Hate Speech Detection EvalIta 2020

Francesca Boccardi, Luigi Podda
Master's Degree in Artificial Intelligence, University of Bologna
{francesca.boccardi, luigi.podda}@studio.unibo.it

Abstract

Identifying hate content in social media is a crucial mission, as online hate speech can be potentially dangerous and harm marginalized individuals or groups. In this perspective, the aim of this study is to compare different methods and architectures, namely ML, RNN and Transformer-based models, to address the tasks of Hate Speech Detection and Stereotype Detection. The experiments were run on a corpus of Italian tweets given by EvalIta for the 2020 HaSpeeDe challenge. Models, trained on tweets, were then tested on both in-domain and out-of-domain new data, i.e. tweets and news headlines respectively. Being pre-processing of input a crucial point in tweets-related tasks, as for the presence of paralinguistic symbols and poorly written text, its impact on the results is also analyzed by feeding ML and RNN models with differently pre-processed input. The results show that, according to macro-F1 measure, the performances of the models are all positive and comparable. However the best outcome is given by UmBERTo, the Transformer-based architecture specifically pre-trained on the Italian corpus of Common Crawl.

1 Introduction

The purpose of this study is to address the dangerous and unpleasant phenomenon of online hateful content by solving two different tasks, closely related to it: Hate Speech Detection and Stereotype Detection on social media and newspaper sites, towards a specific type of target.

One of the most widely used social networking sites, Twitter, has transformed how people connect and exchange information, opinions, and ideas. However, the open nature of Twitter, and more generally of social media, has led to an increase in the use of it for the propagation of violent and racist information. Hate speech has no official definition, but there is a general idea on its meaning. It is defined as any language that disparages an individual or a group because of a characteristic like race, color, ethnicity, gender, or religion. Its subtler shade that hiddens behind the stereotype, defined as a "widely held but fixed and oversimplified image or idea of a particular type of person or thing" (1), can nonetheless have a negative impact on the people concerned.

In the modern era, social media are more and more important from an interaction point of view, however more and more users abuse of hate and bad speech in their posts and comments. This is a huge problem also considering the increasing number of cyberbullying events. Detect and report this type of language can help to control and make social media and chats a safer place.

In this perspective, EvalIta 2020 proposed the HaSpeeDe competition (2), focusing its attention on Hate Speech (HS) and Stereotype (ST) Detection on online Italian texts, as to build models capable of efficiently detect such offensive/stereotyped language.

A popular way to address these tasks is the combination of feature extraction methods, such as the Bag-of-Words approach, and classical machine learning algorithms, like Naïve Bayes classifier, Support Vector Machines or Logistic Regression.

After the introduction of embeddings, which proved useful combination with machine learning algorithms, other Deep Learning methods (12) have been adopted for the hate speech detection, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). An important step was then the introduction of Transformers (11), particularly BERT, which proved to be able to highly outperform other methods.

In this study, both HS and ST detection tasks are approached using different combinations of word features extraction methods and architectures (10): two machine learning (ML) algorithms, i.e. a Support Vector Classifier and a Logistic Regressor (LR), with TF-IDF as words representation; two Deep Learning models, based on RNN ar-

chitectures with Bi-LSTM layers, fine-tuning a pre-trained FastText embedding for words representation; two Transformer-based models, UmBERTo and DistilBERT-base multilingual, the first of which pre-trained specifically on Italian corpus, while the second on several languages, including Italian.
Given the particular structure of online texts like tweets, which often contain poorly written text and paralinguistic symbols such as hashtags and emoticons, pre-processing of the input is something that must be designed with care. Therefore, each ML and RNN model was trained on differently pre-processed input, in order to make an analysis on which procedure might be the most suitable for the task. For transformers, on the other hand, it was decided not to apply any kind of pre-processing and, in order to obtain more robust results, experiments were carried out using two different seeds.
The final outcomes show that the Transformer-based approach generally achieves better results, as transformers can process sentences as a whole rather than word by word and the self-attention mechanisms can provide information about the relationship between different words, helping predictions. In particular, UmBERTo proved to be better than DistilBERT, probably due to its specific pre-training only on Italian corpus.

2 System description

To address both the HS and ST detection tasks, different combinations of words representation and architectures have been implemented:

TF-IDF + ML model: two standard and straightforward approaches have been chosen for the ML variants: a Support Vector Classifier and a Logistic Regressor. Before feeding the models, the text is first converted into numbers by using the previously computed TF-IDF value of each word. TF-IDF gives more weight to words that have an high number of occurrences within the same tweet but a low number of occurrences between different tweets.

