Fontana-Unipi @ HaSpeeDe2: Ensemble of Transformers for the Hate Speech Task at Evalita

Michele Fontana
Dipartimento di Informatica
Università di Pisa
m.fontana12@studenti.unipi.it

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa
attardi@di.unipi.it

Abstract

We describe our approach and experiments to tackle Task A of the second edition of HaSpeeDe, within the Evalita 2020 evaluation campaign. The proposed model consists in an ensemble of classifiers built from three variants of a common neural architecture. Each classifier uses contextual representations from transformers trained on Italian texts, fine tuned on the training set of the challenge. We tested the proposed model on the two official test sets, the in-domain test set containing just tweets and the out-of-domain one including also news headlines. Our submissions ranked 4th on the tweets test set and 17th on the second test set.

1 Introduction

The spreading of hateful messages on social media has become a serious issue, therefore techniques of hate speech detection have become quite relevant. The goal of the Hate Speech Detection task (Sanguinetti et al., 2020) at Evalita 2020 (Basile et al., 2020) is to improve the automatic detection of hate messages in Italian tweets. The organizers provided to the participants the dataset HaSpeeDe2, which consists of 6,837 Italian tweets, containing, besides the raw text, also hashtags and emojis. The Task A can be cast into a binary classification task: the model has to predict whether a given message contains hate speech or not.

Approaches based on transformer models have become quite popular recently and have proved effective in reaching state-of-the-art scores on major NLP tasks such as those of the GLUE benchmark (Wang et al., 2018). With our experiments we try to assess the effectiveness of transformers trained on Italian documents in a task involving Italian texts from different sources. We experiments with both a transformer model trained specifically on Italian tweets and one trained on generic web documents.

We combine several instances of classifiers based on these transformers, in order to address the problem of over-fitting due to the small size of the training set.

For this edition of the Evalita HaSpeeDe task, the organizers released two test sets, an in-domain one consisting of tweets and an out-of-domain one containing also news headlines.

The ensemble model of our official submission achieved a competitive score of 78.03 Macro-F1 on the in-domain test set but did not perform as well on the second test set.

We make available the source code for our experiments as Open Source at `https://github.com/mikelefonty/Haspeede2`.

2 Related Work

The first edition of HaSpeeDe was held in 2018. The results produced during this contest were the starting point of our research. As described in (Bosco et al., 2018), most of the systems were based on neural networks and used word embeddings, such as FastText (Grave et al., 2018) or word2vec (Polignano and Basile, 2018) in the first layer of their architecture. The embeddings layer was usually followed by a Recurrent Network or a Convolutional Neural Network to get an internal representation of the input text. This hidden representation was provided as input to a series of dense layers to obtain the final classification result.

Over the last couple of years, the trend in approaches to language analysis has changed considerably, as can be seen by examining the models used in competitions like SemEval 2020 OffensE-

val 2 (Zampieri et al., 2020). In these new models, to get a better text representation, the embedding layer is often replaced by a Transformer (Vaswani et al., 2017) such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), or Multilingual BERT (Devlin et al., 2019).

We followed this trend but we also focused our attention on the problem raised by the small size of the dataset. As Risch and Krestel (2020) mention, transformer models tend to have a high variance with respect to the input dataset, that often leads to overfitting. The authors therefore suggest to implement an ensemble of classifiers to reduce the variance and consequently improve the generalization capabilities of the trained model.

In the following, we describe a similar approach based on the Bagging technique (Breiman, 1996), where we apply three different transformer-based classifiers to populate the ensemble and to get the final prediction.

3 System Architecture

During the design phase of our classifier, we looked for a transformer trained directly on a significantly large collection of Italian texts and particularly on Italian tweets, in order to compensate for the small size of the training data. We found two possible models based on BERT: AlBERTo (Polignano et al., 2019) [1] and DBMDZ [2]. The former is trained on TWITA (Basile et al., 2018), a 191 GB collection of Italian tweets gathered by the authors, and tested on the SENTIPOLC task during the EVALITA 2016 campaign, where it achieved state-of-the-art accuracy in subjectivity, polarity, and irony detection on Italian tweets. We considered this model suitable for hate speech detection, since its source are Italian tweets and the SENTIPOLC task is a classification task similar to ours. DBMDZ instead is trained on a more general domain, from a 13 GB dataset, which includes a dump of Italian Wikipedia and texts from web pages selected from the Opus Corpora. [3] We decided to test both transformer models, assessing their performance through a validation phase on a development set.

