Machine Learning For Model Driven Engineering: Toxic Comment Classification

Martina Nolletti*, Giordano Tinella†, Giacomo Sfratato‡
Department of Engineering and Computer Science and Mathematics, University of L'Aquila, Italy

February 2, 2024

Abstract

Social media users often encounter abuse, harassment, and insults from other users on most online communication platforms such as Facebook, Instagram, YouTube, etc., due to which many users stop expressing their ideas and opinions. The solution to this problem is to create an effective model that can identify the level of toxicity of comments, such as threats, obscenities, insults, racism, etc. [8], thus promoting a peaceful environment for online dialogue. In this article, we will better understand the multi-label classification of toxic comments and create a model to classify comments into different toxicity labels. The multi-label classification [7] is a supervised machine learning approach where a single instance can be associated with multiple labels simultaneously. It allows the model to assign zero, one, or more labels to each data sample based on its characteristics. In the context of toxic comment classification, a comment or text can be labeled with multiple toxicity categories if it contains various forms of harmful language.

*martina.nolletti@univaq.it
†giordano.tinella@univaq.it
‡giacomo.sfratato@univaq.it

1 Introduction

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.
The Conversation AI team, a research initiative founded by Jigsaw and Google (a part of Alphabet) is working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments [4] (i.e. comments that are rude, disrespectful, or otherwise likely to make someone leave a discussion). So far they have built a range of publicly available models served through the Perspective API, including toxicity. But the current models still make errors, and they don't allow users to select which types of toxicity they're interested in finding (e.g. some platforms may be fine with profanity, but not with other types of toxic content).
A competition was set up where users had to build a multi-headed model that could detect different types of toxicity, such as threats, obscenities, insults, and identity-based hatred, better than current Perspective models. A dataset of comments from changes to the Wikipedia talk page [6] was available.
Improvements to the current model will hopefully help online discussions become more productive and respectful.
The paper is structured as follows: Section 2 explains the dataset considered, Section 3 defines the machine learning techniques, Section 4 defines the k-fold cross-validation, Section 5 explains the results obtained and finally Section 6 defines the conclusions of our paper.

2 Dataset Description

A large number of Wikipedia comments are provided that have been labeled by human evaluators for their toxic behavior. The types of toxicity are:

- **toxic**, comments displaying negativity or hostility, but not reaching extreme levels;
- **severe_toxic**, comments with highly aggressive or harmful content, surpassing typical toxic behavior;
- **obscene**, comments containing offensive or vulgar language, gestures, or imagery;
- **threatening**, comments conveying explicit threats or intentions of harm towards individuals or groups;
- **insult**, comments containing disrespectful or demeaning language directed at others;
- **hateful_identity**, comments displaying discrimination or prejudice based on identity factors such as race, gender, religion, etc.

2.1 Multi-label Classification

Multi-label classification involves predicting zero or more class labels. Unlike normal classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or "labels." Deep learning neural networks are an example of an algorithm natively supporting multi-label classification problems. Neural network models for multi-label classification tasks can be easily defined and evaluated using the Keras [3] deep learning library. Classification is a predictive modeling problem that involves outputting a class label given some input. It is different from regression tasks that involve predicting a numeric value. Typically, a classification task involves predicting a single label. Alternately, it might involve predicting the likelihood across two or more class labels. In these cases, the classes are mutually exclusive, meaning the classification task assumes that the input belongs to one class only. Some classification tasks require predicting more than one class label. This means that class labels or class membership are not mutually exclusive. These tasks are referred to as multiple-label classification or multi-label classification for short. In multi-label classification, zero or more labels are required as output for each input sample, and the outputs are required simultaneously. The assumption is that the output labels are a function of the inputs.

2.2 Dataset Analysis

Once the necessary libraries, including TensorFlow, Pandas, and scikit-learn, have been imported to facilitate data processing, modeling, and evaluation, the Google Colab drive is mounted to access and store datasets and other files. Next, a CSV dataset containing toxic comments and their labels is loaded and a **clean_text** function is defined to clean the comment text. The cleaning process involves converting text to lowercase, handling contractions, removing non-alphabetic characters, and eliminating stop words. In addition, a mapping of contractions in the English language with their expanded forms is created. This mapping is used during the text-cleaning process.

The NLTK library is used to download a set of stop words, which are then removed from the comment text during the cleaning process, and TensorFlow's **TextVectorization** to convert the cleaned text into numerical vectors suitable for machine learning models. Next, the feature matrix $\mathbf{X}$ containing the cleaned comment text and the target matrix $\mathbf{y}$ containing the associated labels are prepared. Once the dataset has been cleaned, the distribution of toxicity classes is analysed. A horizontal bar graph displays the frequency of the different toxicity labels, providing a clear overview of the prevalence of each class. This step is crucial for understanding the composition of the dataset and potential imbalances between classes.

