Political and Religious Hate Speech Detection from HaSpeeDe3 at EVALITA 2023

Gabriele Benanti (550552), **Luca Marini** (578543), **Francesco Mitola** (655383)
{g.benanti, l.marini11, f.mitola2}@studenti.unipi.it

Abstract

This report outlines our approach for the HaSpeeDe3 at EVALITA 2023, that is focused on text pre-processing and pre-trained BERT-based encoders (UmBERTo, GilBERTo, AlBERTo) with LSTM, BiLSTM, or CNN layers. For Task A - Contexual, we enriched the textual data with the most of the available metadata, and for Task B, we extended the training set with HaSpeeDe2 data. Our methodology surpassed the baseline performances, with the UmBERTo + BiLSTM and UmBERTo + LSTM models achieving first place in the in-domain task and second place in the cross-domain task, respectively.

1 Introduction

The rapid increase in social media usage, particularly on platforms like Facebook and Twitter, has led to a rise in toxic language. These platforms have revolutionized communication by making it faster and, at times, anonymous, which has facilitated the dissemination of harmful content. Social media also plays a critical role in shaping public debates, particularly in the political arena. Political leaders utilize these platforms for propaganda purposes, while a significant segment of the population relies on them for news. However, discussions on sensitive topics often devolve into verbal aggression and the spread of hateful messages. Specifically, in Italy, Twitter users tend to focus on politically biased sources, occasionally circulating hate messages against policymaking. To address this issue, researchers are developing automated artificial intelligence models to detect and curb online harm. Here is where the HaSpeeDe3, an Italian-language hate speech detection challenge at EVALITA 2023, comes in, to test the in-domain performance of systems for detecting political hate speech, as well as the out-of-domain performance on religious topics.

For each subtask in HaSpeeDe3, our team performed extensive text pre-processing, then used three pre-trained BERT-based encoders, namely UmBERTo, GilBERTo and AlBERTo, on top of which it also inserted LSTM or BiLSTM or CNN layers or a combination of them, exploring their hyper-parameters through a grid search on a hold-out validation set. In particular, for Task A - Contextual, we integrated metadata concatenating them with the final features extracted by the models and then we fed them into a dense layer for the final classification. For Task B, we extended the original training set for Task A - Textual including also the HaSpeeDe2 training set. All the models selected from the model selection, perform better than the baselines, and among them, two models, in particular UmBERTo + BiLSTM and UmBERTo + LSTM outperform the majority of the competitors, reaching the first place on Task A/ Task B - XPoliticalHate and the second place in Task B - XReligiousHate, respectively.

The paper is divided into seven sections: in section 2, we introduce a brief overview of the work of the other challenger team, in section 3 we describe the dataset, the task and the scoring system used in the competition. In section 4, we outline the data pre-processing techniques used and the architectures of the models designed. Finally, in section 5, 6 and 7 we describe the experimental setting, we analyze the results and draw our conclusions.

2 Related Works

Since we decided to address a public task, we had at our disposal the works of the different teams

already submitted for the challenge. The team ExtremITA[1] designed two different approaches, the first consists in an T5-based encoder-decoder model, while the second is an instruction-tuned decoder-only model built upon the LLama foundational models. A different methodology was addressed by LMU[2], which performed a two steps fine-tuning of different pretrained language models (such as AlBERTo, UmBERTo, XLM-R and the multilingual mBERT). The first step of the fine-tuning was carried out on datasets external to the challenge, while the second step on the in-domain political dataset provided by the challenge. A particular effort was made by O-Dang Team[3] which implemented entity-linking (using O-Dang knowledge graph and the Davinci OpenAI model) combined with AlBERTo to leverage knowledge about named entities in the training data and their association to online abusive language.

Our project was inspired mainly by CHILab[4] and BERTicelli[5]. They are the only two teams in the competition that applied some form of preprocessing, maintaining the most frequent hashtags and normalizing some entities (e.g. user id became <user>). About the models, CHILab used pretrained AlBERTo and fastText, while BERTicelli made a fine-tuning of UmBERTo and BERT-base-ita cased.

In this context, we step up proposing our approach based on a stronger and more personalized data pre-processing with respect to the other teams. After this step, we employed some popular and state-of-art transformer-based encoders (UmBERTo, GilBERTo and AlBERTo) combined with deep learning models (BiLSTM, LSTM and CNN) capable of efficiently capturing dependencies in sequences. This allowed us to achieve a solid trade-off between saving computational time and gaining accuracy. In fact, we got very good results on the different tasks.

3 Data

The challenge focuses on detecting hate speech on social media, in particular political and religious topics on Twitter. The proposal consists in two binary classification tasks, Task A, an in-domain task and Task B, a cross-domain task.

3.1 In-domain Task

Task **A** is divided in two sub-tasks. The first one is a textual sub-task in which only the tweets texts from PolicyCorpusXL[6] dataset can be used. This dataset contains 7,000 tweets sampled from the ones with hashtags `#dpcm`, `#legge` and `#leggedibilancio` between March and May 2020 and between April and July 2021. The corpus has the following structure:

- `anonymized_tweet_id`: pseudo-random numerical identifier of the tweet;
- `anonymized_text`: text of the tweet after the application of anonymization techniques;
- `label`: attribute which tells if the tweet is hate speech (1) or not (0)
- `dataset`: attribute which specifies the domain of the tweet (political or religious)

PolicyCorpusXL was divided in a development set of 5,600 records and a test set of 1,400 tweets called XPoliticalHate.

