applied sciences

MDPI

Article

Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification

Viera Maslej-Krešňáková, Martin Sarnovský *, Peter Butka and Kristína Machová

Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, 040 01 Kosice, Slovakia; viera.maslej.kresnakova@tuke.sk (V.M.-K.); peter.butka@tuke.sk (P.B.); kristina.machova@tuke.sk (K.M.)

* Correspondence: martin.sarnovsky@tuke.sk

Received: 16 October 2020; Accepted: 27 November 2020; Published: 2 December 2020

Abstract: The emergence of anti-social behaviour in online environments presents a serious issue in today's society. Automatic detection and identification of such behaviour are becoming increasingly important. Modern machine learning and natural language processing methods can provide effective tools to detect different types of anti-social behaviour from the pieces of text. In this work, we present a comparison of various deep learning models used to identify the toxic comments in the Internet discussions. Our main goal was to explore the effect of the data preparation on the model performance. As we worked with the assumption that the use of traditional pre-processing methods may lead to the loss of characteristic traits, specific for toxic content, we compared several popular deep learning and transformer language models. We aimed to analyze the influence of different pre-processing techniques and text representations including standard TF-IDF, pre-trained word embeddings and also explored currently popular transformer models. Experiments were performed on the dataset from the Kaggle Toxic Comment Classification competition, and the best performing model was compared with the similar approaches using standard metrics used in data analysis.

Keywords: natural language processing; toxic comments; classification; deep learning; neural networks

1. Introduction

Nowadays, the World Wide Web is an environment where the users can create and share the information with almost minimal restrictions. The majority of the users use the web responsibly and effectively. However, there is a group of users, which act with the type of behaviour, that could be described as anti-social. Numerous definitions of the anti-social behaviour currently exist [1], but there are two major types of such behaviour present:

- Misinformation spreading—this type of actions usually include creation and sharing of misleading content in various forms, e.g., hoaxes, fake or biased news, fake reviews, etc.
- User reactions—this type of behaviour usually occurs in user conversations and has many different forms, e.g., discussion manipulation, cyber-bullying, hate speech, trolling, spamming and other.

Both forms of anti-social behaviour present a serious issue, as their consequences can be significant, also in the real-world. Internet users often communicate with each other in real-time; the discussions usually involve a considerable number of users. Such massive communication supported by modern technologies which enable partial anonymity also leads to the new threats in form non-proper user reactions. Anti-social user reactions in online discussions are often related to the use of abusive

language. There are numerous different definitions of such behaviour and it could be difficult to find the exact definition of such phenomenon and is even a more significant challenge to do so in the online environment [2]. However, toxic comments in an online discussion, in general, can be defined as a response in an aggressive way, which forces the offended participants to abandon it (e.g., personal attacks, verbal bullying) [3]. As the vast majority of those data are in the form of text, various techniques of natural language processing (NLP) can be utilized to their processing.

With a growing number of textual data generated in online environments, there is a strong need to detect and eliminate the various forms of anti-social behaviour effectively. Currently, manual techniques are still frequently used in the detection of such behaviour in online communities (discussion forums, social networks, etc.). Using human moderators responsible for finding and revealing the anti-social behaviour in online environments can be very time consuming and also biased by moderators themselves. In general, there is a strong need to design and implement the new methods able to detect the anti-social behaviour from the content automatically using the NLP, machine learning and artificial intelligence techniques. The overall goal of these approaches is to utilize the results of such methods for both, prevention and elimination of negative impacts of anti-social behaviour in online communities, for example by enabling the fully-automated detection and prediction of different types of anti-social behaviour from the user-created content. However, ML and NLP methods can still suffer from learning from the data which are often human-labelled. Measurement and mitigation of unintended bias is a non-trivial problem, which has been also studied in the area of toxicity detection [4,5].

The work presented in this paper focuses on exploring the use of currently popular deep learning architectures to predict the toxicity in the comments. While several studies were dealing with the problem of using deep learning to predict the toxicity of the comments, they are inconsistent in terms of pre-processing, model application and evaluation. Toxic comments are often written in specific language and style from both perspectives, content and form. Texts are relatively short, written using non-standard language, often using offensive language with a lot of grammatical and spelling errors and punctuation marks. Some of them represent just the common typos, but many of them are written purposely by their authors, to avoid the automatic on-line filtering of the abusive language [6]. In other sentiment analysis tasks, the effect of the pre-processing is well studied and proven, that the right selection of pre-processing may lead to performance improvement [7–9]. In this particular domain, we can assume, that it would require minimal pre-processing techniques to ensure that the information contained in the comment text and form would be preserved. On the other hand, there are word embeddings, as a way of text representations, which are currently frequently being used when training deep learning models. Those embeddings are usually pre-trained using various sets of text corpora. Some of the pre-trained embeddings are built using mostly clean, regular words and are more suitable for processing of standard texts while other ones fit better to short on-line communication. In the toxic comments classification task, it would also be interesting to train the word embeddings from scratch using the dataset related to the task. Therefore, in this research, we aimed to compare multiple currently popular deep learning methods and transformer language models and study the effects of different text representations and basic pre-processing techniques applied in the data preparation.

