UO @ HaSpeeDe2: Ensemble Model for Italian Hate Speech Detection

Mariano Jason Rodriguez Cisnero
Universidad de Oriente
Santiago de Cuba, Cuba
mjasoncuba@gmail.com

Reynier Ortega Bueno
Universidad de Oriente
Santiago de Cuba, Cuba
reynier@uo.edu.cu

Abstract

English. This document describes our participation in the Hate Speech Detection task at Evalita 2020. Our system is based on deep learning techniques, specifically RNNs and attention mechanism, mixed with transformer representations and linguistic features. In the training process a multi task learning was used to increase the system effectiveness. The results show how some of the selected features were not a good combination within the model. Nevertheless, the generalization level achieved yield encourage results.

1 Introduction

Modern societies found easy and interesting ways for sharing information via Social Media. Users discover freedom to express themselves through online communication. Even if the ability to freely express oneself is a human right, some users take this opportunity to spread hateful content. A dangerous and hurtful potential arises with this kind of information. Recognizing automatically such content is an interesting topic for researchers.

Creative methods have been proposed to tackle the fascinating task of recognizing hate in texts (De la Pena Sarracén et al., 2018; Gambäck and Sikdar, 2017). Some of those works face the problem using feature extraction (Schmidt and Wiegand, 2017) and classification algorithms like SVM (Santucci et al., 2018). In the last years, Deep Learning approaches have become one of the most successful research areas in Natural Language Processing (NLP). There are exciting investigations about this topic, such as (Cimino et al., 2018), involving LSTM (Liu and Guo, 2019) and transformers (Vaswani et al., 2017) that gain attention in NLP community due to their results.

We propose a model based on multiple representations learned by means of deep learning techniques and linguistic knowledge. Particularly a Long Short Term Memory architecture mixed with linguistic features and language model representations given by a special kind of transformer model, BERT.

The paper is organized as follows. The Section 2 introduces a brief description of HaSpeeDe Task. Our hate detection system is presented in Section 3. The experiments and results are discussed in Section 4. Finally, in Section 5 the conclusions and future directions are given. The code of this work is available on GitHub: https://github.com/mjason98/evalita20_hate

2 HaSpeeDe2 Task

Hate speech and stereotypes recognition on social media have become an attractive research area from the computational point of view. In the second edition of HaSpeeDe (Sanguinetti et al., 2020) at Evalita 2020 (Basile et al., 2020), the organizers proposed to address three subtasks. The main subtask is the subtask A, which aims at determining the presence or absence of hateful content in a text. The dataset is composed by 6839 short texts, 2766 labeled as hate speech and 4076 as not hate speech. In this work we focused only on subtask A. The subtask B consists of a binary classification problem oriented to stereotypes' detection. The last subtask C is a sequence labeling task aims at recognizing Nominal Utterances in hateful tweets.

3 Our Proposal

We dealt with hate detection task as a text classification problem to classify "hateful" or "no hateful" categories. We train a deep learning model based on attention mechanism and Recurrent Neural Networks, specifically a Bidirectional Long Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) mixed with linguistic features and transformers representations by means of an interpretable multi-source fusion component (Karimi et al., 2018).

In Section 3.1 and Section 3.2 we describe the linguistic features and the transformer representation used in this work. The Section 3.3 presents the preprocessing phase. Finally, the neural network model and the feature ensemble are described in Section 3.4.

3.1 Linguistic Feature

To build the hate detection model, we start by extracting several sets of linguistic features:

WordNet Features: We count the number of verbs, adverbs, nouns and adjectives. Also, for every word, we calculated the average of its similarity with respect to the others using the $similarity\_path$ function provided by the wordnet[2] corpus. Furthermore, we consider the degree of lexical ambiguity by counting the number of *synsets* of each word within the text.

Hurt and Sentiment content: HurtLex (Bassignana et al., 2018) is a lexicon of offensive, aggressive, and hateful words in over 50 languages. The words according to the 17 categories offered by the lexicon are counted and added as linguistic features jointly with polarity and semantic values obtained from SenticNet (Cambria et al., 2018) corpus.

Information Gain: Information gain (Lewis, 1992) had been a good feature selection measure for text categorization. It takes into account the presence of the term in a category as well as its absence and can be defined by:

$$IG(t_k, C_i) = \sum_{C} \sum_{t} p(t, C) \cdot \log_2 \frac{p(t, C)}{p(t) \cdot p(C)}$$

where $C \in \{C_i, \bar{C_i}\}$ and $t \in \{t_k, \bar{t_k}\}$. In this formula, probabilities are interpreted on an event space of documents, where $p(\bar{t_k}, C_i)$ is the probability that, for a random document d, term t_k does not occur in d who belongs to category C_i. In our case, categories were two: hateful and no hateful, and the term is the word's lemma.

