining of their weights is possible.
(4) We add a concatenation layer and a classification layer, effectively transforming the separate models of Figure 1 into the architecture of the combined model (Figure 2).
(5) We train again the combined model. Only the final layer's weights are trainable.

Note: this model resembles an ensemble but with a key difference: the input is not the final class probabilities (num_{class}+num_{class} features) but features learned by the previous layer of the pre-trained models (128+128 features).

Transfer learning with fine tuning (FT): This approach is the same as above, except we do not freeze the weights on the original networks. The practical result of this is that our pre-training serves only to initialize the weights, which the fused network can later adapt when we merge the two paths.

Combined learning with interleaving (Interleaved): As discussed, standard training of the whole network at once may lead to poor performance due to the interaction of updating both data paths at once. The training approaches presented in the previous sections try to mitigate this problem by training the two paths separately and then concatenating the two pre-trained models together. Instead, here we introduce a way that allows us to *train the full network simultaneously while mitigating the aforementioned drawbacks* achieving increased performance.

To do this, we can design our training in a way that, at each mini-batch, data flow through the whole network, but only one of the paths is updated, i.e., we train the two paths in an alternating fashion. E.g., on even-numbered mini-batches the gradient descent only updates the text path whereas on odd-numbered batches the metadata ones. Finally, between epochs we also alternate the paths so both paths get a chance to observe the whole dataset.

To implement this interleaved approach (e.g., in Keras), we initialize two identical models A and B with the architecture shown in Figure 2. However, before compiling the models, we introduce a single difference: the text-path of A is defined as non-trainable ('frozen') and, similarly, the metadata-path on B is also 'frozen.' During training, at each mini-batch we alternate these models:

- If ($batch\_number$ + $epoch\_number$) is even, then use model A (else, use B). Notice, the input will pass through both paths, however, the gradient will only update the weights of a single path.
- Copy the newly updated weights to the unused model. Now both models have equal weights.
- Repeat for next mini-batch.

At the end of this process, we have two identical models, each trained one-path-at-a-time. Our empirical results shown that mini-batches of 64 to 512 samples perform similar on all datasets and we chose 512 as it speeds up training.

This results in a more optimal, balanced network as the gradient is only able to change one path at a time, thus avoiding unwanted interactions. At the same time, the loss function is calculated over the whole, combined, network (notice that the input does pass through the whole network).

The interleaving architecture, originally introduced in [19], used parallel training on two RNNs for multimedia and textual features, and applied for video and product recommendations. However, our work is the first one to introduce it on text classification.

4 DATASET

Here, we first describe the features we extract. Then, we analyze the datasets used in the experiments and how they fit the scope of our analysis. Table 2 summarizes the basic properties (e.g., number of tweets, involved users) and the available metadata per dataset.

Dataset	Tweets	Classes	Users	WV	CF	UF	NF
Cyberbullying	6,091	8.5% Bully 5.5% Aggressive 86% Normal	891	✓	✓	✓	✓
Hateful	16,059	12% Racism 20% Sexism 68% None	1,236	✓	✓	✓	
Offensive	24,783	6% Hate 77% Offensive 17% Neither		✓	✓		
Sarcasm]	61,075	10.5% Sarcastic 89.5% Normal	60,255	✓	✓	✓	
Abusive	85,984	31.7% Abusive 5.7% Hate 62.6% Normal	81,448	✓	✓	✓	✓

Table 2: A descriptive analysis of the datasets with information about: the # of tweets/users, the distribution of the classes, and if we have the correspondent word vector (WV), content- (CF), user- (UF) and network-based (NF) metadata.

4.1 Feature Extraction

Text-based features: Word Vectors (WV). Are representations of words into a vector space (word2vec). As explained earlier, and using the GloVe method [29], the words from tweets are mapped, or embedded, into a high-dimensional vector of 200 dimensions.