FastText Embedding + RNN model: before feeding RNN models, words are represented through a FastText embedding for Italian language, which associates to each word a vector of 300 floating numbers. For computational reasons, the FastText model used is a compressed version of the original one (3). Starting from the FastText vocabulary, a costumed one is created by adding to it an embedding for each Out-Of-Vocabulary (OOV) term of the corpus used. This procedure is handled directly by the Compress-fastText package (4), used to load the pre-trained FastText model, which obtains the embeddings for OOV words by summing up vectors for its component char-ngrams if at least one of the char-ngrams was present in the train data, creating an embedding vector of 0s otherwise. Then, each vocabulary word is converted into an index according to its position in the vocabulary, such that each text becomes a sequence of indexes. Since not all texts have the same length, some of them are padded and some other truncated according to the maximum length computed as to handle the 99% of the train tweets.
Then, two different RNN architectures are implemented. They both contain an embedding layer, initialized with FastText embedding weights previously mentioned. Due to the large number of OOVs, it has been decided to leave the layer trainable, as to let the network to fine-tune the embedding weights. Also, the layer allows the network to ignore padding.
Both two models make use of Bidirectional Long Short-Term Memory (Bi-LSTM) layers with final Dense layer, differing however on how many of those layers they employ.
In the first model, the initial Embedding layer is followed by a Dropout layer with 0.5 as dropout rate, one Bi-LSTM layer of size 64, a 1D Global Max Pooling and two Dense layers, of size 64 and 1 respectively, with another 0.5 Dropout layer between the two.
The second one instead consists of an initial Embedding layer, followed by a 0.5 Dropout layer, plus two Bi-LSTM layers both with size 16. A 1D Global Max Pooling layer preceeds the last Dense layer of size 1.
Both employ Sigmoid as final activation function.

Transformer-based models: the two pre-trained HuggingFace (6) transformers employed are UmBERTo (13), a Roberta-based Language Model trained on the cased version of the Italian subcorpus of OSCAR (5), and DistilBERT-multilingual (14), which is a distilled version of the BERT base multilingual and is trained on the concatenation of Wikipedia in 104 different

Text	HS	Stereotype
La mia storia dal campo rom alla Sorbona. Bisogna farsi valere contro il razzismo URL	0	0
#Roma #Torino, grazie al collegamento con i campi #rom di #QuintaColonna, su #mediaset #tv, stasera calo esponenziale dei #furti. #DelDebbio	0	1
@user te li immagini i tagliagole islamici difensori dei maiali, gli inglesi hanno vietato il cartone di peppa pig è tutto Dire	1	0
#londonattack chiedete ai buonisti del cavolo cosa vogliono fare con i islamici perché io se chiedo mi vengono solo insulti sulla bocca	1	1

Table 1: Dataset tweets examples

languages. It is important to notice that while the first model is specific for Italian language the second one is a multilingual model, meaning that it shares its available resources across different languages and is not focused on a specific one. Both models are fine-tuned on the EvalIta dataset, once for each task.

3 Data

The dataset on which experiments were run is composed by texts targeting Italian minorities, such as Immigrants, Muslims and Roma communities. It has been built for the EvalIta 2020 challenge by collecting tweets and newspaper titles containing specific keywords related to the domain of interest. The training set is composed by 6839 tweets, while the test set contains both in-domain and out-of-domain texts, namely 1263 tweets and 499 news. For each message, the *HS* boolean variable specifies the presence or the absence of hate speech against those Italian minorities mentioned before, while the *stereotype* one indicates whether the text expresses stereotypes or not, always towards those targets. An example of dataset tweets and labels is shown in Table 1

The training set shows a slightly unequal distribution of classes, both for HS and ST, as shown in Table 2 and Table 3.

HS	Not HS	Total
2766	4073	6839

Table 2: Hate Speech train set distribution

Stereotype	Not Stereotype	Total
3042	3797	6839

Table 3: Stereotype train set distribution

After splitting the provided training set into 80% for train and 20% for validation, data are tokenized. As for Transformers, data are transformed in model inputs using proper BERT-based tokenizers, which truncate or pad tweets according to the maximum length computed as to handle the 99% of them.

The tokenization of input feeding the ML models is instead managed by the TfIdfVectorizer, which converts all characters to lowercase before tokenizing.

For what concerns RNN models, tweets are tokenized by the TweetTokenizer of the NLTK Python package (7), which replaces repeated character sequences of length ≥ 3 with sequences of length 3, removes Twitter handles and downcases everything.

