Detecting Hate Speech for Italian Language in Social Media

Valentino Santucci, Stefania Spina
University for Foreigners of Perugia
{valentino.santucci, stefania.spina}@unistrapg.it

Alfredo Milani
University of Perugia
alfredo.milani@unipg.it

Giulio Biondi, Gabriele Di Bari
University of Florence
{giulio.biondi, gabriele.dibari}@unifi.it

Abstract

English. In this report we describe the hate speech detection system for the Italian language developed by a joint team of researchers from the two universities of Perugia (University for Foreigners of Perugia and University of Perugia). The experimental results obtained in the HaSpeeDe task of the Evalita 2018 evaluation campaign are analyzed. Finally, a suggestion for future research directions is provided in the conclusion.

Italiano. *In questo documento descriviamo il sistema di hate speech detection per la lingua Italiana sviluppato da una squadra di ricercatori dell'Università per Stranieri di Perugia e dell'Università degli Studi di Perugia. I risultati sperimentali ottenuti nel task HaSpeeDe, organizzato nell'ambito di Evalita 2018, sono riportati e analizzati. Infine, una possibile direzione di ricerca è fornita nelle conclusioni.*

1 Introduction

In the recent years there was an exponential growth of social media that has revolutionized communication and content publishing. However, social media are also increasingly exploited for the propagation of hate speech. This issue motivates the recent research on hate speech detection systems (Zhang and Luo, 2018; Waseem and Hovy, 2016; Del Vigna et al., 2017; Davidson et al., 2017; Badjatiya et al., 2017; Gitari et al., 2015).

In this paper, we provide the description of our hate speech detection system for the Italian language. The system, namely HSD4I_PG, has been developed by a joint team of researchers from the University for Foreigners of Perugia and the University of Perugia. The code of HSD4I_PG is provided online at the url https://github.com/Gabriele91/HSD4I_PG.

The rest of the paper is organized as follows. The main system architecture is provided in Section 2, while the single software components are described in Sections 3-6. Experimental results are provided in Section 7, while conclusion and future lines of research are depicted in Section 8.

2 Architecture of the Hate Speech Detector

The hate speech detector we have developed, namely HSD4I_PG, is composed by several software components:

- a *tokenizer* for Italian posts from social media,
- the popular *FastText* tool (Bojanowski et al., 2016) used to generate a word embedding model,
- a *features generator* that generates a vector of numeric features for each post to be classified,

- a (trainable) *classifier* that, for each post, predicts its class label.

Moreover, the following resources have been adopted:

- the *Ita_Twitter* corpus (Spina, 2016) that includes 1,234,865 tweets extracted from the Italian timeline in a time span of seven months (November 2012 - May 2013). The tweets were extracted randomly, 2,000 per day, using the R package TwitteR (https://cran.r-project.org/web/packages/twitteR/);
- the *Italian Lexicon of Hate Speech* that was collected based on an Italian monolingual dictionary, Il Nuovo De Mauro, which is also available online (https://dizionario.internazionale.it);
- the *Sentix* italian lexicon for sentiment analysis (Basile and Nissim, 2013);
- the training sets of 3,000 Facebook posts and 3,000 tweets available for the "Haspeede" task of Evalita 2018.

As any other supervised classifier system, HSD4I_PG requires a training stage, that is depicted in Figure 1. The word embedding model is trained by FastText using the Ita_twitter corpus. Numeric features are obtained by aggregating the FastText features and by generating some ad-hoc extra-features. These numeric features are finally fed to a Support Vector Machine (SVM) (Cortes and Vapnik, 1995) in order to generate a classifier model.

After the SVM classifier has been trained, the prediction of (unlabeled) posts is performed following the scheme depicted in Figure 2.

3 The Tokenizer

A tokenizer for the Italian language adopted in social media has been designed by modifying the output produced by the "TweetTokenizer" class of the popular Python library NLTK (Bird et al., 2009).