Metadata Features. Similar to the work presented in [6] a set of metadata is considered, either content-, user-, or network-based, since they have been proven effective for a similar task. More specifically, these metadata are of three general categories:

Content-based (CF): some common and basic textual data are considered, frequently used on Twitter; namely the amount of hashtags and mentions of other users; how many emoticons exist in the tweet; how many words there are with uppercase letters only; the amount of URLs included. Moreover, tweets' sentiment (i.e., positive/negative score), specific emotions (i.e., anger, disgust, fear, joy, sadness, and surprise), and offensiveness scores are considered.

User-based (UF): for the author level, we extract a few basic metadata regarding his popularity (i.e., number of followers/friends). Also, we consider his activity based on the number of posted and favorited tweets, the subscribed lists, and the age of his account.

Network-based (NF): we analyze a user's network by considering his followers (i.e., someone who follows a user) and friends (i.e., someone who is followed by a user). Based on [6], the considered metadata indicate a user' popularity (i.e., the number of followers and friends, and the ratio of them), the extent to which a user tends to reciprocate the follower connections he receives, the power difference between a user and his mentions, the user's position in his network (i.e., hub, authority, eigenvector and closeness centrality), and his tendency to cluster with others (i.e., clustering coefficient).

4.2 Cyberbullying Dataset

The dataset provided by [6] was collected for the purpose of detecting two instances of abusive behavior on Twitter: cyberbullying and cyberaggression. In addition to a baseline, the authors collected a set of tweets between June and August 2016, using snowball sampling around the GamerGate controversy, which is known to have produced many instances of cyberbullying and cyberaggression. The 9,484 tweets were grouped into 1,303 per user "batches" and labeled via crowdsourced workers into one of four categories: 1) bullying, 2) aggressive, 3) spam, or 4) normal. The authors are careful to differentiate between aggressive and bullying behavior. An aggressor was defined as "someone who posts at least one tweet or retweet with negative meaning, with the intent to harm or insult other users" and a bully was defined as "someone who posts multiple tweets or retweets with negative meaning for the same topic and in a repeated fashion, with the intent to harm or insult other users." The aggressive and bullying labels make up about 8% of the dataset, spam makes up about 1/3, with the remainder normal. For our purposes, we remove the batches labeled as spam, as they can be handled with more specialized techniques [6]. We note that the authors were focused on identifying bullying and aggressive *users*, but we are interested in classifying individual tweets and thus, we break up each batch into individual tweets, each labeled with whatever label their batch was given. In addition to word vectors (WV), this dataset includes all types of metadata, i.e., CF, UF, NF.

4.3 Hateful Dataset

The dataset provided by [42], is focused on racism and sexism. Collected over a two-month period, the authors manually searched for common hateful terms targeting groups, e.g., ethnicity, sexual orientation, gender, religion, etc. The search results were narrowed down to a set of users that seemed to espouse a lot of racist and sexist views. After collection, the data were preprocessed to remove Twitter specific content (e.g., retweets and mentions), punctuation, and all stop words *except* "not." The tweets were labeled as racist or sexist according to a set of criteria, e.g., if the tweet attacks, criticizes, or seeks to silence a minority, if it promotes hate speech or violence, or if there is use of sexist or racial slurs.

Data were manually annotated (not via crowdsourcing) and resulted in $2k$ racist and $3k$ sexist tweets, out of $16k$ total. This dataset is a good benchmark for the present work as it has been used by several similar studies, e.g., [2, 15, 28]. When working with this dataset, except from the word vectors (WV), we also employ both CF and UF metadata. We do not use network-related metadata (NF), due to time limitations (it takes a significant amount of computation and network effort to crawl users' profiles with Twitter API rate limits).

4.4 Offensive Dataset

Tweets characterized as hateful, offensive, or neither are provided by [10]. Here, hate speech is defined as language that is used to express hatred, insult, or to humiliate a targeted group or its members. Offensive language is less clearly defined as speech that uses offensive words, but does not necessarily have offensive meaning. Thus, this dataset makes the distinction that offensive language can be used in context that is not necessarily hateful. Of 80 million tweets they collected, a $25k$ were labeled by crowdsourced workers, with a resulting intercoder-agreement score of 92%. 77% of the tweets were labeled as offensive, with only 6% labeled as hateful. The authors only made the text of the tweets available, and so we have no metadata to use, other than WV.

