Toxicity Detection: Does the Target Really Matter?

Stanford CS224N Custom Project - Winter 2020-2021

Roland DUFFAU
Department of Computer Science - Stanford University
rduffau@stanford.edu
CS224N staff mentor: Rachel GARDNER

Abstract

With the continuous rise of online social media platforms, efficient moderation of toxic content has became an indispensable solution to keep the Internet safe. But this classification task remains challenging, with false positives driving to excessive moderation and public dissatisfaction. In this project, we aim to study if the toxicity level of a written comment can be better assessed by also considering its **target** (e.g. towards whom it is directed). Working with a professional French dataset and latest NLP deep learning models, we managed to confirm this hypothesis. We also showed that the target itself can be accurately predicted from the comments. Many things remain to discover and test in this sensitive and complex challenge, but our results already confirm that better professional moderation is possible thanks to deep learning-based tools. Our code is available here.

1 Approach

2 Introduction

Detecting and moderating toxic written content is a challenging task, given the variety of language forms used in social media typically (slang, mispellings, emojis, etc.).
Over the past few years, some researchers have investigated the automated detection of specific types of toxic content (such as sarcasm [1], racism [2], aggression and misogyny [3]). Meanwhile, others have focused on ways to improve the level of detection whatever the form of abusive language: for instance in [4], leveraging the *context* of the post on top of its own content.

Our project analyzes a derivative of this last approach: since the **target** of a toxic content can influence its meaning and intention, can it also influence its level of toxicity? As an example, "go kill yourself" is much more toxic than "she told him: go kill yourself". We had thus **two main objectives** in this project: first, check if *using the target on top of the comment improves the prediction of toxicity level*. Second, *accurately identify the true target of the comment*. Since we couldn't find previous research tackling specifically this second problem, we investigated several approaches detailed below.

To date, research on forms of abusive language detection is mainly focused on English. Few datasets exist in other languages (such as Greek, Arabic or German), but none seem to be yet available in **French**. So we partnered with the company **Bodyguard** [5], which develops an expert moderation solution based on static rules. They provided us a labelled dataset of real-word samples from French social media accounts, to perform our analysis and benchmark deep learning models for our specific objectives.

3 Related Work

Toxic content is an umbrella term for many sub-levels of severity. In [6], up to 46 different variations are defined and used: hateful, offensive, fearful, abusive, etc. Using computational resources to detect

and handle such content is a challenge which has gained lots of interest in the recent years. This is both due to the continuous rise of social media, and to the emergence of new algorithms.

Initially, the first classifiers for toxic content used approaches such as dictionaries look up [7] or bag of words [8]. If they had the benefit of being easy to interpret, these solutions also generated high false positive rates, and suffered from data sparsity issues. Some progress later resulted from the incorporation of additional features such as N-gram graphs [9] or Part of Speech [10], allowing more subtle analysis of semantic content typically by Support Vector Machines (SVMs).

As more data became publicly available, specialized datasets were released to help detect toxic content. They enabled researchers to develop more complex approaches, typically leveraging deep learning models or graph embedding techniques [11]. They also benefited from advanced word embedding techniques such as Word2Vec [12] and GloVe [13], which produce continuous and dense representations of the content.

The first deep learning models leading to a significant rise in the prediction scores where Recurrent Neural Networks (RNNs), particularly as they made it possible to model larger sequences of text. Gated RNNs such as LSTMs [14] and GRUs [15] have been shown to be very efficient at representing long term dependencies. In 2017, Transformers [16] made a breakthrough by capturing contextualized embeddings for a sentence thanks to the attention-mechanism. Even more recently, BERT [17] was released and achieved state-of-the-art performance in text classification, question answering, and language inference without substantial task-specific modifications. Based on the transformer architecture, this model produces contextualized embeddings extendable to classification tasks with an additional output layer.

