Neutralize the toxin: Implementation and Comparison of Algorithms for Classification of Toxic Comments

1st Amandeep Kaur Department of Information Technology
Indira Gandhi Delhi Technical University for Women, New Delhi, India
amandeep022btit18@igdtuw.ac.in
1st Drishti Gupta Department of Information Technology
Indira Gandhi Delhi Technical University for Women, New Delhi, India
drishti018btit18@igdtuw.ac.in
1st Srinidhi Ayyagari Department of Information Technology
Indira Gandhi Delhi Technical University for Women, New Delhi, India
srinidhi026btit18@igdtuw.ac.in
1st Aayushi Bansal Department of Information Technology
Indira Gandhi Delhi Technical University for Women, New Delhi, India
aaysuhi063btit18@igdtuw.ac.in
Rishabh Kaushal Department of Information Technology
Indira Gandhi Delhi Technical University for Women
New Delhi, India
rishabhkaushal@igdtuw.ac.in

Abstract

With the increase in number of online forums, presence of toxic comments has become prevalent as they end up hurting the sentiments of people participating on those forums. In this paper, our goal is to identify and classify toxic online comments. We present different computational approaches used for classification of toxic comments into six categories. To evaluate, we use a large corpus of Wikipedia Talk Page comments from the Kaggle Toxic Comment Classification Challenge. We evaluate the efficacy of data augmentation using Easy Data Augmentation and Back Translation in prediction task.

Index Terms

Multi-Label Classification, Wikipedia Talk Pages, Easy Data Augmentation, Back Translation, NLP, machine learning, deep learning

I. Introduction

Online communication has enabled humans to participate in varied interactions and engagements instantaneously. It is a great way to network and connect along with discovering valuable information. But with debates, discussions and disagreements negative comments have become almost unavoidable.

Negative online behaviour and conversational toxicity has become tremendously common in social networking sites and online communities. Disrespectful and toxic comments can lead to increased stress, anxiety, uneasiness and low self esteem and users end up participating in online discussions. Due to this, many communities are forced to limit the conversation or shut down user comments completely.

This project focuses on implementing various approaches to detect different types of toxicity like severe-toxic, obscenity-based, threats, insults, and identity-based hate. We perform Data Augmentation and analyse its efficacy. Algorithms were evaluated on a corpus of Wikipedia comments from the Kaggle Toxic Comment Classification Challenge.

The toxic behavior of comments have been labeled by human raters. The data has been visualized through graphical representations in order to detect any outliers and anomalies.

Thus in this project, we predict the probability of each type of toxicity for each comment and perform Easy Data Augmentation and Back-Translation. The 6 types of toxicity in the dataset are: toxic, severe-toxic, obscene, threat, insult and identity-hate. The structure of paper is as follows: Section II gives an overview of the past work done to classify toxic comments. In Section III, we describe our infrastructure that is the description of dataset, and the Exploratory Data Analysis performed. Section IV describes our experimental setup including feature engineering and data augmentation techniques. Further in Section V,

implementation details are described: the vectorization methods, machine learning and deep learning models. We compare and analyze our results in Section VI. Finally we conclude and describe our future work in Section VII. [1]

II. Literature review

Table I gives an overview of the base papers we followed as well the methods used by them.

TABLE I
BASE PAPER ANALYSIS

Ref	Title	Description
1	Can We Achieve More with Less? Exploring Data Augmentation for Toxic Comment Classification and Wikipedia	In this paper, the author experimented with Easy Data Augmentation (EDA) and Backtranslation, and implemented Logistic Regression, Support Vector Machine (SVM), Bidirectional Long Short-Term Memory Network (Bi-LSTM) models.
2	Convolutional Neural Networks for Toxic Comment Classification	In this work, the authors compare CNNs against the traditional bag-of-words approach for text analysis combined with a selection of algorithms including kNN, Naive Bayes and Support Vector Machines
3	Challenges for Toxic Comment Classification: An In-Depth Error Analysis	The authors have studied two datasets: Google Jigsaw during Kaggle's Toxic Comment Classification and a Twitter Dataset by Davidson et al. (2017) [9]. The authors have used various word embeddings and applied classifiers, such as Logistic Regression, bidirectional RNN and CNN. Based on these, they created and analysed an ensemble that improved results in detail.