4.5 Sarcasm Dataset

In the dataset by [31], tweets are characterized as sarcastic or non-sarcastic. Sarcasm, in this work, is defined as 'a way of using words that are the opposite of what you mean in order to be unpleasant to somebody or to make fun of them.' In some online settings such as Twitter or Facebook, with not enough context on the topic of discussion or interest to be civil, sarcasm can be considered impolite and even aggressive behavior. Though this dataset is slightly different from the rest, considering the task at hand, we believe it can bring a useful dimension to the plurality and complexity of abusive behavior, and can inform our method to detect such language.

The data collection was conducted based on self-described users' annotations. Specifically, the authors collected only tweets that contained the hashtags #sarcasm and #not and filtered out tweets that did not contain these hashtags at the end of them, to eliminate tweets that referred to sarcasm but were not sarcastic. They also removed non-english tweets, retweets, tweets with less than three words, and tweets that contained mentions or URLs.

The final dataset consists of almost $91k$ tweets, where 10% are sarcastic. Since not all data were still publicly available through Twitter API, we ended up with $60k$ - preserving the portion of sarcastic tweets. Finally, during the classification, we removed the #sarcasm and #not hashtags. Similar to the Hateful dataset, we employ WV, CF and UF, but no NF. We use the original highly imbalanced dataset since it adapts better to real cases. Hence, we compare our classification performance with the imbalanced results of the baseline.

4.6 Abusive Dataset

In the abusive dataset provided by [14] tweets are characterized as abusive, hateful, spam, and normal. To conclude to the aforementioned inappropriate speech categories, the authors initially proceeded with a series of annotation rounds where different types of abusive behaviors were considered, i.e., offensive, abusive, hateful speech, aggressive, bullying, spam, and normal. Then, an exploratory study took place, to identify the most representative labels related to the types of abusive content. Based on statistical analysis the authors either merged or removed a set of labels to conclude to the most representative set.

Overall, $100k$ tweets were characterized using the Figure Eight platform. Similar to the Cyberbullying dataset, the spam-related tweets were removed to proceed with the following analysis. For this dataset, apart from the WV, we also used all types of metadata, i.e., CF, UF, and NF. For the NF only the number of followers, friends, and the ratio of them was used, since the users' two hop friendship network was not available.

5 EVALUATION

Here, we describe in detail our experimental setup and results while testing the performance of our method on the different datasets used. All results shown here are based on 10-fold cross validation.

5.1 Experimental Setup

For our implementation we use Keras[2] with Theano[3] as back-end for the deep learning models implementation. We use the functional API to implement our multi-input single-output model and run the experiments on a server equipped with three Tesla K40c GPUs.

In terms of training, we use *categorical cross-entropy* as loss function and *Adam* as the optimization function. A maximum of 100 epochs is allowed, but we also employ a separate validation set to perform *early stopping*: training is interrupted if the validation loss did not drop in 10 consecutive epochs and the weights of the best epoch are restored.

It is important to notice that the same model (with the same architecture, number of layers, number of units and parameters) is used for all datasets, as we want to demonstrate the performance of this architecture across different tasks. The performance of the algorithm might increase even further if the parameters are tuned specifically for each task (e.g., using a larger network when there are more training samples). Overall, the model, excluding the pre-trained word embeddings, contains approximately $250k$ trainable parameters (i.e., weights).

