Toxic Comment Classification

Antonio Di Mauro - 599785 - a.dimauro3@studenti.unipi.it - MSc in Artificial Intelligence

Domenico Tupputi - 585794 - d.tupputi@studenti.unipi.it - MSc in Artificial Intelligence

Human Languages Technologies, Academic Year: 2020/2021

September 30, 2021

Abstract

A large percentage of online comments on public domains are generally constructive, however a significant percentage are toxic in nature. In this article, we focus on the different approaches regarding the problem of the classification of toxic comments online. This task is a research initiative founded by Jigsaw and Google (both part of Alphabet) who are working on tools, which use Natural Language Processing techniques, to help improve online conversation.

Keywords — Natural Language Processing, Bert, Neural Network, Toxic comment classification, Deep learning, Preprocessing.

1 Introduction

Keeping online conversations constructive and inclusive is a crucial task for platform providers. Automatic classification of toxic comments, such as hate speech, threats, and insults, can help in keeping discussions fruitful. In addition, new regulations in certain European countries have been established enforcing to delete illegal content in less than 72 hours.

Active research on the topic deals with common challenges of natural language processing, such as long-range dependencies or misspelled and idiosyncratic words.

We will use a dataset of comments on Wikipedia talk pages presented by Google Jigsaw during Kaggle's Toxic Comment Classification Challenge [2]. These sets include common difficulties in datasets for the task: They are labeled based on different definitions; they include diverse language from user comments; and they present a multi-class and a multi-label classification task respectively. In this work, we apply and compare different classifiers. Each classifier, such as Logistic Regression, Multi-layer Perceptron, is meant to tackle specific challenges for text classification. We apply the same classifiers to a dataset of validate our results.

Unfortunately for the problem, but fortunately for the Wikipedia community, toxic comments are rare. Just over 10% of this dataset is labeled as toxic, but some of the subcategories are extremely rare making up less than 1% of the data.

Because of this imbalance, accuracy is a practically useless metric for evaluating classifiers for this problem.

Author	Models	AUC
HOON BENG [5]	Logistic Regression	0.9741
AMIR ASHRAFF [4]	ensemble models	0.98705
BOJAN TUNGUZ [21]	Bi-GRU-LSTM Dual Embedding	0.98702
Jay Speidell [19]	NB-SVM	0.9747
SMU Data Science Review [18]	SVM	0.9725
Chun Ming Lee [10]	Vanilla Bi-GRU	0.9885

Table 1: State Of The Art

The Kaggle challenge based on this dataset uses ROC/AUC, or the area under a receiver operating characteristic curve, to evaluate submissions.

Each metrics that we will use gives valuable insight into a model's performance, but they fail to show the whole picture and have weaknesses where bad models get high scores. Predicting all positive values will bring recall up to 100%, while missing true positives will be penalized. Precision will harshly penalize false positives, but a model that predicts mostly negative can achieve a high precision score whether or not the predictions are accurate.

The F1 score is a harmonic average between precision and recall. This combines the strengths of precision and recall while balancing out their weaknesses.

In summary, the following metrics will be shown:

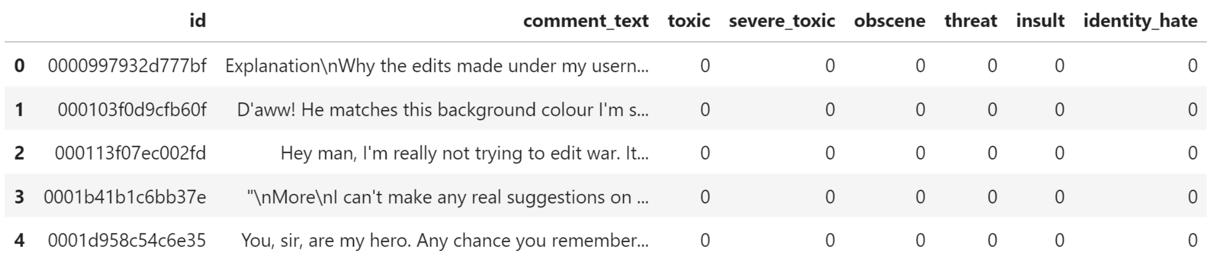
1. Precision (P): the ability of the classifier not to label as positive a sample that is negative.
2. Recall (R): the ability of the classifier to find all the positive samples.
3. F1-score (F1): the relative contribution of precision and recall (best value at 1 and worst score at 0).
4. AUC: the conventional area-under-the-curve score, as implemented in scikit-learn. It is a measure of separability of classes.
5. Accuracy (A): is the ratio of number of correct predictions to the total number of input samples.