Figure 1 focuses on analysing the distribution of labels in our dataset. Initially, the labels are identified within the DataFrame and the occurrence count of each label is calculated.

Labels	N° Occurrences
threat	478
identity_hate	1405
sever_toxic	1595
insult	7877
obscene	8449
toxic	15294

Table 1: Exact values for each class

The resulting graph shows the number of occurrences on the horizontal axis and the class labels on the vertical axis. To improve understanding of the graph, labels are added to the x-axis and y-axis, specifying the number of occurrences and class labels, respectively.

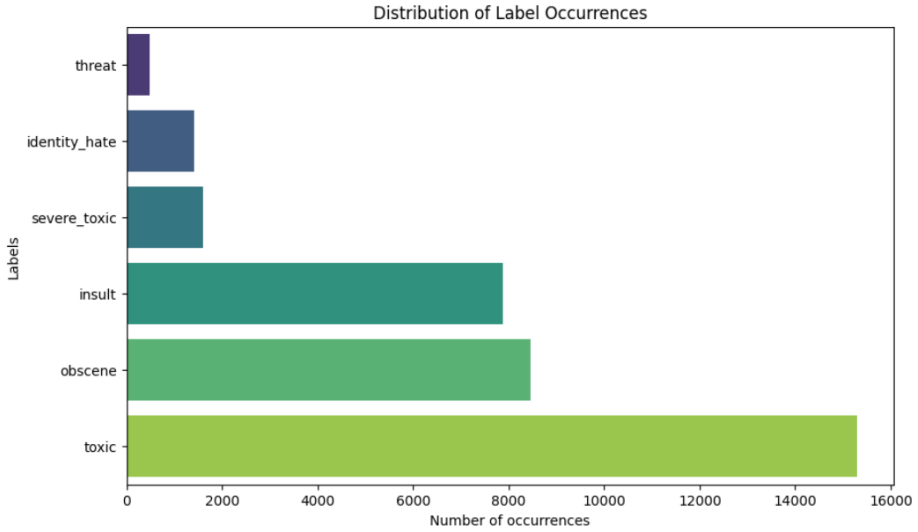

Figure 1: Distribution of Label Occurrences

In Table1, the exact values for each class are defined, and sorted according to the number of occurrences. This provides a detailed view of the distribution of the labels.
Both define a comprehensive and clear approach to analysing the distribution of the labels in the dataset, enabling a quick and thorough understanding of the relative frequencies of the different classes.
Figure 2 focuses on analysing the word frequency of our dataset represented by the variable X, which contains comments or text. Initially, a list of words is created through list comprehension that iterates over each comment in X and divides the comment into words. This process helps construct a complete list of

all the words in the comments. Next, the frequency of each word is calculated, and the graph representing the top 10,000 most frequent words is displayed. The vertical bars represent the frequency of the words, indicating the differences in frequency distribution.

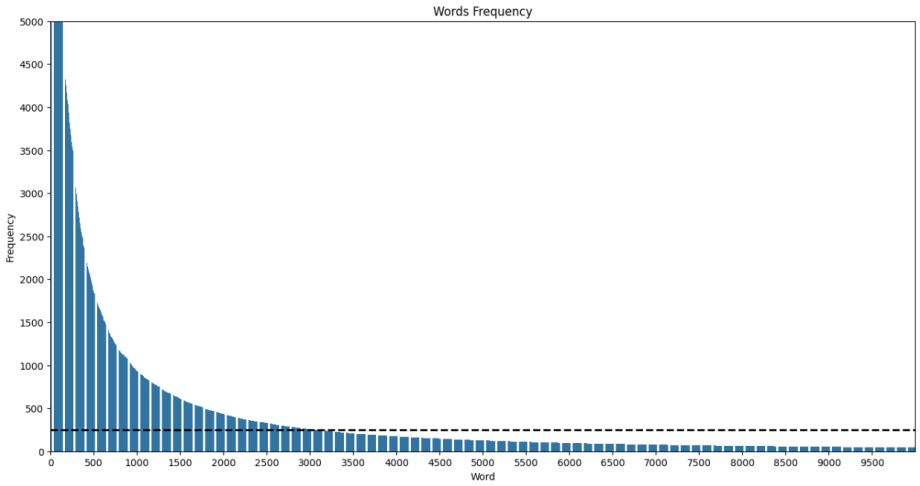

Figure 2: Frequency of words

Figure 3 aims to analyse the distribution of the length of the comments in the dataset represented by the variable X. The length of each comment is calculated using list comprehension. The length of a comment is defined as the number of words it contains, obtained by dividing the comment into words and calculating the length of the result. Subsequently, a histogram is created to display the distribution of comment lengths. This analysis can be useful to understand the variety of lengths of texts in the corpus and may influence pre-processing or subsequent analysis choices.

