In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy.

Hate me, hate me not: Hate speech detection on Facebook

Fabio Del Vigna[12], Andrea Cimino[23], Felice Dell'Orletta[3], Marinella Petrocchi[1], and Maurizio Tesconi[1]

[1] Istituto di Informatica e Telematica, CNR, Pisa, Italy.
{f.delvigna, m.petrocchi, m.tesconi}@iit.cnr.it
[2] University of Pisa, Pisa, Italy
[3] Istituto di Linguistica Computazionale, CNR, Pisa, Italy
{andrea.cimino, felice.dellorletta}@ilc.cnr.it

Abstract

While favouring communications and easing information sharing, Social Network Sites are also used to launch harmful campaigns against specific groups and individuals. Cyberbullism, incitement to self-harm practices, sexual predation are just some of the severe effects of massive online offensives. Moreover, attacks can be carried out against groups of victims and can degenerate in physical violence. In this work, we aim at containing and preventing the alarming diffusion of such hate campaigns. Using Facebook as a benchmark, we consider the textual content of comments appeared on a set of public Italian pages. We first propose a variety of hate categories to distinguish the kind of hate. Crawled comments are then annotated by up to five distinct human annotators, according to the defined taxonomy. Leveraging morpho-syntactical features, sentiment polarity and word embedding lexicons, we design and implement two classifiers for the Italian language, based on different learning algorithms: the first based on Support Vector Machines (SVM) and the second on a particular Recurrent Neural Network named Long Short Term Memory (LSTM). We test these two learning algorithms in order to verify their classification performances on the task of hate speech recognition. The results show the effectiveness of the two classification approaches tested over the first manually annotated Italian Hate Speech Corpus of social media text.

1 Introduction

Social Network Sites (SNSs) are an ideal place for Internet users to keep in touch, share information about their daily activities and interests, publishing and accessing documents, photos and videos. SNSs like Facebook, Twitter, Ask.fm and Google+ give the ability to create profiles, to have a list of peers to interact with and to post and read what others have posted. It comes as no surprise that, overall, SNSs - together with search engines - are among the most visited websites[1].

Unfortunately, SNSs are also the ideal plaza for proliferation of harmful information. Cyberbullying, sexual predation [25], self-harm practices incitement [6] are some of the effective results of the dissemination of malicious information on SNSs. Many of these attacks are often carried by a single individual, but they can be also managed by groups. The target of the *trolls* are often selected victims but, in some circumstances, the hate can be directed towards wide groups of individuals, discriminated for some features, like race or gender. Such campaigns may involve a very large number of *haters* that are self excited by hateful discussions, and such hate might end up with physical violence or violent actions.

[1] http://www.alexa.com/topsites - All websites have been lastly accessed on October, 23, 2016.

Work in [21] characterises the attacker and it provides a definition of *trolls*, i.e., online users pretending to sincerely strive for be part of an online community, but whose real intentions are to cause disruption and exacerbate conflict, for the purposes of their own amusement. Thus, sexists, religious fanatics, political extremists massively use SNSs to foster *hate* against specific individuals/organizations, by causing a sounding board effect, which may critically damage the targets of the hate campaign, by using both psychological and physical violence. Although more experienced users could be able to face threats and trolls, the great majority of them cannot easily bear the attacks, especially minors and those who might get exposed mediatically to public judgment. Media frequently report evidences about the (unfortunately extreme in some cases) consequences that naïve and emotive users have faced to[2].

This work aims at containing and preventing the alarming diffusion of massive online hate campaigns on SNSs and it focuses on Italian texts. The issue has been tackled in the past with different approaches, laying somehow in the middle between pre-emption and mending. One approach aims at mitigating chat conversations through ad hoc filters, like in [38], by semantically detecting offensive content and removing it. The second approach operates on published content and tries to remove the offending one, often leveraging the analysis on multiple messages, as in [3, 7, 36].