A variety of corrections have been introduced. The most important ones are:

1. two or more consecutive occurrences of the same vowel have been replaced by a single occurrence (e.g., "ciaooo" is replaced with "ciao"),
2. alternative spellings of some bad words have been normalized (e.g., "vaffa" is replaced with its most popular form),
3. some common mispellings and abbreviations have been corrected (e.g., "cmq" is replaced with "comunque"),
4. hashtags have been split into multiple tokens using the Python library "compound-word-splitter",
5. apostrophes have been considered as token separators,
6. tokens composed by digits characters have been replaced with the token `NUM`,
7. tokens corresponding to Twitter mentions have been replaced with the token `MEN`,
8. tokens corresponding to web links have been replaced with the token `URL`,
9. emojis have been kept as tokens on their own, while other punctuation characters have been removed,
10. all the textual tokens have been replaced with their stemmed form by using the NLTK implementation of the Snowball stemming algorithm for the Italian language (Porter, 1980).

Moreover, in order to provide additional experimental results, we have also tried a lighter variant of the tokenizer that only perform the tasks numbered from 5 to 10.

4 The Word Embedding Model

A word embedding model is generated by FastText (Bojanowski et al., 2016) using the skipgram technique.

Fed with the Ita_Twitter corpus, FastText produces a numeric vector representation for every n-gram contained in the corpus' posts in such a way that the n-grams belonging to tokens appearing in similar contexts are close to each other in the continuous numerical space.

After the model has been generated, a numeric representation for a given token w can be simply computed by summing up the numeric representations of the n-grams that compose w.

Since out-of-vocabulary words are quite common in social media texts, we think that the subwords information contained in the n-grams is particularly useful in our scenario.

5 The Features Generator

The word embedding model allows to generate a numeric representation for every token. Therefore, in order to produce a (constant length) numeric representation of the whole post, we need to aggregate the vectors corresponding to the tokens of the post. Six different aggregation functions have been considered: average (`avg`), standard deviation (`std`), minimum (`min`), maximum (`max`), median (`med`), and sum (`sum`). Any combination of these aggregators can be adopted, thus the features generator requires an experimental tuning (see Section 7).

Moreover, 20 additional extra-features have been introduced:

- number of hateful tokens, computed using the Italian Lexicon of Hate Speech (Spina, 2016),
- average sentiment polarity and intensity, computed using the Sentix lexicon (Basile and Nissim, 2013),
- number of web links,
- number of mentions,
- a boolean flag to indicate if it is a reply tweet or not,
- number of hashtags,
- maximum length of an hashtag (in characters),
- a boolean flag to indicate if it is a retweet or not,
- the percentage of capital letters,
- the percentage of tokens whose letters are all in capital case,
- number of exclamation marks,
- number of tokens composed by three or more dots,
- number of punctutation characters,
- number of emojis,
- number of repeated consecutive vowels,
- percentage of tokens representing a correct Italian word,
- post length in number of characters,
- post length in number of tokens.

As an illustrative example, let consider that: FastText has generated numeric vectors of size 300 for every single token w of a post p, and that the combination of the three aggregators `sum, min, max` has been chosen. Then, the numeric vector representing p has $300 \times 3 + 20 = 920$ dimensions and it is formed by concatenating the three vectors, each one of size 300, given by every chosen aggregator together with the 20 extra-features.

Finally, in the case the number of features is too large for the classifier, during the training phase we are able to reduce the dimensionality to a given number k by selecting the features having the largest mutual information with respect to the class labels.

6 The Classifier

After some preliminary experiments, we have decided to adopt a Support Vector Machine (SVM) classifier (Cortes and Vapnik, 1995). SVM is a supervised technique for training a classifier model by efficiently computing a separation hyperplane (between the two classes to be predicted) in a (implicitly) higher dimensional space (with respect to the features dimensionality). The SVM implementation of the Python's library Scikit-Learn (Pedregosa et al, 2011) has been used.

Compared to the popular neural network model, the SVM technique has less parameters to be tuned, it is computationally more efficient, and it generally obtains comparable performances.

Finally, it is important to note that, before the training phase, all the training features have been standardized in such a way that their means and variances, across all the training instance, are, respectively, 0 and 1.

7 Experiments

7.1 Experimental Setting

The parameters of the different software components of HSD4I_PG have been tuned using a grid search approach and a 10-folds cross-validation scheme.

