Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

# Multilabel toxic comment classification

*Authors:*

Christian Bernasconi - 816423 -
c.bernasconi11@campus.unimib.it

Marco Ripamonti - 806785 -
m.ripamonti@campus.unimib.it

25/01/2021


UNIVERSITA' DEGLI STUDI DI MILANO
BICOCCA

**Abstract**

In the era of social medias online harassment has become a serious issue and new technologies needs to be implemented to keep the Internet a safer and healthier place for everyone. In this work different deep learning techniques and models have been applied for a multilabel classification on an imbalanced dataset to classify different types of toxicities. A two phases approach, which combines a binary and a multilabel classification, has been adopted to solve the task at hand. Multiple models involving CNN, LSTM, GRU and BERT have been evaluated and compared for a final evaluation and discussion of the strategy implemented.

# 1 Introduction

With the advent of social medias, public forums, comment sections and online chats the amount of toxic, harmful and unsafe messages has increased. Many people daily struggle with abuse and harassment online and new and more robust techniques are required to correctly identify and stop those malicious behaviours. In this work deep learning technologies have been applied to not only identify toxicity in messages but also to identify to which categories of toxicity a message belongs to. Threat, insult, identity hate all are different types of toxicity which need to be taken care of with the correct sanction. The dataset used was provided by a Kaggle challenge[1] proposed in 2017, This challenge has been chosen for two main reasons: applying deep learning to natural language has been a topic of interest and also a contribution to make the *Internet* a better place for everyone can be given.

The approach adopted in this work consists mainly on the use of different types of deep learning architectures applied to natural language for multilabel text classification: *Recurrent Neural Networks*, *Convolutional Neural Networks* and *Transformers* based architectures.
Instead of building a single model, the approach undertaken splits up in two main phases. Given the composition of the dataset, which will be presented in section 2, two models have been built. A first one to correctly separate toxic comments from non-toxic comments and the second one to correctly identify

[1]Kaggle challenge: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

the categories to which a toxic comment belongs to. This solution has been chosen to provide a better identification of toxicity given the strongly imbalanced nature of the dataset.
Before going into the details of the approach some of the main concepts and theory used in the whole process are given.

### Word Embeddings

*Word Embeddings* encapsulate large sparse vectors of words into a lower dimensional space that maintains the semantic relationships between them. To map high dimensional data into a lower dimensional space multiple techniques have been developed and used. One of the most common is *Word2Vec*. Word2Vec models are known as *Continuous Bag of Words (CBOW)* and *Skipgram* [1]. In this work multiple pretrained embeddings have been used like *GloVe*[2], *fastText*[3] and a transformer encoder based language model with *BERT*[4].

### Convolutional Neural Network

Convolutional Neural Networks are deep learning architecture which have been and are still used in the state of the art for image classification. This type of architecture allows a hierarchical features extraction using convolutional filters combined with dense layers to learn the classification of those features.
Even though the main focus of the CNN is to capture spatial and temporal dependencies in an image, some attempts have been proposed to extract features and temporal dependencies from vectors representing sentences. As described by Yoon Kim CNNs can be used to achieve great results in sentence classification. He also presents a variant of CNN which uses multiple filters of different sizes on the same input which outperforms standard CNNs performances [2]. This architecture has also been used in this work.

### Recurrent Neural Networks

Recurrent Neural Networks have the peculiar characteristic that theirs decisions are influenced by what they have learnt from the past. RNNs training

[2]GloVe Stanford: https://nlp.stanford.edu/projects/glove/
[3]fastText Facebook: https://fasttext.cc/
[4]DistilBert: https://huggingface.co/distilbert-base-uncased

works a lot like traditional fully connected networks, but in addition, they remember information learnt from previous inputs while they produce outputs. This behaviour makes them really useful in analysis of text sequences where every word is temporal dependent from previous words. Of course those architectures can be made deeper. Some state of the art deep learning RNNs architectures are the so called gated RNNs: LSTM and GRU. Those use gates to efficiently "forget" or carry on information learnt from the past during training. In this work both of these architectures will be used.

