Automatic Misogyny Identification

Martina Trigilia
m.trigilia@studenti.unipi.it
Data Science & Business Informatics
Text Analytics (635AA), Academic Year: 2021/22

September 7, 2022

Contents

1 Introduction **2**

2 Data Understanding and Pre-processing **2**

3 Data Classification **6**
3.1 Naive Bayes, Logistic Regression, SVM, Decision Tree, Random Forest . 7
3.2 CNN 9
3.3 LSTM 12
3.4 AlBERTo - Italian BERT model 13
3.5 Final results on External Test Set 14

4 Appendix **16**

1 Introduction

Twitter is a social media platform used by hundreds of millions of people around the world to network and share information. The platform is not immune to many of the abuses, harassment and violence that women face offline, resulting in misogynistic comments, often with little accountability due to anonymity.
Therefore, it is important to identify and remove this type of content from social networks. It is also important to guarantee the fairness of models, referring to errors due to unintended bias.
For the 7th evaluation campaign EVALITA 2020 it was presented a shared task on Automatic Misogyny Identification, which proposed the resolution of two main sub tasks:

- **Subtask A - Misogyny & Aggressive Behaviour Identification:** a system must recognize if a text is misogynous or not, and in case of misogyny, if it expresses an aggressive attitude.
- **Subtask B - Unbiased Misogyny Identification:** a system must discriminate misogynistic contents from the non-misogynistic ones, while guaranteeing the fairness of the model (in terms of unintended bias) on a synthetic dataset.

The objective of this project is to address Subtask A. A raw dataset containing tweets in italian language is provided for this specific task resolution. Tweets in this dataset are already hand-classified according to the definition of misogynous and aggressive in the Automatic Misogyny Identification paper.
To evaluate the generalization abilities of the systems, a test set with data collected in a different time period is also released: this is characterized by higher language variability with respect to the training data.

2 Data Understanding and Pre-processing

The dataset is a collection of 4409 tweets with 4 columns, which are:

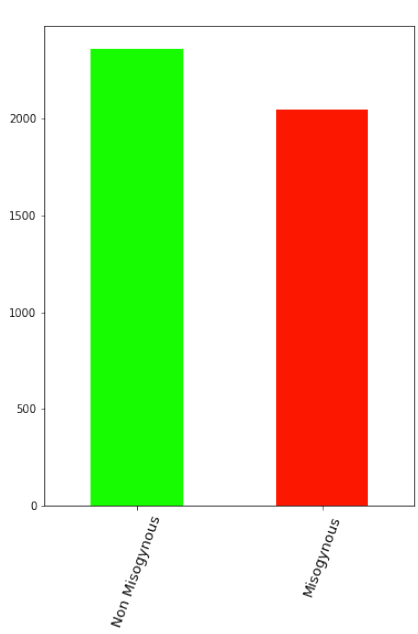
- **id:** unique identifier which is removed from the dataset.
- **text:** content of the tweet written in Italian language.
- **misogynous:** binary variable indicating whether a tweet is misogynous or not.
- **aggressiveness:** binary variable indicating whether a misogynous tweet has an aggressive tone or not.

Therefore, a tweet can only be classified as Non-Misogynous, Non-Aggressive Misogynous or Aggressive Misogynous. To better encode this information, an additional variable ***Target*** is added, which takes values 0 (Misogynous = 0 & Aggressiveness = 0), 1

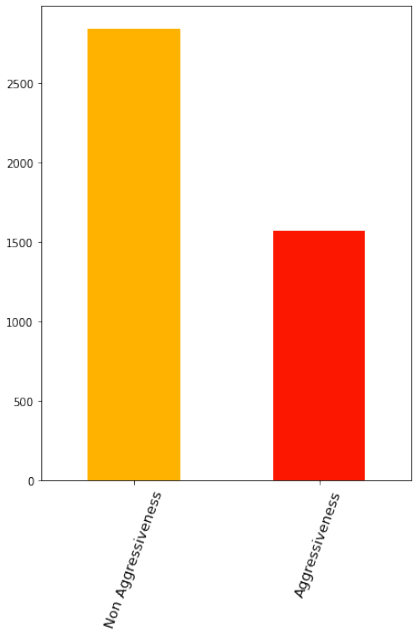
(Misogynous = 1 & Aggressiveness = 0), or 2 (Misogynous = 1 & Aggressiveness = 1).