The second subtask is a contextual task in which the tweet texts from PolicyCorpusXL and their corresponding metadata can be used as development set. In particular, between the metadata datasets provided by the challenge, we focused on `training_contextual` and `test_contextual` ones. From these dataset we used the following attributes:

- *retweet_count*: the number of times the tweet has been retweeted;
- *favorite_count*: the number of times the tweet has been liked;
- *statuses_count*: the number of tweets posted by the author of the tweet;
- *followers_count*: the number of followers of the author of the tweet;
- *friends_count*: the number of users the author of the tweet follows;
- *anonymized_description*: the self-description of the author of the tweet;
- *dataset*: attribute which specifies the domain of the tweet (political or religious).

For using the metadata, we merged the PolicyCorpusXL development set and test set with `training_contextual` and `test_contextual` datasets respectively, on the attributes *anonymized_tweet_id* and *dataset*.

3.2 Cross-domain Task

Task **B** provides two test sets from two different domain, political and religious. The former is the previously mentioned *XPoliticalHate*, the same as in Task A. The latter is *XReligiousHate*, which is the Italian part of a bigger hate speech dataset [7]. XReligiousHate consists of 3,000 tweets collected between December 2020 and August 2021 with keywords concerning Christianity, Islam and Judaism.

The Task B does not impose a particular development set, in fact, the participants can use any external dataset, including also the textual and contextual PolicyCorpusXL development data. For this reason, to enhance the performance of the models for the Task B - XReligiousHate, we extended PolicyCorpusXL textual development data with a corpus of tweets from the development set of HaSpeeDe2[8]. The main topics of these tweets are about hate against Muslims, Roma and immigrants. From the HaSpeeDe2 corpus development set we only used the anonymzed tweet text attribute and the label attribute indicating presence (1) or absence (0) of hate speech.

3.3 Required metrics

To submit valid results, the challenge requires participants to calculate performance in terms of average F1-score computed over the two binary classes (hate speech and non-hate speech).

4 Methodologies

4.1 Data Pre-processing

In order to maximize our models' performance, we designed a detailed data pre-processing pipeline for the textual data, namely the tweet strings in `anonymized_text` and `anonymized_description` attributes of our datasets. As the first step of our preprocessing, we developed custom functions to replace the disguised bad words and elongated words with their normal forms (e.g. of disguised bad words replacement: `caxxaro` to `cazzaro`; e.g. of elongated word replacement: `ciaoooo` to `ciao`). Then, we removed the punctuation marks, and all the Unicode emojis. All the words have been lower cased as well.

To enrich our text preprocessing pipeline, we decided to use *Ekphrasis* library[9], a popular tool comprising a custom NLP pipeline. This library was used to perform the following steps:

- Substitute URLs, emails, percents, money signs, time-formatted strings, numbers and phone numbers to their corresponding tags. For example, it uses `<user>` for the users (e.g. `@mariorossi` mapped to `<user>`), `<url>` for the urls and so on;
- Unpack hashtags and extract words within it. For example, an hashtag such as `#governodeipeggiori` becomes `<hashtag> governo dei peggiori </hashtag>`

In order to split correctly words comprised in hashtags, a word segmentation algorithm using a Viterbi algorithm implementation from Beautiful Data book (Segaran and Hammerbacher, 2009) [1] is employed. The algorithm is composed by four steps:

1. define a probabilistic model by assigning a probability to each string in the corpus provided;
2. define a set of candidates that are the different ways of splitting a merged string (the hashtags) into a set of words (even without any meaning);
3. compute each set of words probability as a multiplication of unigram or bigram according to the probabilistic model;
4. select the set of words with the highest probability.

As emerged from its description, in order to segment Italian merged string, this algorithm needs word statistics extracted from an Italian corpus, so we decided to compute the statistics using the built-in *Ekphrasis* script *generate_stats.py* with a subset of ten million Italian tweets taken randomly from TWITA dataset [10]. To avoid incorrect segmentations, each tweet within this subset has

[1] http://norvig.com/ngrams/ch14.pdf

been preprocessed in order to contain only correct words, so all bad words and elongated words have been substituted as described before, while URLs, mentions and hashtags have been removed.

Furthermore, we applied the tokenizer module of each selected encoder and we investigated the tokenized resulting texts. We discovered that UmBERTo and GilBERTo tokenizers cannot identify "<" symbol, used as first character in tags introduced after the text preprocessing. This results in tokenizer mapping all the occurencies of that symbol to [UNK] token. In order to reduce [UNK] token and to make tags understandable to UmBERTo and GilBERTo, we replaced "<" and ">" symbols with "[" and "]" symbols, respectively. In this way, tags took the following form: "[entity type]".

4.2 Architecture specification

The architecture adopted to solve the competition tasks consists of three main modules: the first is a BERT-base encoder to generate contextual embeddings from the input tweet texts; the second consists of a model capable of processing and capturing dependencies in sequences; finally, the third is a MLP for the final classification.

All the architectures were implemented using Keras library [11].

4.2.1 Encoders

As first module of our architecture, we decided to explore three pretrained BERT-based model, AlBERTo, UmBERTo, GilBERTo.

We decided to set the maximum number of tokens to take as input to each model to the number of tokens of the longest tweets in input. In this way, we avoided any input truncation. For Task A - Textual and Task B, we set the maximum input length to 290 tokens for the AlBERTo[12] encoder and 357 tokens for UmBERTo [13] and GilBERTo [14] encoders. Since in Task A - Contextual, we concatenated to each tweet its *anonymized_description* attribute text value, we used the maximum input length available for the BERT-based models, namely 512.