To create the information gain feature (IgF), we calculated the IG for every word and the highest ones are chosen[3]. Then, the occurrence of those selected words in the text are counted.

3.2 Italian BERT

Finally, we use a pre-trained BERT[4] to accomplish the calculation of a deep representation of the text. One of the most widely used auto-encoding pre-trained Language Models (PLMs) is BERT (Devlin et al., 2018). BERT is trained using the masked language modeling task that randomly masks some tokens in a text sequence, and then independently recovers the masked tokens by conditioning on the encoding vectors obtained by a bidirectional Transformer.

Inside BERT, the information is passed forward crosswise transformer layers. In this work, we used a specific output from one of those layers, this operation can be expressed by:

$$\begin{aligned} h_0 &= Hl_0(text_{tok}) \\ h_i &= Hl_i(h_{i-1}) \\ h_n &= Hl_n(h_{n-1}) \end{aligned}$$

where $text_{tok}$ is the text after its tokenization[5], h_i is the output of the i^{th} transformer layer(Hl_i) called $hidden\_state$ and n is the total transformer layers in BERT. Then, for an specific i, from the tensor of order 2 h_i it is computed the vector f_{bert}, as a deep representation of the initial text who will act as PLM feature.

$$v = \sum_{k=0} h_i[k, :] \qquad f_{bert} = \frac{v}{||v||}$$

3.3 Preprocessing

In the preprocessing step, firstly stopwords were removed . Then, the hashtags composed of many words are split (e.g: #NessunDorma becomes # nessun dorma). We use a regular expression[6] algorithm to archive this step.

Secondly, using the FreeLing[7] tool we obtain for each word it lemma, and non alphanumeric characters are removed. Finally, the remaining words are represented as vectors using a pre-trained word embedding generated by Word2Vec model (Mikolov et al., 2013).

[2]The *wordnet* came from the python library *nltk*

[3]We selected the top 50 words with highest IG value.

[4]https://huggingface.co/dbmdz/bert-base-italian-cased

[5]The text is represented as a vector of integers using the *tokenizer* function in BERT Model

[6]The automaton was created using the *re* library from python and the words from an italian corpus.

[7]http://nlp.lsi.upc.edu/freeling/index.php

3.4 The Deep Ensemble Model

The standard LSTM receives sequentially at each time step a vector x_t and produces a hidden state h_t. Each hidden state h_t is calculated as follow:

$$
\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \\
u_t &= \sigma(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \\
c_t &= i + t \oplus + f_t \oplus c_{t-1} \\
h_t &= o_t \oplus tanh(c_t)
\end{aligned} \tag{1}
$$

Where all $W^{(*)}$, $U^{(*)}$ and $b^{(*)}$ are parameters to be learned during training. Function σ is the sigmoid function and $\otimes$ stands for element-wise multiplication.

Bidirectional LSTM, on the other hand, makes the same operations as standard LSTM but, processes the incoming text in a left-to-right and a right-to-left order in parallel. Thus, it output become $\hat{h_t} = [\overrightarrow{h_t}, \overleftarrow{h_t}]$ for the two directions.

By adding an attention mechanism, we allow the model to decide which part of the sequence "attends to". First, lets define the *softmax* function $\pi(v)$ for a vector $v = [v_0, \cdots, v_{n-1}]$ as:

$$\pi(v) = \frac{e^v}{\sum_{i=0} e^{v_i}}$$

Then, let $I \in \mathbb{R}^{N \times L}$ be the matrix of input vectors, where L the size of then and N the length of the given sequence. We define the attention layer (AttLSTM), as a regular LSTM layer like (1) with extra operations described as follow:

$$
\begin{aligned}
a_{k,t} &= \pi(W_k \cdot h_{t-1}^T + b_k) & \alpha_{k,t} &= a_{k,t}^T \cdot I \\
\beta_t &= [\alpha_{0,t}, \cdots, \alpha_{S-1,t}] & x_t &= W_a \cdot \beta_i + b_a
\end{aligned} \tag{2}
$$

Here $k \in [0, S-1]$ represents the number of attention's heads, $W_k \in \mathbb{R}^{N \times M}$ where M is the size of the hidden state vector h_t, $W_a \in \mathbb{R}^{M \times SM}$, b_a and b_k are learnable parameters. The $(*)^T$ is the transpose operation and the output of the layer is $O = [h_0, ..., h_t, ..., h_N]$, a concatenation of the hidden states produced by the AttLSTM at each time step.