Distribution of target variables

As can be seen in the Figure 1a, the dataset is fairly balanced with respect to the misogynous variable: around 53.60% of the tweets are classified as Non-Misogynous (2362 tweets). Instead, the distribution of Aggressiveness variable, in Figure 1b, is quite unbalanced, with only 35.59% of the tweets classified as aggressive (1569 tweets). Analysing the distribution of the new variable Target, in Figure 2, it can be seen that this is even more unbalanced: in fact, 76.65% of the misogynous tweets are aggressive, so only 10.8% of the total tweets are classified as Non Aggressive Misogynous.



(a) Misogynous



(b) Aggressiveness

Figure 1: Binary Classes Distribution

Tweets Analysis

The raw tweets in the dataset are not clean as they contain various emojis, special characters, numbers, punctuation, upper case words, hashtags, mentions (represented in the dataset by the key words <MENTION_1>, <MENTION_2>, etc.), and urls (also represented by the key word <URL>).

Before cleaning the tweets, an analysis of of these is carried out to see whether the noise within them has an impact on their classification. The largest percentage of hashtags is contained in non-misogynous tweets, while the most common hashtag is different from misogynous and non-misogynous tweets (most common hashtag in non-misogynous tweets *#Amici17*; most common hashtag in non-aggressive misogynous and aggressive

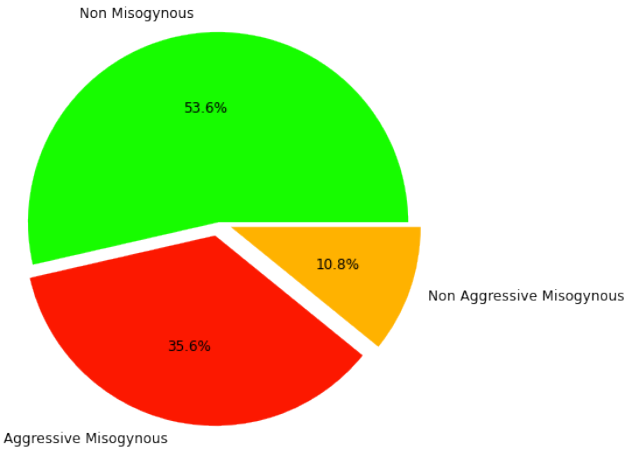


Figure 2: Target Distribution

misogynous tweets: *#GF15*). With regard to upper case words, these are usually used to emphasise a feeling, but they are present in a higher percentage in non-misogynous tweets. Punctuation is evenly distributed within the different classes, but the use of exclamation marks and question marks at the end of the tweets is more present, in percentage terms, in both non aggressive misogynous and misogynous ones. Moreover, mentions are more frequent in misogynous tweets in non misogynous ones.
Each tweet consists, on average, of 20 terms and 124 characters; if, on the other hand, we analyse the length of the tweets by class, we can state that misogynous tweets are, on average, shorter than non-misogynous tweets (17, 16 and 23 are, respectively, the average lengths per Aggressive Misogynous, Non-Aggressive Misogynous and Non-Misogynous tweet): this could be because, generally speaking, offensive messages are more concise and direct, and less articulate.

Pre-Processing
After the analysis, text cleaning was carried out, with the aim of improving the performance of the various models for the project resolution. The following steps were then performed:

1. Characters were lower-cased.

2. Punctuations were removed, except exclamation and question marks at the end of the tweets (they are be somewhat correlated with classification) which are replaced with <puntuaction>.

3. Emoticons and unicode emojis are replaced with <emoji>; also numbers are replaced with <number>.

4. Double spaces, symbols and special characters are removed, besides some of them which are transformed, e.g. € in 'soldi'. Also, note that the symbol #, which represents hashtag, is removed and the word after is not, as it may have a meaning, despite the fact that some of the terms used as hashtags are often made up of distinct words not separated by space.