2 State of the art

In this section we will highlight the state of the art (SOTA) of this task, according to participants in the Toxic Comment Classification Challenge. You can see a brief summary in Table 1. The models involved range from Logistic Regression to an ensemble of multiple models. Many models to suppress data imbalance have unified the labels 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate' in a single label.

Others have tried a more agnostic approach using stylometric characteristics (ex: word frequencies, POS tags, N-grams, ...) and also classic characteristic sets such as token or character n-gram representation. Many other approaches using BERT models. As for the resulting performances for all the listed works, it is possible to consult the Table 1.

As for the resulting performance for all the listed works, as noticeable, they all have a very high AUC, but looking at F1 score, it's really very low, averaging around 0.5.



	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Table 2: Example of data from the Kaggle dataset

3 Dataset

The task of toxic comment classification lacks a consistently labeled standard dataset for comparative evaluation (Schmidt and Wiegand, 2017) [3]. While there are a number of annotated public datasets in adjacent fields, such as hate speech (Ross et al., 2016; Gao and Huang, 2017) [11], racism/sexism (Waseem, 2016; Waseem and Hovy, 2016) [13] or harassment (Golbeck et al., 2017) [9] detection, most of them follow different definitions for labeling and therefore often constitute different problems.

Class	# of occurrences
Clean	201,081
Toxic	15,294
Obscene	8,449
Insult	7,877
Identity Hate	1,405
Severe Toxic	1,595
Threat	478

Table 3: Class distribution of Wikipedia dataset. The distribution shows a strong class imbalance.

We analyse a dataset published by Google Jigsaw in December 2017 over the course of the 'Toxic Comment Classification Challenge' on Kaggle [1]. It includes 223,549 annotated user comments collected from Wikipedia talk pages and is the largest publicly available for the task. These comments were annotated by human raters with the six labels 'toxic', 'severe toxic, 'insult', 'threat', 'obscene' and 'identity hate'.

Comments can be associated with multiple classes at once, which frames the task as a multi-label classification problem. Jigsaw has not published official definitions for the six classes. But they do state that they defined a toxic comment as "a rude, disrespectful, or unreasonable comment that is likely to make you leave a discussion.

The dataset features show an unbalanced class distribution, shown in Table 2 and in Table 3. 201,081 samples fall under the majority 'clear' class matching none of the six categories, whereas 22,468 samples belong to at least one of the other classes. While the 'toxic' class includes 9.6% of the samples, only 0.3% are labeled as 'threat', marking the smallest class. Comments were collected from the English Wikipedia and are mostly written in English with some outliers, e.g., in Arabic, Chinese or German language. The domain covered is not strictly locatable, due to various article topics being discussed. Still it is possible to apply a simple categorization of comments as follows:

1. 'community-related':

   Example: "If you continue to vandalize Wikipedia, you will be blocked from editing."

2. 'article-related':

   Example: "Dark Jedi Miraluka from the MidRim world of Katarr, Visas Marr is the lone surviving member of her species."

3. 'off-topic':

   Example: "== I hate how my life goes today == Just kill me now."

4 Experimental setup

The experiments involve models from scikit-learn and tensorflow libraries.
Those from scikit-learn library are: Multinomial Naive Bayes, Linear Support Vector Classifier, Logistic Regression, K-Nearest Neightbours and Multilayer Perceptron. A tf-idf vectorization, using stopwords from nltk library, of the input sentences are provided as input to all the models. A One-vs-the-rest

(OvR) multiclass strategy is used with all the models and it consists in fitting one classifier per class and for each classifier, the class is fitted against all the other classes.
Those from tensorflow library are: Deep Echo State Network and distilBERT. The models are built using the keras build-in functions of tensorflow.
A grid search was done to find the final hyperparameters of each model except for distilBERT because of computational and time constraints.

4.1 Multinomial Naive Bayes

This model is implemented using the MultinomialNB class that implements the naive Bayes algorithm for multinomially distributed data. The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, ..., \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and $\theta_y = P(x_i|y)$ of feature i appearing in a sample belonging to class y.
The parameter θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_y = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{1}$$

where N_{yi} is the number of times feature i appears in a sample of class y.
With the parameter fit_prior=True is assumed a uniform prior, with the parameter class_prior=None the prior is adjusted according to the data [17].