Contributions. Our aim is not censoring online content, as we mostly address its classification for the Italian language to pinpoint anomalous waves of hate and disgust. Using Facebook as a benchmark, we classify the content of comments appeared on a set of public pages. We contribute along the following dimensions:

- we design and develop the first hate speech classifier for the Italian language and we compare two different approaches based on state-of-the-art learning algorithms for sentiment analysis tasks;
- starting from classifying the single comment on a Facebook page, the results proposed in this paper constitute the prelude to the detection of violent discussions as a whole. This with the ultimate goal to promptly detect waves of hate, which several users may take part to, as it happened recently and unfortunately on Facebook pages[3,4];
- we introduce a taxonomy of a variety of hate categories, expanding the classes proposed in [19] and specifically considering the *subject of the hate*, like, e.g., hate for religious, racial, socio-economical reasons; while not directly employed here in the classification, the definition of such taxonomy is a step towards more refined classification tasks.

Next section introduces the corpus for hate speech detection. Section 3 presents our classification techniques and reports its performance results. In Section 4, we discuss related work on detecting textual aggressions on social media. Finally, Section 5 concludes the paper.

2 Hate Speech Corpus

This section reports on the retrieval and annotation phase of our hate speech Italian corpus.

[2] http://osservatorio-cyberbullismo.blogautore.repubblica.it/ (La Repubblica - Italian newspaper online edition)

[3] https://www.facebook.com/Black-block-430370993643807/ (Facebook page)

[4] https://goo.gl/jYJPoZ (Il Tempo - Italian Newspaper online edition)

2.1 Data crawling

Aiming at monitoring the "hate level" across Facebook, we have built a corpus of comments retrieved from the Facebook public pages of Italian newspapers, politicians, artists, and groups. These pages typically host discussions spanning across a variety of topics.

We have developed a versatile Facebook crawler, which exploits the Graph API[5] to retrieve the content of the comments to Facebook posts. The crawler leverages the flexibility of the Laravel framework to deploy a wide set of features, like flexibility, code reuse, different storage strategies and parallel processing. Implemented as a Web service, it can be controlled through a Web interface or using a cURL command[6]. The tool requires a set of registered application keys and some target pages to crawl. It is capable of storing data in the filesystem either as JSON[7] files, or in Kafka[8] queues or in Elasticsearch[9] indexes. According to the number of application keys provided to the application, it is able to crawl multiple pages in parallel. Starting from the most recent post, the crawler collects all the information related to the posts, up to comments to comments. For the sake of simplicity, in this work we have however limited our analysis to direct comments to the posts.

2.2 Data annotation

The crawler was used to collect comments related to a series of web pages and groups, chosen since we suspected to possibly contain hate content.

Title of Facebook page	Annotated posts	Comments	Annotations
salviniofficial	19	5404	15298
matteorenziufficiale	2	158	584
lazanzarar24	10	307	1253
jenusdinazareth	2	132	460
sinistracazzateliberta2	7	79	234
ilfattoquotidiano	11	126	135
emosocazzi	4	73	75
noiconsalviniufficiale	14	223	270

Table 1: Dataset description and annotations

Overall, we collected 17,567 Facebook comments from 99 posts crawled from the selected pages: 6,502 of them have been annotated at least once (spanning over 66 posts) and at most received 5 annotations from distinct human annotators. We asked to 5 bachelor students to annotate comments, and the majority of comments received more than one annotation. Students annotated 5742, 3870, 4587, 2104 and 2006 comments respectively. In particular, among the annotated comments, 3,685 received at least 3 annotations. On average, each annotator annotated about 3,662 comments.

The annotators were asked to assign one class to each post, where classes span over the followings levels of hate: *No hate*, *Weak hate*, and *Strong hate*.

We then divided hate messages into distinct categories: *Religion*, *Physical and/or mental handicap*, *Socio-economical status*, *Politics*, *Race*, *Sex and Gender issues*, and *Other*.

[5] https://developers.facebook.com/docs/graph-api
[6] https://curl.haxx.se
[7] http://json.org
[8] http://kafka.apache.org
[9] https://www.elastic.co/products/elasticsearch

Given that the majority of comments has been annotated by more than one annotator, we have also computed the Fleiss' kappa κ inter-annotator agreement metric [20], which measures the level of agreement of different annotators on a task. The level of agreement among annotators conveys the level of the difficulty of a task. In our case, considering the 1,687 comments that received annotations from all the 5 annotators, we obtain $\kappa = 0.19$ when discriminating over three hate classes, while $\kappa = 0.26$ over two classes (where *Strong Hate* and *Weak Hate* have been merged together). Such low κ values testify that the annotation task was really hard for our students.