#### Transformers

A transformer, like LSTMs and GRUs, is a sequence to sequence architecture which is a network that transforms a sequence of elements into another sequence *(e.g. machine translation)*. An important concept involved in a transformer is the attention mechanism: given an input sequence, for each item of the sequence decides which other parts of the sequence are important to it. A transformer is mainly composed by two components: an encoder and a decoder[3]. BERT is a transformer architecture that makes use of just the encoder part of the original transformer. BERT reads an entire sequence of words at once and learns to map a word by its context by looking at its surroundings words.
In this work transfer learning has been applied to a distilled version of BERT (*i.e., DistilBert*) which uses only 40% of parameters and runs 60% faster than BERT while preserving over 95% of its performances [4].

# 2 Datasets

## 2.1 Data exploration

The dataset provided by the challenge is very basic. It's composed by only one feature that corresponds to the text written by a user in the comment sections of Wikipedia edit pages. The language of the dataset is english. Starting from this feature, a comment could be classified with zero or more labels identifying a type of toxicity: *toxic, severe toxic, obscene, threat, insult* and *identity hate*. The number of instances contained in the training set is about 160K.

By exploring the data, the first considerations could be done on the content

of the comments. First of all, it has been noticed that the text contained was really raw and needed to be preprocessed before using it for the classification task. The detailed preprocessing strategy will be presented in the section 2.2.
Then to see how the words contained in a comment discriminates between categories a search of the most common words has been performed. The top words (that will not be reported due to their vulgarity) have shown that there is a big difference between the two macro categories of toxic and non-toxic comments. Also between the different toxicity types there are representative words.

The second considerations that have emerged during the exploration phase are related to the targets distributions. The training data contains a very imbalanced set of labels. The imbalance was present both in terms of toxic / non-toxic macro categories and between the labels of specific toxicity types. The figure 1 shows the high imbalance between toxic and non-toxic comments. In fact, the amount of toxic comments is only about the 11% of the training data. The figure 2, instead, shows the distributions between the labels. As you can see, while the *toxic* label is the dominant, the *severe toxic, threat* and *identity hate* labels have a very low number of instances if compared to the others. In particular, it has been found out that most of the minority labels belongs to comments that are also labeled has *toxic* (*e.g., severe toxic comments never appear alone*). The non-mutuality of these labels did not make possible to perform an effective data balancing for the poorest categories. To limit these imbalance problems, a data augmentation has been performed in order to have a richer training set in term of toxic comments. The details of the data augmentation will be discussed in the section 2.3.

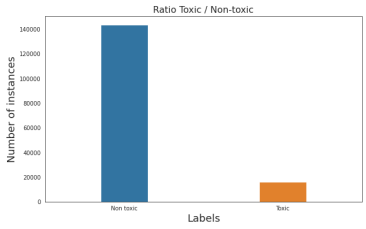


Figure 1: Toxic and non-toxic

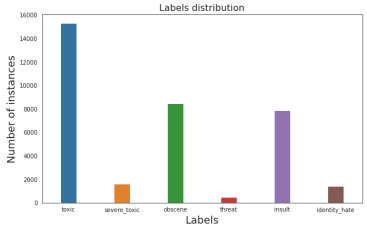


Figure 2: Toxic labels

## 2.2 Data preprocessing

The preprocessing of the data has been done according to the embeddings used in the classification models. The implemented strategy involves a customizable preprocess for transforming text in lower case, removing char repetitions, replacing contracted forms, removing stopwords, replacing emojis with the corresponding words, correcting misspelled bad words and keeping specific punctuation.
Once the data has been preprocessed, a vocabulary for the embeddings has been created with the words of the cleaned comments.

For the BERT model a sentence is preprocessed using its standard methodology[5], which also creates sub words for out of vocabulary words.

## 2.3 Data augmentation

In order to augment the amount of toxic examples in the dataset, two main strategies have been taken into account.
The first strategy was the one based on **back translations**. With this strategy each toxic comment has been oversampled with the text obtained translating it to another language and then translating it back to english. This strategy has permitted to create different examples with new words without losing the real meaning of the comments. The used back translations were in French, Spanish and German[6].

The second strategy consisted on an **augmentation based on synonyms**. In this case it was possible to create a new example by replacing the words in a comment with their synonyms according to a thesaurus. The source used to get words synonyms is *WordNet*[7]. The implemented method allowed to generate a comment by substituting every adjective, verb and noun of each sentence with a synonym randomly extracted from a pool of synonyms used in the same part of speech. The toxic examples have been tripled using this strategy. Compared to the previous one, this approach can introduce a higher bias into the data because more words are altered during the process.

