Six Things I Hate About You (in Italian) and Six Classification Strategies to More and More Effectively Find Them*

Tiziano Fagni, Leonardo Nizzoli, Marinella Petrocchi, and Maurizio Tesconi

National Research Council (CNR), Institute of Informatics and Telematics (IIT)
via G. Moruzzi 1, 56124, Pisa, Italy
{tiziano.fagni, leonardo.nizzoli, marinella.petrocchi,
maurizio.tesconi}@iit.cnr.it

Abstract. While favouring communications and easing information sharing, Online Social Networks are increasingly used to launch harmful campaigns against specific groups and individuals. Although providers struggle to keep pace by manually removing hate content published on their platforms, recent research efforts rely on automatic text classification techniques, whose performances are usually measured on annotated corpora. In this work, we propose six distinct machine learning classification strategies: three based on conventional machine learning approaches, three based on neural networks. The latter are able to process texts almost from scratch, avoiding the need of i) NLP tools specialised for a specific language, ii) the phase of time-consuming feature engineering, and iii) the high computational cost usually derived from processing a huge amount of features. Thus, the main goal of the paper is to investigate whether it is possible to rely on neural networks and to achieve performance results at least comparable with those of NLP-based classifiers. The performances of the six configurations are evaluated over an annotated dataset consisting of 4,000 Italian tweets and 4,000 Italian Facebook comments. By comparing the classification results, we demonstrate that relying on deep learning techniques for hate speech detection is more than encouraging. In particular, a deep learning model, based on an ensemble approach, obtains a F1 score of 0.786 on the Twitter data and 0.775 on the Facebook ones, the best results, compared to the ones obtained with the other tested configurations.

1 Introduction

The more and more massive usage of Online Social Networks (OSNs) leads undeniable benefits, among others the opportunity for users to easily interact for

*Research partly supported by the EU H2020 Program under the schemes INFRAIA-1-2014-2015: Research Infrastructures grant agreement #654024 *SoBigData: Social Mining & Big Data Ecosystem* and MSCA-ITN-2015-ETN grant agreement #675320 *European Network of Excellence in Cybersecurity (NECS)*.

a myriad of goals, which range from planning social events to engaging in commercial transactions. Unfortunately, since years social platforms also represent a fertile ground for ill-intentioned people, whose goals spam from maliciously influencing the public opinion, by diffusing polarized content on important societal topics – like politics, terrorism, immigration – (see, e.g., [6, 9, 27]) to stalking, cyberpranking, and slyly acquiring sensitive information by circumventing the most defenseless users[1].

Furthermore, the enormous degree of freedom for knowledge creation and sharing on OSNs allows the publication of content promoting '*violence or hatred against individuals or groups based on certain attributes, such as: race or ethnic origin, religion, disability, gender, age, veteran status, sexual orientation and gender identity*'[2]. Such violent content is usually referred as *hate speech*. Although the most popular social networks managers announced in their policies to strive to fight against hate speech [21], current solutions mainly rely on manually checking and possibly removing the targeted content once published, upon appropriate signaling by OSNs users. Despite the massive enrollment of content moderators by social media providers [31], the recent past has seen the dramatic growth of hate attacks, affecting the mental and even the physical status of the victims (see [19, 20, 1], just to mention a few deplorable events).

A promising stream of research to fight haters, by quickly and automatically discriminating between hate and no hate speech, is the training of automated classifiers based on manually annotated corpora. The majority of the existing methods rely on supervised document classification tasks [29]. In their turn, these tasks are divided into two main categories: classic and deep learning methods [35]. While the former depend on the extraction and engineering of (mainly) textual features, successively given as input by classifiers such as Support Vector Machines, Naive Bayes, and Logistic Regression, see, e.g., [7], the latter employ neural networks to learn various level of abstract features from raw text [24]. As of late, the trend in automatic detection of hate speech is relying on deep learning. This is mainly due to the fact that classic methods strongly depend on Natural Language Processing (NLP) approaches, which require a notable effort for extracting the features in input to the classifier and, moreover, are strongly dependent from the considered language.