Lastly, the recent multiplication of datasets in languages other than English led to the development of multilingual classifiers. They take several forms and address various challenges, such as classification in low resource setting [18], benchmarks for zero-shot multilingual classification [19], or online prediction tools such as [20] or Jigsaw's Perspective API [21].

4 Approach

4.1 Baselines

Since Bodyguard built our dataset, including both "Target" and "Toxicity Level" labels, they represent for us a baseline of **100% precision** and **0% False Positive Rate**.
We also used **Perspective API** toxicity score on our dataset as a baseline. This score indicates how likely it is that a reader would perceive the comment provided in the request as toxic. We thought this was an acceptable proxy, splitting the score range (from 0 to 1) in 5 equal segments to reflect our 5 toxicity levels, but averaging prediction results for each segment to the closest 2 levels (to account for nuances of level definition). Given Perspective API quota restriction (1 request per second), we computed their score for 100000 samples from our dataset. It resulted in a **global precision of 62%**.

4.2 Model architectures and pipelines

As in [4], we experimented a small set of models, specialized for text analysis. We chose a **Bi-LSTM** (128 units) [22], a **Transformer** (1 block, 2 attention heads) [23] and **CamemBERT** [24]. CamemBERT was developed by Facebook, based on the RoBERTa architecture and pretrained on the French subcorpus of OSCAR [25]. We built our models from scratch with Keras API, except CamemBERT that we replicated from `this colab`, based on Hugging Face Transformers API and PyTorch.

For the first part of our primary objective, which was to *predict the Toxicity Level directly from the Comment*, we used **CamemBERT architecture for classification**. This is the normal CamemBERT model with an added single linear layer on top for classification, that we use as a sentence classifier. As we feed input data, in our case the comments in raw text, the entire pre-trained CamemBERT model and the additional untrained classification layer are fine-tuned on our specific task. For implementation, Hugging Face Transformers API provides this model under the name CamembertForSequenceClassification.

For the second part of our primary objective, which was to *also take into account the Target in our predictions*, we **concatenated both the Comment and the Target via distinct layers**, before

applying a standard classification layer to the output. As describe in Figure 1, the Comment was first passed into a regular version of CamemBERT pretrained model, named CamemBERTModel in Hugging Face Transformers API. In its output, we extracted the CLS embedding and used it as input for the concat function.
In parallel, the Target was encoded as one hot vector and passed into a Linear layer of same size as the CLS embedding, before feeding the concat function. The output of the concat function was then fed into an ultimate Linear layer, and passed to a Softmax layer. Its output was the probability distribution per Toxicity Level.

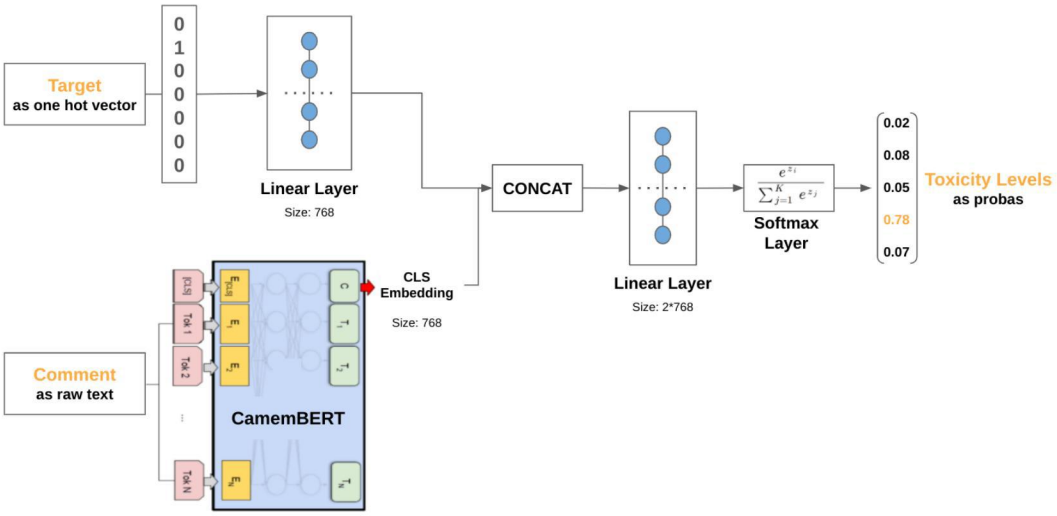


Figure 1: Model architecture to predict Toxicity level from Comment and Target.