[5]https://huggingface.co/transformers/preprocessing.html
[6]https://github.com/PavelOstyakov/toxic/tree/master/tools
[7]https://wordnet.princeton.edu/

# 3 The Methodological Approach

The main idea of the approach adopted for this multilabel classification problem consists on splitting the task in two phases to better handle the imbalance of the dataset. As shown in figure 3 a first model is trained to discriminate toxic comments from non-toxic comments and a second model is trained to identify the toxicity types of the toxic comments coming from the first classification. Finally, the results of the two classifiers are then merged.

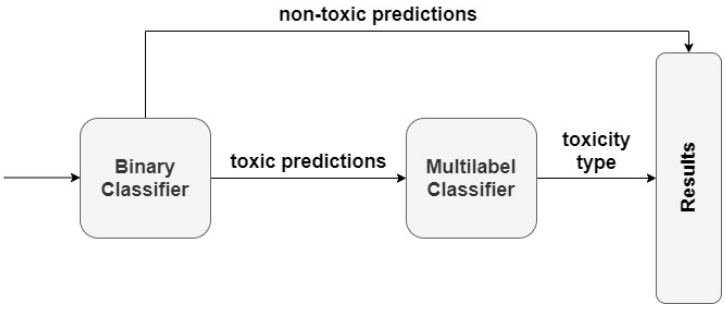


Figure 3: Pipeline classification task

The initial part of this work tried to solve the high imbalance towards the non-toxic examples and between toxic labels. Using the back translations as strategy for data augmentation achieved the best results. To address the toxicity types imbalance it has been tried to use different class weights for giving more relevance to minority labels during the training, but due to an open issue[8] for multilabel usage of the used library this strategy has been shelved.

Following the data augmentation, a preprocessing phase has been performed. The configuration that led to the best results was the one that involved all implemented steps previously described, except the emojis replacement and the bad words correction. For the punctuation it has been kept only the *'!'* symbol which seemed to help the classification.
Then a vocabulary for the embeddings has been created, which is constituted by the top 20K most common words out of almost 150K distinct words contained in the training set. Since the number of words for each comment was

[8]https://github.com/keras-team/keras/issues/8011

really variable, each text has been truncated at the length of 150 words with a bottom padding for those comments of lower length. To build an embedding the pretrained vectors of FastText have been extracted corresponding to those of the words in the vocabulary. FastText has been chosen over the GloVe pretrained embedding because it gave better results.

After having defined a solid embedding, multiple models have been studied for the training process of the actual classification task. The models defined include a bidirectional LSTM based model, a bidirectional GRU based model and a CNN based model. The structure is shared between all three models and between binary and multilabel classifiers and shown in figures 4 and 5. It's important to notice that in some cases (the most in the multilabel classification), the combination of both a Global Max Pooling[9] and a Global Average pooling, applied to the output of LSTM or GRU layers, boosted the performances of the models. Dropout and early stopping have been applied during training as regularization techniques to prevent overfitting.
In addition to the fact that the binary classifiers were trained on the whole training set and the multilabel classifiers were trained only on toxic comments, the main difference between the two is the output layer where respectively are present one and six units. For both models the output function is a sigmoid and a binary cross entropy is used as loss function, which is minimized with the adam optimizer and 0,001 as learning rate. While in the first model the output represents the probability of a comment being toxic, in the second one each output unit maps the probability of a specific toxicity type.

A solution using an embedding layer generated by a distilled version of BERT, called DistilBERT, has been explored. With a very similar architecture to the one shown in figure 4 the embedding layer is replaced by the DistilBERT one.

For training the models 75% of data have been used as training, with the remaining 25% used as validation set. Due to the data augmentation with back translation, an important aspect to notice is that it's been important to pay attention on which data goes into the training set and validation set. Having the original data and generated data in different sets may lead to overfitting. This is because examples of the two sets could be very similar

[9]A global pooling layer is an ordinary pooling layer with pool size equals to the size of the input.