3 Machine Learning Techniques

Machine Learning [9] is an Artificial Intelligence technique that teaches computers to learn from experience. Machine Learning algorithms use computational methods to 'learn' information directly from data, without needing a predetermined equation as a model.
Algorithms improve their performance adaptively as the number of samples available for learning increases. Deep Learning is a specialized form of Machine Learning. Machine Learning uses two types of techniques [5]: supervised learning, which trains a model based on known input and output data in such a way as to predict future outputs, and unsupervised learning, which finds hidden

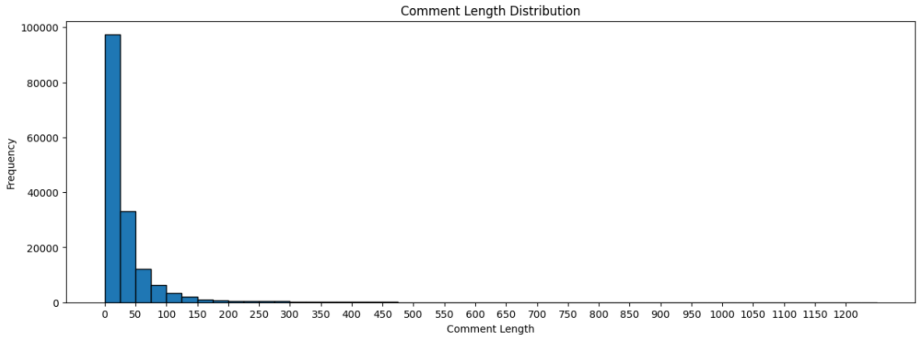

Figure 3: Distribution of Comment Length

patterns or intrinsic structures in the input data.
Deep learning [2] is a subset of machine learning (ML), in which artificial neural network algorithms are modeled to function like the human brain, learning from large amounts of data.
Deep learning is based on layers of neural networks that are algorithms modeled loosely on the way human brains work. Training with large amounts of data is what configures the neurons in the neural network. The result is a deep learning model that, once trained, processes new data.
Deep learning models acquire information from multiple data sources and analyze this data in real-time, without the need for human intervention. In deep learning, graphics processing units (GPUs) are optimized for training models because they can process multiple calculations simultaneously.

3.1 Implementation

To address the problem of toxicity in online conversations, a customized neural network was created to classify toxicity. The process starts by uncovering the essence of the language in the dataset through text vectorization. Here, each word is transformed into a numerical representation, allowing the model to understand textual nuances.
The **TextVectorization** layer of TensorFlow, with parameters defined as **MAX_FEATURES** and **MAX_SEQUENCE**, plays a key role in this conversion. The dataset is then divided into training and test sets to facilitate the evaluation of the model.
The choice of MAX_FEATURES = 3000 was made based on Figure 2 where the number of most frequent words is given, while the choice MAX_SEQUENCE = 100 was made based on Figure 3 according to the length of the comments. The result will be a dataset that has 100 columns and the value of each column can vary from 0 to 3000.
To capture sequential dynamics in language, a bi-directional short-term memory **LSTM** layer is introduced. The latter makes it possible to capture dependen-

cies within the text. Next come two Dense layers, which further unleash the neural power of the model.
These layers play a crucial role in learning complex patterns and representations. Finally, there is a last Dense layer, designed for multi-label classification. Its role is to decode toxicity by predicting the probability of each toxicity class independently.
The neural network is then compiled with the meticulous selection of loss functions, evaluation metrics, and optimizers. The binary cross-entropy loss, accuracy as an evaluation metric, and the optimizer Adam collectively forge a path to a model capable of discerning toxic from non-toxic content.

4 K-Fold Cross Validation

Cross validation [1] is an evaluation method used in machine learning to find out how well your machine learning model can predict the outcome of unseen data. It is a method that is easy to comprehend, works well for a limited data sample, and also offers a less biased evaluation, making it a popular choice. The data sample is split into a 'k' number of smaller samples, hence the name: K-fold Cross Validation. You may also hear terms like four-fold cross-validation, or ten-fold cross-validation, which essentially means that the sample data is being split into four or ten smaller samples respectively.