For our second main objective, e.g. *Target identification*, we relied on two approaches: the first one was basic **deep learning classification**. We fed the CamembertForSequenceClassification model with the comments and fine-tuned it to detect the Target. Even if it was the simplest solution, it turned our to be the most effective as well.
Our second approach relied on a **custom pipeline for syntactic features identification**. After a close analysis of our dataset, we noticed that several patterns seemed to be more related to specific targets: comments with one "@" sign were 10 to 15 more likely to be addressed to "User" or "User family" than to any other target, and comments with 2 "@" signs saw this ratio rise to 25. A second important pattern was the presence and nature of certain personal pronouns (subject or object): for instance, the presence of the pronoun 'vous' (or its misspellings: 'vou', 'vos') increased the likelihood of the comment to be addressed to "Everyone" or "Group" by a factor 10 versus other Targets. Our pipeline involved 4 of these patterns.
To detect the occurrences of the pronouns, we used CamemBERT pretrained model for **Part-of-Speech (PoS) tagging**. Again, Hugging Face provides a relevant implementation named French-Camembert-Postag-Model [26], pretrained on the free-french-treebank dataset. This model detects 29 different tags, based on [27], but we regrouped them into 11 essential categories for clarity. Due to French slang and misspellings, the model had troubles to identify all relevant PoS tags. For instance, most pronouns written via abbreviations (such as "t" for "tu es", meaning "you are") were tagged as common name. We found static rules (based on substring research in the token lists) more effective to identify all relevant PoS tags, and used their outcomes as input for concatenation with the output of a CamemBERT model.

5 Experiments

5.1 Data

The Bodyguard dataset includes **190.000 examples of toxic comments** in raw text coming from French real social media accounts. They are labelled with 3 fields: *Target, Toxicity Level* and *Toxicity Type*. The dataset also contains **200.000 examples of non toxic comments** in raw text, without label.

Toxicity Level and Target were the two labels we focused on. As shown in Table 1, their distribution is imbalanced, but Bodyguard confirmed that it reflects the real world they see. We encoded them as one-hot vectors, to use them as input of our concatenated layers.

Toxicity Level	Target						
	Author of comment	User	User family	Single Person	Group	Everyone	**Total**
Very High	171	39.856	1.098	9.000	9.000	7.776	**66.901**
High	189	27.380	3.573	9.000	9.000	3.951	**53.093**
Medium	837	24.316	1.413	9.000	9.000	2.124	**46.690**
Low	1.350	5.445	378	9.000	9.000	2.736	**27.909**
Total	**2.547**	**96.997**	**6.462**	**36.000**	**36.000**	**16.587**	**194.593**

Table 1: Distribution of toxic examples per Target and Toxicity Level

As social media language is full of misspellings, slang and emoji, so were our input comments. We thus had to perform **data preparation** on the whole dataset: comments cleaning via custom functions (case lowering, removal of URL, HTML and punctuation), Emoji removal (but kept apart for further analysis). We left misspellings and slang untouched due to their large quantity and diversity, as well as kept stopwords and frequent words since our tests shown they gave better results. Once cleaned, we **tokenized** our inputs via NLTK Word Tokenizer, and padded sequences to 150 characters.

