Text analysis for hate speech detection in Italian messages on Twitter and Facebook

Giulio Bianchini
University of Perugia
Italy
giulio.bianchini@studenti.unipg.it

Lorenzo Ferri
University of Perugia
Italy
lorenzo.ferri@studenti.unipg.it

Tommaso Giorni
University of Perugia
Italy
tommaso.giorni@studenti.unipg.it

Abstract

English. In this paper, we present a system able to classify hate speeches in Italian messages from Facebook and Twitter platforms. The system combines several typical techniques from Natural Language Processing with a classifier based on Artificial Neural Networks. It has been trained and tested on a corpus of 3000 messages from the Twitter platform and 3000 messages from the Facebook platform. The system has been submitted to the HaSpeeDe task within the EVALITA 2018 competition and the experimental results obtained in the evaluation phase of the competition are presented and discussed.

Italiano. *In questo documento presentiamo un sistema in grado di classificare messaggi di incitamento all'odio in lingua italiana presi dalle piattaforme Facebook e Twitter. Il sistema combina diverse tecniche tipiche del Natural Language Processing con un classificatore basato su una Rete Neurale Artificiale. Quest'ultimo stato allenato e testato con un corpus di 3000 messaggi presi dalla piattaforma Twitter e 3000 messaggi presi dalla piattaforma Facebook. Il sistema stato sottomesso al task HaSpeeDe relativo alla competizione EVALITA 2018, e sono presentati e discussi i risultati sperimentali ottenuti nella fase di valutazione della competizione.*

1 Introduction

In the last years, social networks have revolutionized in a radical way the world of communication and the publication of contents. However, if on one hand social networks represent an instrument of freedom of expression and connection, on the other hand they are used for propagation and incitement to hatred. For this reason, recently, many softwares and technologies have been developed to reduce this phenomenon (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).

Specifically, approaches based on machine learning and deep learning are used by large companies to stem and stop this widespread fact. Despite the efforts spent to produce systems for the English language, there are very few resources for Italian (Del Vigna et al., 2017). In order to bridge this gap, a specific task (Bosco et al., 2018) for the detection of hateful contents has been proposed within the context of EVALITA 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian. The EVALITA team provided the participants with the initial starting data sets, each consisting of 3000 classified comments taken respectively from Facebook and Twitter pages. The objective of the competition is to produce systems able to automatically annotate messages with boolean values (1 for message containing Hate Speech, 0 otherwise).

In this paper we describe the system submitted by the Vulpecula team. The system works in four phases: preprocessing of the initial dataset; encoding of the preprocessed dataset; training of the Machine Learning model; testing of the trained model. In the first phase, the comments were cleaned by applying text analysis techniques and some features have been extrapolated from these; then in the second phase, using a trained Word2Vec model (Mikolov et al., 2013), the comments were coded in a vector of 256 real num-

bers. In the third phase, an artificial neural network model was trained using the encoded comments as input along with their respective extra features. In order to have better and reliable results, to train and evaluate the model a cross-validation was used. In addition, together with accuracy and evaluation of the error, the F-measure were used to evaluate the quality of the model. Finally, in the fourth and last phase, the test set comments provided by EVALITA were classified. The rest of the paper is organized as follows. A system overview is provided in Section 2, while some details about the system components and the external tools used are provided in Section 3. Experimental results are shown and discussed in Section 7, while some conclusions and ideas for future works are depicted in Section 8.

2 System overview

The system has a structure similar to (Castellini et al., 2017); it has been organized into four main phases: preprocessing, encondig, training, testing, as also shown in Figure 1.

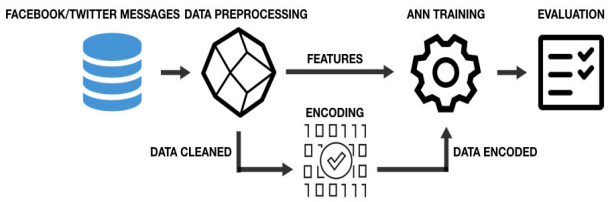

Figure 1: System architecture.

- In the first phase the corpus of 3000 Facebook comments, and the corpus of 3000 Twitter comments are cleaned and prepared to be encoded. In parallel within the cleaning we have extrapolated some interesting features for each comment. The entire phase is explained in details in section 4 and 5.