FastText parameters have been chosen in the following ranges: number of epochs `epoch` $\in \{5, 20, 50, 100\}$, the initial learning rate `lr` $\in$

$\{0.05, 0.1\}$, the negative sampling `neg` $\in \{5, 20, 50\}$, the window size `ws` $\in \{5, 10\}$. Moreover, the skipgram model has been considered, while other FastText parameters that have been set to constant values are: `dim` = `300`, `minCount` = 1, `minn` = 3, and `maxn` = 6.

Regarding the features generator (see Section 5), a combination of the six aggregators has to be chosen. Importantly, for combinations resulting in more than 1,000 features, the filtering procedure described at the end of Section 5 is performed.

After some preliminary experiments, we have decided to use the following ranges in order to tune the SVM parameters: `kernel` $\in$ {`rbf`, `linear`}, `C` $\in \{1.8, 2, 2.2, 2.4\}$. Moreover, the `gamma` and `class_weight` parameters have been set to, respectively, `auto` and `balanced`.

The best parameter setting resulting from the experimental tuning is provided in Table 1.

	Parameter	**Value**
FastText	`epoch`	50
	`lr`	0.05
	`ns`	50
	`ws`	5
Features Generator	aggregators	`sum`
		`min`
		`max`
SVM	`kernel`	`rbf`
	`C`	2.2

Table 1: Tuned parameter setting

This setting has been used to generate the results submitted as "run 2" at the Haspeede task of Evalita 2018 by the team "Perugia1". For a mistake, we have submitted a wrong file as "run 1". Anyway, in the following section we also provide the results of three additional executions of HSD4I_PG:

Execution A) It uses the same setting of Table 1 except that $C = 2$,

Execution B) It uses the same setting of Table 1 except that the lighter variant of the tokenizer (see Section 3) has been adopted,

Execution C) It uses the same setting of Table 1 except that $C = 2$ and the lighter variant of the tokenizer (see Section 3) has been adopted.

7.2 Experimental Results

Table 2 provides the results obtained by HSD4I_PG in the four proposed tasks. In particular, the Macro-Average F1 score for each subtask is shown, along with the difference from the best competitor in the subtask.

SubTask	**HSD4I_PG**	**Distance from best**
HaSpeeDe-FB	0.7841	0.0447
HaSpeeDe-TW	0.7744	0.0249
Cross-HaSpeeDe-FB	0.6279	0.0262
Cross-HaSpeeDe-TW	0.5545	0.1440

Table 2: Subtask results of HSD4I_PG

Table 2 shows that HSD4I_PG achieved results comparable to the best competitors, except in the task Cross-HaSpeeDe-TW. The complete results for all the tasks are available in (Bosco et al., 2018). Besides, in Tables 3 and 4, three additional rows corresponding to the new executions A,B,C previously discussed (and performed after the official HaSpeeDe evaluation) are provided.

Interestingly, the results in Table 4 show that HSD4I_PG, tuned with different parameter settings, would have ranked 3rd in the HaSpeeDe-TW subtask (see (Bosco et al., 2018)).

8 Conclusion and Future Work

In this paper we have introduced a system for the hate speech detection of social media texts in Italian language. The results we have obtained for the HaSpeeDe task of the Evalita 2018 campaign are provided.

It is worth to point out that the results of most participants are very similar and quite far from being fully accurate. The question is whether hate annotation is objective or subjective. Few of the posts in the datasets looks to be difficult to annotate even for a human being. Indeed, we think that different people can produce different annotations. Therefore, it can be interesting to model the subjective perception of hatefulness and exploit such information in the detection task, perhaps, taking inspiration by recommender system techniques.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web*