AlBERTo model

AlBERTo[12] is a BERT-based model specifically designed to understand Italian language as used on social media, particularly Twitter. It adopts BERT[15] training strategy, leveraging a masked learning approach on a 12-layer Transformer Encoder. This process involves hiding a percentage of terms using a [MASK] token and training the model to guess these terms, optimizing network weights via back-propagation.

The model's architecture and training configurations include essential hyper-parameters such as dropout probabilities for attention and fully connected layers, the size of encoder and pooler layers, and the maximum sequence length the model can handle. Notably, AlBERTo's vocabulary is significantly larger than the original BERT's, including 128,000 lowercase words, allowing it to better encode slang, incomplete, or uncommon words found on social media. AlBERTo was trained on a randomly selected subset of 200 million tweets (about 191 GB) from the TWITA dataset [10], a large corpus of Italian tweets, that was pre-processed to normalize and tag text elements such as URLs, hashtags, and mentions effectively.

UmBERTo model

UmBERTo[13] builds upon the foundational RoBERTa base model architecture [16]. This model advances RoBERTa's capabilities through the adoption of two novel methodologies: SentencePiece and Whole Word Masking. The SentencePiece Model is a subword tokenizer and detokenizer that operates independently of language, tailored for text processing in neural networks. It generates subword units by adjusting to the specified vocabulary size and the language of the dataset. In contrast, Whole Word Masking ensures that masking occurs at the word level rather than the subword level. It applies a mask to an entire word if any part of that word, as segmented by the SentencePiece Tokenizer, was initially selected for masking. This approach ensures that only full words are obscured, avoiding the masking of individual subword components.

We used the UmBERTo pre-trained version on cased Commoncrawl ITA corpora (about 69 GB) from OSCAR [17]. The model excels in NLP tasks such as POS tagging and NER and it represents a significant advancement in processing Italian text, offering state-of-the-art results for Italian language understanding.

GilBERTo model

GilBERTo[14], an Italian language model based on RoBERTa[16], uses the CamemBERT[18] approach for tokenization. It leverages subword masking and was trained over roughly 71GB of Italian text, with a vocabulary of 32k Byte Pair Encoding (BPE) subwords generated via SentencePiece tokenizer. GilBERTo has been evaluated on Part-of-Speech tagging and Named Entity Recognition tasks, demonstrating competitive performance.

4.2.2 Sequence processing and final classification

For Task A - Textual, on top of the encoders we try different popular types of layer for sequence processing such as LSTM or BiLSTM or CNN or BiLSTM + CNN.

Among the different architecture we tried, there are three relevant ones that have distinguished features.

In particular, the first one is the LSTM/BiLSTM model architecture, shown in Figure 2, which consists of an encoder, followed by the LSTM/BiLSTM layer, a global max-pooling and a dropout layer.

The second one is the CNN model architecture, shown in Figure 3, which starts with an encoder followed by a convolutional layer using Rectified Linear Unit (ReLU) as its activation function. After this, a max-pooling layer is employed to diminish the dimensions of the output. Lastly, a dropout layer is incorporated as an effective method for regularization.

In the last architecture, the BiLSTM + CNN one, shown in Figure 4, we merged the two previous architectures.

Moreover, all the architectures finish with a fully connected layer and an output layer.

For Task A - Contextual, we modified the three model structures mentioned earlier in order to use tweet metadata. First of all, in order to obtain contextual and meaningful embeddings of the tweet text and the user description metadata, we concatenated them together inserting the [SEP] token between the end of the tweet and the start of the description. Then, we modified also the input to the classification layers (MLP), combining the numerical normalized data to the pooled output of the LSTM or BiLSTM or BiLSTM + CNN or CNN layers, as shown in Figure 5 and Figure 6 and Figure 7).

Since the metadata did not improve the performance of our models and the CNN-based models proved to be the worst performers, as it will be explained in section 6, for Task B, we tested the same architectures used in Task A - Textual, excluding the CNN-based one (see to Figure 2 and Figure 4).

5 Experiments

Since a large number of model architecture variants were tested, all the experiments were run on different platforms such as Google Colab [2], Kaggle [3] and a UniPi server with *NVIDIA Tesla P100 PCIe 16GB*, in order to use as many GPUs as possible and to parallelize the different grid searches to speedup the model selection.

For each task in the competition, we have performed an extensive model validation. In particular, for each task we applied an hold-out split of the original development set, stratified over the class attribute, using 80% of the data as training set and the remaining data, 20%, as validation set. To search model hyper-parameter space, we employed a grid search using Keras `GridSearch` methods [11].

During the experiments, we have preferred to freeze the pre-trained encoder parameters. This choice was due to the fact that two out of three tasks of the challenge, the in-domain tasks, only provide little training data (5600 records). Moreover, some of the pre-trained transformer-based encoders used, such as AlBERTo, have already been trained specifically on a large corpus of tweets or, in case of UmBERTo and GilBERTo on a large selection of Italian texts. Proceeding in this way, we managed to spare a lot of computational time and explore a larger range of architectures.

We also decided to fix Adam as training optimizer, which is computationally efficient, has little memory requirement and is well suited for problems (as ours) that are large in terms of parameters [19]. Moreover, since we were solving a problem of binary classification, we decided to employ

[2] https://colab.google/

[3] https://www.kaggle.com/

binary cross entropy as training and validation loss, keeping track also of the F1-Score values.