4.2 Linear Support Vector Classifier

In a Linear Support Vector Machines we have a training dataset with n points of the form $(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)$, where the y_i are either 1 or -1, each indicating the class to which the point $\boldsymbol{x_i}$ belongs. Each $\boldsymbol{x_i}$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\boldsymbol{x_i}$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\boldsymbol{x_i}$ from either group is maximized.
Any hyperplane can be written as the set of points $\boldsymbol{x}$ satisfying $w^T x - b = 0$, where $\boldsymbol{w}$ is the (not necessarily normalized) normal vector to the hyperplane. $\frac{b}{\|w\|}$ determines the offset of the hyperplane from the origin along the normal vector $\boldsymbol{w}$ [16].
This model is implemented using the LinearSVC class that implements the Support Vector Classifier with linear kernel. It is capable of performing binary and multi-class classification on a dataset.
The parameter class_weight=balanced to automatically adjust weights inversely proportional to class frequencies in the input data.

4.3 Logistic Regression

The Logistic Regression is a simple model described by the equation $\frac{1}{1+e^{-wx}}$, in which we basically learn to binary separate the samples by softening the Perceptron's step-function with the continuous differentiable logistic equation previously described.
This models is implemented using the LogisticRegression class that implements regularized logistic regression. The solver is setted to "lbfgs" [20].

4.4 K-Nearest Neighbours

The K-nearest neighbours is a non-parametric classification method in which the input consists of the k closest training examples in the dataset. The output is a class membership in which an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors [12].
The model is implemented using the KNeighborsClassifier class that implements the k-nearest neighbors vote. The number of neighbors is set to 5.

4.5 Multi-layer Perceptron

The Multi-layer Perceptron is a layered model made of different layers of artificial neurons, such that each unit receives input only from units in the immediate preceding layer. Each layer has a set of parameters $\mathbf{W}$ to be learned through a proper learning algorithm. Each unit computes a weighted sum of its input $net_j = \sum_{i=0}^{n} w_{ij} a_i$, where a is the output of the previous layer, and then it applies an activation function f (e.g: sigmoid) such that the output $a_j = f(net_j)$. [14]
This model is implemented using the MLPClassifier class that optimizes the log-loss function. The solver used is "Adam", with one hidden layer with 100 units and "ReLu" activation function. The learning rate is setted to "0.001" with a maximum iterations of 100.

4.6 Deep Echo State Network

The Echo State Network is a Reservoir Computing approach that uses a Recurrent Neural Network (RNN) with a sparsely connected hidden layer named "reservoir" with fixed and randomly assigned weights. The reservoir is constructed in such a way that respect the Echo State Property (ESP) that guarantees asymptotic stability to the reservoir in a dynamical

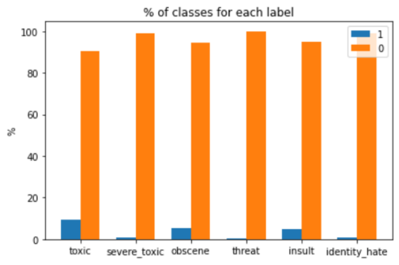
system perspective; practically the recurrent weight matrix of the reservoir is scaled with the desired spectral radius, that in theory is 1. The weights of output neurons, called "readout", can be learned so that the network can produce or reproduce specific temporal patterns.
This model is implemented using the DeepRC.py of Prof. Gallicchio for the reservoir and keras for the readout.
The reservoir has 10 layers and each one with 200 units. Each reservoir layer computes the state of the RNN at each time step t as $h(t) = (1-\alpha)h(t-1) + \alpha tanh(\boldsymbol{U}x(t) + \boldsymbol{W}h(t-1))$, where $\boldsymbol{U}$ (with the same dimension of the input) is the input weight matrix, $\boldsymbol{W}$ (with dimension NxN where N is the number of recurrent units) is the recurrent weight matrix and α is the leaking rate ($\alpha \in (0,1]$). In a multi-layer setting, the output of a reservoir layer is the input of the next reservoir layer [8].
The readout take as input the output of the reservoir and is made by 2 dense layers with, respectively, 100 and 50 units and "ReLu" activation function with two dropout layers after each dense layer. The output is a dense layer with 6 units (as the number of labels) and "sigmoid" as activation function.
The model is optimized minimizing the binary crossentropy using "Adam" as optimizer with a learning rate of "0.001". The total number of trainable parameters are 25,456 and the number of epochs used for the training is 1.
A tf-idf vectorization of the inputs are provided to the model using 10 most common words.