5. Links and special tag rt (re-tweet) are removed as they add little information.

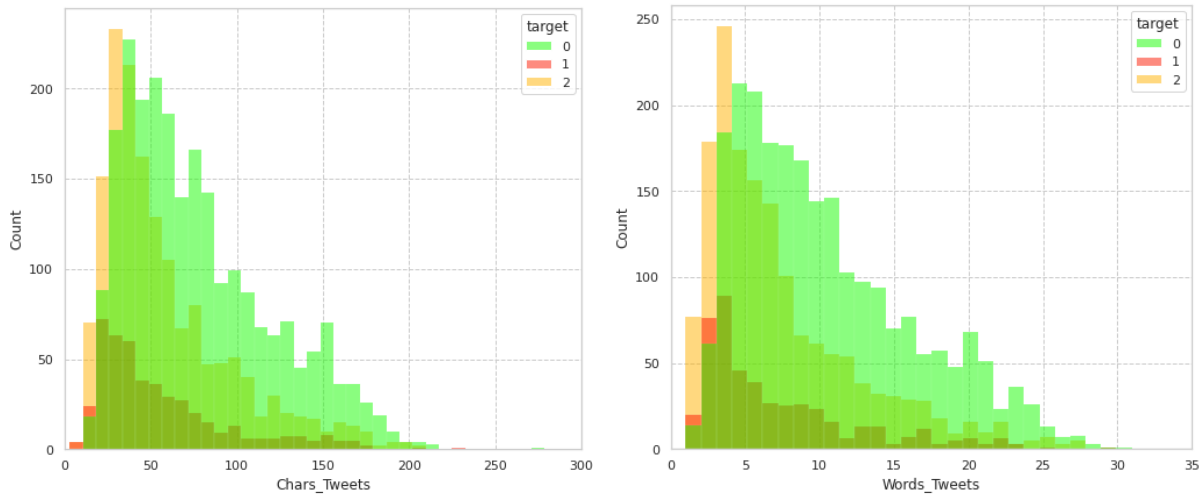
6. Some of the most common italian abbreviations are replaced with their corresponding full word, e.g. 'xkè' with 'perchè'.

7. Repeated consecutive words are transformed into one, this happens, for example, when more mentions are present in the tweet.

8. Repeated consecutive vowels and more than two repeated consecutive consonants are corrected, e.g. 'bellissimaaa' in 'bellissima'.

After tokenization of tweets, italian stopwords were removed as well as one-character long words. Finally, words are lemmatized, in particular verbs are converted to their base form. These steps are performed through a pre-trained italian pipeline available on the spaCy library (*it core news lg*).
In spite of this initial cleaning phase of the text, it soon became evident that a lot of words were still misspelled, also due to the very informal language used in the platform. In particular, 23.4% of the set of words (11.879 distinct words, not considering tags) are misspelled, according to the AlBERTo dictionary, even if they appear with low frequency (8% frequency over total words). An attempt to further improve the quality of text was then made with the use of PyspellChecker library, which give a suggested correction for words out of AlBERTo vocabulary. After this operation, words out of vocabulary dropped to 7.7% with 3% of frequency.

Analisys after preprocessing
After the preprocessing phase, the number of distinct words decreased considerably, for a total of 10244 unique words, without considering tags. Most of the tweets contain less than 20 words (Figure 3b), and the same considerations before pre-processing apply for the different classes. In particular, only 33 tweets exceed 35 words. Also, tweets consist of an average of about 80 characters (Figure 3a). In Figure 4a, 4b and 4c are shown Word Cloud plots for non-aggressive misogynous and non-misogynous tweets: in non-misogynous tweets the size of words is smaller than than the ones in misogynous tweets, so we can say that in those tweets words are repeated less frequently and, therefore, there is a greater variety of words used. Finally, frequency of bigrams was also analysed, with respect to the different classes (Figure 6a, 6b and 6c in the Appendix).



(a) Chars per Tweets

(b) Words per Tweets

Figure 3: Words Statistics.

(a) Non Aggressive Misogynous

(b) Non Misogynous

(c) Aggressive Misogynous

Figure 4: Words Cloud.