The paper is organized as follows: Section 2. provides an overview of the abusive language and toxic comments field and application of different machine learning methods to their detection. Section 3 describes the deep learning methods used for text classification. The following section presents the data used in the experiments and their preparation; Section 5 then describes the performance metrics used in the experiments, followed by the section describing implemented models and their settings. The next section is dedicated to the experimental evaluation and describes achieved results.

2. Toxic Comments Classification

Sentiment analysis in general considered a research area which combines NLP and text mining to automatically detect and identify the opinions contained in the text and determine the writer's

opinion or attitude with regards to a particular topic [10]. Although multiple approaches have been applied in this field, most of them are based on the application of machine learning methods. A specific sub-section of sentiment analysis is a detection of abusive language in the conversational content. Use of aggressive or offensive language in online discussions may occur in various forms. Various studies address different aspects of the abusive language in the online discussions, often differentiated by the types of aggression expressed. Therefore, when considering the abusive language detection from the texts, various related tasks are explored, including detection of cyber-bullying, hate or hate speech, online harassment, flaming, toxic comments, extremism, etc. Those tasks are often not clearly distinguishable, often are overlapping, and despite the differences between the concepts, often similar methods are utilized to tackle those problems [11]. However, there are studies trying to establish the common typology of the different abusive language detection tasks [12].

Toxic comments detection can be considered as a specific sub-task of approaches mentioned above, which aims to detect and identify the toxicity in the conversational text. It is usually solved as a text classification task, where the input features are extracted from the piece of text. As multiple types of toxicity could be contained in the text (e.g., insults, obscene language, hate, etc.), therefore toxic comments detection is usually considered as a multi-class classification task where the target class describe the particular type of the toxicity contained in the text. In this case, the problem of unbalanced data is a common issue, as the frequency of occurrence of the different toxicity types may vary.

Recently, the essential source of the data used to build the toxicity detection models come from social networks. Data are usually extracted from the discussions, comments or social network posts and typically represent the user reactions to a particular topic [13]. During recent years, several datasets became publicly available, containing labelled data from different social platforms and areas, e.g., Twitter dataset [14] contains 25,000 manually annotated tweets containing hate speech. Youtube dataset [15] consists of 3221 manually labelled comments from YouTube discussions [16] or very popular Wikipedia talk page corpus also used in this work. However, different datasets are often labelled non-consistently, which could be the effect of the different problem understanding and will require a more integrated approach when collecting the data in the future [17].

To detect the toxicity in the conversational data, both traditional machine learning methods, as well as advanced deep learning techniques, have been utilized. Traditional machine learning approaches include the use of various classifiers, e.g., Decision Trees [18], Logistic Regression [19], Support Vector Machine models [15] or Ensemble Models [20]. Traditional machine learning models are frequently used and popular in the detection of other types of anti-social behaviour, such as fake reviews detection. For example, work Naive Bayes and Random Forests have been used in the detection of the fake reviews obtained from Amazon [21] using data describing the seller, website, product, reviewer and review content. Authors in [22] answered interesting questions, if the performance of the classification methods for fake reviews filtering are affected when they are used in real-world scenarios that require online learning. Regarding the toxicity detection, authors in [23] monitored and analyzed the most recently published comments to detect whether an aggressive action emerges in a discussion thread. The authors experimented with various forms of representations of input texts in combination with Radial Basis Function, Support Vector Machines and Hidden Markov Model classifiers. The work [24] is focused on fake reviews detection and the influence of a length of the text data on a measure of the effectiveness of the learned models. The results of experiments showed that the models learned from the whole body of texts are more effective than models learned only from the headlines. Similarly, in [25] authors have examined the influence of a length of the text data on the effectiveness of machine learning models trained for recognition of authors of toxic posts. The paper describes an approach to suspicious authors identification based on the training a specialized dictionary of the toxic author and also the training of Naive Bayes and Support Vector Machine models.