3 Text Classification

This section describes the classification approaches and gives their results. On the annotated dataset, we compute a series of features, described in detail in the following. A series of lexicons used to derive part of the features is described in Section 3.1.1. Comments in our dataset are then represented as a vector of features, given as input to the classifier, along with the result of the annotation. In the training phase, the classifier learns to classify a comment according to the values of its features and the annotation result. In the test phase, the classifier takes decision and tags comments as expressing hatred or not, according to the learned model.

3.1 The classifier

We tested two different classifiers based on different learning algorithms: the first one based on Support Vector Machines (SVM) and the second one on a particular Recurrent Neural Network named Long Short Term Memory (LSTM). While SVM is an extremely strong performer, hardly to be transcended, unfortunately these types of algorithms capture "sparse" and "discrete" features in document classification tasks. This makes the detection of relations in sentences really hard, while this is often the key factor in detecting the overall sentiment polarity of a document [33]. On the contrary, LSTM networks are a specialization of Recurrent Neural Networks (RNN), which are able to capture long-term dependencies in a sentence. This type of neural network has been recently tested on sentiment analysis tasks [33, 37], reaching outperforming classification performance [29], with even a 3-4 points improvement with respect to commonly used learning algorithms. In this work, we use the Keras [8] deep learning framework and LIBSVM [5] to generate, respectively, the LSTM and the SVM statistical models. Since our approach relies on morpho-syntactically tagged texts, the hate speech corpus was automatically morpho-syntactically tagged by the Part-Of-Speech tagger described in [14].

3.1.1 Lexical resources

To improve the overall accuracy of our system, we used both sentiment polarity and word embedding lexicons. Sentiment polarity lexicons[10] were already successfully tested for the classification of positive, negative and neutral sentiment of Italian social media posts [2]. We used the ones described in [9] which include a manually created lexicon for Italian [35], two automatically translated sentiment polarity lexicons originally created for English [23, 35], an automatically created Twitter sentiment polarity lexicon and two word similarity lexicons automatically created using *word2vec*[11] [28], starting from two Italian corpora: (i) PAISÀ [26], a

[10] We downloaded the lexicons from www.italianlp.it
[11] http://code.google.com/p/word2vec/

large corpus of authentic contemporary Italian texts; and (ii) a lemmatized corpus of 1,200,000 tweets, automatically collected.

In addition to these resources, we created two Word Embedding lexicons, to overcome the issue that lexical information in a short text can be very sparse. For this purpose, we trained two predict models using *word2vec*. These models learn lower-dimensional word embeddings. Embeddings are represented by a set of latent (hidden) variables and each word is a multidimensional vector that represent a specific instantiation of these variables. The first lexicon was built using a tokenized version of the itWaC corpus[12]. The itWaC corpus is a 2 billion word corpus constructed from the Web, limiting the crawl to the *.it* domain and using medium-frequency words from *La Repubblica* corpus and basic Italian vocabulary lists as seeds. The second lexicon was built from a tokenized corpus of tweets. This corpus was collected using the Twitter APIs and it is made up of 10,700,781 Italian tweets.

3.1.2 The SVM classifier

The SVM classifier exploits a wide set of features, ranging across different levels of linguistic description. With the exception of the *word embedding combination*, these features have been already used in sentiment polarity classification tasks [9] showing their effectiveness. The features are organised into three main categories: *raw and lexical text features*, *morpho-syntactic and syntactic features*, and *lexicon features*.

Raw and Lexical Text Features. *Number of tokens*: number of tokens occurring in the analyzed text; *Character n-grams*: presence or absence of contiguous sequences of characters in the analyzed text. *Word n-grams*: presence or absence of contiguous sequences of tokens in the analyzed text. *Lemma n-grams*: presence or absence of contiguous sequences of lemma occurring in the analyzed text. *Repetition of n-grams chars*: presence or absence of contiguous repetition of characters in the analyzed text. *Punctuation*: checks whether the analyzed text finishes with one of the following punctuation characters: "?", "!".

