Deep learning for hate-speech detection: Evalita-2018 HaSpeeDe

Matteo Pinna
matteo.pinna@hotmail.com
University of Pisa
Pisa, Italy

ABSTRACT

The task of hate speech detection has gained considerable popularity in recent years due to a widespread increase in harmful contents on social media platforms. The fast, consistent, and accurate identification of hate speech is therefore of great importance in mitigating discrimination, misogyny, cyberbullying, and various forms of hate. This paper presents four systems that have been implemented and/or fine-tuned to solve the Evalita-2018 HaSpeeDe challenge. HaSpeeDe consists of three distinct sub-tasks related to hate-speech detection in Italian comments on Facebook and Twitter. The systems of choice include a dynamic convolutional neural network, an architecture based on convolutions and gated recurrent units, a bidirectional LSTM with attention, and finally a pre-trained BERT that will be fine-tuned w.r.t. to the data at hand. While more traditional approaches, such as convolutions and LSTMs, perform reasonably well on provided test data, they fail to achieve state-of-the-art performance. On the other hand, the fine-tuned BERT outperforms, in the majority of sub-tasks, all other systems that have been selected, as well as those proposed by previous HaSpeeDe-2018 participants. This underscores the effectiveness and relevance of pre-trained BERT architectures in handling complex NLP tasks.

KEYWORDS

deep learning, natural language processing (NLP), neural networks, hate-speech detection, evalita

ACM Reference Format:
Matteo Pinna. . Deep learning for hate-speech detection: Evalita-2018 HaSpeeDe. In *Proceedings of (Human Language Technologies)*. Pisa, Italy, 14 pages.

1 INTRODUCTION

In the last decade, the exponential increase in social media interactions has been paralleled by a surge in harmful contents, often targeted at specific public figures, such as celebrities or politicians, or communities, including minorities, religions, or specific genders. In response, contemporary research efforts in the field of Natural Language Processing (NLP), which encompasses a broad spectrum of topics tied to processing and interpretation of human language, has focused on developing methods for automated hate speech detection [1]. In fact, given the inherent intricacies of human languages - different forms of hatred, different interpretations for words or sentences based on context - hate-speech detection presents itself as a complex task with multiple challenges. Events like SemEval 2020-22 [29] [12], or HaSpeeDe 2018-23 [25] [24] [16] underscore the significance of this task and have substantially contributed to the development of novel methods, references, resources, and tools for addressing such challenges.

The domain of Natural Language Processing (NLP) has also undergone important shifts in recent years as a consequence of the incorporation of Deep Learning (DL) approaches in traditional NLP tasks [13] [19]. DL techniques are powerful tools for modeling, analyzing, and understanding complex data patterns. In the last decade, DL has achieved remarkable advancements thanks to increases in computational capabilities and parallelization [19], data collection techniques becoming more sophisticated, and the development of novel methods for artificially generating large, representative datasets [23]. As a consequence, a widespread adoption of DL-based systems has been seen in safety-critical and security-sensitive domains [18], such as self-driving vehicles [14] or robotics [30], and also in NLP.

Paper outline. This paper discusses four distinct architectures, ranging from Convolutional Neural Network (CNN) to Long Short-Term Memory (LSTM), to (pre-trained) Bidirectional Encoder Representations from Transformers (BERT). The first section provides a concise survey of past and present state-of-the-art in efficient DL-based models applied to NLP. Then, the tasks and corresponding datasets that are going to be used as a valuable benchmark for the proposed methods are going to be introduced. Subsequently, we provide a comprehensive description of the methodology that has been employed to carry out our experiments and benchmarks, as well as the four architectures of choice. Finally, the results, comparison and ablation studies are presented for each system.

2 STATE OF THE ART

Recent developments in the domain of NLP have witnessed an increasing trend in the use of deep learning methodologies [9] [4], thus demonstrating the crucial role of DL-based techniques in achieving efficient, consistent and reliable hate speech detection. NLP-related tasks typically range from understanding the structural characteristics of words and sentences (i.e., synctactic analysis) to comprehending their contextual meaning (i.e., semantic analysis). In both scenarios, the main challenge is how to translate words to a suitable format and/or representation for AI-based models, which require the input to be represented as a fixed length feature vector. As far as text is concerned, one common representation has been the bag-of-words (BoW), however despite its popularity it presenst major weakness as they not only lose the ordering of the words but they also ignore semantics of the words. In the last decade, word embeddings have been introduced [17], where words

Human Language Technologies,

or characters from a given vocabulary are converted into vector representations, capturing semantic and contextual information. Popular techniques include Word2Vec, Global Vectors for Words Representation (GloVe), and FastText.

Convolutional Neural Networks (CNNs), although originally designed for computer vision applications, have emerged and stay to this day as efficient architectures for addressing NLP tasks [6]. They provide noteworthy effectiveness and fast training times, despite showing limitations in capturing long-term dependencies. In fact on of the main challenges in analyzing sentences lies in the temporal dynamics of textual data, where the sequential arrangement of words assumes paramount importance in comprehending the contextual nuances.

To confront this challenge, Recurrent Neural Networks (RNNs) have gained popularity. RNNs are designed to maintain an ordered record of input words and retain critical information within their memory. Nevertheless, as the length of input sequences extends, the training of RNNs can become computationally demanding, rendering them less adept at effectively capturing long-term dependencies. This concern bears substantial significance in NLP, given the necessity to unravel the context and interrelationships between words across extended spans of text. This major drawback led to the development of more complex RNNs: Long Short-Term Memory (LSTM). LSTMs consists of a more complex, gate-based internal structure w.r.t traditional RNN cells. The gates allow LSTMs to decide which information to retain and which to discard as data is processed. At the same time there highway for gradient flow, similar to skip connections in ResNet, ensures there is always a path for information to go through so that longer sequences cannot cause gradient problems. Popular LSTM-based architectures include the Gated Recurrent Unit (GRU) and the Bidirectional LSTM (BiLSTM).

However, both word embeddings and traditional DL methods have a crucial drawback in their inability to capture the context within a sentence. Thus, new techniques have emerged, such as Generative Pre-trained Transformer (GPT/GPT-2) [22], and Bidirectional Encoder Representations from Transformers (BERT) [10], which replace static word embeddings with contextualized word representations. In this way, they address the limitation of their predecessors by leveraging a language understanding model which is capable of providing contextual, task-independent term representations [11]. Specifically, BERT employs a transformer-based architecture to analyze contextual relationships within a sentence bidirectionally. Pre-trained on large text corpora, they mitigate the need for both word embeddings and extensive pre-processing pipelines. Fine-tuning them for specific tasks enables BERT to adapt its learned representation to a wide variety of NLP tasks.

3 TASK AND DATASET

The Evalita-2018, HaSpeeDe challenge [3] consists of three subtasks:

- **Task 1: HaSpeeDe-FB**, only the Facebook dataset can be used to classify the Facebook test set;
- **Task 2: HaSpeeDe-TW**, only the Twitter dataset can be used to classify the Twitter test set;
- **Task 3: Cross-HaSpeeDe**, further divided into two subtasks:
  - **Task 3.1: Cross-HaSpeeDe-FB**, only the Facebook dataset can be used to classify the Twitter dataset;
  - **Task 3.2: Cross-HaSpeeDe-TW**, only the Twitter dataset can be used to classify the Facebook dataset.

Dataset. Facebook dataset comprises a collection of Italian Facebook posts created in 2016 by the CNR-Pisa group [8], while the Twitter dataset consists of a collection of Italian tweets developed in 2018 by the group from the Computer Science Department of Turin University [20] [25]. These datasets were randomly divided into development and test sets, each containing 3.000 and 1.000 comments, respectively. To ensure we are aware of potential biases introduced by class imbalances in the development and test sets, the distribution of the labels has been computed for each dataset. In fact, if one class is significantly underrepresented, models may perform poorly on the minority class. Given the class imbalances observable in Table 1, we're going to adopt appropriate approaches when splitting the development data.