While transformer models showed better performance with simple tokenization of the original texts, i.e. which were not subjected to any previous cleaning, the ML and RNN models required that the input be first pre-processed and then tokenized in order to achieve reasonable results.

Thus, to better figure out how to deal with available data, it turned out to be useful to deeply analyze the content of texts. In particular, both training and test tweets show to contain an high number of *#hashtags*, emoticons, *@usernames* and URLs, while the test news are almost devoid of them.

Differently from *@usernames* and URLs which do not carry relevant information, *#hashtags* could usefully help categorize a message as containing hate speech, stereotyping or otherwise, as their purpose in social network is precisely to identify the content of the message as belonging to a specific category and thus carrying a certain nuance of meaning. Thus, it has been decided to only remove *@usernames* and URLs, while furtherly process *#hashtags*, as to make them more understandable, by removing the # character and, as far as possible, splitting those containing multiple words attached.

Moreover, emoticons in text might express the emo-

	Basic pre-processing			Spell-checking		
	Val	Test Tweets	Test News	Val	Test Tweets	Test News
SVC	0.7557	0.7173	0.6631	0.7517	**0.7181**	0.6647
LR	0.7462	0.7200	0.6532	0.7425	0.7182	0.6705
1 LSTM + 2 DENSE	0.7591	**0.7259**	0.6550	0.7550	0.7249	0.6655
2 LSTM + 1 DENSE	0.7586	0.7283	0.6704	0.7517	**0.7308**	0.6820

Table 4: F1-scores of ML and RNN models on Hate speech texts, using basic pre-processing and spell-checking.

	Lemmatization			Spell-checking & Lemmatization		
	Val	Test Tweets	Test News	Val	Test Tweets	Test News
SVC	0.7493	0.7133	0.6845	0.7517	0.7142	**0.6855**
LR	0.7506	0.7237	0.6739	0.7540	**0.7238**	**0.6739**
1 LSTM + 2 DENSE	0.7425	0.7219	0.6723	0.7440	0.7136	**0.6819**
2 LSTM + 1 DENSE	0.7533	0.7243	**0.7000**	0.7487	0.7189	0.6994

Table 5: F1-scores of ML and RNN models on Hate speech texts, using lemmatization and spell-checking combined with lemmatization.

	Seed: 42			Seed: 2023		
	Val	Test Tweets	Test News	Val	Test Tweets	Test News
UmBERTo	0.7828	**0.7941**	0.6393	0.7839	0.7941	0.6410
DistilBERT Multilingual	0.7583	0.7478	**0.6651**	0.7500	0.7431	0.6497

Table 6: F1-scores of Transformers on Hate speech texts, using two different seeds.

tional engagement of the user who wrote it, maybe supplementing the meaning that other simple words already express. However, it has been decided to remove them anyway, as the user feelings are not really what matter for the task at hand and they might bring conflicting information.

Then, punctuation and Italian stopwords of NLTK have been removed too, while being careful, however, not to remove the stopword *"non"*, the absence/presence of which can greatly influence the meaning of a message.

Additionally to the basic pre-processing just described, other stages, namely spelling correction (9) and lemmatization (8), are applied. Indeed, spell-checker might be an effective tool for correcting possible bad written words, which are very common in social media contents. Lemmatization instead, by reducing words to their base form, may also help in lowering the number of OOVs terms for the FastText embedding.

4 Experimental setup and results

In order to investigate the contribution of spelling correction and lemmatization separately, all ML and RNN models are fed with differently pre-processed input: in addition to experiments on input processed with basic pre-processing, tests were carried out by adding to it the spelling correction alone, the lemmatization alone, or both spelling correction and lemmatization at the same time.

Experimental results showed that the best configuration for each type of model is as reported below. For the Logistic Regressor, the maximum number of iterations taken for the solvers to converge was set to 1000.

Both RNN models were trained for a maximum of 100 epochs and with a batch size of 32, sharing the same following setup: the early stopping technique with patience equal to 3; Adam as optimizer with a learning rate of 0.0001; the accuracy as metric during the training process and the binary crossentropy loss. Final evaluation was made on the test set using the macro F1-score.

All the obtained results for the ML and RNN models are summarized in Table 4 and 5 for the HS detection task, in Table 7 and 8 for the ST detection task.