The research question of this work is the following: *Can we avoid the adoption of NLP-based classifiers for the task of hate speech detection, moving towards deep-learning techniques and achieving at least comparable performance results?*

To answer the question, we compare the performances of NLP and deep learning-based classifiers to automatically discriminate whether, and to what extent, two Italian corpora express hate. Our benchmark consists of two annotated datasets: i) a collection of Facebook posts and comments, created and firstly used in [8], and ii) a Twitter corpus recently introduced in [26, 28].

[1] `https://www.mobistealth.com/blog/facebook-misuse-stats/` All URLs accessed on November 16, 2018.

[2] `https://support.google.com/youtube/answer/2801939?hl=en` YouTube Hate Speech Policy.

Overall, we test six classifier configurations: three based on conventional machine learning approach, three based on neural networks. The outcome of the analysis, given as usual in terms of standard classification metrics, testify that the best classification results are obtained via deep learning techniques. This means that an effective hate speech detection system can be successfully built without the need of NLP tools. The main advantages of this achievement are:

- a language-agnostic solution: there is no need of NLP tools optimised for specific languages;
- a very limited feature engineering phase: basically, we use tokenization and stop words removal only;
- a sizable reduction of the time needed for both the learning and the classification phase: at the same classification performances, deep learning-based classifiers need a number of features which are 3 orders of magnitude less than those needed by NLP-based classifiers;
- there is no need for a complete re-training of the learning model, when feeding the classifier with new labeled data: neural networks models can be updated online, and incrementally, on the arrival of new data. Remarkably, this limits the issue known as concept drift [11].

As a further achievement, although for both the Twitter and the Facebook datasets the best results are obtained with the deep learning methodologies, we discuss the differences of such results passing from one methodology to the other. This gives us some remarkable insights on the characteristics of the corpora taken into consideration, and consequently, the capability to discern how much is the relative gain in the use of each methodology for a specific dataset.

Remainder: Next section describes the annotated corpora and how the text is pre-processed and, next, it introduces the six classification configurations adopted throughout the paper. Section 3 describes the choice of parameters of the learning methods and critically discusses the obtained performances. Finally, Section 4 illustrates related work in the area of hate speech detection, while Section 5 concludes the paper.

2 Methodologies

We propose a comparison between several methods of increasing complexity which can be used to build an effective hate speech detection system. The main aim is to show that we can detect hate speech by completely avoiding the "feature engineering problem" (adopting textual representations based on language modeling techniques [16]) and using popular deep learning algorithms instead. Each of the proposed methods is a binary classifier which outputs 1 in case of an input document expressing hate and 0 otherwise. Before describing each method, we introduce the dataset used throughout this study, and we describe how we process documents before applying machine learning methods to them.

2.1 Dataset

The dataset is the result of a joint effort of two research teams on unifying the annotation previously applied to two different datasets, in order to allow their exploitation for the HaSpeeDe (Hate Speech Detection) task [2], organized within Evalita 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian[3].

The first sub-dataset is a collection of Facebook comments created in 2016 and firstly presented in [8]. The content of the comments to Facebook posts were retrieved through a versatile Facebook crawler, which exploited the Facebook Graph API[4]. The collected comments are related to posts published on a series of public Italian web pages and groups, mainly involved in politics, and possibly featuring hate content. The original set in [8] consisted of 17,567 Facebook comments from 99 posts.

The second sub-dataset is a Twitter corpus [26, 28] comprising tweets against immigrants, Muslims and Roma. To obtain such data, a phase of data filtering was previously enacted, by means of a keyword-based approach (neutral keywords frequently associated to the three targets). This led to 236,193 tweets. After further processing and restricting the dataset so obtained, the final version of the corpus was made of 6,928 tweets.

Recently, the Facebook and Twitter corpora have been re-annotated and made publicly available for participating to the Evalita 2018 task on Hate Speech Detection. In the brand new annotated dataset, the annotation format consists of a simplified version of the schemes adopted in [8, 26, 28]. The resulting annotated dataset comprises the tweet or Facebook comment along with the tag resulting from the annotation, 1 and 0, expressing the presence or not of hate speech in the text. Overall, the renewed Facebook and Twitter dataset consists of 4,000 comments and 4,000 tweets. Each corpus is divided into two distinct sets (3,000 documents for training and 1,000 for test). For Facebook, as training set we have 1,618 comments tagged as 0 (No Hate) and 1,382 tagged as 1 (Hate), while as test set we have 323 comments tagged as 0 and 677 tagged as 1. For Twitter, as training set we have 2,028 tweets tagged as 0 and 972 tagged as 1, while as test set we have 654 tweets tagged as 0 and 346 tagged as 1.