For each task, the maximum number of epochs was set to a small number (60), since we observed over-fitting typically after 8 to 10 epochs, regardless of the model configuration. In order to avoid over-fitting, an early stopping technique based on the validation set loss was employed with a patience of 3 or 5 epochs, and different dropout percentages were experimented.

Once the model selection was completed and the resulting best models according to validation loss and validation F1-score values were obtained, we evaluated them on the gold test set and measured their performance in terms of F1-Score averaged over the two classes of the dataset, as required by the challenge.

Finally, for reproducibility purposes, we set a random seed (42) for the development set stratified split and for the weight initialization of the tested models.

In the following subsection, we analyze some preliminary trials and the experimental setup for each task in detail.

5.1 Preliminary trials

In order to reduce the hyper-parameter search space, we performed some preliminary experiments involving one or more layer of LSTM, CNN, BiLSTM and densely connected NN layer, with a number of units between 16 and 128. We have also explored a wide range of learning rates varying between 1e-6 and 1e-2 and dropout rates between 0.3 and 0.7.

In most cases, the best results were obtained with the simpler architectures consisting mainly of a single layer of LSTM, BiLSTM, CNN or BiLSTM+CNN, and a densely connected NN layer with units between 16 and 64, learning rate between 1e-2 and 1e-3, dropout between 0.5 and 0.7.

Therefore, we set a finer grid search for each task.

5.2 Hyper-parameter tuning setup

After performing different preliminary trials on larger ranges for each relevant hyper-parameter, as described in the previous subsection (5.1), we decided to build a finer grid search shared by every task, testing several hyper-parameters which ranges can be read in Table 1. This table describes the hyper-parameters values explored varying the type of architecture used to solve each task. As we have frozen the weights of the BERT-based encoders, no hyper-parameters tuning was needed for them. Reading the table, we can also observe that all the architectures share some hyper-parameters, namely learning rate, the units for the dense layer and the norm that weights must not exceed, used for gradient clipping. The architectures with BiLSTM or LSTM layers require the tuning of specific hyper-parameters such as the number of the units for that specific layer, the dropout rate and the recurrent dropout, while architectures with 1D convolutional neural network layer include hyper-parameters to search such as the number of filters and the kernel size.

Finally, we also tested BiLSTM + CNN architectures. For these architectures we had to tune hyper-parameters of both the LSTM and CNN layers.

6 Result and Analysis

In this section we discuss the best model obtained after the model selection and their results on the test sets for each task.

6.1 Task A - Textual

For this task, the best performing models out of the model selection were mainly the one composed by UmBERTo encoder and BiLSTM or LSTM, which performed also surprisingly well on the test sets. In particular, the model composed by UmBERTo and BiLSTM achieved the best validation loss (0.226) and the best average F1-Score on validation set (0.89) (see Table 2). This model also overtakes the first currently ranked model for the task, in terms of F1-Score results on the test set, as we can see in Table 5. This shows us that the combination of an accurate text preprocessing pipeline, the use of state-of-art transformer encoders to generate context aware embeddings, and the use of powerful models for processing sequences such as LSTM or BiLSTM, allows to achieve the best performance of the competition even without necessarily fine-tuning the encoders or building very complex architectures on top of them. For the sake of completeness, we have also reported the learning curves of this

Encoder	Model	hyper-parameter name	hyperparameter range
AlBERTo or UmBERTo or GilBERTo	all	learning rate	[1e-2, 1e-3, 1e-4]
		dense layer units	[16, 32, 64]
		clipnorm	[1, 3, 5]
	BiLSTM or LSTM	units	[32, 64, 128]
		dropout rate	[0.3 - 0.7]
		recurrent dropout	[0, 0.3, 0.5]
	CNN	filters	[32, 64, 128]
		kernel size	[3, 7, 9]

Table 1: Grid Search hyper-parameters varying the transformer-based enconder and the classifier architecture on top of it.

model (see Figure 1). Not far from the top are the UmBERTo-LSTM and GilBERTo-BiLSTM models, which achieve competitive results (see Table 2). On the other hand, the use of CNN-based models or the mixing of BiLSTM with CNN layers, even if they outperformed the baseline models performance proposed by the authors of the challenge (0.85-0.87 average F1-Score versus 0.846 of the baseline, a Support Vector Classifier with a unigram textual representation [20]) did not show relevant performance.

6.2 Task A - Contextual

For this task, enriching the input with metadata did not bring any particular benefits. On the contrary, it often worsened the performance of the best models found for the textual part of the same task. For example, as we can see in Tables 2 and 3, the best model, UmBERTo + BiLSTM, remains always one of the best also for Task A - Contextual but it scores 0.2508 of validation loss, 0.8714 of average validation F1-score and 0.9065 of average test F1-score, against 0.226, 0.890 and 0.919 on the Task A - Textual dataset.

Moreover, the same considerations made for all the other models in Task A - Textual can be applied also to this task.

For this reason, and because a run submission for Task A - Textual also meets the requirements of Task A - Contextual [20], we have included our best model on Task A - Textual, namely UmBERTo + BiLSTM, also for the contextual task in the Table 5, which shows the results of each team in the competition.

6.3 Task B

Since as reported in section 3.2, there was no dataset constraint in this task, and since adding metadata did not seem to give any advantage in the performance of the models, we decided to train each model on the task A - Textual dataset extended with HaSpeeDe2 dataset. This proved to be particularly useful for performance on the XReligiousHate test set, where as can be seen in Table 4, our best model, UmBERTo + LSTM, achieved 0.80 and 0.64 of average validation and test F1-Score, respectively. This allows our team to achieve the second place on the competition for Task B - XReligious-Hate (see Table 5). On the other hand, performance on the XPolitical-Hate test set did not improve with the extension of the development set, and since the runs for Task A - Textual are also valid for Task B - XPolitical-Hate, the best model on Task - A Textual was also included in Table 5 for XPoliticalHate.