Due to the specificity of our training data (French social media content), we had limited choice for the **word embeddings** used to feed our Bi-LSTM model. We found that building our own vectors via Word2Vec (Gensim implementation) provided the best results, versus FastText (either Gensim or pretrained French vectors). We chose to create 200-dimensional vectors, which allowed a good performance trade-off.
Out-of-vocabulary words are drawbacks of word embeddings. But in our tests, even slang and misspelled words were encoded since we created our own vectors by applying the Word2Vec framework (not using their pretrained vectors). We analyzed the quality of some slang or misspelled word embeddings by looking at their most similar words. For most of them, these similar words made sense, so we assessed that words embeddings were satisfactory enough for our project.

In order to deal with the **imbalanced distribution** of our dataset, we tested several solutions. First, we tried to simply use *class weights to influence the loss function* calculation. We computed these weights via a custom function, which basically boosts the less represented class by a factor function of their size relative to the total number of comments. But it didn't provide much positive impact.
Second, we tried *data augmentation via a Google Translate loop* (FR -> EN -> FR) applied to our input comments. We tested various intermediate languages but English seemed to produce the best balance between wording variation and global meaning preservation. To confirm that, we computed sentence similarity between input and output of the loop, with SentenceTransformers [28], a Python framework for state-of-the-art sentence and text embeddings. This implementation follows the work described in [29]. This technique allowed us to create 128253 new inputs on less represented classes, of which 97240 had a similarity score between 0.7 and 1. We appended these latter to the dataset with the same labels as their respective original comment. Figure 2 illustrates some samples of the dataset with their English translation as provided by Google Translate.

Sentence	Type	Level	Target	Google_Translate
@Michou t un mytho ton pseudo aurais changé sinon	Hate	MEDIUM	USER	@Michou t un mytho your nickname would have ch...
J'suis une merde, de A à Z	Insult	LOW	AUTHOR_OF_COMMENT	I'm a shit, from A to Z
Oh des gays moches 🤣🤣🤣🤣🤣	Homophobia	HIGH	GROUP	Oh ugly gays 🤣🤣🤣🤣🤣
Allez vous faire foutre bande de connards	Insult	VERY_HIGH	EVERYONE	Fuck you all you assholes
On va bien rigoler	Neutral	NONE	UNDEFINED	We'll have some fun

Figure 2: Dataset extract. Last column illustrates English translation obtained from Google Translate

Finally, we tried a widely used approach to synthesizing new examples, called the *Synthetic Minority Oversampling Technique (SMOTE)* [30]. SMOTE works by selecting examples that are close in the input feature space, drawing a line between the examples in the feature space and choosing a new

sample at a point along that line. SMOTE allowed us to pass from 295098 to 597290 examples, perfectly balanced since all Level classes had an equal number of examples (119458).

Lastly, **emojis** are clearly part of social media culture, and as such often used to enrich our written comments (almost 80% of our dataset comments included at least one emoji). Luckily, they are often redundant with the spirit of the comment, in which case excluding them from the comment doesn't significantly change its meaning. Yet, for the cases where the emoji adds a strong level of toxicity to a neutral comment, omitting them is penalizing. That's why we applied a simple pipeline to take them into account: first, we converted them into their text description, taken from the official UTS Unicode Emoji dictionary [31]. Then, we leveraged this additional feature on top of the text comment, via two approaches: first, by concatenating both text segments, before vectorization and input into one of our model. Alternatively, we vectorized the text description of emojis and passed it through a separate dense layer concatenated before the final softmax. Despite these various attempts, we didn't manage to get improvement in the performance metrics of our classifiers.

5.2 Evaluation method

Since the ultimate objective of our project was to improve the quality of toxic content moderation, we definitely wanted to avoid over-moderation. In terms of metrics, and given the dataset is imbalanced, this translates into a focus on **Precision** and **False Positive Rate**. As an aggregated metric for each, we used the **macro-average** computation since we do value the minority classes (toxic examples).

5.3 Experimental details

We trained our models over 10 epochs, with the global same set of **hyperparameters** at first: Adam optimizer with default learning rate (0.001) and parameters, Categorical Cross-entropy loss, Dropout (at start 0.2), Early Stopping.
Using a Nvidia V100 GPU, training for 10 epochs took approx 5 hours for CamemBERT, 3 hours for Bi-LSTM, and 15 minutes for Transformer. Given these duration, we run most of our tests with the Transformer model, and kept the two others to fine-tune our best test scenarios.