2.2 Text pre-processing and document representation

We use two different approaches for the textual representations of documents, i.e., *bag of words* and *word embeddings* (more on these later). Before processing the data with machine learning methods, we tokenize the original text of the Twitter and Facebook input datasets, by adopting the following approach. After lowering the text and removing the stopwords, we delete all the URLs. Next, we include all tokens which belong to one of these categories: a) normal words, b) emoticons, and c) special characters (e.g., '!' and '?'). In particular, we remove

[3] `http://www.di.unito.it/~tutreeb/haspeede-evalita18/index.html\#`
[4] `https://developers.facebook.com/docs/graph-api`

'#' from hashtags and consider the remaining characters as single words, while we include all the usernames i.e., (words beginning with '@') as single tokens.

The first textual representation we use is the classic bag-of-words (BoW)[30], which is probably the most popular approach used in the past years. With this representation, we analyze the training data to build a dictionary of valid words (tokens) and use them to encode the documents by generating high sparse vectors for their representations. The importance of each token contained in a document is usually weighted according to a specific policy. Here, we adopt the well known TF-IDF [30], whose main idea is to give more importance to terms that are frequent in a document included in the corpus under evaluation, but, at the same time, tend to appear in few documents.

The second textual representation is based on word embeddings, a set of neural language modeling techniques[12] based on unsupervised learning and used to build effective vector representations of textual contents. Here, we consider the popular word2vec algorithm[23] implemented in `gensim` software[5] as the language modeling technique to represent text. This technique extracts low-dimensional dense word vectors from analyzed text which keep track of semantic/syntactic relationships existent between words. Such vectors can thus be used in algebraic operations to point out some specific characteristic of the data (e.g., $W(\text{“}queen\text{”}) \cong W(\text{“}king\text{”}) - W(\text{“}man\text{”}) + W(\text{“}woman\text{”})$).

We use three different word2vec models to encode documents. The first one is a model built over the complete document collection of the Italian Wikipedia[6], using *skipgram* architecture, a window equals to 10 words and embeddings size equals to 300. The main problem of this model is that the Wikipedia articles used to train it are very different in terms of lexical and syntactical forms from documents available on input datasets (i.e., short texts like tweets and comments on Facebook). To overcome this limitation, we also consider a second embeddings model, which is based on the first one but it is updated considering the documents collection coming from the Twitter dataset. Thirdly, we build a third word2vec model based on the Wikipedia one but updated with data coming from the Facebook dataset.

In order to encode the documents, we compute the document vector d_i as

$$d_i = 1/lw_i \sum_{j=1}^{lw_i} we(idx(i,j))$$

where lw_i is the number of valid words[7] included in document i, $idx(i,j)$ is the function which returns the index of embeddings vector corresponding to the word at position j in document i, and we is the function which returns the embeddings vector corresponding to the specified index. This document encoding is desirable while analyzing short textual documents, such as in our scenario, because the resulting vector does not suffer from a diffuse information loss, due to the mixing and averaging of vectors of many words (typical case for long documents).

[5] A Python implementation of word2vec: `https://radimrehurek.com/gensim/`.

[6] The pre-trained embeddings model: `http://hlt.isti.cnr.it/wordembeddings/`.