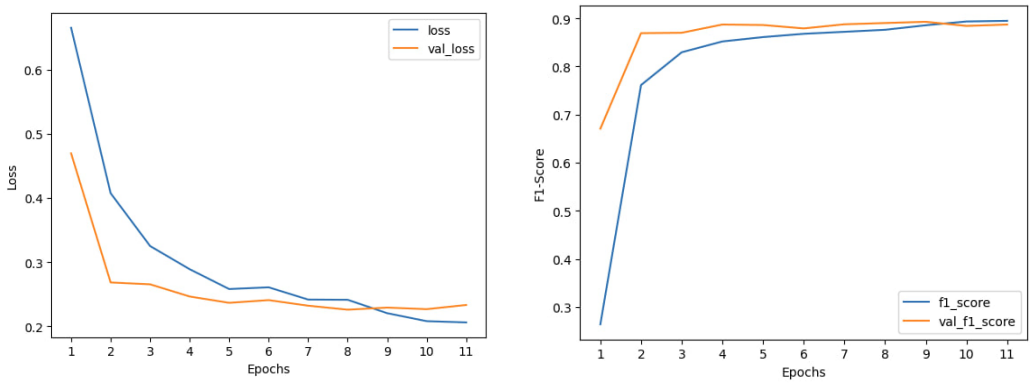
7 Conclusion

The analysis of the results shows that a significant increase in performance can be achieved through a combination of different factors:

- a careful and meticulous text pre-processing;
- the use of state-of-the-art transformer-based encoders specifically pre-trained on tweets (AlBERTo) or on a large number of Italian texts (UmBERTo and GilBERTo), even without fine tuning;

Models	hyper-parameters	validation loss	validation F1-score	test F1-score
UmBERTo + BiLSTM	LSTM units: 64, dense units: 32, dropout rate: 0.6, l.r.: 0.01, clipnorm: 3	**0,2260**	**0,8902**	**0,9192**
UmBERTo + LSTM	LSTM units: 64, dense units: 32, dropout rate: 0.6, l.r.: 0.01, clipnorm: 3	0,2380	0,8858	0,9093
GilBERTo + BiLSTM	LSTM units: 64, dense units: 32, dropout rate 0.5, l.r.: 0.01, clipnorm: 3	0,2458	0,8754	0,9092

Table 2: Top three model for **Task A - Textual**. The omitted hyper-parameters are set to 0.



(a) (b)

Figure 1: Training and validation loss (a) and average F1-Score (b) curves of the best model for **Task A - Textual**, namely UmBERTo + BiLSTM. The early stopping was at epoch 8, preventing the over-fitting.

- the use of powerful architectures for sequence processing such as LSTM with a small number of layers, which favors computationally efficient and extensive testing.

These factors proved to be crucial in solving the tasks of the challenge, and even though we did not use very complex architectures, we were able to achieve first position in all in-domain subtasks and the second position in the cross-domain task in terms of average test F1-Score (see Table 5).

References

1. Hromei, C., Croce, D., Basile, V. & Basili, R. Extremita at evalita 2023: Multi-task sustainable scaling to large language models at its extreme. *CEUR Workshop Proceedings.* `https://iris.unito.it/bitstream/2318/1945514/1/paper13.pdf` (2023).
2. Hangya, Viktor & Alexander, F. *LMU at HaSpeeDe3: Multi-Dataset Training for Cross-Domain Hate Speech Detection* in *The Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)* (CEUR Workshop Proceedings, 2023). `https://ceur-ws.org/Vol-3473/paper24.pdf`.
3. Bonaventura, C. D., Muti, A. & Stranisci, M. A. *O-Dang at HODI and HaSpeeDe3: A Knowledge-Enhanced Approach to Homo-*

Model	hyper-parameters	validation loss	validation F1-score	test F1-score
UmBERTo + BiLSTM	LSTM units: 64, dense units: 32, dropout rate: 0.5, learning rate: 0.01, clipnorm: 3	0,2508	0,8714	0,9065
UmBERTo + LSTM	LSTM units: 64, dense units: 16, dropout rate: 0.5, learning rate: 0.01, clipnorm: 3	0,2579	0,8646	0,9098
GilBERTo + BiLSTM	LSTM units: 64, dense units: 32, dropout rate: 0.5, learning rate: 0.001, clipnorm: 1	0,2655	0,845	0,8924

Table 3: Top three models for **Task A - Contextual**. The omitted hyper-parameters are set to 0.

Model	hyper-parameters	validation loss	validation F1-score	test F1-score
UmBERTo + LSTM	LSTM units: 64, LSTM rec. dropout: 0.3 dense units: 64, dropout rate: 0.6, learning rate: 0.01, clipnorm: 3	**0,3492**	**0,8031**	**0,6476**
UmBERTo + BiLSTM	LSTM units: 64, LSTM rec. dropout: 0.3 dense units: 64, dropout rate: 0.6, learning rate: 0.01, clipnorm: 3	0,3496	0,7948	0,6276
AlBERTo + LSTM	LSTM units: 64, dense units: 32, dropout rate: 0.6, learning rate: 0.001, clipnorm: 1	0,3738	0,7955	0,6312

Table 4: Top three models for **Task B**. The omitted hyper-parameters are set to 0.

transphobia and Hate Speech Detection in Italian in (CEUR Workshop Proceedings, 2023). https://ceur-ws.org/Vol-3473/paper29.pdf.