However, recently, deep learning techniques proved to be successful in the detection of various types of anti-social behaviour on the web. For example, deep neural networks were used to detect the cyber-bullying within the user posts on the Twitter [26]. Multiple topologies of Convolutional

Neural Networks (CNN) were evaluated to find the most suitable model when handling this task [27]. Besides tweets, other data sources can be utilized to train the cyber-bullying detectors. Authors in [28] used transfer learning within different datasets of conversational data (e.g., Wikipedia, Twitter) and then compared the performance of deep learning models. Multiple deep networks were successfully used also in hate speech detection [29], including deep learning ensemble models [30]. Multi-label toxic comments classification was also addressed by different deep learning models [31]. In [32], authors used CNN for multi-label classification of the comments and experimented with different word embeddings, in [33], authors compared the performance of CNN to Long Short-Term Memory (LSTM) network, and authors in [34] presented the capsule network approach. When monitoring social networks, an interesting aspect would be tracking the temporal aspects of toxicity in the comments. In [35] authors present the CNN model able to detect the toxic tweets. Authors also utilize the hashtags from the tweets related to toxic tweets and also are able to monitor the toxicity propagation over time. Several previous works approached toxicity detection as the binary classification problem. However, deep learning models are also used in more complex, ensemble approaches. In [36] an ensemble model consisting of CNN, BiLSTM and GRU is presented, which determines whether the text is toxic or not in the first step and then classifies the toxic comments into a more specific category representing the particular type of the toxicity.

3. Deep Learning Methods for Text Processing

Neural networks are considered to be one of the best-performing machine learning algorithms. They have brought great success in the field of artificial intelligence, such as in the field of computer vision, where their task is image processing and pattern recognition, and, for example, in sound processing and speech recognition. In this section, we took a closer look at how neural networks can be used to work with the textual data.

3.1. Feedforward Neural Network

Deep forward neural networks known as the feedforward neural networks (FFNN) or multilayer perceptrons are basic models of deep learning. Feedforward networks became popular in 1986 when Rumelhart, Hinton, and Williams introduced a method of training forward neural networks using the error back-propagation [37]. The goal of feedforward neural networks is to approximate the function f^*. For example, the function $y = f^*(x)$ maps the input x to the value y. FFNN defines the mapping $y = f(x;\theta)$ and finds the value of the parameters θ, which leads to the best approximation of the function. The flow of information in FFNN is forward; in practice, this means that the computational model represents an acyclic graph.

The basic model of a neuron is called perceptron. The perceptron receives input signals $\bar{x} = (x_1, x_2, \ldots, x_{n+1})$ via synaptic weights, which form the vector $\bar{w} = (w_1, w_2, \ldots, w_{n+1})$. The perceptron output is given as the scalar product of the input vectors transformed using the activation function f, to which the bias is added. Bias b is a constant that does not depend on the input parameters and serves to influence the activation function [38,39].

$$output = f(\bar{w} \cdot \bar{x}) = f\left(\sum_{i=1}^{n+1} w_i x_i\right) + b. \quad (1)$$

For the best classifiers in our work are used following hyper-parameters and settings:

- Activation function on input and hidden layers: ReLU:

  In proposed solution we used ReLU (Rectified Linear Unit) [40] activation function. ReLU belongs to one of the most frequently used activation functions applied in deep networks. It is defined as:

$$f(x) = \max(0, x) \quad (2)$$

which means, that it transforms negative inputs to 0 and leaves positive inputs without transformation.

- Loss function: binary cross entropy

  Error function is used to estimate the error of the model during the training. Using the error backpropagation [41], the neuron weights in particular layer are updated in such manner, that the error rate decreases in following evaluation. We used Binary Cross-Entropy (BCE), which can be defined as:

$$\text{BCE} = -\left(y \log\left(\hat{y}\right)\left(1-y\right) \log\left(1-\hat{y}\right)\right), \tag{3}$$

  where y represents the ground truth and $\hat{y}$ represents predicted value.

- Optimization: Adam

  To minimize the error rate of the model in the prediction, optimization function is used. We used Adam (Adaptive Moment Estimation) [42]. Adam is an optimization function, which computes the learning estimation for each parameter. In addition to storing an exponentially decaying average of past squared gradients $v(t)$ like RMSprop [43], Adam also keeps an exponentially decaying average of past gradients $m(t)$, similar to momentum [44]. Both moving averages are initialized to 0, which leads to the moments' estimation biased towards zero. Such situation occurs mostly during the initial phases when decay parameters ($\beta 1$, $\beta 2$) have values close to 1. Such biased can be removed using modified estimations $\hat{m}_t$ a $\hat{v}_t$:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{4}$$

  The parameters are updated according to the formula:

$$\theta_{t+1} = \theta_t \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon} \cdot \hat{m}_t. \tag{5}$$

  Default setting for the parameter β_1 is 0.9 and $\beta_2 = 0.999$, learning rate $\alpha = 0.001$ and 10^{-8} for the ϵ parameter. Adam is considered as a suitable optimization method in practical tasks. When comparing Adam to other methods, its advantage is faster convergence and training speed is also higher. It also removes certain issues of other optimization techniques, such as slow convergence, or high variance of the parameters, which could lead to variations in the error function.