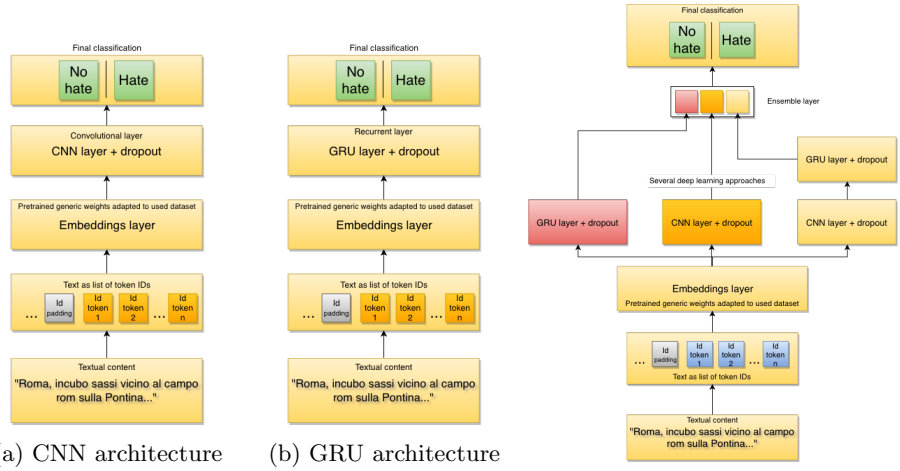
[7] A word is valid only if it is available as entry in the dictionary of embeddings model.

2.3 Machine learning methods

In the following we present the methods tested usable for building effective hate speech detection systems. First we present 3 methods based on SVM algorithm but using different textual data representations. Next we describe 3 different deep learning algorithms, of increasing complexity, which show very competitive results and avoid almost completely the feature-engineering process, except for the pre-processing part previously described.

As first method (called 'SVM-TfIdf' in Table 2), we use a classifier based on the SVM algorithm[14][8]. This algorithm has been successfully used in several works related to text classification[15, 32, 10], especially if combined with BoW textual representations and TF-IDF feature weighting. SVM, even when used with default parameters (e.g., with a simple linear kernel) usually performs reasonably well on various text classifications tasks. For this reason, we argue that this algorithm, together with BoW and TF-IDF, could be a quite strong baseline to compare with, giving us evidence of how the other tested methods perform.

To improve the effectiveness of the algorithm, we use other two variants of SVM, by combining it with a textual representation based on embeddings, obtained as described in Section 2.2. The first variant ('SVM-GenEmb' in Table 2) encode documents using a generic embeddings model built over the Italian Wikipedia collection. A second variant ('SVM-SpeEmb' in Table 2) try to take advantages from a specialized version of the previous embeddings model, enriched with the Facebook and Twitter datasets. We therefore generate two embeddings models, one specific for Twitter and one for the Facebook data.



(a) CNN architecture (b) GRU architecture

(c) Ensemble architecture

Fig. (1) Tested deep learning architectures.

[8] SVM implementation provided by scikit-learn, `http://scikit-learn.org/stable/`.

We subsequently concentrate on deep learning (DL) methods, which, in the last years, have proven to perform very well on text classification tasks. The first architecture (referred as 'CNN' in Table 2) is shown in Figure 1a. The core is a Convolutional Neural Network (CNN) [17], a fast and popular DL method, often used for image classification tasks, but also proven effective within NLP domains [34]. The proposed architecture encodes a raw text into a list of token IDs, which translates in activating the corresponding tokens vectors into an embeddings layer, pre-loaded before that the training starts, with weights learned in the embeddings model[9]. The encoded document is then fed to a CNN layer that tries to learn some space-invariant high level features, able to catch interesting patterns for final hate speech detection phase.

CNN does not take into account the order and dependencies of words while learning potential interesting features. Thus, the same identified pattern is given the same weight, independently from contextual dependencies and position within the document. To overcome this limitation, we test a second DL architecture (Figure 1b). It includes a Gated Recurrent Unit (GRU) layer [5] ('GRU' in Table 2). GRU is a recurrent neural network that, before analyzing new incoming data, maintains an internal state used to record the context that has been previously met. The current internal state thus influences how the network reacts in terms of generated output, giving relevance to the temporal characterization of the text (i.e., to the order the analyzed words appear in the text).

The last tested method, referred as 'Ensemble' in Table 2 and Figure 1c, tries to merge advantages from the previous DL architectures. Here, there is an ensemble classifier whose decisions are based on 3 different sub-modules. The first two modules are composed by two separated layers, as discussed above (GRU and CNN networks). The third one is a composition of layers, first a CNN layer and then a GRU layer. The main intuition for such a configuration is that the CNN and GRU single sub-modules could produce independent information about a text, to be classified according to the specific nature of the network (spatial for CNN, temporal for GRU), while the third sub-module could produce a high level and dependent spatial-temporal representation of the text. The contribution of each sub-module is in turn sent to an ensemble layer, which can appropriately measure each feedback, before taking the final decision.