4.7 DistilBERT

The distillation is a compression technique in which a compact model, the student, is trained to reproduce the behaviour of a larger model, the teacher, or an ensemble of models, in some sense it is similar to a posterior approximation.
Bert ("bert-base") is an encoder-decoder network. Both encoder and decoder sections of transformer are a stack of 6 identical layers of multi-head attention and feed forward sublayers with 768 hidden units. The decoder has an additional sublayer which performs attention over output.
DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than "bert-base-uncased", runs 60% faster while preserving over 95% of BERT's performances. It does not have token-type embeddings, pooler and retains only half of the layers of BERT and it has 66M parameters [15].
The model is implemented using TFDistilBertModel based on "distilbert-base-uncased" of huggingface library. The output of the [CLS] token is fed in input to a dense layer with 50 units and "ReLu" activation function, a dropout of the 10% and finally to a dense layer of 6 units (as the number of labels) with a "sigmoid" activation function.
The model is optimized minimizing the binary crossentropy using "Adam" as optimizer.
There are two versions of the model: the one with distilBERT layer with non trainable and fixed weights with only 38,756 trainable parameters over 66,401,636 and the other with distilBERT finetuned with 66,401,636 trainable parameters.
The first model is trained for 4 epochs, instead for the second have been used 2 epochs.
It is used the class DistilBertTokenizerFast of huggingface for the tokenization of the inputs. The inputs are passed in batches to the model and accepts also the attention mask of the corresponding inputs. The input sequences has a maximum length of 100.

5 Results and discussion

Figure 1: Average percentage of classes per label.



AUC score uses True Positive Rate (TPR or Recall) and False Positive Rate (FPR). Precision-Recall analysis, on the other hand, exchanges FPR for Precision. Then, while AUC uses all the cells (TP, FP, TN, FN) of the Confusion Matrix, Precision-Recall disregards the True Negatives, which have a high impact on an imbalanced problem, since almost all of the data is of the negative class. Therefore, Precision-Recall gives more weight to the majority class (the negative class) than the AUC. This is why the AUC is more suitable for heavily imbalanced problems [6] [7].
The best model is the Logistic Regression with an AUC score of "0.9734" far from the Kaggle SOTA AUC of "0.9885" and a low accuracy of "0.8130" that is not significant for this task as stated before.
The second best model is distilBERT with an AUC score of "0.9411" and an accuracy of "0.9713".
Overall the two models are slightly similar w.r.t. F1 score.

Model	P	R	F1	AUC	Accuracy	
Multino-mialNB	0.90	0.11	0.19	0.8482	0.8986	**VL**
	0.84	0.12	0.20	0.8482	0.8986	**TS**
LinearSVC	0.67	0.77	0.72	0.8263	0.8950	**VL**
	0.47	0.81	0.59	0.8429	0.8325	**TS**
Logistic Regression	0.61	0.85	0.70	0.9782	0.88	**VL**
	0.42	0.88	0.56	0.9734	0.8130	**TS**
KNeighbors Classifier	0.81	0.17	0.27	0.5710	0.9012	**VL**
	0.63	0.18	0.28	0.5729	0.8999	**TS**
MLP-Classifier	0.73	0.66	0.69	0.9594	0.9001	**VL**
	0.54	0.69	0.60	0.9564	0.8524	**TS**
DeepESN	0.21	0.01	0.01	0.7145	0.9629	**VL**
	0.17	0.00	0.01	0.6779	0.9621	**TS**
distilBERT	0.82	0.54	0.64	0.9567	0.9790	**VL**
	0.65	0.51	0.56	0.9411	0.9713	**TS**
distilBERT (fine-tuned)	0.84	0.54	0.65	0.9544	0.9790	**VL**
	0.66	0.52	0.57	0.9403	0.9711	**TS**

Table 4: Validation and Test scores (Kaggle SOTA AUC: 0.9885).

6 Conclusion

This task was very challenging in terms of training a model on sparse toxic comments.
It was a surprise to notice how a simple model in a OvR setting performs better than complex models as distilBERT.
Probably a model selection phase and the using of more epochs should increase the performances of distilBERT, but training a transformer is heavy and requires a lot of time.
In our opinion the models performances are far from the State of the Art because of the imbalanced dataset, as shown in Figure 1, that led to have low value of P/R and F1 scores.

As a future work, 5 experiments could be done to handle unbalanced data as follows:

1. Define higher weight for label '1': Depending on how imbalanced the data is, we put higher weight on the minority label to force the classifier to work harder on label '1'. For example, if the ratio of label '1' to the label '0' is 1:9 we will put 0.9 as the weight factor on label '1'.