4. Siragusa, I. & Pirrone, R. *CHILab at HaSpeeDe3: Overview of the Taks A Textual* in (CEUR Workshop Proceedings, 2023). https://ceur-ws.org/Vol-3473/paper23.pdf.
5. Grotti, L. & Quick, P. *BERTicelli at HaSpeeDe 3: Fine-tuning and Cross-validating Large Language Models for Hate Speech Detection* in (CEUR Workshop Proceedings, 2023). https://ceur-ws.org/Vol-3473/paper25.pdf.
6. Celli, F., Lai, M., Duzha, A., Bosco, C. & Patti, V. *Policycorpus XL: An Italian Corpus for the Detection of Hate Speech Against Politics* in (CEUR Workshop Proceedings). https://ceur-ws.org/Vol-3033/paper38.pdf.
7. A., R., B., T., S., T. & E., J. Addressing religious hate online: from taxonomy creation to automated detection. *PeerJ Computer Science* **8,** e1128. https://doi.org/10.7717/peerj-cs.1128 (2022).
8. Sanguinetti, M. *et al.* HaSpeeDe 2 @ EVALITA2020: Overview of the EVALITA 2020 Hate Speech Detection Task. https://pdfs.semanticscholar.org/ebbd/1ba976c33ab1ed71e7f03cbb38b6c9359572.pdf (2020).
9. Baziotis, C., Pelekis, N. & Doulkeridis, C. *DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis* in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (eds Bethard, S. *et al.*) (Association for Computational Linguistics, Vancouver, Canada, Aug. 2017), 747–754. https://aclanthology.org/S17-2126.
10. Et al., V. B. *TWITA* http://valeriobasile.github.io/twita/about.html.
11. Chollet, F. *et al. Keras* https://github.com/fchollet/keras. 2015.

Team	Task A	Task B	
	textual/contextual	XPoliticalHate	XReligiousHate
our team	**0.9192**	**0.9192**	0.6476
BERTicelli	0.8976	0.8976	0.5401
CHILab	0.8516	0.8516	-
extremITA	0.9079	0.9079	**0.6525**
INGEOTEC	0.8845	0.8845	0.5522
LMU	-	0.9014	0.6461
odang4	0.9128	0.9128	0.5213

Table 5: Summary table showing the average test F1-Score of the best model of each team, including ours, for each task of the competition (only the best results over two possible runs per each team).

12. Polignano, M., Basile, P., de Gemmis, M., Semeraro, G. & Basile, V. *AlBERTo: Italian BERT language understanding model for NLP challenging tasks based on tweets* in. **2481** (2019).
13. Parisi, L., Francia, S. & Magnani, P. *UmBERTo: an Italian Language Model trained with Whole Word Masking* `https://github.com/musixmatchresearch/umberto`. 2020.
14. Ravasio, G. & Perna, L. D. *GilBERTo: An Italian pretrained language model based on RoBERTa* `https://github.com/idb-ita/GilBERTo`. 2020.
15. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* 2019. arXiv: `1810.04805 [cs.CL]`.
16. Liu, Y. *et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach* 2019. arXiv: `1907.11692 [cs.CL]`.
17. Ortiz Suárez, P. J., Sagot, B. & Romary, L. *Assembling the Jigsaw: Extracting Multilingual Resources from the Web* in *Proceedings of the 12th Language Resources and Evaluation Conference* (Marseille, France, 2020), 2023–2030. `https://oscar-project.org`.
18. Martin, L. *et al. CamemBERT: a Tasty French Language Model* in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics, 2020). `http://dx.doi.org/10.18653/v1/2020.acl-main.645`.
19. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* 2017. arXiv: `1412.6980 [cs.LG]`.
20. Lai, M. *et al. HaSpeeDe3 at EVALITA 2023: Overview of the Political and Religious Hate Speech Detection task* in (Sept. 2023).

Appendix

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 320)]	0	[]
attention_mask (InputLayer)	[(None, 320)]	0	[]
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 320, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	184345344	['input_ids[0][0]', 'attention_mask[0][0]']
bidirectional (Bidirectional)	(None, 320, 128)	426496	['tf_bert_model[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 128)	0	['bidirectional[0][0]']
dropout_37 (Dropout)	(None, 128)	0	['global_max_pooling1d[0][0]']
dense (Dense)	(None, 32)	4128	['dropout_37[0][0]']
outputs (Dense)	(None, 1)	33	['dense[0][0]']

Total params: 184776001 (704.86 MB)
Trainable params: 430657 (1.64 MB)
Non-trainable params: 184345344 (703.22 MB)

Figure 2: Encoder with BiLSTM layer on top for Task A textual or Task B

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 94)]	0	[]
attention_mask (InputLayer)	[(None, 94)]	0	[]
tf_bert_model_4 (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 94, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	184345344	['input_ids[0][0]', 'attention_mask[0][0]']
conv1d_2 (Conv1D)	(None, 88, 32)	172064	['tf_bert_model_4[0][0]']
max_pooling1d_2 (MaxPooling1D)	(None, 44, 32)	0	['conv1d_2[0][0]']
dropout_187 (Dropout)	(None, 44, 32)	0	['max_pooling1d_2[0][0]']
flatten_1 (Flatten)	(None, 1408)	0	['dropout_187[0][0]']
dense_1 (Dense)	(None, 16)	22544	['flatten_1[0][0]']
outputs (Dense)	(None, 1)	17	['dense_1[0][0]']