III. Infrastructure

A. Dataset

We have performed our analysis on the "Wikipedia Toxic Comments" dataset, from Kaggle Comment Classification Challenge. Approximately 158K wikipedia comments are present with 6 labels for the nature of hate speech - toxic, severe-toxic, obscene, threat, insult and identity-hate. For the dataset, toxic comments of the above categories constitute the positive class while the rest of the lot makes up the negative class. After analysing the data and plotting the same using matplotlib library, Figure 14 (supplementary) shows that comments range from 1 to 1750 characters and maximum comments contain less than 500 characters. From Figure 15(supplementary) it is clear that the number of words in a comment are between 1 to 400 and most sentences have less than 100 words. Figure 16 (supplementary) depicts that the average word length ranges between 2 to 10 with 5 being the most common word length. Does it mean that people are using really short words in comments? One reason for this can be, stopwords("the"/"a"/"an") . Due to which the average word length might be incorrectly left-skewed. Thus, in the data pre-processing state,important processes to do in order to better understand the data are:

- Lowercasing: Like converting "The" to "the"
- Remove stopword and punctuation: Such as "the", "to", "of", "and" and "a"
- Tokenize: Convert sentences into a list of tokens
- Lemmatize/Stemming: Reduce the inflectional forms of words to its stem, like "who" and "whose"

B. Exploratory Data Analysis

After analysing the dataset, we found that the toxicity level is unevenly spread across the classes, indicating that this dataset suffers from class imbalance as seen in Figure 1, and many comments are a part of multiple classes as seen in Figure 2, which indicates the problem is of *multilabel classification.*
There are 159,571 comments in the dataset and the distribution is as follows : [('toxic', 15294), ('severe_toxic', 1595), ('obscene', 8449), ('threat', 478), ('insult', 7877), ('identity_hate', 1405)]. Figure 3 depicts correlation which helps us in finding relationship/dependencies.

- "Toxic" and "severe toxic" are weakly correlated.
- "Obscene" comments and "insult" comments are highly correlated.

[1]Find the code base : https://bit.ly/2SefRju

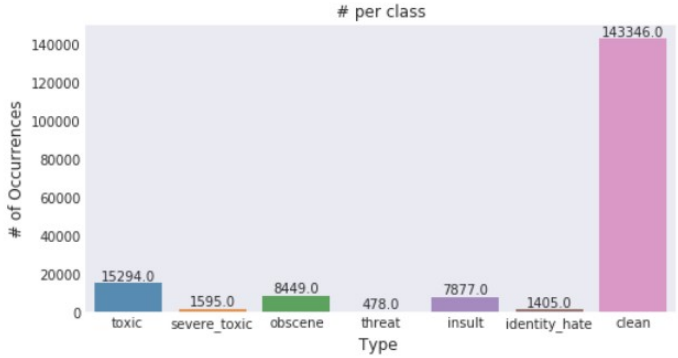

Fig. 1. Distribution of comments across various labels

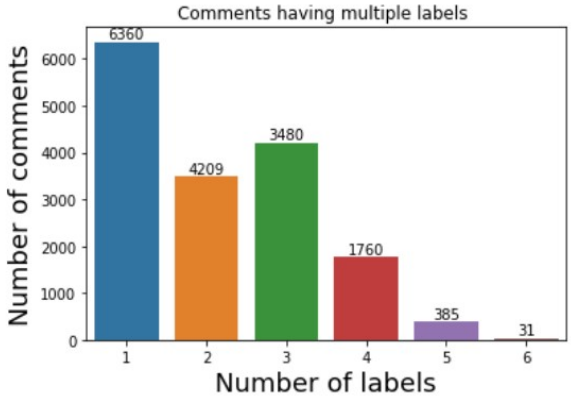

Fig. 2. Distribution of comments with multiple labels

Word Cloud - Representation of text data in visual format. The words more often mentioned within a given text are bigger and bolder. We see that the Word Cloud of clean comments in Figure 4 highlights the words talk, page, article, think etc. which are explanatory since the frequency of non-toxic comments is much higher than toxic comments and Wikipedia is a platform for communication, normal words are more prominent. Figure 5 shows the word cloud for ”toxic” comments, words like fuck, hate, nigger and other derogatory terms are highlighted. As we can observe the comments which are severe toxic are a super set of toxic comments, thus the same derogatory words repeat again. Threatening words like must die, kill etc are more prevalent. We observe that insulting comments are more related to body shaming (fat), racism (Jew, Nigger), gender based (faggot), thus indicating the various types of categories of insults.