3 Data Classification

The pre processed dataset was split into a training set and an internal test set, according to the 80-20 rule, maintaining the proportion of the variable Target in the two sets. The internal test set was used to evaluate all the final models. The models that obtained the highest scores on the internal test set were subsequently re-trained and tested in the test set provided by the challange, which, as mentioned previously, contains tweets with a different variety of vocabulary. The metric used to evaluate model performances is f1 score, as it is the official one for the challange.
The classification task was then approached with two different strategies:

1. A model for the **binary classification of the mysogenous variable**, followed by another model for the **binary classification of the aggressiveness variable** (e.g. one DecisionTree followed by another DecisionTree), both trained on the same training set. Due to the way the problem is formulated, during the testing

phase, the second classifier should check the presence or absence of aggressiveness only on those tweets marked as misogynous by the first classifier, however, the results are worse than when the entire test set is used to evaluate the models (and thus considering the two tasks as independent problems); results of both testing strategies are reported.

2. A **single multinomial model** for the classification of the Target variable, containing three classes.

3.1 Naive Bayes, Logistic Regression, SVM, Decision Tree, Random Forest

As a first approach to the classification task, some of the more classical algorithms were tested, namely Naive Bayes, Logistic Regression, SVM, DecisionTree and Random Forest. Before these models could be applied, it is necessary to perform some transformations of the tweets, which were then processed by a pipeline of methods:

1. Through *CountVectorizer*, text is vectorized and transformed into a matrix, which indicates the frequency of vocabulary words for each tweet.

2. Feature selection is performed using *SelectKBest* and *chi2* as a score function, which returns the k features most related to the target variable, and so more likely to provide important information, based on chi2 statistic values.

3. *TfidfTransformer*, which transforms the resulting count matrix to a normalized tf (term-frequency) or tf-idf (term-frequency times inverse document-frequency) representation.

Hyperparameters Tuning
For each of the models considered, a tuning phase of the hyperparameters was carried out in order to optimize the f1 macro metric. The algorithm used is **GridSearchCV**, with CV = 5. Table 12 in the Appendix shows the grid of hyperparameters used for all the models.

Results
In Table 1 is reported the **f1-score** obtained for each classifier, which were then re-trained with the best hyperparameters obtained from GridSearchCV. All classifiers perform better on the binary task of mysogenous classification, achieving satisfactory results. For the task of binary classification of the variable Aggressiveness, two different testing strategies were performed, as explained before: if the classifiers are tested only in tweets that were previously classified as mysogenous, they perform significantly worse than when the entire test set is used, due to the class Non-Aggressive Mysogenous. This is probably because, as explained in DU, the vast majority of misogenous tweets are aggressive, and

thus there is an imbalance between non aggressive misogenous and aggressive mysogenous tweets. Thus, when the classifiers are tested only in tweets that have previously been labelled as misogenous, they struggle to distinguish between tweets which are simply misogenous and the aggressive ones; whereas when the two problems are tested as if they were independent, the results are better because the non-aggressive tweets include the non-misogenous ones. This trend is even more evident if we look at the results obtained by the classifiers for the multinomial classification task of the Target variable, in Table 2,with the class 1 (Non-Aggressive Mysogenous) scores being extremely low. In all cases, we can state that **LogisticRegression** is the classifier that obtains, overall, the highest results. In the attempt to improve LR performance over the Target variable, training set was *re-balanced* with RandomOverSampler and LR was tre-trained; however performance is almost unchanged and macro avg f1 score only gets slightly better (0.70 instead of 0.69).

	Binary Mysogenous Variable			Binary Aggressiveness Variable - Testing Strategy 1			Binary Aggressiveness Variable - Testing Strategy 2		
	0	1	macro avg	0	1	macro avg	0	1	macro avg
NaiveBayes	0.89	0.87	**0.88**	0.84	0.74	**0.79**	0.30	0.78	**0.54**
LogisticRegression	0.91	0.90	**0.90**	0.86	0.74	**0.80**	0.53	0.78	**0.65**
LinearSVC	0.88	0.87	**0.87**	0.86	0.74	**0.80**	0.53	0.79	**0.66**
DecisionTreeClassifier	0.87	0.86	**0.87**	0.83	0.75	**0.79**	0.34	0.79	**0.57**
RandomForestClassifier	0.88	0.88	**0.88**	0.84	0.75	**0.79**	0.45	0.79	**0.62**