- In the second phase we trained a Word2Vec model, starting from 200k comments we download from some Facebook pages known to contain hate messages. Each of the initial data set comment has been encoded in a vector of real values by submitting it to the Word2Vec model. This phase is explained in details in the section 5.

- In the third phase we trained a multi-layer feed-forward neural network using the 3000 encoded comments and the respective features we extracted in the first phase. The description of the ANN is in the section 6.

- In the last phase the test set comments provided by EVALITA were classified and we joined the competition. This phase is explained in details in the section 7.

The source code of the project is provided online[1].

3 Tools Used

The entire project was developed using the Python programming language, for which several libraries are available and usable for the purpose of the project. Specifically, the following libraries were used for the preprocessing phase of the dataset:

- nltk: toolkit for natural language processing;

- unicode_emoji: library for the recognition and translation of emoticons;

- treetaggerwrapper: library for lemming and word tagging;

- textblob: another library for natural language processing;

- gensim: library that contains word2vec;

- sequence matcher: library for calculating the spelling distance between words;

For the training phase of the ML model the following libraries were used:

- keras (Chollet and others, 2015): High-level neural network API;

- sklearn (Pedregosa et al., 2011): Simple and efficient tools for data mining and data analysis ;

Finally, some corpora have been used:

- SentiWordNet (Baccianella et al., 2010);

- dataset of badwords, provided by Prof. Spina and research group of the University for Foreigners of Perugia;

- dataset of italian words;

[1] https://github.com/VulpeculaTeam/Hate-Speech-Detection

- dataset of 220k comments downloaded from Facebook pages (*Italia agli italiani stop ai clandestini, matteo renzi official, matteo salvini official, noiconsalvini, politici corrotti*)

4 Preprocessing

In the Knowledge Discovery in Databases (KDD) one of the crucial phases is data preparation. In this project this phase was tackled and the comments given to us were processed and prepared. Specific text analysis techniques have been applied in order to prepare the data in the best possible way in order to extract the most important information from them. All the operations performed for the data cleaning and for the extra-feature extraction are listed below. Each operation is iterated for all the 3000 comments of the data set.

- Extraction of the first feature: length of the comment.

- Extraction of the second feature: percentage of words written in CAPS-LOCK inside the comment. Calculated by the number of words written in CAPS-LOCK divided by the number of words in the comment.

- Replace the characters '&', '@' respectively in the letters 'e', 'a'.

- Conversion of disguised bad words. An interesting function added to the preprocessing is the recognition of censored bad-words, i.e. bad-words where some of their middle letters are replaced by special character (symbol, punctuation...) to make it recognizable by an human but not by a computer. At this scope we don't use a large vocabulary but it's better a simple list of most common bad-words censored (because only a small group of bad words is commonly censored). At this python function we pass an entire sentence creating a list splitting this by space. We scan the list of sentence words and we control if the first and last characters are letters and not number or symbols. Then we take this word without first and last letters and control if this middle sub-word is formed by special symbols/punctuation or by letter x (because "x" is often used for hiding bad-words). If yes, this middle sub-word is deleted from the censored bad-word, taking the top and end part of this formed by letters. At the end we scan the list of bad-words and we control if this top and end part matching with one of this scanned bad-words. If yes, this is replaced by the real word.

- Hashtag splitting. One of the most difficult cleaning phases is the Hashtag Splitting. For this we used a large dictionary of italian words in .csv format. First, we scan every word in this file and we control if these word is in the hashtag and then (for convenience we avoid the words of lenght 2) saving it in a list. In this phase will be taken also useless words not contextualized to the hashtag, so we will need to filter them. For this, first we sort all found words in decreasing length and we scan the list. So, starting to the first word on, we delete it from the hashtag. In this way the useless words in the list contained in larger words are found, saved in another list, and deleted from the beginning list containing all the words (both useful and useless) in the hashtag. In the final phase for each word in the resulting list we find its position within the hashtag and with this we create the real sentence, separating every word with a space.

- Removal of all the links from the comment.

- Editing of each word in the comment by this way: removal of nearby equal vowels, removal of nearby equal consonants if they are more than 2. Examples: from *"caaaaane"* to *"cane"*, from *"gallllina"* to *"gallina"*.