IV. Experimental Setup

A. Feature Engineering

Natural Language Processing is usually applied for classification of text data. In this problem statement, categories are assigned to text data according to its semantic meaning.The crucial step of text classification is feature engineering which is the task of creation of features for an ML model from the raw text data we have. As part of the feature engineering task, we use N-grams.

- Unigrams : A 1-gram (or unigram) is a one-word sequence. For example, ”This is an AI and NLP based project”. The unigrams of this sentence are ”This”,”is”,”an”,”AI”,”and”,”NLP”,”based”,”project”. Unigrams do not contribute much to the meaning of the sentence.
- Bigrams : A 2-gram (or bigram) is a two-word sequence. For example, ”This is an AI and NLP based project”. The bigrams of this sentence are ”This is ”,”is an”,”an AI”,”AI and ”,”and NLP”,”NLP based”,”based project”. Bigrams contribute moderately to the meaning of the sentence.
- Trigrams : A 3-gram (or Trigram) is a three-word sequence. For example, ”This is an AI and NLP based project”. The bigrams of this sentence are ”This is an”,”is an AI”,”an AI and”,”AI and NLP”,”and NLP based”,”NLP based project”. As we can anaylze that trigrams are able to bring efficient meaning of the sentence, we are able to capture important word sequences like ”AI and NLP” and ”NLP based project” which maybe further used for various analysis like word auto-complete etc. Figure 6 shows the trigram chart for Wikipedia data before pre-processing. The trigram analysis is more

insightful. Phrases like ”thank you for”, ”if you have”, ”my talk page”, ”please do not”, and ”would like to” dominate most comments.

B. Training Strategy

1) Easy Data Augmentation based [1][7][8]: Easy Data Augmentation is done by performing any of 4 kinds of operations on the sentences in the dataset. The words to be operated upon are chosen at random. The 4 operation performed by us to augment our data are:

- Synonym Replacement (SR): Randomly choose n tokens from the sentence(not stop words), and replace chosen token by one of its randomly chosen synonym.
- Random Swap (RS): Randomly choose tokens from the sentences, and swap their positions.
- Random Insertion (RI): Randomly choose n tokens from the sentence (not stop words), and insert a synonym (picked randomly) in place of each word.
- Random Deletion (RD): For each word in a sentence, delete words with probability greater than p.

Each of the above operations tackle some problems encountered in small datasets. SR helps in maintaining the same syntactic and semantic meaning [6] as the original sentence while generating new words that might not be present and will help the model to learn. RI and RS maintain the words in the sentence, but introduce disorder that helps the model identify and process unknown patterns. RD helps in reducing over-fitting, by deleting words randomly, it ensures that an ML model is not ”memorizing” particular patterns and is exploring all features.

We implement EDA such that number of tokens being operated upon, in every iteration, are proportional to the length of the sentence. This is because long sentences are more robust to noise than short sentences.[1] A parameter (a) decided the number of tokens(n) that will undergo either of the mentioned operations. For simplicity, we take the probability in case of RD also equal to a. Thus, a acts as a hyperparameter, which can be tuned for better performance. Finally, we call our four functions, one corresponding to each operation, uniformly on each randomly chosen sentence to generate an augmented sentence in its place.
In our case,

- sample dataset size = 7979 (5% of original dataset)
- n = number of tokens augmented
- l = 10 (avg length)
- a = 0.1
- So, n = a*l = 0.1* 10 = 1
  implying that 1 token will be changed for each instance.
- Also, we generate 5*4 sentences for each original sentence So total number of augmented sentences are:
  4 * 5 * 7979 = 159580 data points.

Having built an augmented dataset, We run all our previously built models on it and compare them to the results obtained on the baseline dataset.

2) Length Based: Length Based Training and testing is done by dividing the original dataset into two separate datasets named df_long and df_short for long and short sentences respectively.