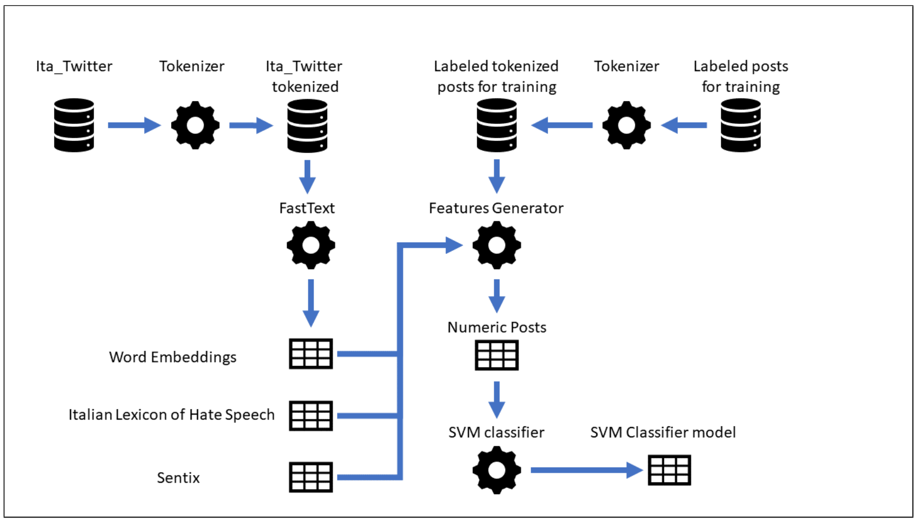

Figure 1: Training in HSD4I_PG

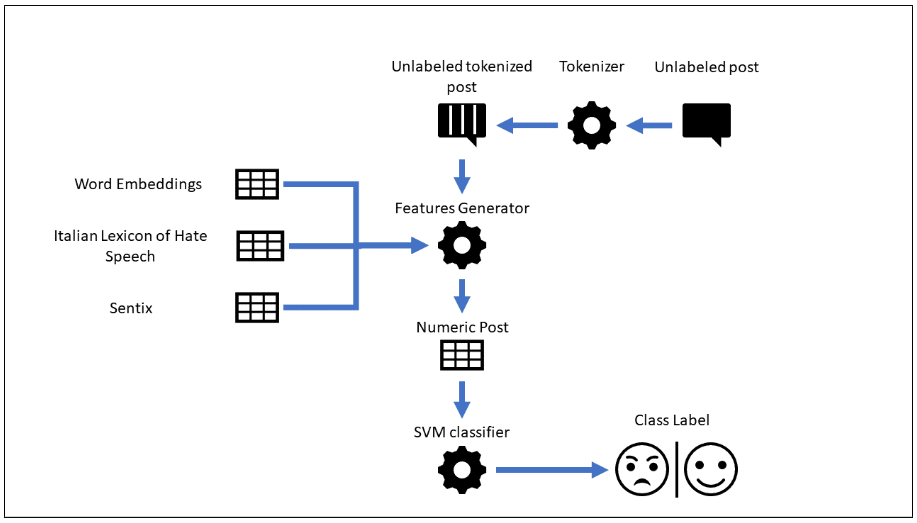

Figure 2: Classification in HSD4I_PG

	Not HS			HS			Macro-Avg F-score
	Precision	Recall	F-score	Precision	Recall	F-score	
A	**0.7261**	**0.6811**	**0.7029**	**0.8522**	**0.8774**	**0.8646**	**0.7838**
B	**0.7219**	**0.6749**	**0.6976**	**0.8496**	**0.8759**	**0.8625**	**0.7801**
C	**0.7166**	**0.6811**	**0.6984**	**0.8514**	**0.8715**	**0.8715**	**0.7799**

Table 3: Additional results in the subtask HaSpeeDe-FB

	Not HS			HS			Macro-Avg F-score
	Precision	Recall	F-score	Precision	Recall	F-score	
A	**0.8489**	**0.8728**	**0.8607**	**0.7180**	**0.6759**	**0.6963**	**0.7785**
B	**0.8545**	**0.8950**	**0.8743**	**0.7568**	**0.6821**	**0.7175**	**0.7959**
C	**0.8575**	**0.8905**	**0.8737**	**0.7517**	**0.6914**	**0.7203**	**0.7970**

Table 4: Additional results in the subtask HaSpeeDe-TW

Companion - WWW '17 Companion, pages 759–760, New York, New York, USA. ACM Press.

Valerio Basile and Malvina Nissim. 2013. Sentiment Analysis on Italian Tweets. In *In Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Atlanta, Georgia, 14 June 2013*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. 7.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the Evalita 2018 Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. 3.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on Facebook. In *CEUR Workshop Proceedings*.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*.

Fabian Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137, 3.

Stefania Spina. 2016. *Fiumi di parole. Discorso e grammatica delle conversazioni scritte in Twitter*. StreetLib, Loreto, Italy.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*.

Ziqi Zhang and Lei Luo. 2018. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter. 2.