We tried several fine-tuning options. Since our Transformer tend to overfit, we tested an increase in Dropout (from 0.2 to 0.5), but it didn't improve the performance. Data augmentation helped a little reduce the overfitting, but not significantly.
As mentioned earlier, we also tried loss balancing via custom class weights, but it didn't improve the overall performance of our models. The utilization of SMOTE was the best option in that case.

5.4 Results

The following table recaps our main quantitative results after fine-tuning, on our two main objectives:

Model	Predict Level				Predict Target	
	from Comment		from Comment & Target		from Comment	
	Precision	FPR	Precision	FPR	Precision	FPR
Bi-LSTM	89%	2.0%	91%	1.7%	94%	1.0%
Transformer	88%	2.1%	89%	2.0%	92%	1.6%
CamemBERT	**91%**	**1.7%**	**93%**	**1.2%**	**96%**	**0.6%**

Table 2: Main quantitative results for our 3 models, computed as Macro-average of label classes

We definitely bet the Perspective API baseline, but didn't manage to perfectly predict all labels from Bodyguard.
Overall, these results provide the answers for our two main objectives: first, they tend to prove that the target of a comment can indeed be used as complementary feature to improve the performance of toxicity level detectors. We obtained an improvement of +2pts in average precision and -0.5pts in FPR for our state-of-the-art model.
Second, they show that the target can even be predicted from the comment by deep learning models with high precision, up to 96% for our best model. However, for this second objective, adding syntactic features didn't allow us to improve the prediction metrics. For instance, our Transformer

model reached an average 92% precision on test set when trained on the comment only, but 90% precision only when pronouns (encoded as one hot vector representing their distribution) were added as complementary input feature. Interestingly, it also reached an average 91% when using three input features: the comment, pronouns, and '@' symbol presence (also encoded as one hot vectors). Further investigation of other possible syntactic features may help improve the average precision.

The pretrained CamemBERT model outperformed the two others, showing that transfer learning worked well despite the specificity of social media language.
In order to more completely assess its potential, we trained the CamemBERT model to predict each of the 3 labels of our dataset (Level, Type, Target) from the comments only. We also added a simple binary label Toxic / Neutral, and trained our classifier to predict it as well. Table 3 summarizes the results.

Model	Predict Level		Predict Target		Predict Type		Pred. Toxic/Neutral	
	Precision	FPR	Precision	FPR	Precision	FPR	Precision	FPR
CamemBERT	91%	1.7%	96%	0.6%	93%	1.5%	**98%**	**0.02%**

Table 3: CamemBERT performance metrics to predict the 4 types of label of the dataset

These results demonstrate that the CamemBERT architecture can be used to predict all labels from the input sentence with more than 90% average precision. Among those results, the performance on the binary prediction is particularly outstanding.

Finally, thanks to data augmentation, and particularly SMOTE, we managed to slightly improve the prediction performance even for less represented classes. Table 4 illustrates the results for the Transformer model, trained on comments only to predict Toxicity Levels.

Toxicity Level predicted	Transformer w/o SMOTE Precision	Transformer with SMOTE Precision
None	95%	**97%**
Low	87%	**88%**
Medium	82%	**83%**
High	79%	**86%**
Very High	83%	**84%**
Average	86%	**88%**

Table 4: Main quantitative results for our 3 models, computed as Macro-average of label classes