- df_long = sentences with greater than 200 characters (81066 data points)
- df_short = sentences with less than or equal to 200 characters (78505 data points)

Our dataset categorization is :

- Df-long [81066 data points]
  [(’toxic’, 5144), (’severe-toxic’, 426), (’obscene’, 2597), (’threat’, 142), (’insult’, 2408), (’identity-hate’, 445)]
- Df-short [78505 data points]
  [(’toxic’, 10150), (’severe-toxic’, 1169), (’obscene’, 5852), (’threat’, 336), (’insult’, 5469), (’identity-hate’, 960)]

We implemented these datasets in three different ways on our various models:

- Train and Test Only on Longer Sentences
- Train and Test Only on Shorter Sentences
- Train On Short Sentences and Test on Long Sentences

Following are some observations we made:

- df-short had a lot more toxic comments than df-long.

- We have achieved much better accuracy by implementing the models separately on the divided datasets than on the original Wikipedia dataset.
- To see if the model was able to predict toxicity-classes from shorter sentences, we trained our models on df-short, and tested it on df-long. This use case had maximum accuracy, maybe because there were more toxic comments in df-short, which resulted in better classification of comments on df-long (with fewer toxic comments).

V. Implementation details baseline and comparative

Here we discuss the various methods and techniques used by us for analysis on both wikipedia dataset and created EDA dataset.

A. Vectorization Methods

- GloVe
  GloVe is an unsupervised machine learning algorithm used to obtain vector representation of words. Here the model is trained on training non-zero entries of " global word-word co-occurrence matrix", which illustrates how frequent words co-occur with each other in a given data sample. We have used glove.6b.200d.txt for our vectorization.

- TF-IDF

  TF-IDF scores the relative importance of a word based on its semantics. It consists of Term Frequency (number of times a word appears in a document divided by the total number of words in the document) and Inverse Data Frequency (IDF) which is the log of the number divided by the number of documents that contain the word w. Finally the TF-IDF is simply the TF multiplied by IDF. Figure 7 depicts the relation between TF-IDF scores and words in each class.

B. Learning Algorithms

Figure 8 depicts the models we use for our analysis. Results obtained are in Results and Discussion section

- Machine Learning based Approaches
  Because our problem is of multilabel classification, we use OneVsRest Approach in our models to perform classification using Machine Learning Techniques. 1. Logistic Regression
  Logistic regression is a statistical ML model used for the purpose of discrete classification(hypothesis statement with more than 1 class). It is a simple technique used in the paper to validate the performance of the data set used, both original Wikipedia comments and augmented dataset. The model is used with "categorical variables" and 100 max-iterations, to predict the probability of occurrence of a variable. The evaluation metrics used are accuracy, precision, recall, F1-score and confusion matrix(supplementary).

  2. Decision Tree
  Decision tree algorithm, as the name suggests, is a tree-like structure based approach where internal nodes represent the feature, branch determines a decision rule with leaf nodes giving the outcome. DT algorithm is considered as a white box ML algorithm as it shares the internal decision-making logic and the training time is much faster than Neural networks. We have used "DecisionTreeClassifier" from sklearn.tree library with default values.

  3. Random Forest Classifier
  The next model used is random forest classification algorithm, which is a supervised learning method used for classification and regression problems. This model creates multiple decision trees on randomly selected data points(bootstrap = True), gets predictions from each tree and selects the best accuracy based solution.

  4. Support Vector Machine (SVM)
  Support Vector Machine, a supervised machine learning algorithm, is used to classify a data point by looking at the extremes of the dataset called 'support vectors' and creating a decision boundary called 'hyperplane' that maximizes the margin between two classes. Thus, this algorithm implies that only support vectors are important while classification and other training samples can be ignored.

  5. K Nearest Neighbours (KNN)
  K-Nearest Neighbour is a Supervised Machine Learning algorithm based on feature similarity. It classifies a new data

point based on it's similarity with existing data points by calculating the shortest distance among it's neighbours. 'K' indicates the number of nearest neighbours the algorithm should consider. This algorithm is effective for large dataset and for classification of non linear data points .

6. NBSVM
The Naive Bayes-Support Vector Machine is an algorithm that combines Multinomial Naive Bayes (MNB) classifier with the Support Vector Machine (SVM). The word count features are replaced with Naive Bayes log-count ratios. It is known to be one of the very powerful algorithms with respect to text classification.

VI. Results and discussion

Following table depicts the results after experimentation, Table II shows results on the baseline dataset, Table III depicts our results on the Augmented dataset, Table IV, V and VI shows the results of Length based training and testing while the Table VII mentions the best results as mentioned in analysed base papers.