Morpho-syntactic and Syntactic Features. *Coarse grained Part-Of-Speech n-grams*: presence or absence of contiguous sequences of coarsegrained PoS, corresponding to the main grammatical categories (noun, verb, adjective). *Coarse grained Part-Of-Speech distribution*: the distribution of nouns, adjectives, adverbs, numbers in the text. *Fine grained Part-Of-Speech n-grams*: presence or absence of contiguous sequences of fine-grained PoS, which represents subdivisions of the coarse-grained tags (e.g., the class of nouns is subdividefalice d into proper vs common nouns, verbs into main verbs, gerund forms, past particles). *Dependency types n-grams*: presence or absence of sequences of dependency types in the analyzed text. The dependencies are calculated with respect to *i)* the hierarchical parse tree structure and *ii)* the surface linear ordering of words *Lexical Dependency n-grams*: presence or absence of sequences of lemmas calculated with respect to the hierarchical parse tree. *Lexical Dependency Triplet n-grams*: distribution of lexical dependency triplets, where a triplet represents a dependency relation as (ld, lh, t), where ld is the lemma of the dependent, lh is the lemma of the syntactic head and t is the relation type linking the two. *Coarse Grained Part-Of-Speech Dependency n-grams*: presence or absence of sequences of coarse-grained Part-of-Speech, calculated with respect to the hierarchical parse tree. *Coarse Grained Part-Of-Speech Dependency Triplet n-grams*: distribution of coarse-grained Part-of-Speech dependency triplets, where a triplet represents a dependency relation as (cd, ch, t), where cd is the coarse-grained Part-of-Speech of the

[12] http://wacky.sslmit.unibo.it/doku.php?id=corpora

dependent, h is the coarse-grained Part-of-Speech of the syntactic head and t is the relation type linking the two.

Lexicon features. *Lemma sentiment polarity n-grams*: for each n-gram of lemmas extracted from the analyzed text, the feature checks the polarity of each component lemma in the existing sentiment polarity lexicons. Lemmas that are not present are marked with the *ABSENT* tag. This is for example the case of the trigram "tutto molto bello" (*all very nice*) that is marked as "*ABSENT-POS-POS*" because *molto* and *bello* are marked as positive in the considered polarity lexicon and *tutto* is absent. The feature is computed for each existing sentiment polarity lexicons. *Emoticons*: presence or absence of positive or negative emoticons in the analyzed text. The lexicon of emoticons was extracted from `http://it.wikipedia.org/wiki/Emoticon` and manually classified. *Polarity modifier*: for each lemma in the text occurring in the sentiment polarity lexicons, the feature checks the presence of adjectives or adverbs in a left context window of size two. If this is the case, the polarity of the lemma is assigned to the modifier. This is for example the case of the bigram "non interessante" (*not interesting*), where "interessante" is a positive word, and "non" is an adverb. Accordingly, the feature "non_POS" is created. The feature is computed 3 times, checking all the existing sentiment polarity lexicons. *PMI score*: for each set of unigrams, bigrams, trigrams, four-grams and five-grams that occur in the analyzed text, the feature computes the score given by $\sum_{\text{i}-gram \in text} score(\text{i}-gram)$ and it returns the minimum and the maximum values of the five values (approximated to the nearest integer). *Distribution of sentiment polarity*: this feature computes the percentage of positive, negative and neutral lemmas that occur in the text. To overcome the sparsity problem, the percentages are rounded to the nearest multiple of 5. The feature is computed for each existing lexicon. *Most frequent sentiment polarity*: the feature returns the most frequent sentiment polarity of the lemmas in the analyzed text. The feature is computed for each existing lexicon. *Sentiment polarity in text sections*: the feature first splits the text in three equal sections. For each section, the most frequent polarity is computed using the available sentiment polarity lexicons. The purpose of this feature is identifying changes of polarity within the same text. *Word embeddings combination*: the feature returns the vectors obtained by computing separately the average of the word embeddings of the nouns, adjectives and verbs of the text. It has been computed once for each word embedding lexicon.

3.1.3 The LSTM classifier

The LSTM unit was initially proposed by Hochreiter and Schmidhuber [22]. LSTM units are able to propagate an important feature that came early in the input sequence over a long distance, thus capturing potential long-distance dependencies. LSTM is a state-of-the-art performer for semantic composition and it allows to compute the representation of a document from the representation of its words, with multiple abstraction levels. Each word is represented by a low dimensional, continuous and real-valued vector.