As mentioned before, we propose a feature ensemble by using an interpretable multi-source fusion component (IMF). The IMF aims to combine features from different sources. A naive way of doing this is concatenating the vector representations into a single vector. This scheme considers all sources equally, but one source may yield a better result than others. With IMF we propose to consider the contribution of every source of feature via an attention mechanism. The IMF can be expressed by:

$$r_i = \tanh(W_{p_i} f_i + b_{p_i})$$

where r_i represents a projection of f_i, the i^{th} feature vector passed to IMF ensuring that every r_i have the same size. In this step, all the W_{p_i}, b_{p_i}, W_a and b_a are parameters to be learned during training, then:

$$
\begin{aligned}
a_i &= W_a r_i + b_a & \alpha_i &= \pi(a_i) \\
\beta_i &= \alpha_i r_i & z &= \sum_{k=0} \beta_k
\end{aligned} \tag{3}
$$

where α_i represents the importance of r_i to the final calculation of z, the IMF outcome.

To increase the learning power of our system, we used a multitask learning (Caruana, 1997) in which we predict the polarity of tweets in parallel with the classes of the hate speech detection subtask. This approach have been developed before (Cimino et al., 2018) in HaSpeede at Evalita 2018 (Bosco et al., 2018). The tweets used to accomplish the multitask learning are extracted from the Sentipolc-2016 (Barbieri et al., 2016) challenge.

Finally we present the composition of the previous layers and features to create our deep ensemble model:

$$
\begin{aligned}
E &= [w_0, w_1, \cdots, w_{N-1}] \\
o_{b1} &= BiLSTM(E)
\end{aligned} \tag{4}
$$

where E represents the vector representation of the text, see Section 3.3. Equation (4) is the first block of our model, and the second block can be described as follow:

$$
\begin{aligned}
A &= AttLSTM(o_{b1}) \\
m_i &= \max_{j=0,\cdots,N-1} A_{j,i} \\
o_{b2} &= [m_0, \cdots, m_{M-1}]
\end{aligned} \tag{5}
$$

The vector o_{b2} is the return of a MaxPool layer over the A vector sequence, then:

$$
\begin{aligned}
F &= [o_{b2}, f_{bert}, f_{wn}, f_{hs}, f_{ig}] \\
o_{b3} &= IMF(F) \\
\hat{y} &= \sigma(W_h o_{b3} + b_h) \\
\hat{y_f} &= \sigma(W_f o_{b3} + b_f) \qquad (6)
\end{aligned}
$$

The third block is described in (6) where W_h, W_f, b_f and b_h are learnable parameters and $\hat{y}, \hat{y_f} \in \mathbb{R}$. The vectors f_{bert}, f_{wn}, f_{hs} and f_{ig} correspond to the BERT, WordNet, Hurt-Sentiment and Information Gain features respectively. The prediction of the tweets polarity is determined by the $\hat{y_f}$ value and the hate value trough $\hat{y}$.

The overall weighted loss of the model is calculated by cross-entropy, with higher importance value for the hate speech predictions that polarity predictions. The overall loss is calculated according to the following formula.

$$
\begin{aligned}
L_1 = -\sum y_i \log(\hat{y_i}) \qquad L_2 = -\sum y_{f_i} \log(\hat{y_{f_i}}) \\
loss = \lambda L_1 + (1-\lambda) L_2 \qquad (0 \leq \lambda \leq 1) \qquad (7)
\end{aligned}
$$

Here $L1$ and $L2$ are the cross-entropy loss of hate predictions and sentiment polarity predictions respectively. The value λ is the main task importance weight. The values y_i and y_{f_i} represents the ground true hate classification and polarity classification respectively. Then, the final loss is obtained as a convex sum of $L1$ and $L2$.

4 Experiments and Results

In this section we show the results of our proposed method in subtask A and discuss about them. The organizers allow a maximum of two submissions for every subtask in the challenge. We named our team UO.

Experiments where conducted in two main directions: Firstly, to investigate the impact of the IMF fusion strategy and secondly, to evaluate the impact of each proposed single-modal representation into our proposal. The results of our experiments are presented in Table 1 and Table 2.