TABLE II
Implemented Techniques and results on Wikipedia dataset

Parameters	***Accuracy***	***Precision***	***F-score***	***Recall***
TFIDF +LR	96.3%	96.3%	96%	96.3%
TFIDF +RFC	66%	79%	64%	56%
TFIDF +DT	62%	63%	61%	60%
TF-IDF SVM	91.763%	83%	57%	68%
KNN	89.5%	70%	29%	19%
NBSVM	95.6%	91.6%	93.6%	95.6%

TABLE III
Implemented Techniques and results on created EDA dataset

Parameters	***Accuracy***	***Precision***	***F-score***	***Recall***
TF-IDF + LR	95%	92.1%	93%	95.5%
TF-IDF + RFC	79%	82%	66%	56%
TF-IDF + SVM	90.52%	85%	68%	57%

TABLE IV
Train and Test Only on Longer Sentences

Parameters	***Accuracy***	***Precision***	***F-score***	***Recall***
TF-IDF + RFC	-	79%	53%	42%
TF-IDF + SVM	93.9%	84%	54%	41%

TABLE V
Train and Test Only on Shorter Sentences

Parameters	***Accuracy***	***Precision***	***F-score***	***Recall***
TF-IDF + RFC	-	81%	72%	68%
TF-IDF + SVM	89.7%	88%	70%	59%

TABLE VI
Train On Short Sentences and Test on Long Sentences

Parameters	***Accuracy***	***Precision***	***F-score***	***Recall***
TF-IDF + RFC	-	62%	55%	53%
TF-IDF + SVM	93.6%	88%	50%	36%

VII. Conclusion and Future Work

The efforts done for this project is intended to understand, detect and classify toxic comments in online discussions in order to counter abuse and harassment online. Detecting different types of toxicity like threats, insults, obscenity, and identity-based hate is incredibly useful in ensuring that online discussions are more polite and respectful. Possible future scope of this work is to improve our algorithms by performing hyper-parameter tuning, trying more augmentation techniques such as back-translation which will also be helpful in performing toxic comment detection and classification on multiple and mixed languages. Another future research proposal could be analyzing the errors in mixed languages like idiosyncrasies and mitigating those to get better results.

References

1) Rastogi, Chetanya, Nikka Mofid, and Fang-I. Hsiao. "Can We Achieve More with Less? Exploring Data Augmentation for Toxic Comment Classification." arXiv preprint arXiv:2007.00875 (2020).
2) "A Machine Learning Approach to Comment Toxicity Classification" arXiv:1903.06765v1 [cs.CL]
3) Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. Convolutional Neural Networks for Toxic Comment Classification. In Proceedings of the 10th Hellenic Conference on Artificial Intelligence (SETN '18). Association for Computing Machinery, New York, NY, USA, Article 35, 1–6. DOI:https://doi.org/10.1145/3200947.32080
4) van Aken, Betty Risch, Julian Krestel, Ralf Löser, Alexander. (2018). Challenges for Toxic Comment Classification: An In-Depth Error Analysis. 10.18653/v1/W18-5105.
5) "Aggressive, Repetitive, Intentional, Visible, and Imbalanced: Refining Representations for Cyberbullying Classification" arXiv:2004.01820v1 [cs.SI]
6) Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level Convolutional Networks for Text Classification". In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 649–657. URL: http://dl.acm.org/citation.cfm?id=2969239.2969312.
7) M. Ibrahim, M. Torki and N. El-Makky, "Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 875-878, doi: 10.1109/ICMLA.2018.00141.
8) Jason W. Wei and Kai Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: CoRR abs/1901.11196 (2019). arXiv: 1901.11196. URL: http://arxiv.org/abs/1901.11196.
9) Davidson, Thomas, et al. "Automated hate speech detection and the problem of offensive language." Proceedings of the International AAAI Conference on Web and Social Media. Vol. 11. No. 1. 2017.