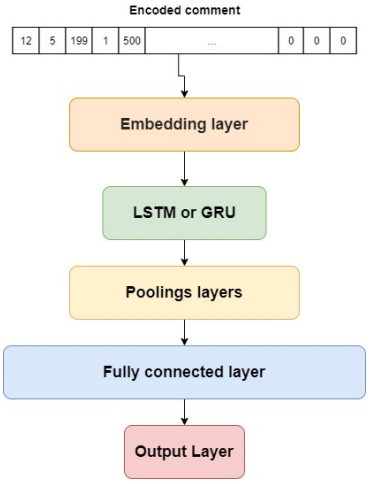


Figure 4: LSTM - GRU models

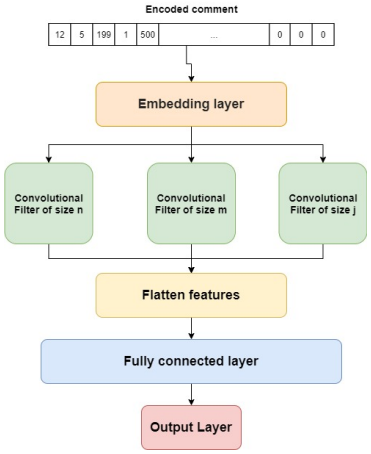


Figure 5: CNN models

and even if, during training, the validation loss continues decreasing it may hide an overfitting problem. For this reason the data split has been forced to keep the back translated comments in the same set of the original ones from which they have been generated. The table 1 shows the training statistics[10] of each model.

Table 1: Training stats of the models for the two classification phases

| **Model** | **Phase** | **# epochs** | **Time** | **Validation Loss** |
|---|---|---|---|---|
| LSTM | binary | 13 | 9m | 0.0940 |
| GRU | binary | 13 | 20m | 0.1032 |
| CNN | binary | 19 | 8m | 0.1001 |
| BERT | binary | 20 | 2h | 0.1115 |
| LSTM | multilabel | 50 | 36m | 0.0713 |
| GRU | multilabel | 42 | 28m | 0.0846 |
| CNN | multilabel | 34 | 4m | 0.0720 |
| BERT | multilabel | 43 | 3h | 0.1131 |

[10]please notice that all models were trained using Google Colab sessions with GPUs runtime, so different models could have been trained on different machines

# 4 Results and Evaluation

The results achieved with the different models are shown in terms of accuracy[11], precision, recall and F1-score using a macro average. In the next sections will be reported respectively the performance of the binary classifier, the multilabel classifier and the final combined classifier. Please notice that the results of the multilabel classifier have been obtained on the subset of examples constituted by only toxic comments of the test set. This because in this way it has been possible comparing the performance between the models without any bias originated by the binary classifier of the first phase (*i.e., avoid including false positives and excluding false negatives*).

The performance of the binary and multilabel classifiers are shown in figures 6 and 7.
The final classifier results have been obtained by chaining the two models based on LSTM, which achieved the highest F1-score during each phase. In figure 8 are shown the performance measures. The final confusion matrices are illustrated in the table 2.

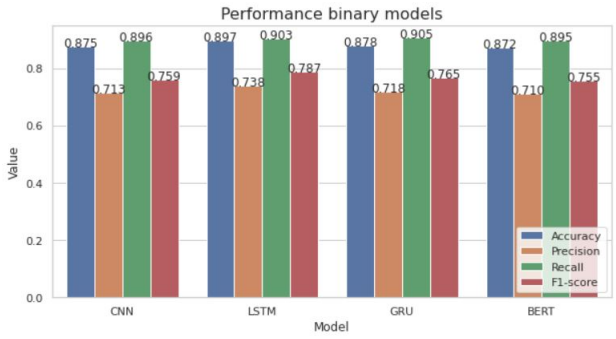


Figure 6: Performance of the binary classifier

[11]the accuracy considers true positive an example if it has all the 6 labels classified correctly