Table 1: Class distribution in Facebook and Twitter dataset.

		# non hate-speech (0)	# hate-speech (1)
Facebook	**dev**	1618 (54%)	1382 (46%)
	test	323 (32%)	677 (68%)
Twitter	**dev**	2028 (68%)	972 (32%)
	test	676 (68%)	324 (32%)

Evaluation metric. The evaluation metric is the macro average F1-score, i.e. the F1-score is computed for the two classes (hateful and non hateful messages) and then the macro averaged F1 is taken into consideration for final results.

4 METHODOLOGY

In this section, we provide an in-depth overview of the methodology that has been used to build the systems and carry out our experiments. Four main steps: (1) data pre-processing, (2) word embedding with a pre-trained Word2Vec, (3) model tuning and training, and finally (4) testing w.r.t. test data.

4.1 Pre-processing

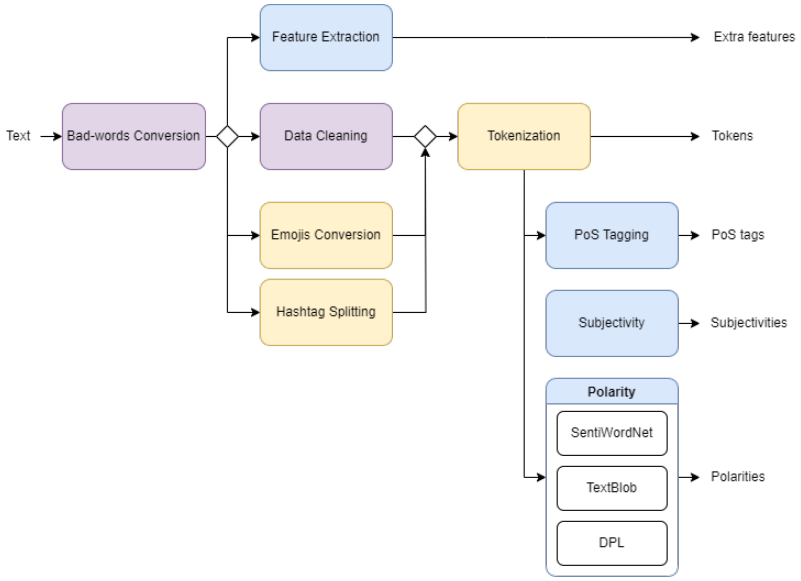
Data preparation is a crucial step aimed at converting raw data into meaningful information with a suitable format for analysis or further processing. In our study, different text pre-processing techniques have been applied to extract the most important features, to remove errors or inconsistencies, and to normalize and enrich the original data.

Our pre-processing approach aligns with the *Pre-processing* section from Bianchini et al. [2]. In Figure 1 a visual representation of the pre-processing pipeline, below a more in-depth description of each step, grouped by type.

A concise list of each operation, grouped by type, is listed below.

Feature extraction. To extract the most pertinent task-related features and avoid potential information loss due to subsequent pre-processing steps, a feature extraction operation is performed.

Figure 1: Pre-processing pipeline: feature extraction (blue), data cleaning (purple), text normalization/enrichment (yellow) steps.



Below a list of all extracted features.

- Length of the comment;
- Percentage of words written in CAPS-LOCK;
- Number of sentences: list of words ending with punctuation such as '.', '?' or '!';
- Number of question marks '?' or exclamations '!';
- Number of '.' or ',';
- Percentage of spelling errors: each word is compared with all words contained in a Italian vocabulary corpora[1] of approx. 900.000 unique Italian words, if not present it is considered a spelling error;
- Number of bad words; each word is compared with all words contained in our Italian bad words corpora - obtained by merging a list of Italian hate-speech terms by Tullio de Mauro [7] and profanity terms from the Italian (profanity) Wiktionary - if present it is considered a bad word;
- Percentage of bad words;
- Hashtags;
- Polarity: computed using three different polarity lexicons, including *SentiWordNet*, *TextBlob*, and the *Distributional Polarity Lexicon* (Castellucci et al. 2016);
- Subjectivity: computed using *TextBlob*;
- Part-of-Speech (PoS) tagging: words are grouped by their grammar class (e.g. noun, verb, adjective, etc.) to provide useful information about a word, and its neighbours, role in a sentence.

Data cleaning. To remove errors, inconsistencies or not meaningful information:

- Removal of mentions and URLs: HaSpeeDe-1 organizers replaced mentions and URLs with anonymized placeholders. For mentions, they used *<PERSONA_i>* for Facebook and *<MENTION_i>* for Twitter. For URLs, *<URL>* was used in both datasets;
- Handling special characters and newlines: special characters '@', '&' are replaced with 'a', 'e' respectively, newlines ' n' are removed;
- Conversion of disguised bad words: bad words in which some of their middle letters have been replaced by special characters are replaced with the corresponding bad word;
- Removal of redundant vowels and consonants: removing nearby equal vowels and nearby equal consonants if more than 2 (e.g., *caneeee* becomes *cane*);
- Removal of punctuation;
- Replacement of abbreviations and acronyms with the respective words;
- Removal of articles, pronouns, preposition, conjunctions and numbers;
- Removal of the laughs;
- Removal of accented characters with the respective unaccented characters.

Text normalization and enrichment. The original text is enriched by converting emoticons, leveraging the *emoji* library, and normalized through hashtag splitting and tokenization, leveraging the *spacy* library and the Italian model *it_core_news_sm*.

- Translation of emoticons: considering the large presence of emojis in tweets or facebook posts, they are translated with their respective Italian translation;
- Hashtag splitting: considering hashtags are often used to compose sentences, we split them (e.g. *#manovradelpopolo* becomes *manovra del popolo*);

[1] https://github.com/napolux/paroleitaliane

- Tokenization: converting sequences of text into individual tokens such as words or sub-words, breaking down complex text into more manageable, smaller units.

4.2 Word embedding

To generate word embeddings for tweets and Facebook comments, as well as for their PoS tags, we employed the *Italian Twitter embeddings* - a pre-trained Word2Vec (W2V) model with 128 features - work of Cimino et al. from the Italian NLP Lab [5]. Although there are potential differences between these two platforms - such as vocabulary, character constraints, or linguistic styles - the Twitter-based embeddings serve as a valuable resource. This is particularly pertinent given we were not able to find a dedicated pre-trained W2V model for Italian Facebook comments.

To ensure an information-guided padding and truncation during the creation of words embedding, sentence length-related statistics have been computed w.r.t. the pre-processed sentences. In Table 2 the aforementioned statistics and in Figure 2 the sentence length distribution for each dataset.

Table 2: Sentence length statistics, w.r.t. pre-processed sentences, for Facebook and Twitter dataset.

		# average	# maximum	# median	# mode
Facebook	**dev**	7	73	5	2
	test	9	74	7	2
Twitter	**dev**	11	29	11	12
	test	11	24	11	12

Given that our datasets are not exceedingly large, it has been decided to pad all sentences to the maximum sentence length observed in each platform. Although this approach does introduce computational overhead - which could be reduced by using a different statistic such as the average sentence length - it is negligible due to the manageable size of our datasets. Additionally, this approach avoids information loss since no sentence is truncated.

4.3 Model tuning and training

Hyper-parameter tuning is a pivotal step when building machine learning models. The dimensionality of hyper-parameter spaces can be vast, and choices within this space significantly affect model performance. When promising hyper-parameter values are identified, the model is ready to be extensively trained and evaluated on training and validation data respectively, based on the best identified values.

Hyper-parameter tuning. A common approach in DL involves an initial coarse search across a wide range of hyper-parameters' values to identify promising sub-ranges. Subsequently, a more zoomed in search is performed within these sub-ranges to fine-tune the final configuration. Popular search techniques include grid search, random search, and more recently, Bayesian optimization.