4.1 Our implementation

K-Fold Cross Validation is an effective method for evaluating the performance of neural network models on moderate to small datasets. This approach reduces the risk of overfitting and provides a more reliable assessment of overall model performance.
Initially, the parameters for K-Fold Cross Validation are configured by determining the number of folds and specifying the randomness of data splitting. A seed ensures reproducibility.
Subsequently, a loop iterates through the generated folds. During each iteration, the data is split into training and validation sets. The model is then trained on each fold with a specified number of epochs and batch size.
After training each fold, the model is evaluated on the validation set, and loss and accuracy results are printed. This provides a detailed performance evaluation on each fold.
Next, the model is trained on the entire training set, ensuring that it can benefit from the use of all available data. Finally, a summary of the model is printed, providing information on its architecture and number of parameters. This approach ensures a more reliable assessment of performance by improving the model's ability to generalize to unobserved data.

Metrics	Value
Precision	0.65
Recall	0.65
Accuracy	0.99
Hamming Loss	0.03
F1-Score	0.65

Table 2: Value of the Metrics

5 Results

Precision, Recall, Accuracy, Hamming Loss, and F1-Score are used to evaluate model performance.
Next, the model is used to generate predictions on the test data set (X_test). The continuous predictions are then binarised, assigning the value 1 to predictions greater than 0.5 and the value 0 to predictions less than or equal to 0.5.
Next, metrics are calculated using the binarised predictions and true class values from the test dataset. The calculated metrics include Precision, Recall, and Accuracy by updating the state of the metrics and calculating the resulting values. The values of the calculated metrics are printed out, providing a quantitative assessment of the model's performance (Table 2). The metrics provide insight into key aspects such as the accuracy of positive predictions, the ability to correctly recover positive instances, and the overall accuracy of the model. The Hamming Loss and F-1 Score provide further information on the overall quality of the model's predictions.
Finally, to construct the confusion matrices, a loop iterates over a list of classification labels. For each label, the corresponding column is extracted from the test data and model predictions.
Next, a confusion matrix is created using the scikit-learn library. The confusion matrix is visualized as a heatmap, where numerical annotations provide information on the distribution of the predictions.