3 Parameters settings and experimental results

This section describes the assumptions and parameters used to perform the experimentation and presents and discusses the experimental results.

3.1 Parameters settings

For all the SVM methods, except for the 'SVM-TfIdf' configuration, we use a k-fold cross validation (with $k = 5$), in order to find those kernels and parameters

[9] We always use the specialized variant of embeddings model.

Table (1) Values of the parameters used in learning methods, obtained through a 5-fold cross validation.

	Method	Kernel type	C	Gamma	NumFiltersCNN	KernelSizeCNN	NumUnitsGRU
Twitter	SVM-Tfidf	linear	1.0	-	-	-	-
	SVM-GenEmb	rbf	1.0	0.3	-	-	-
	SVM-SpeEmb	rbf	1.0	0.3	-	-	-
	CNN	-	-	-	50	3	-
	GRU	-	-	-	-	-	50
	Ensemble	-	-	-	50	3	50
Facebook	SVM-Tfidf	linear	1.0	-	-	-	-
	SVM-GenEmb	rbf	1.0	0.1	-	-	-
	SVM-SpeEmb	rbf	1.0	0.3	-	-	-
	CNN	-	-	-	50	3	-
	GRU	-	-	-	-	-	50
	Ensemble	-	-	-	50	3	50

that guarantee the best effectiveness on a validation set opportunely extracted from the training set[10]. In the case of 'SVM-Tfidf', for which the computational cost necessary to perform the optimization is very high[11], we decide to use the default SVM parameters, i.e., a linear kernel with C equals to 1. In all the deep learning architectures, we use a learning procedure including a batch size of 32 documents, a maximum input sequence length of 100 words, opportunely padded if needed, a maximum number of epochs equals to 20[12], a fixed dropout value of 0.2 and the binary cross-entropy loss function.

A summary of the parameters values is in Table 1. The values `C` and `Gamma` are the standard parameter names used in the SVM algorithm for tuning its behaviour during the learning phase, `NumFiltersCNN` is the number of convolutional units used in the corresponding layer, `KernelSizeCNN` is the size of the sliding window used in convolutions, and `NumUnitsGRU` is the number of GRU units used in the corresponding layer.

3.2 Experimental results

The experimental results are in Table 2. We use the Precision (Pr), Recall (Re) and F1 metrics [30] as the standard way to measure the effectiveness of the proposed methods.

We report in bold the best F1 values obtained for the two datasets, for the single classes (No hate and Hate) and for the average F1 obtained weighting both the labels according to the distribution of the documents in the two classes.

[10] The training set has been divided in two distinct sets. The first 90% used as real training in the optimization process, the remaining 10% as validation data used to measure the effectiveness of the built model.

[11] The document vector is very sparse and long, in the order of tens of thousands distinct words.

[12] As early stopping criterion, in order to prevent overfitting, we stop the learning process when the loss on validation set is less than 0.02 after two consecutive epochs.

Table (2) Effectiveness of tested methods over the test sets.

		No hate			Hate			Weighted avg results		
	Method	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
Twitter	SVM-TfIdf	0.805	0.875	0.838	0.702	0.564	0.625	0.771	0.774	0.769
	SVM-GenEmb	0.810	0.870	0.839	0.679	0.574	0.622	0.767	0.774	0.769
	SVM-SpeEmb	0.817	0.867	0.841	0.682	0.596	0.636	0.773	0.779	0.775
	CNN	0.798	0.935	**0.861**	0.788	0.506	0.617	0.795	0.796	0.782
	GRU	0.830	0.851	0.840	0.671	0.636	0.653	0.778	0.781	0.779
	Ensemble	0.839	0.848	0.843	0.675	0.660	**0.668**	0.786	0.787	**0.786**
Facebook	SVM-TfIdf	0.520	0.782	0.624	0.861	0.663	0.749	0.750	0.701	0.708
	SVM-GenEmb	0.609	0.638	0.623	0.823	0.805	0.814	0.754	0.751	0.752
	SVM-SpeEmb	0.602	0.669	0.633	0.833	0.789	0.810	0.758	0.750	0.753
	CNN	0.636	0.681	**0.658**	0.843	0.814	0.828	0.776	0.771	0.773
	GRU	0.658	0.619	0.638	0.823	0.846	**0.835**	0.770	0.773	0.771
	Ensemble	0.647	0.659	0.653	0.836	0.829	0.832	0.775	0.774	**0.775**