In our study, we start with a coarse Bayesian optimization, followed by a fine-tuning random search. Bayesian optimization provides an informed and guided search across the spaces, and ensures that we learn an approximate landscape of the objective function before zooming into our hyper-parameters' intervals. Should Bayesian optimization get stuck in a local minima or overlook important regions, potentially due to bias, the subsequent random search - conducted within the previously identified sub-intervals - provides a mechanism to mitigate such bias and possibly explore overlooked regions. In both steps an early stopping mechanism with patience to avoid wasting training time when the average validation F1-score is not improving.

Training. Initially, the model is trained/evaluated on the entire training/validation data to (approximately) assess the model performance, potentially identify overfitting, and visualize the learning curves. Taking into account class imbalance, as previously shown in Table 1, the development set is split using stratification, ensuring more representative training and validation sub-sets. Then, k distinct model instances are trained utilizing a (Stratified) K-fold, where k represents the number of splits. A stratified K-fold is employed due to class imbalances, also a relatively high number of splits ($k = 10$) has been chosen to make the most out of the available data, considering the small size of our dataset. As in the tuning phase, an early stopping mechanism is employed, this time not only to save time but also to improve generalization through prevention of overfitting. Moreover, upon conclusion of training, the model weights from the epoch which achieved the best average validation F1-score are restored. After several iterations of K-Fold training, the two runs that achieve the highest average macro F1-score across the respective validations sub-sets are selected. This methodology aligns with the original challenge constraints, where it was possible to submit at most two runs for final evaluation, and where the gold set was not available before the submission deadline.

4.4 Model testing

The two best runs from training are taken into consideration. For each run, the k models models trained on different splits of the development set are used to perform Ensemble prediction on the test set. Ensemble prediction is a technique that involves leveraging distinct models to make a simple class prediction on each test set sample, it is particularly useful to achieve a more robust and generalized performance. To combine the predictions we implement a majority vote mechanism, i.e. the final class label for a given test samples depends on whether the majority of the k models predict it as class 0 (non hate-speech) or 1 (hate-speech). In case of ties, class 0 is the default label. Note that, all the results that are going to be presented are the best performance between two runs.

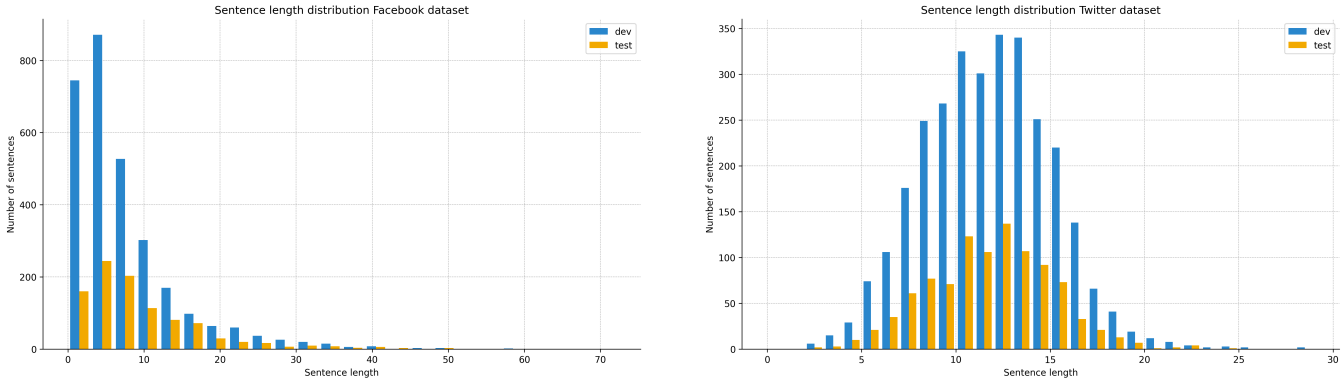
The final results represent the run which achieved the highest macro F1-score on test data.

5 MODELS

In this section, the DL-based architectures of choice are going to be discussed: a *Convolutional Neural Network with Dynamic k-Max Pooling (DCNN)* [15], a *Convolution GRU-based Network (Conv-GRU)* [32], a *Attention-based BiLSTM (A-BiLSTM)* [31], and the pre-trained (Italian) BERT-xxl-uncased made available by the MDZ Digital Library team (dbmdz) at the Bavarian State Library[2].

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

Figure 2: Sentence length distribution of Facebook (left) and Twitter (right) datasets.



Input. The first three models take three sources of information, thus comprising three input layers: one for embedded, tokenized sentences, one for embedded PoS tags, and a third for all additional features acquired during the feature extraction phase. Note that, PoS tags and extra feature were included, alongside the actual sentences, in order to provide supplementary information and possibly enhance model performance and robustness. Conversely, the pre-trained BERT model takes only the original (data-cleaned) text: transformer-based models such as BERT have the advantage of mitigating the need for extensive pre-processing, as well as traditional word embeddings.

5.1 DCNN

The (dynamic) convolutional neural network introduced by Kalchbrenner et al. employs a sequence of convolutional layers, folding, and dynamic k-max pooling. For simplicity, in our implementation the dynamic k-max pooling is replaced by a custom semi-dynamic corresponding version.

Below a brief description of how folding and the semi-dynamic k-max pooling work.

Folding. Folding is a simple technique used to reduce the size of feature maps by element-wise summation of every two consecutive rows, consequently simplifying the network without introducing additional parameters. By halving the size of feature maps, it establishes dependencies between feature detectors of different orders, enabling more complex relationships between rows within the same feature maps. Folding is applied after each convolution and before each (semi-dynamic) k-max pooling operation.

Semi-dynamic k-Max pooling. The original dynamic k-max pooling operation computed the value of k dynamically based on the sentence length and network depth. For each step and level of depth l, k_l is calculated as follows:

$$k_l = \max\left(k_{top}, \lceil \frac{L-l}{L} s \rceil\right) \tag{1}$$

where k_{top} is the fixed pooling parameter for the topmost convolutional layer, L is the total number of convolutional layers in the network, l is the number of the current convolutional layer to which the pooling is applied, and s denotes length of the current sentence.

In our semi-dynamic k-max pooling implementation k_l is still computed based on network depth, however instead of using sentence length, the average sentence length avg_s is considered. Thus, Equation 1 is adjusted as follows:

$$k_l = \max\left(k_{top}, \lceil \frac{L-l}{L} avg\_s \rceil\right) \tag{2}$$

where avg_s denotes the average sentence length for the specific dataset.

Our DCNN implementation comprises two distinct DCNNs: one processes tokenized sentences embeddings, while the other handles PoS tags embeddings. Their outputs are concatenated with the extra features, and dropout is applied to mitigate overfitting. Finally, the result is fed into a sigmoid classification layer with L2 regularization. The model is then compiled using Adam as optimizer and BCE as the loss function. In Figure 3 the architecture of our (double) DCNN.

5.2 Conv-GRU

The Convolutional + Gated Recurrent Unit (GRU) architecture introduced by Zhang Z. et al. applies dropout to the word and PoS embeddings before they're fed into a sequence of 1D convolution, 1D max pooling, a gated recurrent unit, and finally a global max pooling. The authors decided to use a GRU layer instead of a traditional LSTM to make training faster and generalization better on small data. In fact, a GRU has only two gates (reset and update gates) instead of three (input, output and forget gates).

As for the DCNN, our implementation comprises two distinct Conv + GRU: one for tokenized sentences embeddings, one for PoS tags embeddings. The respective outputs are merged along with the extra features and fed into a sigmoid classification layer with elastic net regularization (i.e., L1 and L2). The Adam optimizer and the binary cross entropy loss function are used. In Figure 4 the architecture of our (double) Conv-GRU.

Figure 3: Architecture of the CNN with dynamic k-max pooling: the core structure of the DCNN for words and PoS embeddings (left) and the concatenation of their results and the extra features (right).