Table 1: NB, LR, SVM, DT and RF F1 Score on Internal Test Set - Binary Classification

	Multinomial Target Variable			
	0	1	2	macro avg
NaiveBayes	0.88	0.40	0.75	**0.68**
LogisticRegression	0.90	0.42	0.76	**0.69**
LinearSVC	0.90	0.41	0.77	**0.69**
DecisionTreeClassifier	0.87	0.36	0.73	**0.65**
RandomForestClassifier	0.88	0.42	0.73	**0.68**

Table 2: NB, LR, SVM, DT and RF F1 Score on Internal Test Set - Multinomial Classification

3.2 CNN

To build the Convolutional Neural Networks, an embedding layer is used in order to have a distributed word representation that allows words with similar meaning to have a similar representation. It was decided to use two different pretrained word embeddings, GloVe with 100 dimention and FastText with 300 as dimension. The structure of the Convolutional Neural Network, Figure 7a in the Appendix, consists of two convolutional layers spaced by max pooling and dropout layers, followed by a fully connected layer and finally an output layer with Sigmoid as activation function. The binary crossentropy and categorical crossentropy loss functions have been used for the binary and multinomial task respectively. The optimiser used is the one that implements the Adam algorithm. Eventually, early stopping technique is used to stop the training of the network after 5 epochs with no improvement of the validation loss, which has to be minimized.

Hyperparameters Tuning
The hyperparameters of the network were fine tuning using the Hyperband function from the keras-tuner library. In particular: number of filters in output, kernel size and L2 regularization factor for convolution layers, fraction of input units to drop for dropout layers, units and L2 factor for the fully connected layer, and learning rate for the optimizer. Grid of hyperparameters is shown in Table 13 in the Appendix.

Results
CNNs were then trained with the best parameters obteined with fine tuning, and the results of the binary and multinomial task are respectively shown in Table 3 and 4. For both configurations and both tasks, the models tend to overfitting, in fact after a certain number of epochs, loss over validation set is greater than the one on the training set, as we can see, for example, during the training of the model with GloVe over aggressive task,

in Figure 5. For the binary classification task on the variable Mysogenous, the configuration with fastText embedding seems to be better, while for the variable Aggressiveness performances are almost equivalent. Again, if the two tasks are executed in pipeline (the second CNN identifies aggressiveness only in those tweets marked as misogynous by the first CNN), the f1 score for non-aggressive misogynous tweets is rather low, for the reasons mentioned in the previous section.
Before executing multinomial task, training set was re-balance to have equal number of tweets in the Aggressive Mysogenous and Non Aggressive Mysogenous classes.

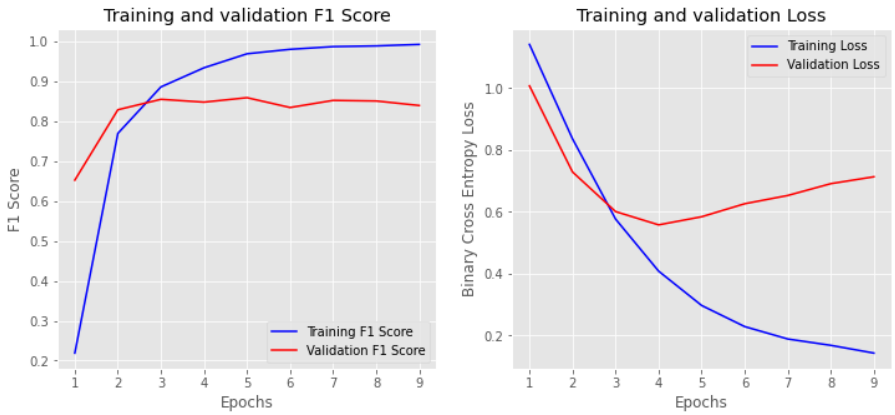


Figure 5: Loss and F1 score during Training of Aggressive Task - GloVe

	Binary Mysogenous Variable			Binary Aggressiveness Variable - Testing Strategy 1			Binary Aggressiveness Variable - Testing Strategy 2		
	0	1	macro avg	0	1	macro avg	0	1	macro avg
GloVe	0.73	0.70	**0.70**	0.78	0.71	**0.67**	0.27	0.72	**0.56**
fastText	0.81	0.77	**0.79**	0.80	0.61	**0.71**	0.30	0.76	**0.53**