6 Analysis

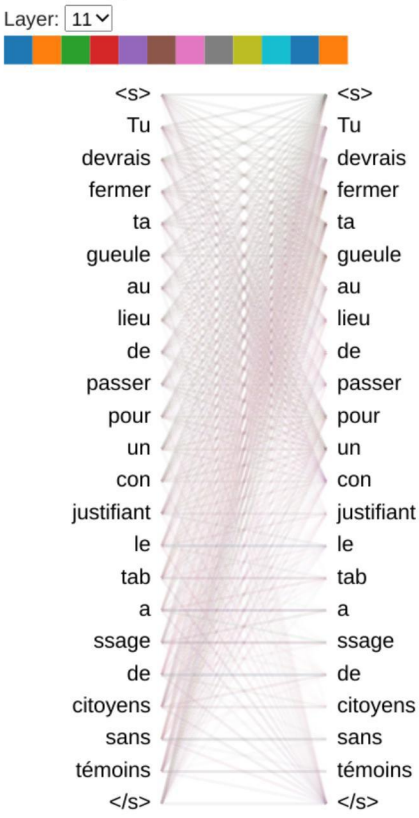
We performed **error analysis** by manually investigating classification errors from our best models. We noticed that, within the top 100 errors with highest prediction confidence, almost 60 had a questionable label. This challenges the 100% baseline of our dataset, and will provide some interesting feedback for Bodyguard. It also means that our best models, even trained on partially mislabelled data, still managed to learn the characteristics of the various toxicity levels. This generalization ability is a very positive output.
Among those top 100 errors, the false negatives (eg ranked as neutral although the comment was toxic) were mostly due to very toxic emojis added to neutral comments. Since our model didn't analyze them, its prediction relied on the text only.

On another note, it is well known that **pretraining** helps a model learn the meaning of words, as well as some broader knowledge. But we could question the ability of such models to also perform well on social media content, due to its language specificity and multiple misspellings.

The good results we got from our CamemBERT models provide some reinsurance on their ability to generalize to such specific content. This may in part be due to the fact that CamemBERT was trained on diverse web crawled data, rather than Wikipedia data.

To further understand how CamemBERT **attention mechanism** supported the efficient classification of toxic content, we performed visual inspection of the attention layers as described in [32]. The example in Figure 3 has been correctly classified as toxic by our model, and the graph illustrates the weights of the last attention layer. As we can see, most of the attention seems to be focused on the first part of the comment, which contains all the toxic words ("gueule", which is a slang word for "mouth", and "con" which is a vulgar version of "stupid") and the target ("tu", meaning "you"). More globally, This is definitely not an exact science, as already stated by [33], and all checked samples were not as convincing. Yet, it provides an interesting perspective to analyze the exact role of the attention mechanism in the good prediction performance obtained by our model.

Figure 3: Visualization of weights in attention layer 11 on toxic comment



From the results shown in previous section, we can also note that the Bi-LSTM systematically got better performance metrics than the Transformer. Beyond the difference of architecture between the two models, which we kept very simple in this project, we can question the **influence of word embedding techniques** as they also differ. In our manual tests mentioned earlier, Word2Vec seemed to provide relevant embeddings for slang and misspellings. To get a broader perspective on how Word2Vec represents toxic content, we computed a K-means of 10 clusters on its vectors. We used a KD Tree to get the 20 words closer to the centroids of our clusters, and displayed them in Word Clouds. Figure 4 illustrates 3 of these clusters. Most of the words inside each cluster tend to have related meaning, as we could expect. But curiously, they don't include much of the most frequent hate words the dataset contained. Overall, the words present in these clusters could also appear in the neutral part of the dataset. Based on this, we may deduct that the dense representations provided by the embeddings doesn't specifically capture toxicity of words.

Figure 4: Toxic content representation by Word2Vec measured by K-means of its embeddings

7 Conclusion

In this work, we managed to show that the toxicity level of a written comment can be better assessed by also considering its target, and that the target can be accurately predicted from the comment by deep learning models.
We also found that large models pretrained on massive amount of public data such as CamemBERT provide state-of-the-art results even on social media language, despite its large amount of slang and misspelling.

We made some findings in terms of syntactic features likely to help better predict the target of toxic content, but we didn't manage yet to leverage them for improved performance of our classifiers.