Finally, except from each dataset's state-of-the-art, we also compare our results with a basic Naive Bayes model, using the TF-IDF weights for each tweet. For this baseline, we only use the raw text. First, we perform some basic preprocessing of the data; we convert all characters to lowercase and remove all stop words for 14 frequently spoken languages, as well as some twitter-specific stop words. Finally, we tokenize the tweet based on some Twitter-specific markers (hashtags, URLs, and mentions) and punctuation. Afterwards, we experiment with both Porter and Snowball stemmers, lemmatization, keeping the most frequent words and the combinations of all the previously mentioned. We find that the most efficient step is keeping only the most frequent words. We also experiment on the amount of frequent words we need to keep and find that the best results are yielded using the top $10k$ words.

5.2 Experimental Results

In this section we present the classification performance of the proposed methodology on the five datasets. Next, we examine which are the inputs that contribute the most. Finally, we discuss how the different training strategies affect the classification performance.

Training methodology: We apply the same model over all five datasets and the results of AUC, Accuracy, Precision, Recall, and F1-score are summarized in Table 3. Here, we test the training methods discussed earlier to choose the best method to compare with the state-of-art. Firstly, we observe that training with the whole network at once (naive training) results to suboptimal performance (e.g., AUC of 0.94 in the cyberbullying dataset). The reason is that allowing the gradient descent to update both paths simultaneously might result in unwanted interactions between the two. For example, one path might converge faster than the other, dominating in the decisions. In fact, when we examine the standalone classifiers, we observe that the text classifier requires 25-40 epochs to converge whereas the metadata classifier only requires 7-12. By training the whole network together, the metadata side can start overfitting.

A step towards the right direction is to train each path separately, as individual classifiers, and then transfer the constructed features to a new classifier (transfer learning). This method slightly improves the results as it reduces the interactions between the two paths. Notice that, due to the fact that most of the network has been already trained, the additional layer converges after just 3-5 epochs.

[2]https://keras.io/
[3]http://deeplearning.net/software/theano/

	AUC	Acc.	Prec.	Rec.	F1
Cyberbullying Dataset (3 classes)					
DL-Baseline Naive Bayes	0.73	0.88	0.88	0.88	0.88
Chatzakou et al. 2017	0.91	0.91	0.90	0.92	0.91
DL-Metadata only	0.93	0.88	0.91	0.88	0.89
DL-Text only	0.92	0.89	0.91	0.89	0.89
DL-Text & Metadata (Naive Train.)	0.94	0.89	0.90	0.90	0.90
DL-Text & Metadata (Tran. Lear.)	0.95	0.90	0.92	0.90	0.90
DL-Text & Metadata (Tran. Lear. FT)	0.95	0.90	0.91	0.90	0.91
DL-Text & Metadata (Interleaved)	0.96	0.92	0.93	0.92	0.93
Hateful Dataset					
Baseline Naive Bayes	0.79	0.81	0.81	0.81	0.81
Waseem and Hovy 2016	-	-	0.74	0.73	0.78
Badjatiya et al. 2017	-	-	0.93	0.93	0.93
Pitsilis et al. 2018	-	-	0.93	0.93	0.93
DL-Metadata only	0.91	0.74	0.81	0.74	0.76
DL-Text only	0.93	0.83	0.84	0.83	0.83
DL-Text & Metadata (Naive Train.)	0.93	0.85	0.86	0.86	0.86
DL-Text & Metadata (Tran. Lear.)	0.95	0.85	0.86	0.85	0.85
DL-Text & Metadata (Tran. Lear. FT)	0.95	0.86	0.87	0.86	0.86
DL-Text & Metadata (Interleaved)	0.96	0.87	0.88	0.87	0.87
Offensive Dataset					
Baseline Naive Bayes	0.71	0.87	0.84	0.87	0.85
Davidson et al. 2017	0.87	0.89	0.91	0.9	0.9
DL-Metadata only	0.75	0.61	0.80	0.61	0.66
DL-Text only	0.91	0.87	0.89	0.87	0.88
DL-Text & Metadata (Naive Train.)	0.90	0.87	0.89	0.87	0.88
DL-Text & Metadata (Tran. Lear.)	0.91	0.87	0.89	0.87	0.88
DL-Text & Metadata (Tran. Lear. FT)	0.90	0.87	0.89	0.87	0.88
DL-Text & Metadata (Interleaved)	0.92	0.90	0.89	0.89	0.89
Sarcasm Dataset					
Baseline Naive Bayes	0.66	0.90	0.89	0.9	0.89
Rajadesingan, Zafarani, and Liu 2015	0.7	0.93	-	-	-
DL-Metadata only	0.96	0.92	0.94	0.92	0.92
DL-Text only	0.81	0.89	0.89	0.89	0.89
DL-Text & Metadata (Naive Train.)	0.97	0.96	0.96	0.96	0.96
DL-Text & Metadata (Tran. Lear.)	0.97	0.95	0.95	0.95	0.95
DL-Text & Metadata (Tran. Lear. FT)	0.97	0.95	0.95	0.95	0.95
DL-Text & Metadata (Interleaved)	0.98	0.97	0.96	0.97	0.97
Abusive Dataset					
Baseline Naive Bayes	0.80	0.39	0.66	0.39	0.31
DL-Text & Metadata (Interleaved)	0.93	0.84	0.85	0.85	0.85