Table 3: CNNs F1 Score on Internal Test Set - Binary Classification

	Multinomial Target Variable			
	0	1	2	macro avg
GloVe	0.80	0.28	0.53	**0.53**
fastText	0.79	0.26	0.53	**0.53**

Table 4: CNNs F1 Score on Internal Test Set - Multinomial Classification

Basic Model

Since all models have over fitting, it was also decided to build and test CNNs with a less complex architecture, with just one convolution layer instead of two (Architecture in Figure 7b in the Appendix), in the hope to have better performances. In this case, no pre-trained word embedding is used, but instead it was learned while fitting the model. Results are shown in Table 5 and 6. For the binary classification of Mysogenous Variable the performance seems better, with losses over training and validation sets decreasing at the same pace as the number of epochs increase, and obtaining f1 scores, for both classes, greater than those obtained from the more complex CNNs. However, performance does not improve much neither for the binary task on the variable Aggressiveness nor for the multinomial task (whose CNN is trained on the re-balanced training set as done previously).

	Binary Mysogenous Variable			Binary Aggressiveness Variable - Testing Strategy 1			Binary Aggressiveness Variable - Testing Strategy 2		
	0	1	macro avg	0	1	macro avg	0	1	macro avg
Basic CNN	0.82	0.79	**0.80**	0.78	0.50	**0.64**	0.41	0.58	**0.50**

Table 5: Basic CNN F1 Score on Internal Test Set - Binary Classification

	Multinomial Target Variable			
	0	1	2	macro avg
Basic CNN	0.80	0.28	0.69	**0.59**

Table 6: Basic CNN F1 Score on Internal Test Set - Multinomial Classification

3.3 LSTM

The problem was also addressed using a Recurrent Neural Network, that is LSTM (Long Short-Term Memory). Along the same lines as the Basic CNNs, a simpler model was preferred in order to avoid overfitting. The structure of the model consists of an embedding layer (the pretrained GloVe), a single LSTM layer and an output layer with Sigmoid as activation function. The binary crossentropy and sparse categorical crossentropy loss functions have been used for the binary and multinomial task respectively. As done for the CNNs, it was used Adam Optimizer and also the early stopping technique to control overfitting.

Hyperparameters Tuning
The hyperparameters of the network were fine tuning using again the Hyperband function. In particular: number of units and fraction of the units to drop for the linear transformation of the recurrent state (recurrent dropout) in the LSTM layer, L2 regularization factor in the Dense Layer, and the learning rate for the optimizer.

Results
We can state that despite using a simpler architecture than the one used for CNNs, overfitting is not prevented. In general, the results are worse than those obtained by the CNNs. However, we can see that by testing in pipeline, the same score is obtained for the binary task of the variable Aggressiveness as the one obtained by testing without the pipeline (testing strategy 1).

	Binary Mysogenous Variable			Binary Aggressiveness Variable - Testing Strategy 1			Binary Aggressiveness Variable - Testing Strategy 2		
	0	1	macro avg	0	1	macro avg	0	1	macro avg
GloVe	0.64	0.60	**0.62**	0.71	0.43	**0.57**	0.59	0.56	**0.57**

Table 7: LSTM F1 Score on Internal Test Set - Binary Classification

	Multinomial Target Variable			
	0	1	2	macro avg
GloVe	0.74	0.14	0.31	**0.40**

Table 8: LSTM F1 Score on Internal Test Set - Multinomial Classification

3.4 AlBERTo - Italian BERT model

Finally, it was also decided to use the BERT model which is considered one of the best models for many NLP tasks. It is a bi-directional model and this allows context-dependent learning. The model used is AlBERTo, a pretrained Italian BERT model focused on the language used on Twitter. The model was only used for solving the multinomial task.