These transformers were used in the input stage of all our architectures, providing contextual embeddings for sentences that were fine tuned during training. We designed three architecture variants, which were employed as the basic building blocks to construct the ensembles:

- **ALB-SINGLE**: It consists of a first layer provided by the AlBERTo transformer, followed by a single neuron with a sigmoid activation function.
- **DB-SINGLE**: It follows the same structure of ALB-SINGLE; it just replaces AlBERTo with DBMDZ in the first layer.
- **DB-MLP**: Compared to DB-SINGLE, it adds a new dense layer, using a ReLU activation function, between the transformer and the output neuron.

The final model is an ensemble consisting of a number of instances of each of the above architectures. For each architecture, e.g. ALB-SINGLE, we construct instances in the following way. After initializing the weights randomly within a given interval and generating the training data by applying the bootstrap technique to the original dataset, we start training the model. When that phase is over, we insert the resulting model in the ensemble. We repeat this process several times with different random weights initialization. Note that, due to the random initialization, no two classifiers in the ensemble are identical to each other. More formally, the model consists of N elements,

$$N = N_{AL} + N_{DB} + N_{MLP}$$

where N_{AL}, N_{DB}, N_{MLP} represent, respectively, the number of instances of *ALB-SINGLE*, *DB-SINGLE* and *DB-MLP* classifiers.

In retrospect, it might have been worth while to consider instances of the architecture obtained varying them more thoroughly than just in the initial weights, for example, by changing in the hyper-parameters or number of layers.

Our classification algorithm is a slight generalization of the most classical one, which collects results from each member of the ensemble and outputs the class which gets the majority of predictions over all iterations. The process, described by Algorithm 1, performs n_{run} iterations. During the ith iteration, the algorithm starts sampling randomly from the ensemble a given number of instances for each type of classifier (line 3-5) and initializing to 0 the variable $class1$, which contains the total number of votes that the $hate$ class

[1] https://github.com/marcopoli/AlBERTo-it

[2] https://huggingface.co/dbmdz/bert-base-italian-uncase

[3] http://opus.nlpl.eu/

Algorithm 1 Classification Algorithm

Input: t: the tweet to classify.
Input: $(n_{AL}, n_{DB}, n_{MLP})$: number of classifiers of each type to be sampled.
Input: $(N_{AL}, N_{DB}, N_{MLP})$: number of classifiers of each type in the ensemble.
Input: n_{run}: number of desired iterations.
Output: c_{final}: predicted class

1: $preds = []$
2: **for** $run = 1$ to n_{run} **do**
3: $albs = sample\_al(n_{AL}, N_{AL})$
4: $dbs = sample\_db(n_{DB}, N_{DB})$
5: $mlps = sample\_ml(n_{MLP}, N_{MLP})$
6: $sampled\_classif = albs \cup dbs \cup mlps$
7: $class1 = 0$ // votes for class 1
8: **for** cl in $sampled\_classif$ **do**
9: $class1 \mathrel{+}= cl(t)$ // cl's classification
10: **end for**
11: $preds[run] = \left(class1 \geq \lceil \frac{n_{AL}+n_{DB}+n_{MLP}}{2} \rceil\right)$
12: **end for**
13: $c_{final} = \left[\left(\sum_{i}^{n_{run}} pred[i]\right) \geq \lceil \frac{n_{run}}{2} \rceil\right]$
14: **return** c_{final}

receives during the iteration (line 7). It then collects the predictions of the selected models on the tweet t (line 8-10). $cl(t) \in \{0, 1\}$ represents the prediction of classifier cl for the tweet t; in particular $cl(t) = 1$ if and only if cl classifies t as hateful. The output of iteration i is the most predicted class (line 11). The final result of the algorithm is then the class $c_{final} \in \{0, 1\}$, which obtained the most votes over all the n_{run} iterations (line 13-14). If $c_{final} = 1$, it means that the tweet t has been classified as hateful.

A simpler variant of the algorithm would be to just add the counts of each class by all classifiers in all iterations and return the class with the highest count. We plan to compare these two approaches in a future work.

4 Experiments

In this section we describe the experiments we performed to tune the hyper-parameters of our model. We will focus on the search to choose the best values for n_{DB}, n_{AL}, n_{MLP}, that is how many instances to select at each iteration in the classification algorithm.

Before starting the experiments, we divided the dataset into two disjoint subsets, a development and an internal test set, in the proportion of 80% and 20%, respectively. The split was done by means of Stratified Sampling, according to the distribution of the target variable hs. We applied the Stratified 3-fold-CV technique to validate our model. Given that we are solving a binary classification problem, we picked the Binary Cross Entropy as our loss. We chose AdamW as our optimizer; we set the first 10% of the total steps as warmup steps. We conducted the experiences on a GPU offered by Google Colab [4]. Our models are implemented in PyTorch (Paszke et al., 2019). To extract as much information as possible from input texts, we preprocessed them through hashtag segmentation by means of *Tweet Preprocessor*.[5] We also converted emojis into their Italian description by using the *emoji* [6] and *Google Translate* [7] libraries.