Regarding transformers, the best training setup was found to employ a batch size of 64 and AdamW as optimizer, with a learning rate of $2e^{-5}$ and a weight

	Basic pre-processing			Spell-checking		
	Val	Test Tweets	Test News	Val	Test Tweets	Test News
SVC	0.6873	0.6873	**0.6789**	0.6900	0.6862	0.6748
LR	0.6882	0.7048	**0.6804**	0.6904	0.6984	0.6723
1 LSTM + 2 DENSE	0.6949	0.7014	0.6392	0.6924	**0.7015**	0.6100
2 LSTM + 1 DENSE	0.6951	0.6957	**0.6748**	0.6914	0.6968	0.6616

Table 7: F1-scores of ML and RNN models on Stereotype texts, using basic pre-processing and spell-checking.

	Lemmatization			Spell-checking & Lemmatization		
	Val	Test Tweets	Test News	Val	Test Tweets	Test News
SVC	0.6847	**0.6981**	0.6689	0.6915	0.6927	0.6765
LR	0.6949	**0.7165**	0.6656	0.6964	0.7126	0.6659
1 LSTM + 2 DENSE	0.7027	0.6917	**0.6715**	0.7071	0.6967	0.6651
2 LSTM + 1 DENSE	0.6854	**0.7113**	0.6494	0.6856	0.7052	0.6590

Table 8: F1-scores of ML and RNN models on Stereotype texts, using lemmatization and spell-checking combined with lemmatization.F1-scores of ML and RNN models on Hate speech texts, using lemmatization and spell-checking combined with lemmatization.

	Seed: 42			Seed: 2023		
	Val	Test Tweets	Test News	Val	Test Tweets	Test News
UmBERTo	0.7538	**0.7524**	0.6863	0.7554	0.7428	**0.6913**
DistilBERT Multilingual	0.6955	0.6943	0.5685	0.6937	0.6713	0.5752

Table 9: F1-scores of Transformers on Stereotype texts, using two different seeds.

decay of 0.01. Both UmBERTo and DistilBERT-base models share the same training setup. Each of them is fine-tuned for 3 epochs with 2 different seeds, namely 42, 2023 and finally evaluated on the test set using the macro F1-score metric.
Their results on test tweets and news are summarized in Table 6 for the HS detection task and in Table 9 for the ST detection task. For each model, their best performances on test tweets and on test news are highlighted in bold.

5 Discussion

5.1 Results

Task A: Tables 4, 5 and 6 show models performances on the HS detection task, according to the F1-score metric. In order to investigate the contribution of spelling correction and lemmatization, for each result of ML and RNN models, the type of pre-processing is also reported.
SVC and LR models roughly obtain comparable results, both performing quite constantly across differently pre-processed test tweets. As expected, results on test news are lower than those on tweets, being the first out-of-domain data. However, results show that all models benefit a bit from spell-checking and lemmatization when trying to classify test news.
Similarly, RNN models performances are not so far one from each other and results on test tweets are quite constant across different type of pre-processing, worsening a bit with the combination of spell-checking and lemmatization. Again, news are worse classified than tweets and out-of-domain data lemmatization seems to contribute positively for both models.
While ML and RNN results are fully comparable, being around 0.72 for tweets and 0.67 for news, transformer models, even without any kind of pre-processing, outperform the others on test tweets, but generally perform worse on test news. UmBERTo proves to be better than DistilBERT on tweets with an F1-score of 0.79, but a bit worse on news, where DistilBERT reaches 0.66. Results are consistent across different seeds, as shown in Table 6.
In general, the performance of all models on validation and test tweets, given the same pre-processing, shows no significant differences.

Text	HS ground truth	Best models prediction
I SOLDI DEI DISABILI AI MIGRANTI La regione Emilia Romagna destina la metà dei fondi destinati alle fasce deboli a progetti per i richiedenti asilo A Reggio Emilia il 69% delle risorse stanziate sono state utilizzate ai richiedenti asilo #PRIMAGLIITALIANI URL	1	0
@user @user L'esatto contrario di quello che sistematicamente fanno da anni i sindaci di sinistra, per loro prima gli immigrati. Mi raccomando votate ancora questi imbecilli di sinistra, come casa vi ritroverete la vostra bella macchina.	0	1

Table 10: Misclassified Hate Speech examples for the three best models.

Text	ST ground truth	Best models prediction
Non esiste l' invasione dei migranti. In Italia ci sono 500 o 600 mila clandestini se fossero regolarizzati ci renderemmo conto che aiutano il paese, pagano i contribuiti e fanno bene all'economia. @user #cartabianca URL	0	1
Basilicata, Salvini sbeffeggia oppositore: "Avrai la gioia di mantenere a vita dieci bei migranti"	1	0

Table 11: Misclassified Stereotype examples for the three best models.