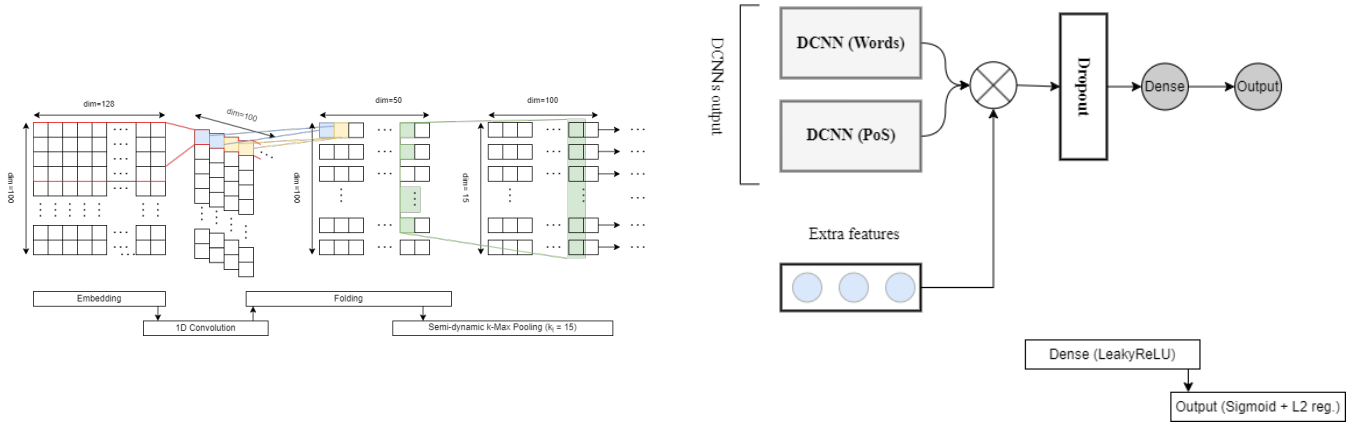
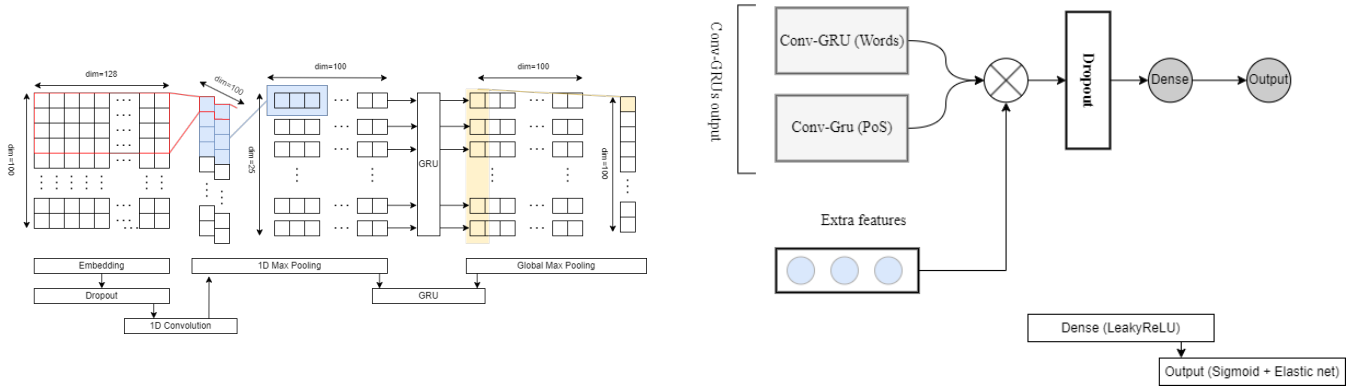


Figure 4: Architecture of the Convolution-GRU: the core structure of the Convolution-GRU for words and PoS embeddings (left) and the concatenation (right).



5.3 A-BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM) is a LSTM variation which has become a popular architecture to tackle NLP problems in recent years. Specifically, BiLSTMs are designed to better capture long term dependencies and solve the gradient vanishing/exploding issue suffered by more traditional RNNs and LSTMs. They are more complex w.r.t. both traditional LSTMs or GRUS as they consists of two LSTMs, one takes the input in a forward direction, the other in the backward direction. This design allows the network to capture information from past and future context simultaneously, improving understandin of sequential data and improving performance in NLP tasks. The Attention-based bidirectional LSTM by Zhang Y. et al. involves a BiLSTM and an attention mechanism.

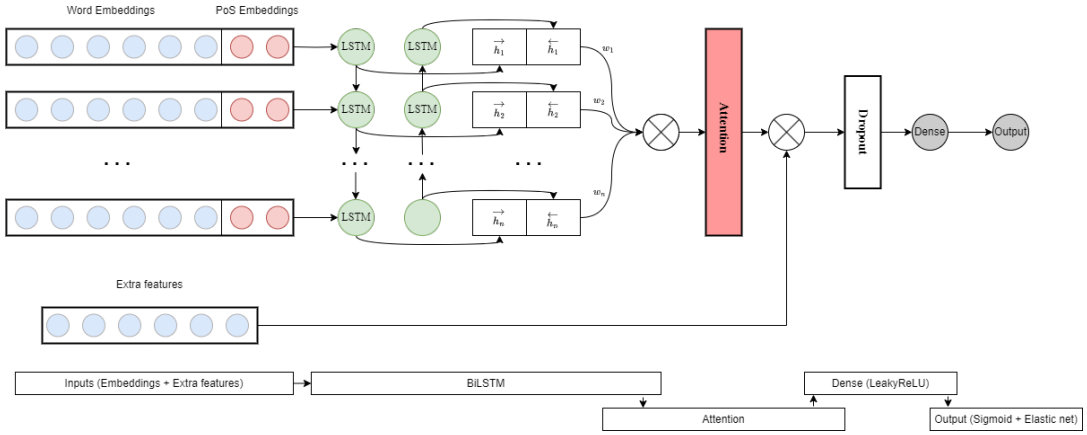
In our implementation, word embeddings and PoS tags embeddings are concatenated and then fed to a BiLSTM, followed by a Attention layer. Then, the output is merged with the extra features and processed by a sigmoid classification layer to generate the final output. The Adam optimizer and the binary cross entropy loss function are used. In Figure 5 the architecture of our BiLSTM with attention mechanism.

5.4 DBMDZ BERT

As previously mentioned, BERT architectures have demonstrated the ability to achieve state-of-the-art performance in numerous NLP tasks. Consequently, we opted to fine-tune the pre-trained Italian DBMDZ BERT, developed by the DBMDZ team. The source data for the model training comprises the OPUS corpora[3] [26] collection, augmented with the Italian portion from the OSCAR corpus[4]. The final corpus consists of 13.138.379.147 tokens. An alternative option would have been to fine-tune AlBERTo [21], a pre-trained BERT model for the Italian language trained on Italian tweets retrieved from TWITA. However, after preliminary experiments, the DBMDZ Italian BERT demonstrated, on average, better performances. Additionally, a study of AlBERTo w.r.t. HaSpeeDe-2018 has already been proposed by Poletto et al. [20].

[3]https://opus.nlpl.eu/
[4]https://oscar-project.org/

Figure 5: Architecture of the Attention-based BiLSTM.



The pre-trained model is loaded and fed the original text into the model after applying some basic data cleaning steps (e.g., removal of punctuation and stop words, elimination of URLs and mention placeholders, etc.). Subsequently, a fine-tuning step is performed by leveraging the development data available. In Figure 6 an example of the BERT architecture, and in Table 3 the configuration of the DBMDZ-BERT-xxl.

Figure 6: Example architecture of BERT.