Table 3: Final results of the baselines and our experiments, for each one of the datasets.

Finally, by employing alternate training (interleaved), we further improve the predictive power of the resulting model reaching 0.96 AUC in the cyberbullying. This shows that multi-input models such as this can benefit from alternate training. The reason is that this methodology avoids any interactions that might result when weights are updated simultaneously in both paths.

Classification performance: Across all datasets and abusive behaviors, the proposed classifier (Interleaved) outperforms in almost all of the cases both the baseline and the state-of-art, as reported in recent publications, for two reasons. First, the proposed approach that combines the raw text and the metadata achieves notably higher performance when compared to the ones using a single set of attributes, as it takes advantage of the additional information from the users' profile and network. This is consistent across all five datasets. Second, the word embeddings allow us to transfer features that were constructed over a billion of tweets, and it enables us to model complex tasks with fewer samples (such as this one).

Looking at the five datasets, on the cyberbullying dataset we observe that using a single set of attributes (text or metadata) we achieve better AUC (0.92 or 0.93, respectively) but worse accuracy (0.89 or 0.88, respectively) than the method proposed in the state-of-the-art (AUC 0.91, accuracy 0.91) [6]. However, the interleaved training of our model substantially outperforms the baseline (AUC 0.96, accuracy 0.92). Having said that, we need to mention here that the comparison with this dataset is not direct. The results reported on [6] are on user-level, while ours are on tweet-level. Therefore, while we can get an understanding on how well their data can be classified with our algorithm, we cannot parallelize the two cases. Nevertheless, the results we achieve on tweet-level are very high, which shows that our model distinguishes very well between the classes, regardless of the comparison.

On the sarcasm dataset, we largely outperform the previous results, as the existing work did not consider metadata. Specifically, we reach an AUC of 0.98 compared to 0.7 of the existing methodology, as in this case the text is not carrying significant information to detect if a tweet is sarcastic. However, the remaining metadata (user, network, content-level metadata) do reveal such information. In case of the hateful dataset, there was no AUC reported in the already existing works, but concerning the precision and recall values we are falling behind for 5% and 6%, respectively. When both text and metadata are combined we reach a precision and recall of 0.88 and 0.87, respectively, when compared to 0.81 of the baseline.

In the offensive dataset the interleaved model is able to reach an AUC of 0.92. Here, even though the precision and recall are similar to those presented in [10], our AUC and accuracy scores still outperform. This is due to the fact that we could not find any user or network metadata related to this dataset and, therefore, our classifier is only using the raw text and the content-based metadata.

Large-scale performance: We run the proposed method on the large-scale abusive dataset to evaluate its performance when executed on larger samples of tweets. Since we are the first that build upon such a dataset, we only compare the interleaved approach with a baseline algorithm, i.e., Naive Bayes. The overall AUC achieved is 0.93, which is significantly higher than the 0.80 achieved with the simple baseline, while also the precision and recall values of 0.85 highlight the efficiency of the proposed approach when executed on larger datasets, and in comparison to the baseline values of 0.31-0.66 for the same performance metrics.