Task B: Table 7, 8 and 9 show instead models F1-score results on the ST detection task, which are generally a bit worse than those of the HS detection one.
Even though ML and RNN models performances are again perfectly comparable and constant across different type of pre-processing, most of them seems to benefit from lemmatization on test tweets and to prefer a simple basic pre-processing on news. Only the LSTM1_Dense2 model takes advantages of the spelling correction on test tweets and of lemmatization on test news, although with minimal improvement gap. Also, it's important to underline that the gap between the results on test news and test tweet decreased compared with the HS detection task.
Regarding transformers, UmBERTo gets the best results on both test tweets and test news, reaching 0.75 and 0.69, respectively. In contrast, DistilBERT shows very poor results, comparable to the 0.71 of ML and RNN models on test tweets, but much lower on test news, with 0.57 versus 0.68. Again, the results prove robustness across several seeds.
For all models, results on validation and test tweets, given the same pre-processing, are very similar.
For both tasks, models struggle when trying to classify out-of-domain data. Intuitively, it happens because the type of language used by Twitter users, on which models were trained, can be really different from the one used in news headlines, which contain specific structures and style. In addition, the meaning of a tweet is often made more evident by explicit expressions, such as insults or bad words in case of hateful ones, while news headlines may have more subtle and nuanced meanings, where it may be more difficult to detect the presence of hate speech or stereotype.
Finally, regardless the task, UmBERTo generally shows the best results, outperforming even DistilBERT. This is likely due to its specific pre-training only on the Italian language, which allowed it employ all available resources to acquire knowledge of the Italian language without sharing them across different languages.

5.2 Error Analysis

For both tasks, to better reason on errors causes, the combinations of model and pre-processing that yield the best results on test tweets are chosen, one per model type. Then, the analysis focuses on samples which are misclassified by all selected

models, in order to investigate what the source of error involved might be.

Task A: The best models were found to be UmBERTo, LR with spelling correction and lemmatization and LSTM2_Dense1 with spelling correction.
For each model, the per-class F1-scores on test tweets are consistent with the macro ones, while it is not the case for test news. Indeed, the per-class F1-scores of all models on test news is higher on the class 0 than on the class 1, which they struggle a lot to classify correctly. This could be influenced by the slight skewing of the training data towards the class 0, but it's more likely due to the difficulty of identifying hate speech within a different type of language than that used for training.
Looking deeper into misclassified samples from all models, some interesting possible causes of error have emerged.
For example, it's really hard for models to detect hate speech in tweets where the hate message is somehow only implied and the meaning is hidden behind nuance of the language. Therefore, tweets like the one in the first row of Table 10 might often be misclassified.
Some errors could also be due to a too heavy pre-processing, which, even if careful, could sometimes change the meaning of the sentence.
Finally, it's interesting to note that the models often misclassify tweets/news containing hate speech but aimed at non-Muslims, Immigrates or Roma targets, whose ground truth is therefore 0 (e.g. tweet in the second row of Table 10). This shows that they have learned to understand the presence of hate speech but do not distinguish with respect to the target the message is are aimed at.

Task B: the best models were found to be UmBERTo, LR and LSTM2_Dense1 with lemmatization.
Again, each model shows consistency between the per-class F1-scores on test tweets and the macro ones, and their per-class F1-scores on test news is always higher on the class 0 than on the class 1. Likewise the hate speech detection task, this may be due to the slightly imbalanced nature of the training data towards the 0 class, but the difference in the way the stereotype is expressed between the training tweets and the test news is what affects it most.
Some peculiar misclassified samples are those probably due to the presence of sarcastic or ironic messages in it. In fact, it is difficult for models to detect stereotype in texts where stereotype is expressed in an implicit form, hidden behind sarcasm (e.g. news in the second row of Table 11).
Texts that contain words that are very frequent in one class but rare in the other may also challenge the classification. Indeed, these words during training can create a misclassification bias toward the class in which the word is more present, letting the model to learn a strong (but sometimes incorrect) relationship between those words and the class. For example, the word *"clandestini"* is one of the 10 most frequent words among all the training tweets marked as containing stereotype but it's not frequent among the non-stereotyped ones. Thus, when it appears in a text, models might misclassified it as stereotype, regardless of everything else (e.g. tweet in the first row of Table 11).