We employed a bidirectional LSTM architecture since it allows to capture long-range dependencies from both the directions of a document by constructing bidirectional links in the network [31]. In addition, we applied a dropout factor to both the input gates and to the recurrent connections in order to prevent overfitting which is a typical issue of neural networks [17]. As suggested in [17], we have chosen a dropout factor value in the optimum range $[0.3, 0.5]$, more specifically 0.45 for this work. Concerning the optimization process, categorical cross-entropy is used as a loss function and optimization was performed by the rmsprop optimizer [34].

To train the LSTM architecture, each input word in the text is represented by a 262-dimensional vector which is composed by:

Word embeddings: the concatenation of the two word embeddings extracted by the two available Word Embedding lexicons (128 components for each word embedding, thus resulting in a total of 256 components), and for each word embedding an extra component was added in order to handle the "unknown word". *Word polarity*: the corresponding word polarity obtained by exploiting the Sentiment Polarity lexicons. This feature adds 3 extra components in the resulting vector, one for each possible outcome in the lexicons (negative, neutral, positive). We assumed that a word not found in the lexicons has a neutral polarity. *End of Sentence*: a component indicating whether or not the sentence was totally read.

3.2 Experiments and Results

We conducted two different classification experiments: the first considering the three different category of hate (*Strong hate*, *Weak hate* and *No hate*) the second considering only two categories, *No hate* and *Hate*, where the last category was obtained by merging the *Strong hate* and *Weak hate* classes.

For the experiments, we used only documents that were annotated at least by three different annotators and where the most annotated class exists. This process resulted in two datasets: the three-class dataset, composed by 3,356 documents - divided into 2,816 *No hate*, 410 *Weak hate* and 130 *Strong hate* documents, and the two-class dataset, composed by 3,575 documents - divided into 2,789 *No hate* and 786 *Hate*. To balance the datasets, we selected a subset of the *No hate* texts, which was limited to the double size of the documents belonging to the *Weak hate* class in the three-class experiment and to the double size of the *Hate class* in the two-class one. To evaluate the accuracy of the two hate speech classifiers in the two experiments, we followed a 10-fold cross validation process: each dataset was randomly split in ten different non overlapping training and test sets. The overall *Accuracy*, *Precision*, *Recall* and *F-score* for each class were calculated as the average of the these values over all the ten test sets. Accuracy, Precision, Recall and F-score are evaluation metrics employed in standard classification tasks. In our scenario: *Accuracy* measures how many comments are correctly identified in the classes; *Precision* measures how many comments, among those classified as expressing hate, have been correctly identified; *Recall* expresses how many comments, in the whole set, have been correctly recognized: a low recall means that many relevant comments are left unidentified and *F-score* is the armonic mean of Precision and Recall.

Table 2 reports the results for the three-class experiment. Both SVM and LSTM are not able to discriminate between the three classes and this is particularly true for the Strong hate one. These results may be due to the small number of Strong hate documents (that is the class with the lowest number of documents) and the low level of annotator agreement. These results lead us to conduct the two-class experiment, whose accuracies are in Table 3. As we expected, the results are much more higher than those in the previous experiment. This is probably due to the higher number of Hate documents with respect to the Strong e Weak classes and to the higher annotator agreement with respect to the three-class experiments.

To evaluate the impact of the annotator agreement in the classification performances, we performed a last experiment, where we selected the documents for which at least 70% of the annotators were in agreement (321 Hate and 642 No-Hate documents). As Table 3 shows, the more agreement yields an increasing accuracy for both the classification algorithms. This improvement is particularly significant for the classification of the Hate class, with F-score of about 72%. These results pave the way to the employment of our system in a real-use context. In addition, the outcome shows that this Hate Speech corpus, filtered with respect to the annotator agreement, allows to build automatic hate speech classifiers able to achieve accuracy

in line with the ones obtained in mostly investigated sentiment analysis tasks for Italian, such as subjectivity and polarity classification [2].