Along this project we touched multiple possibilities offered by current NLP techniques, but several additional experimentation remain possible in the future. As potential next steps, we'd like to further study the influence of emojis on the toxicity level of comments, and their potential to improve the detection by assessing and leveraging their own toxicity level. We'd also be keen to further investigate the interpretability of our classifiers, as many gray areas remain in this domain. Another possible next step would be to test character-level modelling and see if it allows even better word representations than Word2Vec in our context.

Finally, we hope our work will help companies such as Bodyguard further improve their own moderation algorithms, and support them in their important mission of keeping the Internet safe.

References

[1] Rolandos Alexandros Potamias, Georgios Siolas, and Andreas-Georgios Stafylopatis. A transformer-based approach to irony and sarcasm detection. *CoRR*, abs/1911.10401, 2019.

[2] Mengzhou Xia, Anjalie Field, and Yulia Tsvetkov. Demoting racial bias in hate speech detection. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 7–14, Online, July 2020. Association for Computational Linguistics.

[3] Niloofar Safi Samghabadi, Parth Patwa, Srinivas PYKL, Prerana Mukherjee, Amitava Das, and Thamar Solorio. Aggression and misogyny detection using BERT: A multi-task approach. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 126–131, Marseille, France, May 2020. European Language Resources Association (ELRA).

[4] John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online, July 2020. Association for Computational Linguistics.

[5] Bodyguard. `https://www.bodyguard.ai/en`.

[6] Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song, and Dit-Yan Yeung. Multilingual and multi-aspect hate speech analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4675–4684, Hong Kong, China, November 2019. Association for Computational Linguistics.

[7] Radhouane Guermazi, Mohamed Hammami, and Abdelmajid Ben Hamadou. Using a semi-automatic keyword dictionary for improving violent web site filtering. In Kokou Yétongnon, Richard Chbeir, and Albert Dipanda, editors, *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, SITIS 2007, Shanghai, China, December 16-18, 2007*, pages 337–344. IEEE Computer Society, 2007.

[8] P. Burnap and M. Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *Epj Data Science*, 5, 2016.

[9] Chrysoula Themeli. *Hate Speech Detection using different text representations in online user comments*. PhD thesis, 10 2018.

[10] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. Detecting offensive language in social media to protect adolescent online safety. pages 71–80, 09 2012.

[11] Manoel Ribeiro, Pedro Calais, Yuri dos Santos, Virgilio Almeida, and Wagner Meira Jr. Characterizing and detecting hateful users on twitter. 03 2018.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 10 2013.

[13] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. cite arxiv:1706.03762Comment: 15 pages, 5 figures.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] Sai Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. Deep learning models for multilingual hate speech detection. 04 2020.

[19] Sayar Ghosh Roy, Ujwal Narayan, Tathagata Raha, Zubair Abid, and Vasudeva Varma. Leveraging Multilingual Transformers for Hate Speech Detection. *arXiv e-prints*, page arXiv:2101.03207, January 2021.

[20] Neeraj Vashistha and Arkaitz Zubiaga. An online multilingual hate speech recognition system. 11 2020.

[21] Google. Perspective api. `https://www.perspectiveapi.com/`.

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

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[24] Camembert. `https://camembert-model.fr/`.

[25] Oscar. `https://oscar-corpus.com/`.

[26] Camembertpostag. `https://huggingface.co/gilf/french-camembert-postag-model`.

[27] Benoît Crabbé and Marie Candito. Expériences d'analyse syntaxique statistique du français. 06 2008.

[28] Sentence transformers. `https://github.com/UKPLab/sentence-transformers`.

[29] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. pages 3973–3983, 01 2019.

[30] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 06 2002.

[31] Unicode emoji. `https://unicode.org/Public/emoji/13.1/emoji-test.txt`.

[32] Jesse Vig. Visualizing attention in transformer-based language models, 04 2019.

[33] Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. Hate speech detection: Challenges and solutions. *PloS one*, 14:e0221152, 08 2019.