Metadata importance: As described earlier, we use a number of metadata features extracted from the tweets and their authors, namely content-based, user-based, and network-based. These metadata play an important role on the improvement of the performance. When combined with the raw text, they substantially increase all the metrics. However, not all of them have the same impact on the performance (some are more essential than others). In order to determine how each one of these metadata affects our model, we experiment with the cyberbullying dataset and calculate their importance. The results on the AUC are presented in Table 4. We chose this dataset as we have all groups of metadata available (user, content, network, text) and, therefore, it is possible to examine how each of them contributes to the model. The results for other datasets are following similar trends and are omitted due to limited space.

Firstly, by examining individual metadata, we observe that models which are built with individual metadata result in the poorest performance. For example, network-level metadata are the least

Metadata Features	Acronym	AUC
Network Only	NF	0.641
Content Only	CF	0.799
User Only	UF	0.806
User & Content	UF+CF	0.887
Network & Content	NF+CF	0.908
Text Only	WV	0.915
User & Network	UF+NF	0.915
All-metadata Only	CF+UF+NF	0.923
Text & Content	WV+CF	0.930
Text & Network	WV+NF	0.931
Text & User & Content	WV+UF+CF	0.933
Text & Network & Content	WV+NF+CF	0.936
Text & User	WV+UF	0.938
Text & User & Network	WV+UF+NF	0.955
All	WV+CF+UF+NF	0.961

Table 4: Metadata Importance. The values are obtained using the Cyberbullying dataset.

descriptive (AUC of just 0.64) whereas content- and user-based are slightly better (AUC 0.8). By far, raw text is the best feature for this task, as it can lead to a model with particularly higher AUC of 0.91.

Furthermore, combining two or more metadata classes together (user, network, content) does increase performance. This indicates that the information provided is not overlapping and it all adds to better performance. When all the metadata are combined, we reach an AUC of 0.923 which is higher than any other metadata combination. Moreover, using all metadata does result in better classification when compared to just using the text, showing that these metadata carry at least as much predictive power as text does.

Nevertheless, the strongest models were built when text is combined with metadata showing how much the raw text contributes in this classification task. For example, just combining text with user-level metadata is enough to reach an impressive performance (0.94 AUC). Adding network data bumps the performance to 0.955.

Finally, the best performance is reached when all attributes are used. This also demonstrates the fact that the metadata information does not overlap the information that can be extracted from raw text, and this is why the proposed model can be quite powerful and outperforms the state-of-art models for these tasks.

Regarding the network performance, text was by far the most important factor decelerating the training of the whole model. Due to the content path being an RNN, training was eminently more slow than the case of experiments where this path was not used. All other metadata have been two orders of magnitude faster than text, without any significant differences between them. Therefore, there is no tradeoff decisions to be made on whether or not some of the metadata should be excluded for slowing down training.

6 GENERALIZING TO OTHER PLATFORMS: TOXIC BEHAVIOR IN ONLINE GAMING

Though in this work we primarily focus on Twitter to demonstrate how our unified approach works with the same set of features, the same methodology can be applied to other domains *with no modifications*. To demonstrate this, we run the same architecture over a dataset from a completely different domain. We acquired the dataset from the authors of [4] who built a classifier to detect toxic behavior in an online video game. Their dataset is collected from a crowdsourced system that presents reviewers with instances of millions of matches, where toxic behavior is potentially exhibited. The match data include a variety of details such as the full in-game chat logs, players' in-game performance, the most common reason the match was reported for, the outcome of the match, etc. Matches are labeled for either *pardon* or *punish* by a jury of other players who cast votes in either direction.