		Strong hate			**Weak hate**			**No hate**		
Classifier	Accuracy (%)	Prec.	Rec.	F-score	Prec.	Rec.	F-score	Prec.	Rec.	F-score
SVM	64.61	.452	.189	.256	.523	.525	.519	.724	.794	.757
LSTM	60.50	.501	.054	.097	.434	.159	.221	.618	.950	.747

Table 2: Ten-fold cross validation results on *Strong hate*, *Weak hate* and *No hate* classes.

		Hate			**No hate**		
Classifier	Accuracy (%)	Prec.	Rec.	F-score	Prec.	Rec.	F-score
SVM	72.95	.625	.568	.594	.778	.817	.797
LSTM	75.23	.640	.6832	.657	.824	.791	.805
		$\geq 70\%$ of Agreement					
SVM	80.60	.757	.689	.718	.833	.872	.851
LSTM	79.81	.706	.758	.728	.859	.822	.838

Table 3: Ten-fold cross validation results on *Hate* and *No hate* classes.

4 Related Work

Here, we briefly revise academic work on trolls and hate speech detection. Interestingly, the connections between the users profiles on SNSs are often strictly related to the connections in their real life [16]. Using machine learning, it was possible to recognize those users that adopt troll profiles in cyberbullying practices [4, 13, 18]. Similarly, text analysis approaches have been used to link together the contents of anonymous users among different opinion websites [1]. Relationships based on profile connections and behaviors have been exploited to effectively identify fake Facebook profiles [10], while lightweight profile features succeeded in recognising fake Twitter followers [11, 12]. Regarding text classification for automatic hate speech detection, a seminal work is in [32]. In [27], the authors propose a rule-based classifier to distinguish between legitimate and abusive information in texts. PALADIN [24] is a pattern mining tool to mine patterns of language, to detect anti-social behaviors of users. The authors of [36] focus on Twitter and propose a semi-supervised approach and statistical topic modeling for the detection of offensive content, while work in [3] presents a supervised machine learning text classifier, trained and tested to distinguish between hateful and antagonistic responses, with a focus on race, ethnicity and religion. Work in [15] adopts neural language models to learn distributed low-dimensional representations of comments. The approach generates text embeddings that can be used to feed a classifier. The authors of in [19] describe the distinction among flame and hate speech (the latter being more directed to groups, rather than individuals). The same work proposes the three level hate classification adopted in this paper (partially suffering for the low IAA too). Some studies concentrate on the users' behaviour. Authors of [30] propose a reputation system, which tracks reputation of users using positive and negative opinions. A behavioral analysis of banned users is in [7], showing a certain degree of similarity in their texts, containing often irrelevant content too.

5 Conclusions

This paper introduced the first hate speech classifier for Italian texts. Considering a binary classification, the classifier achieved results comparable with those obtained in mostly investigated sentiment analysis tasks for Italian. Encouraged by such promising outcome, we leave for future work the refinement of the classifier results when considering distinction (i) among hate levels (whereas the current classifier fails to achieve satisfactory results) and (ii) among different types of hate (which we defined in the paper and worked with at the annotation level). We will carry out a thorough analysis of the results of the two classifiers to investigate whether they can be combined in order to increase the performance. In addition, we are enlarging the annotation process, both to increase the corpus size and to collect more annotations for a single comment. We are testing new annotation methods, evaluating the inter-annotator agreement for validating the annotation on the different degrees of hate. Besides, we will investigate the effect of sarcasm on the classifier performance. From the classification of single comments, the hate classifier may evolve to detect bursts of hate, thus preventing that virtual discussions give rise to severe injures to people and assets. Given that human moderators cannot monitor the huge user generated texts on social networks, we believe this work represents the basis to track divergent states of Italian texts in online conversations.

Acknowledgements. The authors would like to thank Salvatore Bellomo and Serena Tardelli for their actionable support.

References

[1] Mishari Almishari and Gene Tsudik. Exploring linkability of user reviews. In *ESORICS*, pages 307–324. Springer Berlin Heidelberg, 2012.

[2] Valerio Basile, Andrea Bolioli, Malvina Nissim, Viviana Patti, and Paolo Rosso. Overview of the Evalita 2014 sentiment polarity classification task. In *EVALITA*, 2014.

[3] Peter Burnap and Matthew Leighton Williams. Hate speech, machine classification and statistical modelling of information flows on Twitter. In *Internet, Policy and Politics*, 2014.

[4] Erik Cambria et al. Do not feel the trolls. In *Semantic Web*, 2010.