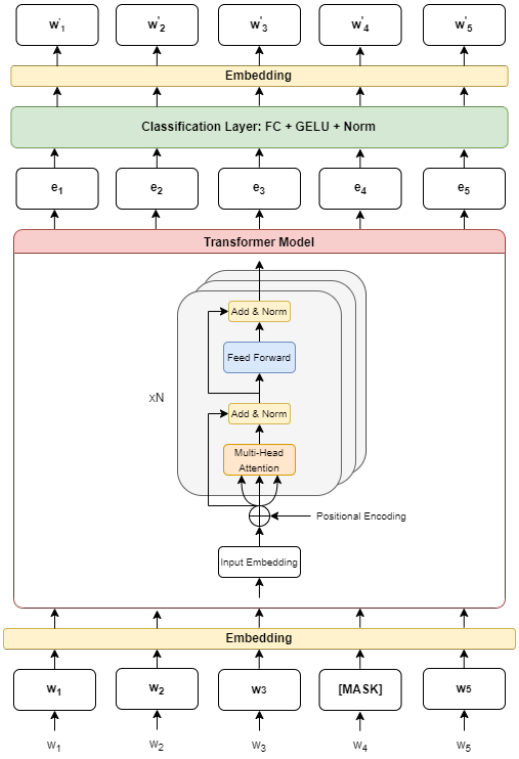


Table 3: DBMDZ-BERT-xxl configuration.

Name	dbmdz/bert-base-italian-xxl-uncased
Architecture	BertForMaskedLM
Parameter	**Value**
attention_probs_dropout_prob	0.1
classifier_dropout	null
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
layer_norm_eps	1e-12
max_position_embeddings	512
model_type	bert
num_attention_heads	12
num_hidden_layers	12
pad_token_id	0
position_embedding_type	absolute
transformers_version	4.35.1
type_vocab_size	2
use_cache	true
vocab_size	32102

6 EXPERIMENTS AND RESULTS

In this section, the results from the tuning step are presented, i.e., the best hyper-parameters configuration identified. Then, the benchmark of each model w.r.t. each sub-tasks and according to the macro averaged F1-score are reported, as well as benchmarks according to ablation studies that we performed on the original architectures.

6.1 Tuning results

Firstly, for each model and development dataset, the hyper-parameters intervals as well as the results of the hyper-parameter tuning phase, i.e the best hyper-parameters configuration found at the end of the Bayesian optimization + Random search, is going to be presented.

In Tables 4, 5, 6, and 7 the hyper-parameters spaces for the tuning phase, for DCNN, Conv-GRU, A-BiLSTM, and DBMDZ-BERT respectively. It has been decided to set very large intervals for each

hyper-parameter to gain the most out of our tuning approach. Besides, such large intervals do not cause an excessively long tuning process since Bayesian optimization ensures a information-guided, faster convergence to optimal configurations (i.e., it doesn't require as many searches as a coarse random search).

Table 4: DCNN tuning hyper-paramaters spaces.

hyper-paramater	intervals
learning rate	$[1e-5, 1e-1]$
filters	$[50, 100, 150]$
dense dim.	$[16, 32, 64, 128]$
dropout	$[0.0, 0.5]$
reg.	$[1e-6, 1e-2]$
batch size	$[16, 32, 64, 128]$

Table 5: ConvGRU tuning hyper-paramaters spaces.

hyper-paramater	intervals
learning rate	$[1e-5, 1e-1]$
filters	$[50, 100, 150]$
GRU units	$[16, 32, 64, 128, 256]$
dense dim.	$[16, 32, 64, 128]$
dropout	$[0.0, 0.5]$
reg.	$[1e-6, 1e-2]$
batch size	$[16, 32, 64, 128]$

Table 6: A-BiLSTM tuning hyper-paramaters spaces.

hyper-paramater	intervals
learning rate	$[1e-5, 1e-1]$
LSTM units	$[16, 32, 64, 128, 256]$
dense dim.	$[16, 32, 64, 128]$
dropout	$[0.0, 0.5]$
reg.	$[1e-6, 1e-2]$
batch size	$[16, 32, 64, 128]$

Table 7: DBMDZ BERT tuning hyper-parameters spaces.

hyper-paramater	intervals
learning rate	$[1e-6, 1e-2]$
weight decay	$[0.0001, 0.001, 0.01, 0.1]$
batch size	$[16, 32, 64, 128]$

In Tables 8, 9, 10, and 11 the best configuration found at the end of the (coarse) Bayesian optimization + (fine-tune) random search for each model, and for Facebook and Twitter dataset.

6.2 Comparison

For each architecture and each development data, the optimal hyper-parameter configuration previously identified is taken into consideration. First, a single model instance is trained and evaluated on the entire development set, then k distinct models are trained using K-Fold cross-validation and tested using Ensemble prediction. This enables us to exploit different data splits to potentially enhance model performance.

Table 8: DCNN best configuration found by the tuning process, for Facebook and Twitter dev datasets.

	Facebook	Twitter
learning rate	6e-3	4e-4
filters	16	25
dense dim.	16	64
dropout	0.5	0.02
reg.	3e-5	1e-3
batch size	128	128
val. macro F1	0.8452	0.8209

Table 9: Conv-GRU best configuration found by the tuning process, for Facebook and Twitter dev datasets.

	Facebook	Twitter
learning rate	3e-3	1e-3
GRU units	32	64
filters	100	100
dense dim.	128	32
dropout	0.1	0.3
reg.	9e-3	4e-3
batch size	16	128
val. macro F1	0.8412	0.8351

Table 10: A-BiLSTM best configuration found by the tuning process, for Facebook and Twitter dev datasets.

	Facebook	Twitter
learning rate	1e-2	4e-3
LSTM units	50	25
dense dim.	64	32
dropout	0.3	7e-2
reg.	5e-2	0.2
batch size	128	64
val. macro F1	0.8521	0.8259

Table 11: DBMDZ BERT best configuration found by the tuning process, for Facebook and Twitter dev datasets.

	Facebook	Twitter
learning rate	2e-5	2e-5
weight decay	0.1	0.1
batch size	16	64
val. macro F1	0.8546	0.8235

In Figure 7, 8, 9 and 10 the learning curves for Binary-cross entropy loss and validation macro avg. F1-score for DCNN, ConvGRU, A-BiLSTM, and DBMDZ-BERT-xxl respectively, Facebook curves on the left and Twitter curves on the right.

Finally, the K-Fold + Ensemble macro F1-scores on the test set for a comprehensive comparison, including the baseline for *Most Common Class (MFC)* and others participants' results. The teams that took part in the competition, as well as a brief summary of the

Figure 7: DCNN training and validation learning curves for Binary cross-entropy loss and macro avg. F1-score, Facebook dev dataset (left) and Twitter dev dataset (right).

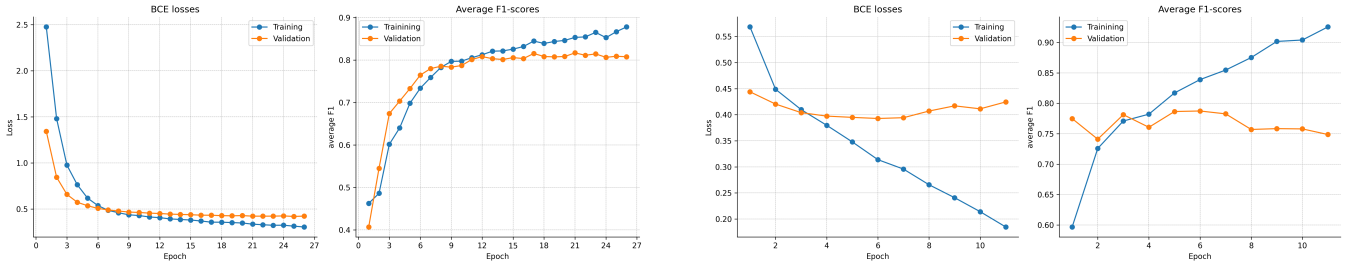


Figure 8: Conv-GRU training and validation learning curves for Binary cross-entropy loss and macro avg. F1-score, Facebook dev dataset (left) and Twitter dev dataset (right).