In those tables, the column named *heads* is the number of attention headers in the Att-LSTM layer. If this space is empty, this layer was not used. Columns *bert* and *ig* correspond to the presence or not of BERT and IG representations. The column *wn-hs* express the presence of Hurt-Sentiment and WordNet based representations. If a cell has a cross, the representation associated to the column were not used in the corresponding run. We used a 10% of the training dataset for validation. We report the accuracy measure computed on this validation data.

Both Tables show that the presence of BERT increase the performance, also almost all the runs have higher values with IMF in contrast to not using it. Increasing the number of attention heads without IMF increase the results, but the opposite occurs in the presence of the IMF.

Name	heads	bert	ig	wn-hs	acc
run1	2				0.764386
run2	-		×	×	0.742690
run3	3				0.767544
run4	2	×			0.713450
run5	2			×	0.763158
run6	-				0.757310
run7	-	×			0.724152
run8	-			×	0.755848

Table 1: Experiment results without IMF.

Name	heads	bert	ig	wn-hs	acc
run1	2				0.795848
run2	-		×	×	0.779101
run3	3				0.764620
run4	2	×			0.720760
run5	2			×	0.774854
run6	-				0.767544
run7	-	×			0.719298
run8	-			×	0.777778

Table 2: Experiment results with IMF.

The pretrained embedding have a size of 300, the number of neurons in the Bi-LSTM and in the AttLSTM was 128. The λ value was equal to 0.75 and the dropout (Srivastava et al., 2014) after the embedding layer was 0.3. The optimizer algorithm to train the whole model was Adam (Kingma and Ba, 2015), with a learning rate of 0.01.

The bold models in Table 2 were chosen as final submission for the subtask. The *run1* uses the attention layer proposed in Section 3.2 and consider all proposed representations. The *run2* does not use attention mechanism and handcraft features, using only the BERT text representation and the rest of the architecture.

The Table 3 shows the official results of our system. The evaluation was performed on two distinct corpora: one conformed by tweets and the other by news headlines.

Runs	macro-F
UO:tweets_run1	0.6878
UO:tweets_run2	0.7214
BEST_RATED:tweets	**0.8088**
UO:news_run1	0.6657
UO:news_run2	0.7314
BEST_RATED:news	**0.7744**

Table 3: Official results.

These results show that between our two models, the simple one get better results. The simplicity is not a condition for a better performance using deep learning. These results also express that some linguistic features decrease the effectiveness of the model, but the similarity between the results in the tweets and news evaluation sets suggest that the system is able to generalize with a good performance.

5 Conclusions and Future Work

In this paper we presented an Ensemble Model for the task Hate Speech Detection (HaSpeeDe2) subtask A at Evalita 2020. Our proposal combines linguistic features and RNNs with transformers representations using an IMF. In the training phase, we used a multi-task learning approaches to recognize hate speech and polarity simultaneously.

The achieved results show that the ability of this ensemble to generalize the detection of hate content in different text genres. Nevertheless, some handcraft features decrements its results. Motivated by this, we plan to explore better features selection, other attention mechanisms and multitask learning techniques to improve the performance.

References

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the evalita 2016 sentiment polarity classification task.

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.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A multilingual lexicon of words to hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Cristina Bosco, Dell'Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9. CEUR.

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Andrea Cimino, Lorenzo De Mattei, and Felice Dell'Orletta. 2018. Multi-task learning in deep neural networks at evalita 2018. *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, pages 86–95.

Gretel Liz De la Pena Sarracén, Reynaldo Gil Pons, Carlos Enrique Muniz Cuza, and Paolo Rosso. 2018. Hate speech detection using attention-based lstm. *EVALITA Evaluation of NLP and Speech Tools for Italian*, 12:235.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hatespeech. In *Proceedings of the first workshop on abusive language online*, pages 85–90.

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

Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. 2018. Multi-source multi-class fake news detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

David D Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50.

Gang Liu and Jiabao Guo. 2019. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Manuela Sanguinetti, Gloria Comandini, Elisa Di Nuovo, Simona Frenda, Marco Stranisci, Cristina Bosco, Tommaso Caselli, Viviana Patti, and Irene Russo. 2020. Overview of the evalita 2020 second hate speech detection task (haspeede 2). In Valerio Basile, Danilo Croce, Maria Di Maro, and Lucia C. Passaro, editors, *Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020)*, Online. CEUR.org.

Valentino Santucci, Stefania Spina, Alfredo Milani, Giulio Biondi, and Gabriele Di Bari. 2018. Detecting hate speech for italian language in social media. In *EVALITA 2018, co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*, volume 2263.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International workshop on natural language processing for social media*, pages 1–10.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.