Considering the weighted average results, the baseline method SVM-TfIdf works quite well, especially for Twitter, where it obtains good performances, comparable to the SVM methods using textual representations with embeddings (F1 equal to 0.769, 0.769, 0.775, *resp.*). Instead, on Facebook, the baseline seems to suffer (F1 equal to 0.708), probably because of a higher average length of the texts which negatively influences the discriminating power of IDF part in feature weighting: this decreases the effectiveness of the classifier. Still on Facebook, the two SVM methods using embeddings (SVM-GenEmb and SVM-SpeEmb) provide useful additional information from the analysis of Wikipedia data and outperform the baseline SVM classifier by a notable margin (F1 = 0.752, 0.753).

The double use of embeddings is comparable in effectiveness, with a slight advantage of SVM-SpeEmb on Twitter data (0.775 *vs* 0.769), where it can exploits local information like emoticons, commonly used in tweets to express emotions.

GRU and CNN show very similar weighted average performances, always better than those obtained with the SVM-based methods. However, they present some differences between each other. In particular, on both datasets, CNN works a lot better than GRU on documents labelled as No hate, while GRU works features good performances in discriminating documents labelled as Hate.

The Ensemble architecture takes advantages from the two DL methods and obtains a compromise generally able to guarantee the best weighted average F1 (0.786 for Twitter, 0.775 for Facebook). Indeed, even if the method does not obtain the best F1 results on all classes and datasets (with the exception of Hate on Twitter, where F1 = 0.668), it is always close to the best performer (0.843 *vs* 0.861 for No Hate on Twitter, 0.653 *vs* 0.658 for No Hate on Facebook, 0.832 *vs* 0.835 for Hate on Facebook). Therefore, the Ensemble configuration results in a always-good solution for the application scenarios considered in this work.

Table (3) Efficiency of the tested methods, in seconds.

Method	**Learning time (s)**	**Classification time (s)**
SVM-TfIdf	422.41	32.56
SVM-SpeEmb	18.62	1.19
Ensemble	138.59	3.58

Table 3 shows how the methods perform *wrt* learning and classification phases on Facebook. For the sake of simplicity, we report only three methods, since the others add no additional information to the discussion. The measures consider both the time to build a classifier with a set of already fixed algorithm's parameters ('Learning time') and the time to classify all the documents in the test set ('Classification time'). SVM-TfIdf is the most costly method because, with a BoW sparse representation, it must process tens of thousands of different features. With embeddings (SVM-SpeEmb), the computational cost is drastically decreased. The most complex tested method (Ensemble) is a compromise in terms of cost at learning time while maintaining very good efficiency at classification time. The additional cost in learning is repaid by the possibility to create online learners, thus allowing incremental learning steps and only requiring to process the entire available training set once, i.e., the first time.

4 Related work

Over the past few years, hate campaigns against individuals or groups, often minorities, have occurred on a variety of online platforms. Given the even extreme consequences that these incidents may cause to the victims, the scientific community has recently spent increasing to realize effective automated techniques able to recognize the presence of aggressive and hateful content within texts published online. To give the flavor of how challenging this task is, the report in [18] summarises the findings of the 'Shared Task on Aggression Identification', a challenge organised in 2018 as part of a popular international conference on Computational Linguistics (COLING 2018). The task was to develop a classifier that could discriminate between three hate classes: Overtly Aggressive, Covertly Aggressive, and Non-aggressive texts. The organisers provided the participants with a dataset of 15,000 annotated comments from Facebook (in English and Hindi). Over a total of 130 participating teams, the best classifier obtained a weighted F-score of 0.64 for both Hindi and English.