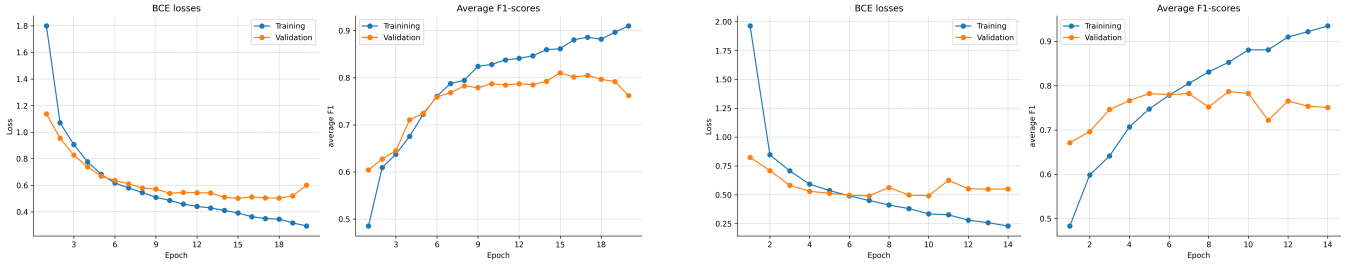


Figure 9: A-BiLSTM training and validation learning curves for Binary cross-entropy loss and macro avg. F1-score, Facebook dev dataset (left) and Twitter dev dataset (right)

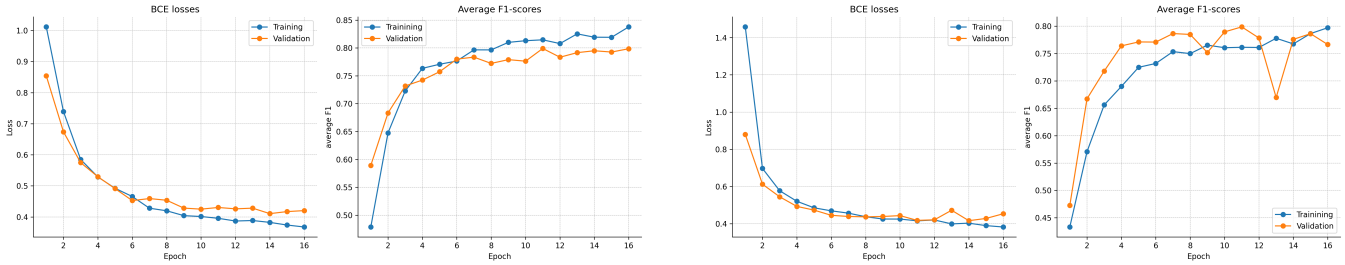
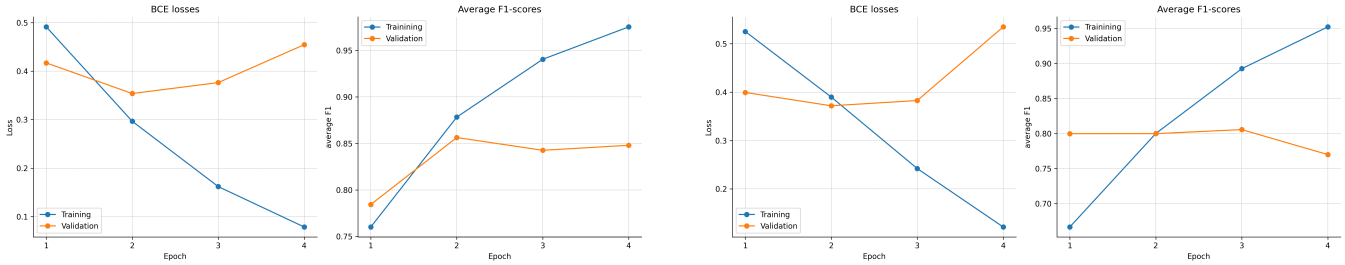


Figure 10: DBMDZ BERT training and validation learning curves for Binary cross-entropy loss and macro avg. F1-score, Facebook dev dataset (left) and Twitter dev dataset (right).



external resources they used is shown in Table 12 (Table 6 from [3]).

In Table 13 the results for **Task 1: HaSpeeDe-FB** and in Table 14 the results for **Task 2: HaSpeeDe-TW**. In bold the models implemented and analyzed by us.

Table 12: HaSpeeDe-2018 participants overview.

Team	Affiliation	External resources
GRCP	Univ. Politècnica de València + CERPAMID, Cuba	pre-trained word embeddings
InriaFBK	Univ. Cote d'Azur, CNRS, Inria + FBK, Trento	emotion lexicon
ItaliaNLP	ILC-CNR, Pisa + Univ. of Pisa	polarity and subjectivity lexicons + 2 word-embedding lexicon
Perugia	Univ. for Foreigners of Perugia + Univ. of Perugia + Univ. of Florence	Twitter corpus + hate-speech lexicon + polarity lexicon
RUG	University of Groningen + Univ. degli Studi di Salerno	pre-trained word embeddings + bad offensive word lists
sbMMP	Zurich Univ. of Applied Sciences	pre-trained word embeddings
StopPropagHate	INESC TEC + Univ. of Porto + Eurecat, Centre Tecn. de Catalunya	-
HanSEL	University of Bari Aldo Moro	pre-trained word embeddings
VulpeculaTeam	University of Perugia	polarity lexicon + list of bard words + pre-trained word embeddings

The fine-tuned DBMDZ-BERT achieves state-of-the-art performance in both HaSpeeDe-FB and HaSpeeDe-TW tasks. The other models that have been implemented show, w.r.t. other participants, better results when trained and tested on Facebook data. Notably, the statistical analysis of sentence lengths per platform, as previously presented, reveals that the Facebook dataset has, as a result of our preprocessing, a substantial portion short comments, and less frequent but considerably longer ones. The predominance of short messages might make the learning and classification process easier. Furthermore, as referenced in [3], Facebook comments tend to exhibit a higher degree of grammatical correctness and are conceivably more straightforward for the systems to interpret. Concluding, both test datasets exhibit significant class imbalances, with the Facebook dataset skewed in favor of hate speech comments (68%) and the Twitter dataset skewed towards non-hate speech comments (68%). This inherent dissimilarity in class distribution may be an additional contributing factor to the observed performance difference.

In Table 15 the result for **Task 3.1: Cross-HaSpeeDe-FB** and in Table 16 the result for **Task 3.2: Cross-HaSpeeDe-TW**. In bold the systems implemented and analyzed by us.

Shifting our focus to the cross-platform sub-tasks, DBMDZ-BERT is able to achieve the state-of-the-art in Cross-HaSpeeDe-FB, however it falls short in Cross-HaSpeeDe-TW, where it is outperformed, aside from other participants' models, by our Conv-GRU and DCNN. The A-BiLSTM achieve the worst performance in all sub-tasks, except for Cross-HaSpeeDe-TW, when compared with our other implementations. In the context of cross-platform evaluation, it is unsurprising to observe sub-optimal performance, as it is anticipated when training and validation procedures are executed on datasets that may significantly differ from the test set in various aspects, which for instance may be platform specific or user base specific. Additionally, it should be noted that the Facebook dataset diverges from the Twitter dataset, manifesting a broader spectrum of hate categories, as highlighted by the organizers. Consequently, the task of identifying hateful content within the Cross-HaSpeeDe-TW sub-task may have proven more formidable due to the relatively diminished availability of potential targets for harmful content within the training dataset in comparison to the test set.

6.3 Ablation studies

In order to understand and estimate the importance of different components or elements within the employed methodology and the traditional model that have been implemented, it has been decided to explore two distinct ablation studies for each architecture. For each model, we start by removing the PoS component from the input, thus assessing the extent to which PoS tags contribute in providing additional (syntactical) information. Then, we propose a more model-specific ablation study for each method: the dropout is removed in the DCNN, the GRU from the Conv-GRU, finally the attention mechanism in the A-BiLSTM. For BERT, we try halving the hidden and intermediate size (i.e., 768, 3.072 to 384, 1.536 respectively), and considering only $\frac{1}{4}$ of the number of attention heads (i.e., 12 to 3) attention heads).