Before proceeding with the application of the model, tweets had to be transformed so that they could be processed correctly. BERT uses special tokens [CLS] and [SEP] to understand input properly, and so [CLS] token was inserted at the beginning of the first sentence and a [SEP] token was inserted at the end of each sentence. Each sentence is then padded to the maximum length, which is set to 35 (we noted earlier that there are very few tweets that exceed that length), this means to add special padding tokens to ensure shorter sequences will have that length. Similarly, longer sequences were truncated. Sentences were then tokenized, and we got an input ids for their numeric representations in italian AlBERTo vocabulary. Finally, attention masks are generated, which tells us which input ids correspond to padding.

It was, thus, instanced a BERT model with encoder weights copied from the AlBERTo-Base-Italian-Twitter-lower-cased-Pytorch model and a randomly initialized sequence classification head on top of the encoder with an output size of three. In order to tune the model, it is used the AdamW optimizer, a stochastic optimization method that modifies

the typical implementation of weight decay in Adam optimizer, by decoupling weight decay from the gradient update. Different configuration of batch size and epochs were tried during evaluation of the model, monitoring f1 score and loss values. Finally, the batch size was set at 32 and the number of epochs at 3.

Results
The results obtained, shown in the Table 9, were slightly better than those obtained with CNNs, and equal to those reported by the LR. Again, the main problem remains the classification of Non Aggressive Mysogenous tweets.

	Multinomial Target Variable			
	0	1	2	macro avg
AlBERTo	0.90	0.41	0.75	**0.69**

Table 9: AlBERTo F1 Score on Internal Test Set - Multinomial Classification

3.5 Final results on External Test Set

Finally, two among the best models were chosen to be evaluated on the external test set proposed by the challenge. In particular, Logistic Regression was evaluated for the binary task and AlBERTo for the multinomial task.
Logistic Regressor was then trained on the entire training data using the best hyper-parameters found previously. AlBERTo was also re-trained on the entire training set, retaining a portion (10%) as validation set. Also, note that the testing phase for LR was carried out in pipeline as requested in the challenge. Results are reported in Table 10 and 11.

	macro f1 score
AlBERTo	**0.49**

Table 10: AlBERTo F1 Score on External Test Set - Multinomial Classification

	macro f1 score
LogisticRegression	**0.56**

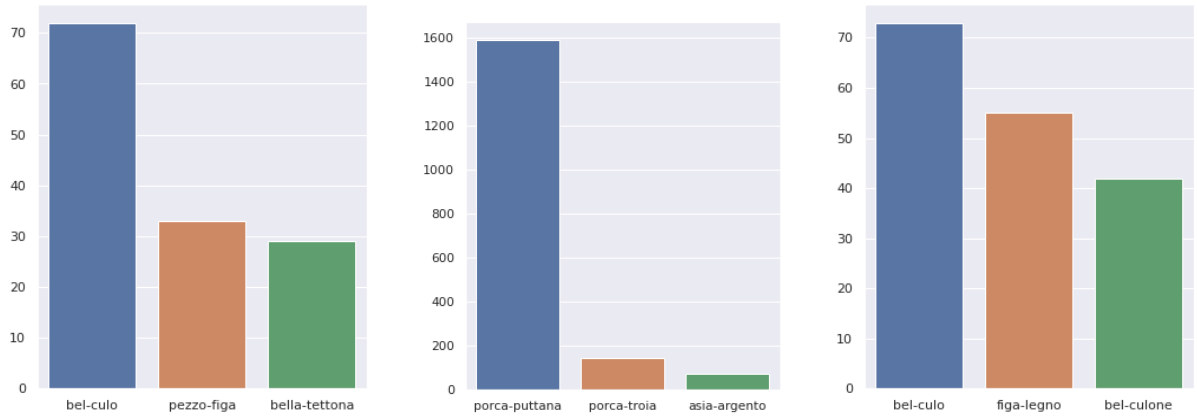
Table 11: Logistic Regression F1 Score on External Test Set - Binary Classification

References

[1] Rosso, Paolo; Nozza, Debora; Fersini, Elisabetta (2020, December 17). AMI 2020 Dataset. Version 1.0.0 (automatically assigned). European Language Grid. [Dataset (Text corpus)].