6 Conclusion

The purpose of this work was to address the Hate Speech detection and the Stereotype detection tasks, towards a given people target, on the HaSpeeDe dataset of EvalIta, by experimenting several combinations of word representation methods and architectures. To get a broader and more complete view, it was decided to employ all of the most popular approaches generally used to solve these types of tasks: classic Machine Learning models, such as SVC and LR; Deep Learning models LSTM-based; Transformers, like UmBERTo and DistilBERT-multilingual. While for feeding ML models, words have been encoded with Tf-Idf features, for RNN models was made use of a pre-trained FastText embedding for Italian language.
Given the complex and unpredictable structure of training texts, which is composed by tweets, a special attention was paid to pre-processing steps. Tweets are in fact by nature much more prone to contain misspelled words, and the presence of paralinguistic symbols, emoticons, urls and links can really challenge the classification.
While transformers showed to prefer unprocessed input, ML and RNN models were trained on differently pre-processed texts, as to investigate which procedure might be more effective. Trained models were then evaluated on the test set, composed by both in-domain and out-of-domain data, i.e. tweets

and news.

From obtained results it turned out that UmBERTo model is the one with the best overall performance on both tasks, reaching on test tweets 0.79 F1-score for HS detection and 0.75 F1-score for ST. It also achieves the best results on test news for stereotype, with 0.69 F1-score, while the highest performance on test news for HS was reached by the LSTM2_Dense1 model with lemmatization, with 0.70 F1-score.

In general, given the same task and the same test set type, all ML and RNN models obtained comparable results. Each model also performed similarly across differently pre-processed inputs, with some minor variations, for which it is worth stating that accurate pre-processing can help with classification.

As expected, for both tasks, models struggled more on classifying out-of-domain data like news, as the type of language they saw at training time with tweets samples is different from the one used in news headlines. Also, the way of expressing hate or stereotype in tweets can be far from that used in news, where it might be subtler and less explicit, then more difficult to identify.

Finally, by deeply inspecting the predictions of only the best models, it could be seen that all of them tend to misclassify tweets that are ambiguous or ironic, which might be hard to classify even for human being. Also, models showed to be generally able to recognize hate speech or stereotype but without making too much difference in the type of target audience the message is aimed at.

To conclude, Hate Speech and Stereotype detection tasks on Italian texts is certainly an interesting and meaningful challenge. However, it often happened to face the difficulty of finding material, tools, libraries, and pre-trained models that could actually be compatible and effective when dealing with the Italian language.

Future works might focus on using different techniques to refine the models as to make them capable of recognizing and differentiating the targets to which the hateful message is addressed. Also, it could be interesting to guide the models to gain more general knowledge about the concept of hate speech, less specific about tweets alone, so that they can make use of what they have learned to do hate detection in broader domains and thus be a more useful and effective tool to fight against hate messages on social media.

7 Links to external resources

The EvalIta 2020 dataset is available for download at the following link. To extract the data from the compressed folder the needed password is *zNw3tCszKWcpDahq*.

References

[1] Stereotype definition from Oxford Languages: (https://languages.oup.com/google-dictionary-en/)

[2] Hate Speech Detection EvalIta 2020: (http://www.di.unito.it/~tutreeb/haspeede-evalita20/index.html)

[3] Compressed fastText Embeddings: (https://zenodo.org/record/4905385#.Y9vPHHbMK3A)

[4] compress-fastText: (https://github.com/avidale/compress-fasttext)

[5] Open Super-large Crawled Aggregated coRpus (OSCAR): (https://oscar-project.org/)

[6] Huggingface: (https://huggingface.co/)

[7] Natural Language Toolkit (NLTK): (https://www.nltk.org/index.html)

[8] Pattern python module: (https://github.com/clips/pattern)

[9] Autocorrect python module: (https://github.com/clips/pattern)

[10] Challenges of Hate Speech Detection in Social Media: (https://link.springer.com/article/10.1007/s42979-021-00457-3#Sec14)

[11] Hate Speech Detection in Twitter using Transformer Methods: (https://www.researchgate.net/publication/346077460_Hate_Speech_Detection_in_Twitter_using_Transformer_Methods)

[12] Classification of Hate Speech Using Deep Neural Networks: (https://hal.science/hal-03101938/document)

[13] UmBERTo Commoncrawl Cased: (https://huggingface.co/Musixmatch/umberto-commoncrawl-cased-v1)

[14] distilBERT base multilingual cased: (https://huggingface.co/distilbert-base-multilingual-cased)