VIII. Supplementary

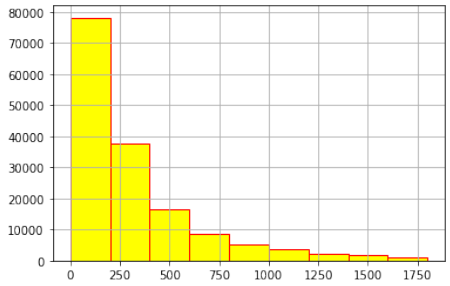

Fig. 3. Characters count in Wikipedia dataset

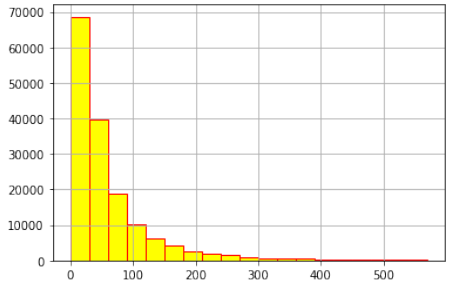

Fig. 4. Words count in Wikipedia dataset

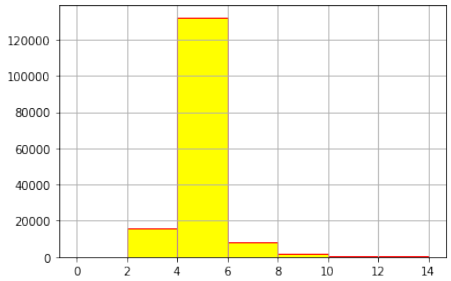

Fig. 5. Average word length in Wikipedia dataset

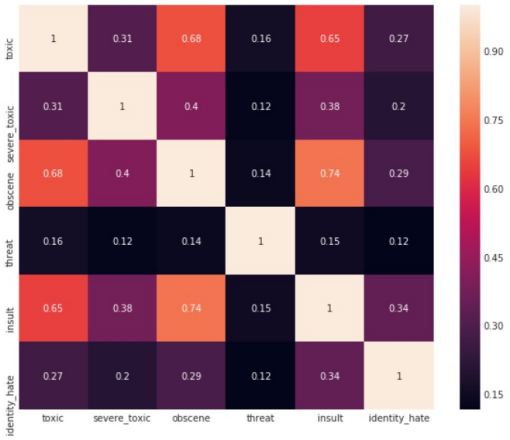

Fig. 6. Correlation plot for the dataset

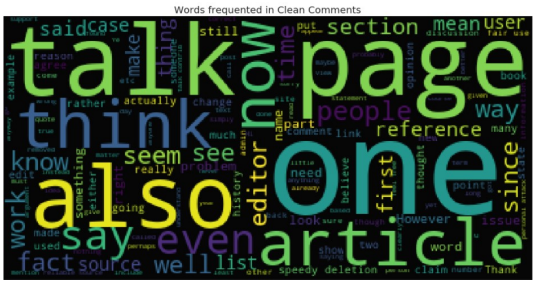

Fig. 7. Word cloud for clean data

Fig. 8. Word Cloud for toxic, severe-toxic, threat, insult comments

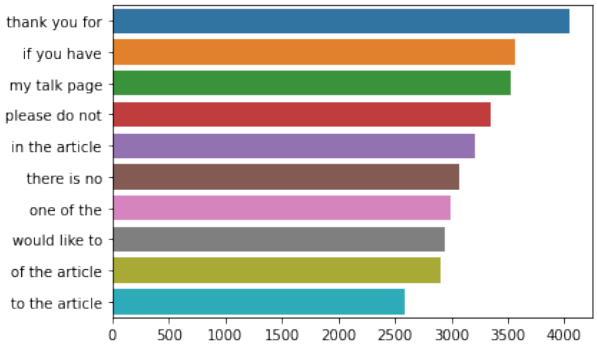

Fig. 9. Trigrams in dataset

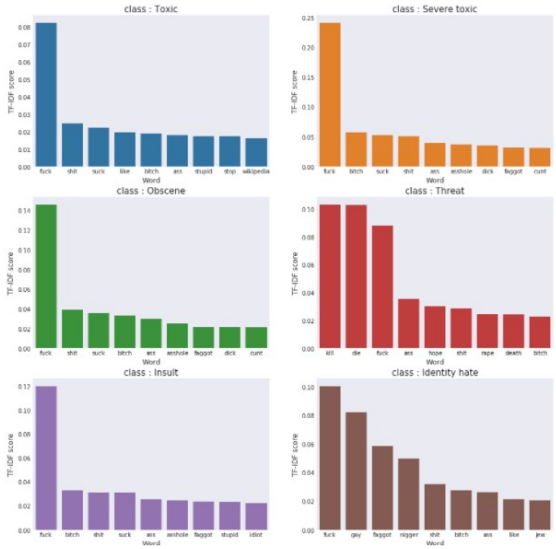

Fig. 10. TF-IDF vs Words relation for each class