The outcomes of these studies are reported in Table 17, 18, 19, and 20 for the DCNN, Conv-GRU, A-BilSTM, and DBMDZ-BERT respectively.
It is interesting to note that the omission of PoS tags leads to a consistent decrease in performance across the majority of sub-tasks, albeit not by a substantial margin. Additionally, on average we can observe that removing a specific component from each architecture results in a smaller performance degradation with respect to the mission of Part-of-Speech (PoS). This insight can be interpreted as indicating that the importance of the additional information provided by the PoS tags outweighs the significance of additional learning capacity. This observation underscores the crucial role of PoS tags in providing additional (syntactical) information within sentences. However, surprisingly the omission of PoS in the A-BiLSTM produces a better results, by a good margin, than the baseline architecture in the Cross-HaSpeeDe-TW tasks. The same can be said when the attention mechanism is removed. The DCNN and the Conv-GRU exhibit a small, anticipated decrease in F1-scores when the dropout and the GRU are removed, respectively, across all sub-tasks. Finally, reducing the hidden and intermediate size in BERT has a larger impact than reducing the number of attention heads.

7 CONCLUSIONS AND FUTURE WORK

This paper has presented the importance and complexity of hate-speech detection as an active and crucial research topic, that aids in identifying harmful comments in text. Due to its intricate nature, hate-speech detection proves to be a challenging research topic, which is also heavily influenced by factors such as social contexts (e.g., difference social media platforms and user base) or ideologies and cultures (e.g., different users' backgrounds). In our research, we have seen how designing an efficient architecture for

classifying harmful comments represents a non-trivial task. This can be attributed to the significant impact choices related to preprocessing techniques, word embeddings approaches, as well as architecture choice, can exert on the outcomes. Additionally, potential bias within the training data, such as class imbalance, can pose as an additional setback to the generalization capabilities of a model when confronted with new, unseen data samples. Three different traditional DL methods have been implemented and analyzed, and we have observed relatively similar levels of performance between them except for the A-BiLSTM which typically achieved the worse results out of the three. This outcome is unsurprising, given that these models all represent developments authored by knowledgeable researchers, and they share common relevant structures, such as convolutions or Long Short-Term Memory (LSTM) networks, or a hybrid of the two. On the other hand, the fine-tuned BERT achieved the best results on three sub-tasks out of three, which is unsurprising as BERT or more generally transformer-based architecture are at the moment are the state-of-the-art in a wide variety of tasks. Furthermore, our findings underscore the complexity of cross-platform tasks, proving the vital importance of training models on data that faithfully mirrors the characteristics of the anticipated (unseen) data that the are expected in real-world or practical applications. Lastly, our ablation studies have provided valuable insights on the impact of PoS tagging and the core components for each model. It is safe to conclude to conclude that PoS-related information plays a pivotal role in NLP-related tasks and should be incorporated, whenever possible, alongside the textual content.

Concerning potential avenues for future research and open challenges in this domain, it would be interesting to conduct a benchmark based on diverse word embedding techniques, such as GloVe or FastText, to assess their influence on hate speech classification performance within the HaSpeeDe-2018 challenge. Additionally, exploring and implementing data augmentation methods on the original dataset, including synonym replacement [27], employing pre-trained language models like GPT-3 to generate supplementary text based on existing data [28], or embracing self-supervised data augmentation strategies, warrants further investigation.

Table 13: Results of the HaSpeeDe-FB sub-task w.r.t. the test set.

	NOT HS			HS			
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Macro F1-score**
baseline_MFC	-	-	-	-	-	-	0.2441
DBMDZ-BERT-xxl	0.8345	0.7183	0.7720	0.8740	0.9321	0.9021	**0.8371**
ItaliaNLP_2	0.8111	0.7182	0.7619	0.8725	0.9202	0.8957	0.8288
InriaFBK_1	0.7628	0.6873	0.7231	0.8575	0.898	0.8773	0.8002
Conv-GRU	0.7310	0.7152	0.7230	0.8655	0.8744	0.8699	**0.7965**
DCNN	0.7432	0.6811	0.7108	0.8537	0.8877	0.8704	**0.7906**
A-BiLSTM	0.7138	0.7028	0.7083	0.8592	0.8656	0.8624	**0.7853**
Perugia_2	0.7245	0.6842	0.7038	0.8532	0.8759	0.8644	0.7841
RuG_1	0.699	0.6904	0.6947	0.8531	0.8581	0.8556	0.7751
HanSEL	0.6981	0.6873	0.6926	0.8519	0.8581	0.855	0.7738
VulpeculaTeam	0.6279	0.7523	0.6845	0.8694	0.7872	0.8263	0.7554
RuG_2	0.6829	0.6068	0.6426	0.8218	0.8655	0.8431	0.7428
GRCP_2	0.6758	0.5294	0.5937	0.7965	0.8788	0.8356	0.7147
StopPropagHate_2	0.4923	0.6965	0.5769	0.8195	0.6573	0.7295	0.6532
Perugia 1	0.3209	0.9907	0.4848	0	0	0	0.2424

Table 14: Results of the HaSpeeDe-TW sub-task w.r.t. the test set.

	NOT HS			HS			
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Macro F1-score**
baseline_MFC	-	-	-	-	-	-	0.4033
DBMDZ-BERT-xxl	0.8717	0.8743	0.8730	0.7360	0.7315	0.7337	**0.8034**
ItaliaNLP_2	0.8772	0.8565	0.8667	0.7147	0.75	0.7319	0.7993
RuG_1	0.8577	0.8831	0.8702	0.7401	0.6944	0.7165	0.7934
Conv-GRU	0.8657	0.8580	0.8618	0.7091	0.7222	0.7156	**0.7887**
InriaFBK_2	0.8421	0.8994	0.8698	0.7553	0.6481	0.6976	0.7837
sbMMMP	0.8609	0.8520	0.8565	0.6978	0.7129	0.7053	0.7809
DCNN	0.8544	0.8595	0.8569	0.7031	0.6944	0.6988	**0.7778**
VulpeculaTeam	0.8461	0.8786	0.8621	0.7248	0.6666	0.6945	0.7783
Perugia_2	0.8452	0.8727	0.8588	0.7152	0.6666	0.6900	0.7744
A-BiLSTM	0.8397	0.8757	0.8573	0.7153	0.6512	0.6817	**0.7695**
StopPropagHate_2	0.8628	0.7721	0.8149	0.6101	0.7438	0.6703	0.7426
GRCP_1	0.7639	0.8713	0.814	0.62	0.4382	0.5135	0.6638
HanSEL	0.7541	0.8801	0.8122	0.6161	0.4012	0.4859	0.6491

Table 15: Results of the Cross-HaSpeeDe-FB sub-task w.r.t. the test set.

	NOT HS			HS			
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Macro F1-score**
baseline_MFC	-	-	-	-	-	-	0.4033
DBMDZ-BERT-xxl	0.8926	0.6272	0.7368	0.5200	0.8426	0.6431	**0.6899**
InriaFBK_2	0.8183	0.6597	0.7305	0.4945	0.6944	0.5776	0.6541
VulpeculaTeam	0.8181	0.639	0.7176	0.483	0.7037	0.5728	0.6452
Perugia_2	0.8503	0.5547	0.6714	0.4615	0.7962	0.5843	0.6279
ItaliaNLP_1	0.9101	0.4644	0.615	0.4473	0.9043	0.5985	0.6068
DCNN	0.8290	0.5163	0.6363	0.4352	0.7778	0.5581	**0.5972**
A-BiLSTM	0.7979	0.4497	0.5752	0.3990	0.7623	0.5239	**0.5495**
GRCP_2	0.7015	0.7928	0.7444	0.4067	0.2962	0.3428	0.5436
RuG_1	0.8318	0.4023	0.5423	0.3997	0.8302	0.5396	0.5409
Conv-GRU	0.8491	0.3580	0.5036	0.3930	0.8673	0.5409	**0.5223**
HanSEL	0.7835	0.2677	0.3991	0.3563	0.8456	0.5013	0.4502
StopPropagHate	0.6579	0.3727	0.4759	0.3128	0.5956	0.4102	0.443

Table 16: Results of the Cross-HaSpeeDe-TW sub-task w.r.t. the test set.