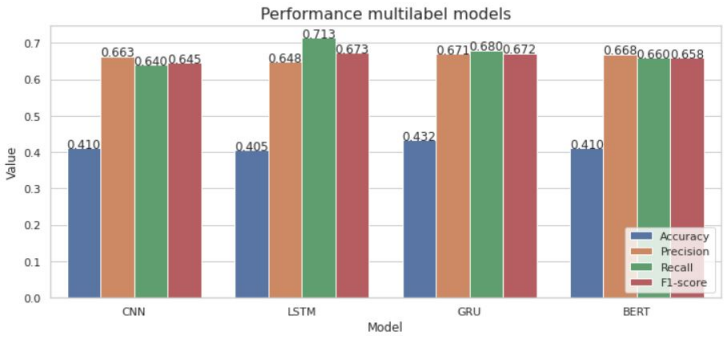


Figure 7: Performance of the multilabel classifier

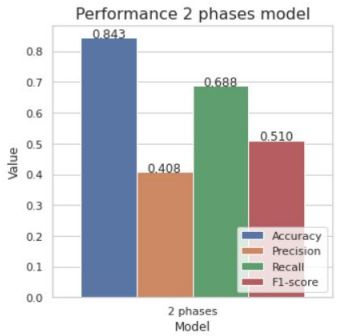


Figure 8: Performance of the final classifier

Table 2: Confusion matrices for the final two phases classification.
*N.B.: for each matrix the element in position (0, 0) indicates true negatives, (0,1) false positives, (1,0) false negatives and (1,1) true positives.*

| **Toxic** | | **Severe toxic** | | **Obscene** | | **Threat** | | **Insult** | | **Identity hate** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 51536 | 5892 | 62559 | 591 | 56648 | 3179 | 63082 | 224 | 57296 | 2795 | 62220 | 585 |
| 604 | 5485 | 168 | 199 | 619 | 3071 | 101 | 110 | 896 | 2530 | 292 | 420 |

# 5 Discussion

The kaggle challenge of this dataset proposed an evaluation based on the ROC AUC score. However, this measure was too optimistic due to the highly

imbalanced nature of the data, so high values would be reachable even prediting all comments as non-toxic. In this work the F1-score has been chosen as the main evaluation measure, computed with a macro average to give the same importance to the minority classes.
Starting from the binary classifiers here are some considerations about the results. Looking at the performance graphs you can see that the model that reached the best results is the LSTM with an F1-score of 0,787 . For every model it's noticeable that there is a great amount of non-toxic examples classified as toxic. By the way, analyzing those false positive comments you can say that lots of them contain profanities or words commonly used in toxic comments and many others have been wrongly labeled.
The LSTM model was the one that performed the best also for the multilabel classifier, with an F1-score of 0,673. In this case you can notice from the confusion matrices that all the models had the major difficulties in classifying correctly the examples belonging to the minority labels.
The results obtained by chaining the two LSTM based models are significantly lower. The F1-score was dragged down to 0,510 due to the high number of false positives passed through the binary classifier, as you can see by the very low precision value.

From these results it's possible to highlight two key points of the proposed approach that could make it more suitable for this task rather than a unique model. The first aspect is that the split of the prediction in two phases allows using different discriminating thresholds for the binary classifier and the multilabel classifier. To compare results a standard 0,5 threshold has been used for both the two phases, but depending on the goal it's possible to change the threshold only for the first classifier to filter in different ways the input of the second classifier. The second aspect is that having two independent model make possible to train, specialize and optimize each of them for its specific task.

There are different improvements that could be done on this approach. A relevant boost to the performance of the model may be achieved by improving the first model in order to reduce the amount of false positive predictions. Another operation that would lead to a better multilabel training and better results could be augmenting only the minority class examples using some multilabel balancing techniques, assigning different weights to the labels, or simply enriching the dataset with new specific examples. Finally, a last im-

provement that could bring better results is a more accurate train and tune of the BERT based model, that achieved quite similar results even with a really poor tuning of the parameters. The training of this model was really expensive and so it has not been possible to find an optimal configuration.

As future works it would be interesting to use ensemble techniques that involve model of different nature. For example a possibility is to combine this approach with models that use TF-IDF for representing the text, or models that uses features extracted from the text (*e.g., number of profanities, number of uppercased words, length, etc.*).

# 6 Conclusions

In this work has been shown that a two phases approach can be used to handle this multilabel classification task and how having distinct models for each one of the two phases an optimal threshold can be chosen with respect to the user needs.
The two main problems that have emerged are the high amount of false positive predictions of the binary classifier and the difficulties to correctly predict the minority labels for the multilabel classifier.
Among different configurations the models who reached better results were the ones based on the use of LSTMs. However, the high labels' imbalance didn't allow to achieve optimal performances, so working in the direction of balancing data could lead to greater effectiveness. Other several future works could be done by taking into account new text features and ensembling different types of models.

# References


[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[4] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.