Classifier	Macro-F1	Std
ALB-SINGLE	76.896	0.7266
DB-SINGLE	77.613	0.3251
DB-MLP	78.562	0.521

Table 1: Results of the experiments comparing the baseline architectures. We report the expected value and the standard deviation of the F1 score computed with respect to the 3 validation folds.

We analyzed the behaviour of the three baseline architectures we planned to include in the ensemble.

We trained each model for a maximum of 4 epochs, using a batch of size 16 and setting the maximum text length to 100. A grid search revealed that the optimal learning rate for DB-MLP is $5 \cdot 10^{-5}$, and $6 \cdot 10^{-5}$ for the remaining models. The optimal number of neurons in the hidden layer of DB-MLP is 50.

Table 1 highlights the following aspect: DB-SINGLE achieves better performance than ALB-SINGLE, even though the dataset used to train AlBERTo was composed by a large collection of tweets. The obtained values of the macro-F1 are the baselines of our work.

We then describe the results obtained through

[4] https://colab.research.google.com/
[5] https://pypi.org/project/tweet-preprocessor/
[6] https://pypi.org/project/emoji
[7] https://pypi.org/project/googletrans/

n_{DB}	n_{MLP}	n_{AL}	Macro-F1	Std
20	**25**	**30**	**80.057**	**0.534**
15	20	25	80.038	0.580
15	30	30	80.036	0.585
15	25	30	80.026	0.563
15	30	15	80.020	0.481

Table 2: Ranking of the 5 best configurations we found, varying the number the number of instances selected from the ensemble. n_{DB} stands for the number of instances of the *DB-SINGLE* model, and similarly for n_{MLP} and n_{AL}. We report the expected value and the standard deviation of the F1 score computed with respect to the 3 validation folds.

n_{DB}	n_{MLP}	n_{AL}	Macro-F1	Std
30	0	0	79.074	0.300
0	30	0	79.581	0.3787
0	0	30	79.482	0.596
30	30	30	79.832	0.525

Table 3: Scores by each architecture, both individually and together in the ensemble. We report the average value and the standard deviation of the F1 score computed with respect to the 3 validation folds.

the ensemble model. To build the classifier, we trained 30 instances of each architecture, keeping the same hyper-parameters obtained from the previous grid search. We thus set:

$$N_{AL} = N_{DB} = N_{MLP} = 30$$

We noted that the generalization capability of the ensemble is strictly related to the triple $(n_{DB}, n_{MLP}, n_{AL})$, so we performed another grid search, looking for the optimal combination of the three parameters. Table 2 shows the five best configurations found by this search. The optimal values for the triple, $(20, 25, 30)$, allow the ensemble to achieve an F1-score of 80.0%, with a gain of about 2 points with respect to the score by a single DB-MLP (see Table 1).

We analyzed the contribution of each architecture individually to the ensemble combination. As shown in Table 3, the best results are obtained with instances of all three architectures. Nevertheless, the results presented in Table 2, show that a more balanced combination achieves better accuracy.

Accuracy	Precision	Recall	F1
79.313	78.510	78.685	78.592

Table 4: Results of the final model on the internal test set.

We picked the first configuration from Table 2 for our final model and tested it on the internal test set, obtaining the results shown in Table 4.

5 Results and Discussion

The results of our final model applied to the data of the two official test sets of the competition are shown in Table 5. The model performs pretty well on the in-domain dataset, reaching the $4th$ position in the rankings. However, it did not rank as well in detecting hate speech on the out-of-domain dataset, obtaining an F1-score of just 65.46. The low recall for the hate class highlights that the model fails too often to identify news headlines containing some form of hate speech. In comparison with the official top rankings, listed in Table 6, our model achieved about 12 points below the top score of 77.44% F1.

Surprised by this fact, we investigated more deeply, looking for an explanation for such poor result on the out-of-domain dataset.

We randomly sampled from the test set some hateful headlines missed by the model, some of which are shown in Table 7.

In these headlines, the qualification as hate is implicit and harder to recognize, since it seems due more to the presence of stereotypes (*nomads*, *asylum seekers*, *Muslims*, *foreigners*), than to the presence of explicit hate expressions.