2. Over-sampling label '1' (minority class): To reduce the imbalance, we simply duplicate samples with label '1' to achieve 1:1 as the ratio of negative samples to positive samples. But there

is a downfall for this method. It might result in over-fitting.

3. Under-sampling label '0' (majority class): To reduce the imbalance, this time we reduce the number of negative samples with label '0' to get the ratio of negative samples to positive samples closer to 1:1. This also has a disadvantage of losing potentially important data.

4. Performing dataset augmentation.

5. Performing stratified sampling during training.

References

[1] Jigsaw/Conversation AI. Toxic comment classification challenge: Data description. *https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data.*

[2] Jigsaw/Conversation AI. Toxic comment classification challenge. *https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview/description*, 2018.

[3] Michael Wiegand Anna Schmidt. A survey on hate speech detection using natural language processing. *https://aclanthology.org/W17-1101/*, 2017.

[4] AMIR ASHRAFF. Fork of ensemble_3_blend. *https://www.kaggle.com/amir78pgd/fork-of-ensemble-3-blend*, 2019.

[5] HOON BENG. Classifying multi-label comments with logistic regression. *https://www.kaggle.com/rhodiumbeng/classifying-multi-label-comments-0-9741-lb*, 2018.

[6] Jason Brownle. Tour of evaluation metrics for imbalanced classification. *https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/*, 2020.

[7] Jason Brownlee. Roc curves and precision-recall curves for imbalanced classification. *https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classification/*, 2020.

[8] Claudio Gallicchio and Alessio Micheli. Deep echo state network (deepesn): A brief survey. *arXiv preprint arXiv:1712.04323*, 2017.

[9] Rashad O. Banjo Alexandra Berlinger Siddharth Bhagwan Cody Buntain Paul Cheakalos Alicia A Geller Quint Gergory Rajesh Kumar Gnanasekaran Raja Rajan Gunasekaran Kelly M. Hoffman Jenny Hottle Vichita Jienjitlert Shivika Khare Ryan Lau Marianna J. Martindale Shalmali Naik Heather L Nixon Piyush Ramachandran Kristine M. Rogers Lisa Rogers Meghna Sardana Sarin Gaurav Shahane Jayanee Thanki Priyanka Vengataraman Zijian Wan Derek Michael Wu Jennifer Golbeck, Zahra Ashktorab. A large human-labeled corpus for online harassment research. *http://www.cs.umd.edu/ golbeck/papers/trolling.pdf*, 2017.

[10] Chun Ming Lee. Kaggle toxic comment classification challenge 1st place. *https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52557.*

[11] Ruihong Huang Lei Gao. Detecting online hate speech using context aware models. *https://www.researchgate.net/publication/320564386_Detecting_Online_Hate_Speech_Using_Context_Aware_Models*, 2017.

[12] Sarah Jane Delany Padraig Cunningham. K-nearest neighbours. *https://www.researchgate.net/publication/228686398*$_k-Nearest_neighbour_classifiers$, 2007.

[13] Marian-Andrei Rizoiu, Gabriela Ferraro Tianyu Wang, and Hanna Suominen. Transfer learning for hate speech detection in social media. *https://arxiv.org/pdf/1906.03829.pdf*, 2019.

[14] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. *https://www.cin.ufpe.br/ tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf*, 2002.

[15] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[16] scikit learn. Linear support vector classifier. *https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.htmlsklearn-svm-linearsvc.*

[17] scikit learn. Multinomial naive bayes. *https://scikit-learn.org/stable/modules/* $naive_bayes.htmlmultinomial - naive - bayes$.

[18] Jeff Leath David Stroud SMU Data Science Review, Sara Zaheri. Toxic comment classification. *Southern Methodist University, 6425 Boaz Lane, Dallas, Texas 75205 szaheri, jleath, jdstroud @smu.edu*, 2020.

[19] Jay Speidell. Toxiccommentclassification-. *https://github.com/jayspeidell/ToxicComment Classification-/blob/master/Report.pdf*, 2018.

[20] in Research Methods for Cyber Security Thomas W. Edgar, David O. Manz. Logistic regression. *https://www.sciencedirect.com/topics/computer-science/logistic-regression*, 2017.

[21] BOJAN TUNGUZ. Bi-gru-lstm dual embedding new test cleaned 5. *https://www.kaggle.com/tunguz/bi-gru-lstm-dual-embedding-new-test-cleaned-5*, 2019.