Interestingly, the research team in [35, 36], in a parallel and independent way by the authors of this paper, considers a detection system combining convolutional and gated recurrent networks and tests it on a large collection of public Twitter datasets (in English), obtaining micro F1 scores that, for most of the datasets, improve the state-of-art performances (values range from 0.83 to 0.94). The considered architecture is different from that presented in Figure 1c. In fact, work in [35, 36] considers a cascade of one CNN layer and one GRU layer, without the ensemble layer here considered.

The authors of [25] propose an ensemble of Recurrent Neural Network classifiers, which incorporates not only features associated with the document, but also user-related information, such as a tendency towards racism or sexism. The approach has been evaluated on a publicly available corpus of 16k tweets in English, manually labelled as containing sexist, racist, or neutral messages [33]. The RNN classifiers ensemble achieves a weighted average F score equal to 0.93, succeeding in distinguishing racism and sexism messages from normal text.

Still focusing on Twitter, work in [3] proposes a multi-attributes approach to detect bullying and aggressive behavior on Twitter. Considering textual, user, and network-based features, it gives a characterization of bullies and aggressors behaviour, showing that the former tend to 'post less, participate in fewer online communities, and are less popular than normal users', while the latter 'are relatively popular and tend to include more negativity in their posts'. Following a standard machine-learning-based classification, the work succeeds in detecting bullies and aggressors, considering a corpus of 1.6M tweets.

There exist some work which focuses on less popular platforms, such as the discussion board 4chan[13], whose sub-board 'Politically Incorrect' has often been linked to the alt-right movement. The authors of [13] rely on standard NLP tools and polarity-annotated lexicons to identify hateful content within the posts of that sub-board. The results of the analysis show an extraordinary level of expressed hatred, particularly for ethnic reasons.

In August, 2014, a harassment campaign, primarily conducted over Twitter with the use of the hashtag #GamerGate, targeted several women in the video game industry, mainly expressing sexist and anti-progressivism opinions. By means of a methodology similar to that followed in [13], work in [4] characterise the users involve in the GamerGate campaign, mainly by analysing their social network and the content of their posts. Findings are that they tend to have more friends and followers, are generally more engaged and post tweets with less joy and more hate than random users.

Still regarding harassment campaigns, a raising and worrying phenomenon, known as raiding, consists of organize and coordinate ad-hoc mobs aimed at disrupt a specific social platform. The authors of [22] employ machine learning techniques to predict those YouTube videos which are likely to be raided by users of third-party hateful communities.

Focusing on the Italian language, [8] designed and developed the first hate speech classifier for Italian, testing two different state-of-art approaches for sentiment analysis tasks (SVM and LSTM algorithms). Initially, the Facebook dataset introduced in Section 2.1 was annotated according to three distinct classes: comments could either be tagged as Strong Hate, Weak Hate, No Hate. Unfortunately, both SVM and LSTM were not able to discriminate well among the three classes. This was particularly true for the Strong Hate class. These results were probably due to the small number of Strong Hate documents in the dataset and the low level of annotators' agreement. To date, as also highlighted by [26, 28], dealing with multiple hate classes lead to a low agreement among the anno-

[13] `http://www.4chan.org/`

tators, and thus, to scarce results in terms of automated classification. Instead, when considering two classes (Hate/No Hate) and only the documents for which at least 70% of the annotators were in agreement, both SVM and LSTM perform better, with a F1 score equal to 0.72 for the Hate class.

5 Conclusions

The growing spread of online hate campaigns against individuals and minorities quests for effective techniques to automatically detect hate content. Parallel to standard text classification methodologies, mainly based on processing texts through conventional machine learning tools, we assist to the flourishing of attempts employing neural networks. In this work, we proposed six different classification configurations and we evaluated their performances over datasets consisting of Italian Facebook posts and tweets. Results showed that those configurations where deep learning is used perform better than the others. This outcome is promising, since deep learning features the noticeable advantages to be independent from i) data re-training, ii) NLP machinery specialised for specific languages, iii) the time-consuming feature engineering phase.

Given the encouraging results, as future work we aim at testing other DL configurations, possibly over larger datasets. Furthermore, in order to assess the strength of our choices *wrt* related work, we will test the proposals available in the literature, including ours, on the same dataset. As an example, aiming at maintaining the dataset considered in this work, a promising starting point is to work on the architectures, and associated results, of the Hate Speech detection task at EVALITA, which took place in late December, 2018.