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[6] S. Chattopadhyay et al. Suicidal risk evaluation using a similarity-based classifier. In *Advanced Data Mining and Applications*, pages 51–61. Springer Berlin Heidelberg, 2008.

[7] Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Antisocial behavior in online discussion communities. *arXiv preprint arXiv:1504.00680*, 2015.

[8] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[9] Andrea Cimino, Stefano Cresci, Felice Dell'Orletta, and Maurizio. Tesconi. Linguistically-motivated and lexicon features for sentiment analysis of italian tweets. In *EVALITA*, 2014.

[10] Mauro Conti, Radha Poovendran, and Marco Secchiero. FakeBook: Detecting fake profiles in on-line social networks. In *Social Networks Analysis and Mining*, pages 1071–1078. IEEE, 2012.

[11] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. A criticism to society (as seen by twitter analytics). In *ICDCS Workshops*, pages 194–200, 2014.

[12] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Sys.*, 80:56–71, 2015.

[13] Maral Dadvar et al. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer, 2013.

[14] Felice Dell'Orletta. Ensemble system for part-of-speech tagging. *Proceedings of EVALITA*, 2009.

[15] Nemanja Djuric et al. Hate speech detection with comment embeddings. In *24th International Conference on World Wide Web*, pages 29–30. ACM, 2015.

[16] Nicole B. Ellison and Danah M. Boyd. Sociality through social network sites. In *The Oxford Handbook of Internet Studies*. Oxford Handbooks Online, 2013.

[17] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv preprint arXiv:1512.05287*, 2015.

[18] Patxi Galán-García et al. Supervised machine learning for the detection of troll profiles in Twitter social network. In *Joint Conf. Soco-Cisis-Iceute*, pages 419–428. Springer, 2014.

[19] Njagi Dennis Gitari et al. A lexicon-based approach for hate speech detection. *Multimedia and Ubiquitous Engineering*, 10(4):215–230, 2015.

[20] Kilem L Gwet. *Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters*. Advanced Analytics, LLC, 2014.

[21] Claire Hardaker. Trolling in asynchronous computer-mediated communication: From user discussions to academic definitions. *Politeness Research*, 6(2):215–242, 2010.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[24] Ralf Klamma, Marc Spaniol, and Dimitar Denev. Paladin: A pattern based approach to knowledge discovery in digital social networks. In *I-KNOW*, volume 6, pages 6–8, 2006.

[25] April Kontostathis. Chatcoder: Toward the tracking and categorization of internet predators. In *Text Mining Workshop of Siam Data Mining (SDM)*, 2009.

[26] Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell'Orletta, Henrik Dittmann, Alessandro Lenci, and Vito Pirrelli. The PAISA corpus of Italian web texts. In *Web as Corpus Workshop (WaC-9)*, 2014.

[27] Altaf Mahmud, Kazi Zubair Ahmed, and Mumit Khan. Detecting flames and insults in text. In *Natural Language Processing*, 2008.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[29] Preslav Nakov et al. Semeval-2016 task 4: Sentiment analysis in Twitter. In *SemEval@NAACL-HLT 2016*, pages 1–18, 2016.

[30] F Javier Ortega. Detection of dishonest behaviors in on-line networks using graph-based ranking techniques. *AI Communications*, 26(3):327–329, 2013.

[31] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[32] Ellen Spertus. Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI*, 1997.

[33] Duyu Tang et al. Document modeling with gated recurrent neural network for sentiment classification. In *Empirical Methods in Natural Language Processing*, pages 1422–1432, 2015.

[34] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[35] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP*, pages 347–354. ACL, 2005.

[36] Guang Xiang et al. Detecting offensive tweets via topical feature discovery over a large scale Twitter corpus. In *Information and Knowledge Management*, pages 1980–1984. ACM, 2012.

[37] XingYi Xu et al. UNIMELB at SemEval-2016 tasks 4a and 4b: An ensemble of neural networks and a Word2Vec based model for sentiment classification. In *SemEval*, 2016.

[38] Zhi Xu and Sencun Zhu. Filtering offensive language in online communities using grammatical relations. In *Collaboration, Electronic Messaging, Anti-Abuse and Spam*, pages 1–10, 2010.