From the 1 million individual matches provided to us, we extracted a set of features. Like the datasets used earlier in this work, we extract the chat logs of each potentially offending player. We also extract a set of domain specific metadata features (e.g., features that describe the offenders' performance, as well as the performance of other players, the outcome of the game, how many reports the match received, the most common report type, etc.). Each match is also labeled with the final decision of the crowdsourced worker (either, pardon or punish).

Even though, at a high level, this dataset is structured similarly to the five datasets presented earlier (i.e., it is divided into text based and domain specific categories), there are important differences. First, the text is *much* larger (on average, offenders use 2, 500 words per match compared to just 30 words per tweet). Next, the domain specific nature of the metadata does not really have an analogue in the Twitter datasets we used. Finally, the language used in the chat logs themselves, while English, is littered with domain specific jargon. Thus, applying our architecture to this dataset makes a strong case for its portability to different domains.

In [4], the authors evaluated several sub-tasks, with varying degrees of difficulty. The first was the general problem of predicting whether a player will be pardoned or punished, where their best model had an AUC of 0.80. They also experimented with trying to predict only overwhelming decisions, an easier problem, and achieve AUCs of 0.88 and 0.75 for overwhelming pardon and punish decisions, respectively.

We tackled the more general problem by running the dataset through the model presented in our Architecture Section. We enabled an attention layer to deal with the length of the text, however, no other changes were made to the architecture. While we expected *reasonable* performance, we achieved an accuracy of 0.93 and an AUC of 0.89, beating the performance of even the easiest task presented in [4]. These results provide a strong indication that our architecture is suitable for finding abusive behavior in a wide variety of domains.

7 SUMMARY

Unified deep learning classifier is possible: In this work, we built and applied the exact same deep learning model architecture in all five datasets and demonstrated that it can efficiently handle each type of abusive behavior. While fine-tuning the classifier parameters for each dataset can squeeze some more performance, the proposed methodology does beat the current state of the art (in almost all cases) in each behavior detection. Importantly, our architecture can handle the imbalance of the various classes without special tuning, and especially for the minority classes.

All inputs help: We demonstrated how each of the attributes (text, user, network, content) contribute in each task, i.e., identification of specific type of abusive behavior. Our proposed architecture can seamlessly combine this input into a single classification model, without particular tuning.

Training methodology: Training a multi-input network is not straightforward. We introduced a methodology that alternates training between the two input paths to further increase performance in all datasets tested. We compared the proposed training paradigm with various other possible training methodologies (ensemble, feature transfer, concurrent training) and show that it can substantially outperform them.

Flexible to other data: In this paper, we show the ability of our approach to combine two different paths: text and metadata. However, one can simply concatenate more input paths to the architecture. For example, in an image classification problem, a CNN-based network can be used to extract image features and it could be joined with text information (tags and user comments) and image metadata (time and location taken, how many pictures the user has taken, the uploader's social network, etc.). Similarly, in an audio classification task, an audio path can be merged with text and metadata. We leave this exploration as future work.

Generalizing to other platforms: Finally, we showed that our proposed architecture can be easily applied, in a plug-and-play fashion, to detect abusive behavior in other online domains beyond Twitter. As an example, we presented results on detecting toxic behavior in an online gaming network, with superior performance over the state of art. We leave further explorations as future work.

REFERENCES

[1] Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of TRAC-2018*.

[2] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *26th ACM WWW Companion*.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] Jeremy Blackburn and Haewoon Kwak. 2014. STFU NOOB!: Predicting Crowdsourced Decisions on Toxic Behavior in Online Games. In *23rd ACM WWW*.

[5] Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet* 7, 2 (2015), 223–242.

[6] Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean Birds: Detecting Aggression and Bullying on Twitter. In *9th ACM WebScience*.

[7] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *IEEE PASSAT & SocialCom*.

[8] Isobelle Clarke and Jack Grieve. 2017. Dimensions of Abusive Language on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*.

[9] Sam Cook. 2018. Cyberbullying facts and statistics for 2016-2018. https://www.comparitech.com/internet-providers/cyberbullying-statistics/.