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

4 Appendix

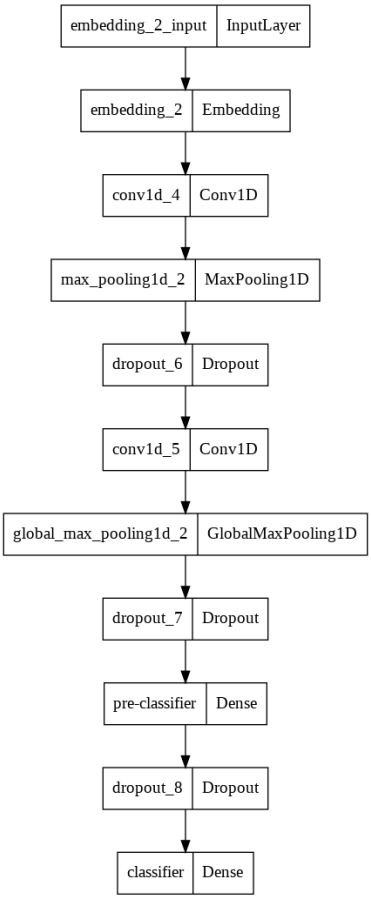


(a) Non Aggressive Misogynous (b) Non Misogynous (c) Aggressive Misogynous

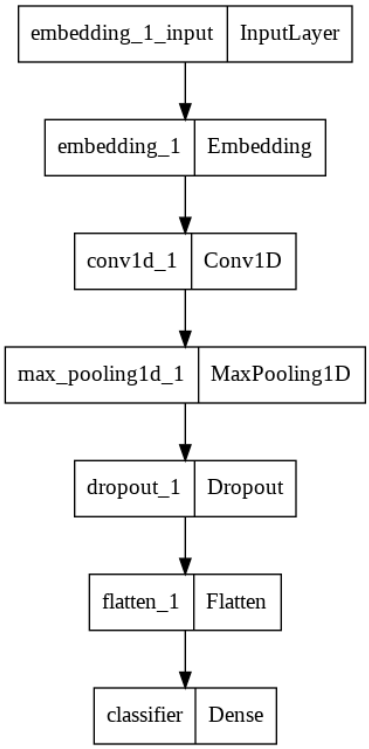
Figure 6: Bigrams Frequency.

Model	Hyperparameters	Values
SelectKBest	selbestk__k	400, 450, 500, 550, 600, 650, 700
TfidfTransformer	norm	l1, l2
	use_idf	True, False
MultinomialNB	alpha	1.0,1e-1,1e-2, 1e-3
LogisticRegression	C	np.logspace(-4, 4, 50)
	penality	l1, l2, ElasticNet
LinearSVC	C	np.logspace(-3, 2, 40)
	penality	l1, l2
	loss	hinge
DecisionTreeClassifier	criterion	Gini, Entropy
	max_depth	None, 2, 5, 10, 15, 20, 25, 30, 35
	min_samples_split	2, 5, 10, 15, 20, 25, 30, 35
	min_samples_leaf	1, 5, 10, 15, 20, 25, 30, 35
RandomForestClassifier	criterion	Gini, Entropy
	max_depth	None, 2, 5, 10, 15, 20, 25
	min_samples_split	2, 5, 10, 15, 20, 25
	min_samples_leaf	1, 5, 10, 15, 20, 25
	n_estimators	5, 10, 15, 20, 25

Table 12: GridSearchCV - Hyperparameters Tuning



(a) CNNs Architecture



(b) Basic CNNs Architecture

Figure 7: Convolutional Neural Networks Architectures

	Hyperparameters	Values
First Conv1D Layer	filters	range(16, 128, step = 16)
	kernel_size	range(3, 11, step = 2)
Layer weight regularizers	kernel_regularizer	range(1e-4, 1e-1)
Dropout Layer	rate	range(0, 0.5, step=0.1)
Second Conv1D Layer	filters	range(32, 144, step = 32)
	kernel_size	range(3, 11, step = 2)
Dense Layer	units	range(64, 128, step=32)
Adam Optimazer	learning rate	[1e-1, 1e-2, 1e-3]

Table 13: Hyperband Tuner - CNN Hyperparameters Tuning