Broadly speaking, we identified some possible reasons for the difference in performance across the two test sets:

- **Linguistic register**: Tweets often exhibit a more informal and colloquial language, while headlines employ a more formal lexicon and a more objective tone. This is a crucial difference in identifying hateful messages: while in tweets the feeling of hatred transpires clearly and directly, in headlines this message is conveyed in a more subtle way, often alluding to concepts from political propaganda or common stereotypes. Prior knowledge about the subject and inference might be necessary

	NOT HATE			HATE				
	Precision	Recall	F1	Precision	Recall	F1	Macro-F1	Position
Tweets	81.93	72.85	77.12	74.89	83.44	78.94	78.03	4
News	71.88	99.37	83.42	96.61	31.49	47.50	65.46	17

Table 5: Results of the submitted model on the official blind test sets.

Tweets		News	
Position	F1 score	Position	F1 score
1	80.88	1	77.44
2	78.97	2	73.14
3	78.93	3	72.56
4	**78.03 (ours)**	4	71.83
5	77.82	5	70.2
6	77.66	17	**65.46 (ours)**

Table 6: Comparison between our final results and the top-5 F1-scores. The values are taken from the official rankings.

Hateful News Headlines
anziana rapinata sull'autobus, i due nomadi in fuga si rifugiano al campo di via Candoni *(elderly woman robbed on the bus, the two fleeing nomads take refuge at the camp on via Candoni)*
Expo: Bordonali, richiedenti asilo in campo base simbolo fallimento governo. *(Expo: Bordonali, asylum seekers in base camp government failure symbol.)*
Il cardinale Müller: "non possiamo pregare come o con i musulmani" *("we cannot pray like nor with Muslims")*
Salvini: "Il calcio? Rimpiango i tre stranieri in campo" *(Salvini: "Soccer? I regret the three foreigners on the field")*

Table 7: Examples of hateful headlines, randomly picked from the out-of-domain test set, that are misclassified by our model.

to decipher the presence of hate. Examining the entire body of the article might have been helpful.

- **Length of text**: Tweets are usually longer than news headlines. Thus, the model has fewer elements to exploit to correctly classify a piece of news.

These difficulties seem to be shared with other submissions which all got lower scores on the out-of-domain dataset. We expected that pretrained contextual embedding would be more effective in addressing the domain adaptation issue. Further experiments would be needed to improve the resilience of our model.

6 Conclusions

We described an ensemble of neural classifiers, relying on contextual embeddings from transformers, for automated detection of hateful content in Italian texts. We presented the general architecture of our base classification models and how they were combined into an ensemble through a bagging technique. We performed extensive experiments to tune our models and the ensemble on a validation test set. The results achieved by our ensemble model on the in-domain test set confirm its ability in detecting hateful tweets; however the same model performed poorly on the out-of-domain dataset, showing particularly an inability to adapt to handling news headlines. We plan to investigate this issue in future research.

References

Valerio Basile, Mirko Lai, and Manuela Sanguinetti. 2018. Long-term social media data collection at the university of turin. In Elena Cabrio, Alessandro Mazzei, and Fabio Tamburini, editors, *Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Torino, Italy, December 10-12, 2018*, volume 2253 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro. 2020. Evalita 2020: Overview of the 7th evaluation campaign of natural language processing and speech tools for italian. In Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro, editors, *Proceedings of Seventh Evaluation Campaign of Natural Language Processing and*

Speech Tools for Italian. Final Workshop (EVALITA 2020), Online. CEUR.org.

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 Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy, December 12-13, 2018*, volume 2263 of *CEUR Workshop Proceedings*. CEUR-WS.org.

L. Breiman. 1996. Bagging predictors. *Machine Learning*, 24:123–140.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Marco Polignano and Pierpaolo Basile. 2018. Hansel: Italian hate speech detection through ensemble learning and deep neural networks. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy, December 12-13, 2018*, volume 2263 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Marco Polignano, Pierpaolo Basile, Marco de Gemmis, Giovanni Semeraro, and Valerio Basile. 2019. Alberto: Italian BERT language understanding model for NLP challenging tasks based on tweets. In Raffaella Bernardi, Roberto Navigli, and Giovanni Semeraro, editors, *Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019*, volume 2481 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Julian Risch and Ralf Krestel. 2020. Bagging BERT models for robust aggression identification. In Ritesh Kumar, Atul Kr. Ojha, Bornini Lahiri, Marcos Zampieri, Shervin Malmasi, Vanessa Murdock, and Daniel Kadar, editors, *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, TRAC@LREC 2020, Marseille, France, May 2020*, pages 55–61. European Language Resources Association (ELRA).

Manuela Sanguinetti, Gloria Comandini, Elisa Di Nuovo, Simona Frenda, Marco Stranisci, Cristina Bosco, Tommaso Caselli, Viviana Patti, and Irene Russo. 2020. HaSpeeDe 2@EVALITA2020: Overview of the EVALITA 2020 Hate Speech Detection Task. In Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro, editors, *Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020)*, Online. CEUR.org.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çagri Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *CoRR*, abs/2006.07235.