	NOT HS			HS			
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Macro F1-score**
baseline_MFC	-	-	-	-	-	-	0.2441
ItaliaNLP_2	0.5393	0.7647	0.6325	0.8597	0.6883	0.7645	0.6985
InriaFBK_2	0.5368	0.6532	0.5893	0.8154	0.7311	0.771	0.6802
Conv-GRU	0.5212	0.7245	0.6062	0.8385	0.6824	0.7524	**0.6793**
DCNN	0.4764	0.8142	0.6011	0.8661	0.5731	0.6898	**0.6455**
DBMDZ-BERT-xxl	0.4717	0.8266	0.6007	0.8710	0.5583	0.6805	**0.6406**
VulpeculaTeam	0.453	0.7461	0.5637	0.8247	0.5701	0.6742	0.6189
RuG_1	0.4375	0.6934	0.5365	0.7971	0.5745	0.6678	0.6021
A-BiLSTM	0.4307	0.8447	0.5705	0.8737	0.4904	0.6282	**0.5993**
HanSEL	0.3674	0.8235	0.5081	0.7934	0.3234	0.4596	0.4838
Perugia_2	0.3716	0.9318	0.5313	0.8842	0.2481	0.3875	0.4594
GRCP_1	0.3551	0.8575	0.5022	0.7909	0.257	0.3879	0.4451
StopPropagHate	0.3606	0.9133	0.517	0.8461	0.2274	0.3585	0.4378

Table 17: Ablation studies results for DCNN.

Model	HaSpeeDe-FB	HaSpeeDe-TW	Cross-HaSpeeDe-FB	Cross-HaSpeeDe-TW
DCNN_baseline	0.7906	0.7778	0.5972	0.6455
DCNN - no dropout	0.7734	0.7677	0.5826	0.6432
DCNN - no PoS	0.7712	0.7668	0.5593	0.6401

Table 18: Ablation studies results for Conv-GRU.

Model	HaSpeeDe-FB	HaSpeeDe-TW	Cross-HaSpeeDe-FB	Cross-HaSpeeDe-TW
ConvGRU_baseline	0.7965	0.7887	0.5223	0.6793
ConvGRU - no GRU	0.7806	0.7805	0.5111	0.6683
ConvGRU - no PoS	0.7794	0.7750	0.5015	0.6641

Table 19: Ablation studies results for A-BiLSTM.

Model	HaSpeeDe-FB	HaSpeeDe-TW	Cross-HaSpeeDe-FB	Cross-HaSpeeDe-TW
A-BilSTM_baseline	0.7853	0.7695	0.5495	0.5993
A-BiLSTM - no attention	0.7799	0.7507	0.4849	**0.6554**
A-BiLSTM - no PoS	0.7770	0.7684	0.5250	**0.6492**

Table 20: Ablation studies results for DBMDZ-BERT.

Model	HaSpeeDe-FB	HaSpeeDe-TW	Cross-HaSpeeDe-FB	Cross-HaSpeeDe-TW
DBMDZ-BERT_baseline	0.8371	0.8034	0.6899	0.6406
DBMDZ-BERT - 1/4 attention heads	0.8204	0.7767	0.6496	0.5447
DBMDZ-BERT - 1/2 hidden & intermediate size	0.8162	0.0.7741	0.6183	0.4351

REFERENCES

[1] 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*. 759–760.

[2] Giulio Bianchini, Lore nzo Ferri, and Tommaso Giorni. 2018. Text analysis for hate speech detection in Italian messages on Twitter and Facebook. *EVALITA Evaluation of NLP and Speech Tools for Italian* 12 (2018), 250.

[3] Cristina Bosco, Dell'Orletta Felice, Fabio Poletto, Manuela Sanguinetti, Tesconi Maurizio, et al. 2018. Overview of the evalita 2018 hate speech detection task. In *Ceur workshop proceedings*, Vol. 2263. CEUR, 1–9.

[4] Junyi Chai and Anming Li. 2019. Deep learning in natural language processing: A state-of-the-art survey. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*. IEEE, 1–6.

[5] 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)* (2018), 86–95.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12, ARTICLE (2011), 2493–2537.

[7] Tullio De Mauro. 2016. Le parole per ferire. *Internazionale. Search in* (2016).

[8] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the first Italian conference on cybersecurity (ITASEC17)*. 86–95.

[9] Li Deng and Yang Liu. 2018. *Deep learning in natural language processing*. Springer.

[10] 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* (2018).

[11] Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512* (2019).

[12] Elisabetta Fersini, Francesca Gasparini, Giulia Rizzi, Aurora Saibene, Berta Chulvi, Paolo Rosso, Alyssa Lees, and Jeffrey Sorensen. 2022. SemEval-2022 Task 5: Multimedia automatic misogyny identification. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. 533–549.

[13] Yoav Goldberg. 2022. *Neural network methods for natural language processing*. Springer Nature.

[14] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2020. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics* 37, 3 (2020), 362–386.

[15] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

[16] Mirko Lai, Stefano Menini, Marco Polignano, Valentina Russo, Rachele Sprugnoli, and Giulia Venturi. 2023. Evalita 2023: Overview of the 8th evaluation campaign of natural language processing and speech tools for italian. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023), CEUR. org, Parma, Italy*.

[17] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR, 1188–1196.

[18] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th international symposium on software reliability engineering (ISSRE)*. IEEE, 100–111.

[19] Daniel W Otter, Julian R Medina, and Jugal K Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems* 32, 2 (2020), 604–624.

[20] Fabio Poletto, Marco Stranisci, Manuela Sanguinetti, Viviana Patti, Cristina Bosco, et al. 2017. Hate speech annotation: Analysis of an italian twitter corpus. In *Ceur workshop proceedings*, Vol. 2006. CEUR-WS, 1–6.

[21] Marco Polignano, Pierpaolo Basile, Marco De Gemmis, Giovanni Semeraro, Valerio Basile, et al. 2019. Alberto: Italian BERT language understanding model for NLP challenging tasks based on tweets. In *CEUR Workshop Proceedings*, Vol. 2481. CEUR, 1–6.

[22] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[23] Vincenzo Riccio and Paolo Tonella. 2022. When and Why Test Generators for Deep Learning Produce Invalid Inputs: an Empirical Study. *arXiv preprint arXiv:2212.11368* (2022).

[24] 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. *Evaluation Campaign of Natural Language Processing and Speech Tools for Italian* (2020).

[25] Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An italian twitter corpus of hate speech against immigrants. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

[26] Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS.. In *Lrec*, Vol. 2012. Citeseer, 2214–2218.

[27] Rong Xiang, Emmanuele Chersoni, Qin Lu, Chu-Ren Huang, Wenjie Li, and Yunfei Long. 2021. Lexical data augmentation for sentiment analysis. *Journal of the Association for Information Science and Technology* 72, 11 (2021), 1432–1447.

[28] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. *arXiv preprint arXiv:2104.08826* (2021).

[29] Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). *arXiv preprint arXiv:2006.07235* (2020).

[30] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. 2015. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791* (2015).

[31] You Zhang, Jin Wang, and Xuejie Zhang. 2018. Ynu-hpcc at semeval-2018 task 1: Bilstm with attention based sentiment analysis for affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. 273–278.

[32] Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 745–760.