References

1. BBC News. Akubra girl Dolly's bullying suicide shocks Australia. `https://www.bbc.com/news/world-australia-42631208`, January, 10 2018.
2. Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. Overview of the EVALITA 2018 Hate Speech Detection task. In *EVALITA*, 2018.
3. Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. Mean Birds: Detecting aggression and bullying on Twitter. In *2017 ACM on Web Science Conference*, WebSci '17, pages 13–22, 2017.
4. Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. Measuring #GamerGate: A tale of hate, sexism, and bullying. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 1285–1290, 2017.
5. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.
6. Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The paradigm-shift of social spambots: Evidence, theories, and

tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 963–972, 2017.
7. Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 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, May 15-18, 2017.*, pages 512–515, 2017.
8. Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on Facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017.*, pages 86–95, 2017.
9. Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Commun. ACM*, 59(7):96–104, June 2016.
10. George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003.
11. João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, March 2014.
12. Yoav Goldberg. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.
13. Gabriel Emile Hine, Jeremiah Onaolapo, Emiliano De Cristofaro, Nicolas Kourtellis, Ilias Leontiadis, Riginos Samaras, Gianluca Stringhini, and Jeremy Blackburn. Kek, Cucks, and God Emperor Trump: A measurement study of 4chan's politically incorrect forum and its effects on the web. In *Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 92–101, 2017.
14. Thorsten Joachims. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.
15. Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.
16. Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.
17. Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
18. Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11. Association for Computational Linguistics, 2018.
19. La Repubblica (in Italian). Hater augura la morte ai figli di Bonucci: Forse un giorno vivremo in un mondo migliore, risponde il calciatore. `https://www.repubblica.it/sport/2018/09/13/news/hater\_augura\_la\_morte\_ai\_figli\_di\_bonucci\_la\_risposta\_del\_calciatore-206310289/`, September 13, 2018.
20. La Repubblica (in Italian). Insulti sui social a Chiara Bordi, la 18enne disabile candidata a Miss Italia. `https://www.repubblica.it/spettacoli/people/2018/09/15/news/insulti_sui_social_a_chiara_bordi_la_18enne_disabile_candidata_a_miss_italia-206553959/`, September 18, 2018.
21. Daily Mail. Zuckerberg in Germany: No place for hate speech on Facebook. `http://www.dailymail.co.uk/wires/ap/`

article-3465562/Zuckerberg-no-place-hate-speech-Facebook.html, February, 26 2016.
22. Enrico Mariconti, Guillermo Suarez-Tangil, Jeremy Blackburn, Emiliano De Cristofaro, Nicolas Kourtellis, Ilias Leontiadis, Jordi Luque Serrano, and Gianluca Stringhini. You know what to do: Proactive detection of YouTube videos targeted by coordinated hate attacks. *CoRR*, abs/1805.08168, 2018.
23. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.
24. Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, 2016.
25. Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. Detecting offensive language in tweets using deep learning. *CoRR*, abs/1801.04433, 2018.
26. Fabio Poletto, Marco Stranisci, Manuela Sanguinetti, Viviana Patti, and Cristina Bosco. Hate speech annotation: Analysis of an Italian Twitter corpus. In *CLiC-it*, 2017.
27. Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Detecting and tracking political abuse in social media. In *Proc. 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.
28. Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. An Italian Twitter corpus of hate speech against immigrants. In *LREC*, 2018.
29. Anna Schmidt and Michael Wiegand. A survey on hate speech detection using Natural Language Processing. In *SocialNLP@EACL*, 2017.
30. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.
31. The Guardian. Facebook not protecting content moderators from mental trauma: lawsuit. https://www.theguardian.com/technology/2018/sep/24/facebook-moderators-mental-trauma-lawsuit, September 24, 2018.
32. Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.
33. Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 88–93. Association for Computational Linguistics, 2016.
34. Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for Natural Language Processing. *arXiv preprint arXiv:1702.01923*, 2017.
35. Ziqi Zhang and Lei Luo. Hate speech detection: A solved problem? The challenging case of long tail on Twitter. *CoRR*, abs/1803.03662, 2018.
36. Ziqi Zhang, David Robinson, and Jonathan Tepper. Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 745–760, Cham, 2018. Springer International Publishing.