[10] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *ICWSM*.

[11] Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of Textual Cyberbullying. *The Social Mobile Web* 11, 02 (2011).

[12] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *24th ACM WWW*.

[13] Paula Fortuna, Ilaria Bonavita, and Sérgio Nunes. [n. d.]. Merging datasets for hate speech classification in Italian. ([n. d.]).

[14] Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. In *12th AAAI ICWSM*.

[15] Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-Speech. In *1st Workshop on Abusive Language Online*.

[16] Hariani and Imam Riadi. 2017. Detection Of Cyberbullying On Social Media Using Data Mining Techniques. *International Journal of Computer Science and Information Security* 15, 3 (2017), 244.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[18] Alex Hern. 2016. Did trolls cost Twitter $3.5bn and its sale? https://www.theguardian.com/technology/2016/oct/18/did-trolls-cost-twitter-35bn.

[19] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *10th ACM RecSys*.

[20] Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? Analysis and classification of ambivalent sexism using twitter data. In *2nd Workshop on NLP and Computational Social Science*.

[21] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[22] Irene Kwok and Yuzhou Wang. 2013. Locate the Hate: Detecting Tweets against Blacks.

[23] Daniel Lowd. 2017. Can Facebook use AI to fight online abuse? http://theconversation.com/can-facebook-use-ai-to-fight-online-abuse-95203.

[24] Estefanía Lozano, Jorge Cedeño, Galo Castillo, Fabricio Layedra, Henry Lasso, and Carmen Vaca. 2017. Requiem for online harassers: Identifying racism from political tweets. In *4th IEEE Conference on eDemocracy & eGovernment (ICEDEG)*.

[25] Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence* 30, 2 (2018), 187–202.

[26] Yashar Mehdad and Joel R Tetreault. 2016. Do Characters Abuse More Than Words?. In *SIGDIAL*.

[27] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *25th ACM WWW Companion*.

[28] Ji Ho Park and Pascale Fung. 2017. One-step and Two-step Classification for Abusive Language Detection on Twitter. *arXiv preprint arXiv:1706.01206* (2017).

[29] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

[30] Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Effective hate-speech detection in Twitter data using recurrent neural networks. *Applied Intelligence* 48, 12 (2018), 4730–4742.

[31] Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *8th ACM WSDM*.

[32] Matthew Rozsa. 2016. Twitter trolls are now abusing the company?s bottom line. https://www.salon.com/2016/10/19/twitter-trolls-are-now-abusing-the-companys-bottom-line/.

[33] Twitter Safety. 2017. Enforcing new rules to reduce hateful conduct and abusive behavior. https://blog.twitter.com/official/en_us/topics/company/2017/safetypoliciesdec2017.html.

[34] Huascar Sanchez and Shreyas Kumar. 2011. Twitter bullying detection. *NSDI*.

[35] Johannes Schäfer. 2018. HIIwiStJS at GermEval-2018: Integrating Linguistic Features in a Neural Network for the Identification of Offensive Language in Microposts. *Austrian Academy of Sciences, Vienna September 21, 2018* (2018).

[36] Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *5th SocialNLP*.

[37] Elizabeth Schulze. 2019. EU says Facebook, Google and Twitter are getting faster at removing hate speech online. goo.gl/XPQzGC.

[38] Vinay Singh, Aman Varshney, Syed Sarfaraz Akhtar, Deepanshu Vijay, and Manish Shrivastava. 2018. Aggression Detection on Social Media Text Using Deep Neural Networks. In *Proceedings of ALW2*.

[39] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*.

[40] William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *2nd Workshop on Language in Social Media*.

[41] Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. *arXiv preprint arXiv:1705.09899* (2017).

[42] Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter.. In *SRW@ HLT-NAACL*.

[43] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *21st ACM CIKM*.

[44] Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *European Semantic Web Conference*. Springer.

8 ACKNOWLEDGMENTS

The authors acknowledge research funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie ENCASE project, Grant Agreement No. 691025.