Total params: 184539969 (703.96 MB)
Trainable params: 194625 (760.25 KB)
Non-trainable params: 184345344 (703.22 MB)

Figure 3: Encoder with CNN layer on top for Task A textual or Task B

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 300)]	0	[]
attention_mask (InputLayer)	[(None, 300)]	0	[]
tf_bert_model_1 (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 300, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	184345344	['input_ids[0][0]', 'attention_mask[0][0]']
bidirectional_1 (Bidirectional)	(None, 300, 128)	426496	['tf_bert_model_1[0][0]']
conv1d_1 (Conv1D)	(None, 298, 128)	49280	['bidirectional_1[0][0]']
max_pooling1d_1 (MaxPooling1D)	(None, 149, 128)	0	['conv1d_1[0][0]']
dropout_75 (Dropout)	(None, 149, 128)	0	['max_pooling1d_1[0][0]']
flatten_1 (Flatten)	(None, 19072)	0	['dropout_75[0][0]']
dense_1 (Dense)	(None, 16)	305168	['flatten_1[0][0]']
outputs (Dense)	(None, 1)	17	['dense_1[0][0]']

Total params: 185126305 (706.20 MB)
Trainable params: 780961 (2.98 MB)
Non-trainable params: 184345344 (703.22 MB)

Figure 4: Encoder with BiLSTM + CNN layers for Task A textual or Task B

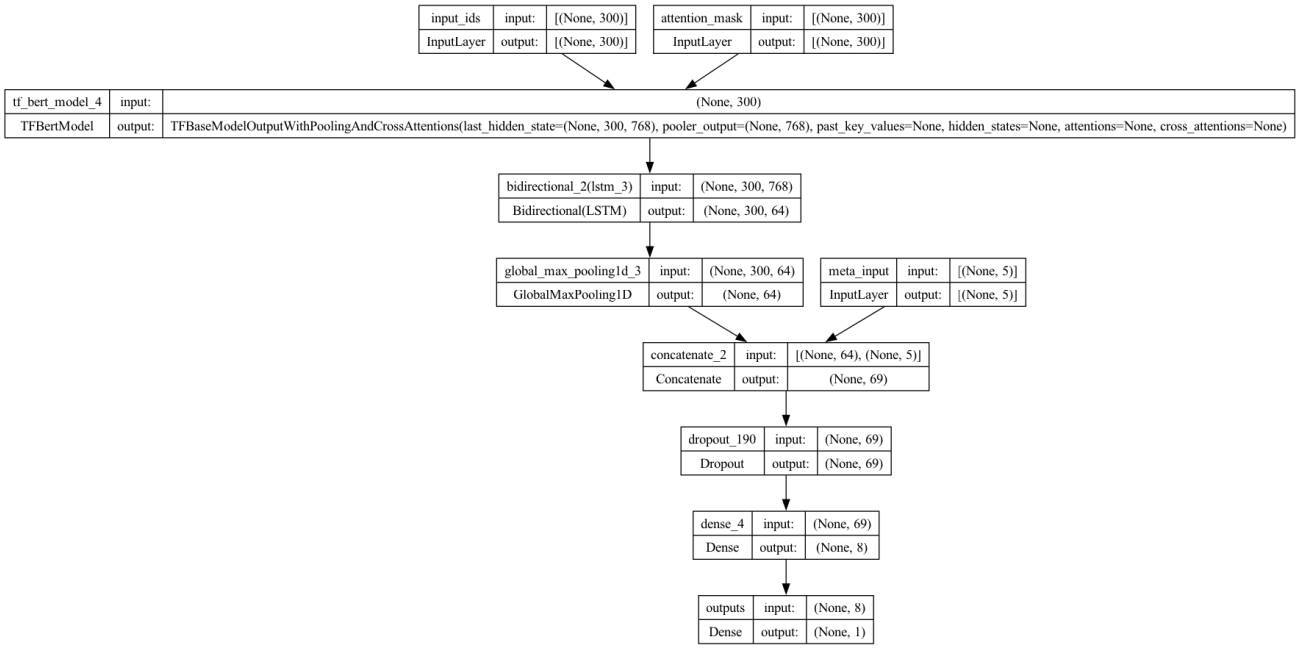


Figure 5: Encoder with BiLSTM layer for Task A contextual

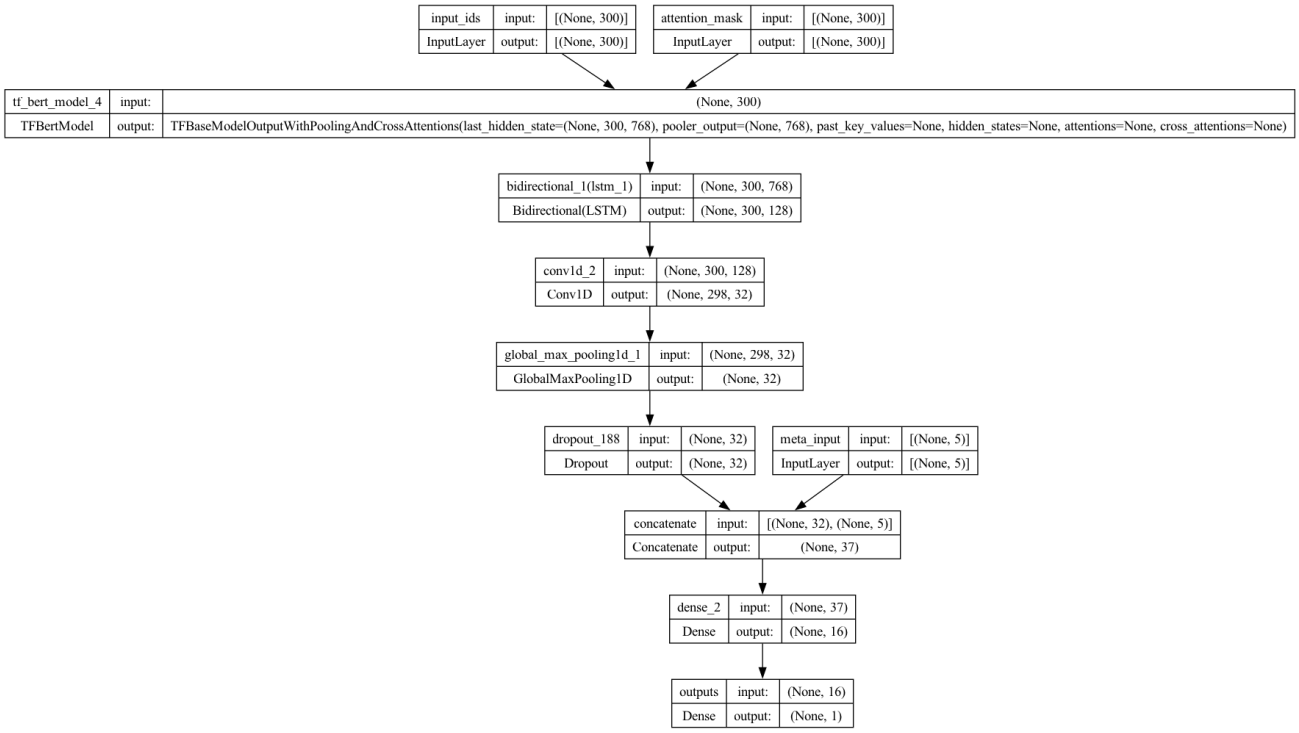


Figure 6: Encoder with BiLSTM + CNN layers for Task A contextual

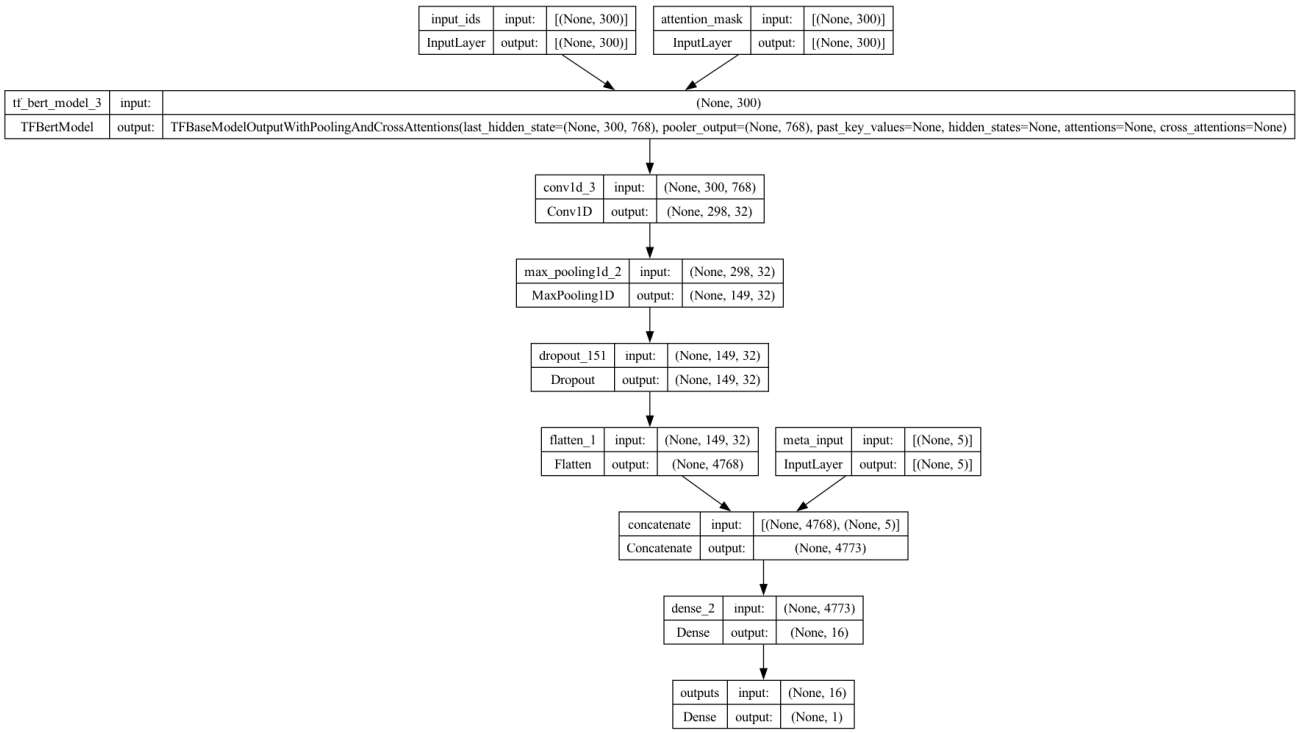


Figure 7: Encoder with CNN layers for Task A contextual