- Extraction of the third feature: number of sentences inside the comment. By sentence we mean a list of words that ends with *'.'* or *'?'* or *'!'*.

- Extraction of the fourth feature: number of *'?'* or *'!'* inside the comment.

- Extraction of the fifth feature: number of *'.'* or *','* inside the comment.

- Punctuation removal.

- Translation of emoticons (for Twitter messages). Given the large presence of emoticons in Twitter messages, it was decided to

translate the emoticons with the respective English translations. To do this, each sentence is scanned and if there are emoticons, these are translated into their corresponding meaning in English. Using the library unicode_emoji,

- Emoticon removal.
- Replacement of the abbreviations with the respective words, using a list of abbreviations created by ourselves.
- Removal of articles, pronouns, prepositions, conjunctions and numbers.
- Removal of the laughs.
- Replacement of accented characters with their unaccented characters.
- Lemmatization of each comment with the treetaggerwrapper library.
- Extraction of the sixth feature : polarity of the message. This feature is compute using the SentiWordNet corpora and his APIs. Since SentiWordNet was created to find the polarity of sentences in English, each message is translated using TextBlob in English and the polarity is then calculated.
- Extraction of the seventh feature : Percentage of spelling errors in the comment. To calculate a spelling error a word is compared with all the words of the Italian Vocabulary corpora; if the word is not present in the corpora there is a spelling error. Calculated by the number of spelling error divided by the number of words in the comment.
- Replacement of spelling error: In parallel with the previous step every spelling error is replaced with the most similar word in the Italian Vocabulary corpora. The similarity between the wrong word and all the other is calculated using a function of SequenceMatcher library. The wrong word is replaced with the most similar word in Italian Vocabulary corpora.
- Extraction of the eighth feature: number of bad words in the comment. Every word in the comment is compared with all the word in the Bad Words corpora; if the word is in the corpora it's a bad word.
- Extraction of the ninth feature: percentage of bad words. Calculated by the number of bad words divided by the number of words in the comment.
- Extraction of the tenth feature : Polarity TextBlob. This value is compute using a TextBlob function that allows to calculate the polarity. Also in this case the message is translated into English.
- Extraction of the final feature : Subjectivity TextBlob. Another value computed with a function in TextBlob.

5 Word Embeddings with Word2Vec

Very briefly, Word Embedding turns text into numbers. This transformation is necessary because many Machine Learning algorithms don't work with plain text but they require vectors of continuous values. Word Embedding has fundamental advantages in particular, it is a more efficient representation (dimensionality reduction) and also it is a more expressive representation (contextual similarity). So we have created a Word2Vec model for word embedding. For the training of the model, 200k messages were downloaded from several Facebook pages. These messages were preprocessed as explained in the previous section 4 and (in addiction with the messages provided by EVALITA's team) were used to train the Word2Vec model. The trained model encode each word in a vector of 128 real numbers. Each sentence is instead encoded with a vector of 256 real numbers divided into two components of 128 elements: the first component is the vector sum of the coding of each word in the sentence, while the second component is the arithmetic mean. At this point each of the 3000 comments of the starting training set is a vector of 265 reals: 256 for the coding of the sentence and 9 for the previously calculated features.

6 Model training

The vectors obtained by the process described in section 5 were used as input for training an Artificial Neural Network - ANN. (Russell and Norvig, 2016) The Articial Neural Network mathematical model composed of artificial "neurons", vaguely inspired by the simplification of a biological neural network. There are different types of ANN, the one used in this research is a feed-forward: this

means that the connections between nodes do not form cycles as opposed to recurrent neural networks. In this neural network, the information moves only in one direction, ahead, with respect to input nodes, through hidden nodes (if existing) up to the exit nodes. In the class of feed-forward networks there is the multilayer perceptron one. The network we have built is made up of two hidden layers in which the first layer consists of 128 nodes and the second one is 56. The last layer is the output one and is formed by 2 nodes. The activation functions for the respective levels are sigmoid, relu and softmax and the chosen optimizer is Adagrad, each layer has a dropout of 0.45. The reason why these parameters have been chosen is because after having tried countless configurations, the best results during the training phase have been obtained with these parameters. In particular, have been tried all the possible combinations of these parameters:

- **Number of nodes** of the hidden layers: 56, 128, 256, 512;
- **Activation function** of the hidden layers: sigmoid, relu, tanh, softplus;
- **Optimizer**: Adagrad, RMSProp, Adam.