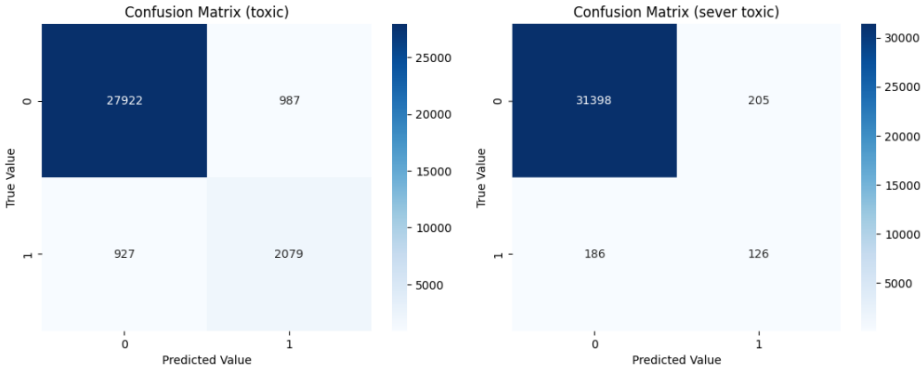

Figure 4: Confusion matrices for toxic and sever toxic

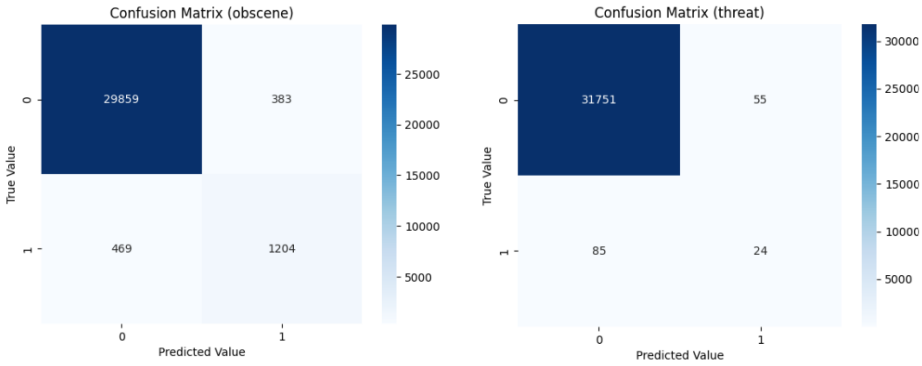

Figure 5: Confusion matrices for obscene and threat

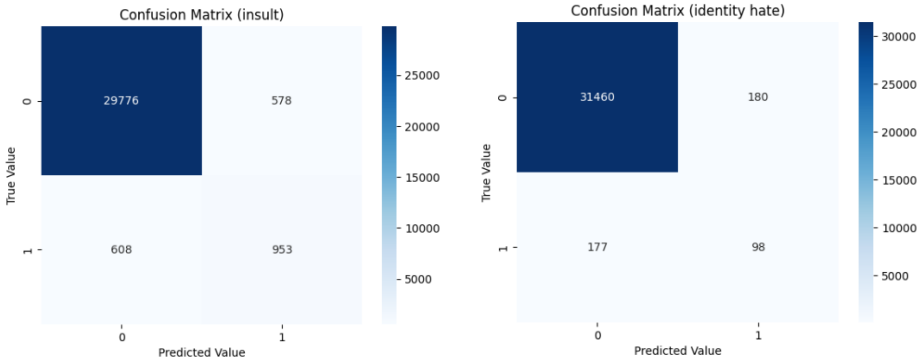

Figure 6: Confusion matrices for insult and identity hate

The x and y axes of the heatmap represent the predicted and actual values, respectively. This visual representation is useful for understanding how the model classifies the different categories, highlighting misclassifications and successes.
The confusion matrices of the labels taken into consideration are shown in the Figures 4, 5, 6.

6 Conclusion

The conclusions of our study reflect a comprehensive analysis of a toxic text multiclass classification problem. Through implementing a comprehensive process, we explored the data, built a machine-learning model, and evaluated its performance.
First, we performed data preprocessing, including cleaning the texts and converting them into numerical vectors. This allowed us to prepare the data for input to the machine learning model.

Next, we built a sequential neural network model including layers such as Embedding, Bidirectional LSTM, and Dense to address multiclass classification. The model was trained using the entire training set and evaluated using several evaluation metrics, including Precision, Recall, Accuracy, Hamming Loss, and F1 Score.
The model's performance was evaluated through a k-fold cross-validation, which confirmed its robustness and generalization ability.
The confusion matrices for each class showed the model's ability to classify different categories of toxic text correctly. This was further confirmed by the evaluation metrics, which indicated a good level of precision, recall, and overall accuracy of the model.
In addition, we wanted to play with the created model by loading a pre-trained model for the classification of toxicity and the corresponding text vectorization level. The text vectorization level is configured and initialized using information stored in a pickle file. This level is crucial for transforming raw text input into a format suitable for the model.
After configuration, there is a 'pretty_prediction' function designed to take a user-supplied comment, vectorize it using the loaded text vectorization layer, and then make predictions about its toxicity using the pre-trained model. The results are presented in a human-readable format, indicating the categories that the model assigns to the input comment.
To make the interaction easier, there is a loop in which the user can enter comments interactively. The loop continues until the user explicitly types 'stop', allowing them to exit the interaction. During this process, the console is cleared to maintain an orderly display.

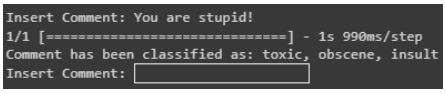

Figure 7: Example of classification of a comment

Figure 7 shows an example of the classification of a comment entered by a user. In essence, an interactive session was created with a toxicity classification model, allowing users to enter comments and observe the model's predictions in a clear and readable manner.
In conclusion, our study provides a robust and comprehensive approach to address the problem of toxic text by analyzing and applying machine learning techniques. We hope that this work can contribute to the understanding and management of online abuse phenomena.

References

[1] Berrar, D., et al.: Cross-validation. (2019)

[2] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)

[3] Moolayil, J., Moolayil, J.: An introduction to deep learning and keras. Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python pp. 1–16 (2019)

[4] Risch, J.: Toxicity. 86272 **12**, 219–230 (2023)

[5] Singh, Y., Bhatia, P.K., Sangwan, O.: A review of studies on machine learning techniques. International Journal of Computer Science and Security **1**(1), 70–84 (2007)

[6] Smirnov, I., Oprea, C., Strohmaier, M.: Toxic comments are associated with reduced activity of volunteer editors on wikipedia. PNAS nexus **2**(12), pgad385 (2023)

[7] Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining (IJDWM) **3**(3), 1–13 (2007)

[8] Zaheri, S., Leath, J., Stroud, D.: Toxic comment classification. SMU Data Science Review **3**(1), 13 (2020)

[9] Zhou, Z.H.: Machine learning. Springer Nature (2021)