Furthermore, the dropout was essential to prevent over-fitting. In fact, dropout consists to not consider neurons during the training phase of certain set of neurons which is chosen randomly. The dropout rate is set to 45%, meaning that the 45% of the inputs will be randomly excluded from each update cycle. As methods of estimation, cross-validation was used, partitioning the data into 10 disjoint subsets. As metrics for performance evaluation, the goodness of the model was analyzed by calculating True Positive, True Negative, False Positive and False Negative. From these the cost-sensitive measures precision, recall and f-score were calculated. These are the best results achieved with the training dataset of Facebook comments obtained during the cross validation:

- **Accuracy**: 83.73%;
- **Standard deviation**: 1.09;
- **True Positive**: 1455;
- **True Negative**: 1057;
- **False Positive**: 163;
- **False Negative**: 325;
- **Precision**: 0.899%;
- **Recall**: 0.817%;
- **F1-Score**: 0.856%;
- **F1-Score_Macro**: 0.856%;

7 Experimental Results

After the release of the unlabelled test set, the new 2000 messages (1000 of them from Facebook and 1000 of them from Twitter) were cleaned as explained in section 4 and the respective features were extrapolated. Then, these new comments were added to the comment's pool used to create the Word2Vec model, and a new Word2Vec model was created with the new pool. Finally, the 2000 comments were encoded as previously explained in section 5 in the 265 component vectors and these were the input of the neural network that classified them. From the training phase, two neural network models were built: one trained with the dataset of 3000 Facebook messages and the other trained with the dataset of 3000 Twitter messages. We call the first model **VTfb** and the second one **VTtw**. EVALITA's task consisted in four sub-tasks that were:

- **HaSpeeDe-FB**: test VTfb with the 1000 messages taken from Facebook;
- **HaSpeeDe-TW**: test VTtw with the 1000 messages taken from Twitter;
- **Cross-HaSpeeDe-FB**: test VTfb with the 1000 messages taken from Facebook;
- **Cross-HaSpeeDe-TW**: test VTtw with the 1000 messages taken from Twitter;

Sub-task	Model	F1	Distance
HaSpeeDe-FB	VTfb	0.7554	0.0734
HaSpeeDe-TW	VTtw	0.7783	0.021
Cross-HaSpeeDe-FB	VTfb	0.6189	0.0089
Cross-HaSpeeDe-TW	VTtw	0.6547	0.0438

Table 1: Team results in the HaSpeeDe sub-tasks.

In Table 1 we report the Macro-Average F1 score for each sub-task together with the differences

with the best result obtained in the competition (column "Distance" in the table). Compared with the results we had in the training phases (section 6), we would have expected better results in the HaSpeede-FB task. However, our system appears to be more general and not specifically targeted to a platform, in fact the differences in the other tasks are minimal.

8 Conclusion and Future Work

In this paper we presented a system based on neural networks for the hate speech detection in social media messages in Italian language. Recognizing negative comments is not easy, as the concept of negativity is often subjective. However, good results have been achieved that are not so far from the results obtained by the best within the competition. The proposed system can certainly be improved, an idea can be to use clustering techniques to categorize the messages (cleaned and with the related features) in two subgroups (positive and negative) and then, for each comment, calculate how much this is more similar to negative comments or positive comments and add it as a feature.

Acknowledgments

The authors would like to thank prof. Valentina Poggioni who has helped and supported us in the development of the whole project. A special thanks to Manuela Sanguinetti, our shepherd in EVALITA competition for all the support she has given to us.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of Lrec 2010*, pages 2200–2204.

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 Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

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 2018)*, Turin, Italy. CEUR.org.

Jacopo Castellini, Valentina Poggioni, and Giulia Sorbi. 2017. Fake Twitter Followers Detection by Denoising Autoencoder. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 195–202.

François Chollet et al. 2015. Keras. `https://keras.io`.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

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 *ITASEC*, volume 1816 of *CEUR Workshop Proceedings*, pages 86–95. CEUR-WS.org.

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*, 10(4):215–230.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Stuart J. Russell and Peter Norvig. 2016. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited.

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*, pages 88–93.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *arXiv preprint arXiv:1803.03662*.