- Regularization: dropout

  The main idea of the dropout is the random removal of specific neurons (along with their connections) from the neural network during the model training [45]. Ignoring, or "dropping-out" of specific neurons can prevent their over-adaptation, which could lead to over-fitting. In each iteration, a new sub-network is created, which contains different neurons than during the previous iteration. The output of such a process is a set of sub-networks, which have a higher chance to capture the random phenomena in the data compared to the single robust network. While using this technique, it is necessary to set the parameter defining the probability of selection of a number of neurons, which will be dropped out from the network.

- Output activation function: sigmoid

  Sigmoid function is a bounded, differentiable real function. It is defined for all real values and has a non-negative derivative in each point [39]. It is mostly used because of its non-linearity and simplicity of the computation. The function is defined as:

$$f(x) = \frac{1}{1+e^{-x}} \tag{6}$$

The output of the sigmoid activation function is in the range between 0 and 1, which makes it suitable for use in the classification tasks.

3.2. Convolutional Neural Network

Convolutional neural networks (CNN) represent a specific type of forward neural networks, which contain a layer of neurons for the convolution operation. The inspiration for the architecture of this network was the function of the ocular nerve. Neurons respond to the input of the surrounding neurons' activations according to a specified size of the convolutional kernel, also called filter. Convolution consists of shifting the convolution kernel over the whole set of values. In this case, the convolution operation represents the multiplication of the convolution kernel and input values (see Figure 1) [38].

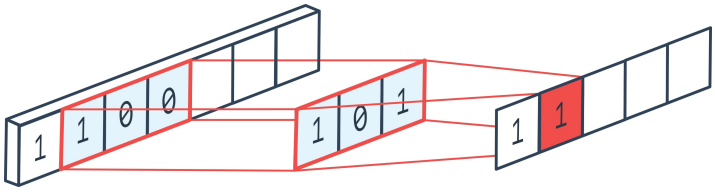

Figure 1. One-dimensional convolutional process. Input data is located on the (**left**), the filter in the middle, and the convolution output on the (**right**).

Pooling layers in convolutional networks are designed to reduce the number of outputs, to reduce the computational complexity, and to prevent the network over-fitting. The sampling layers are usually applied just behind the convolution layers, as the duplicate data are created when the convolution kernels are shifting through the individual inputs. Excess data is removed using the pooling layers.

In our work, we use the Global pooling layer, in which we distinguish between the Global average pooling layer and the Global max pooling layer. These layers work according to the same principle as the traditional average pooling (max pooling) layer. The difference is, that the average (maximum) is not calculated only for a given area, but for the entire input.

3.3. Long Short-Term Memory

Long Short-Term Memory (LSTM) [46] is a type of a recurrent neural network. LSTM has a more complex structure, which makes it suitable to deal with the vanishing gradient problem. Using the LSTM in any sequential task will ensure that long-term information and context is maintained (see Figure 2).

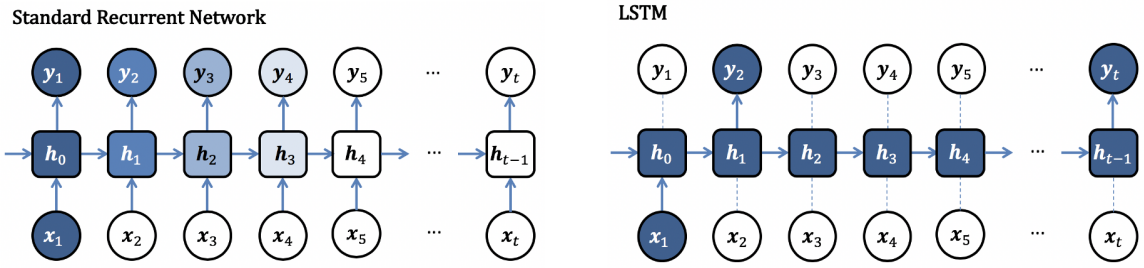

Figure 2. The first figure shows the standard recurrent neural network and the vanishing gradient problem, which results in the loss of context. In contrast, the second figure shows the preservation of information and context in LSTM [47].

Comparing to other types of neural networks, the LSTM network does not consist of interconnected neurons, but of memory blocks that are connected in layers. The block contains gateways that manage the state and output of the block and the flow of information. Gateways can

learn which data in a sequence is important and needs to be preserved. There are four memory block elements performing the following functions (see Figure 3):

- Input gate—it is used to control the entry of information into the memory block.
- Cell state—it is used to store long-term information.
- Forget gate—it is used is to decide what information will be discarded and what information will be kept.
- Output gate—based on the input and the memory unit it is used is to decide what operation to perform on the output.

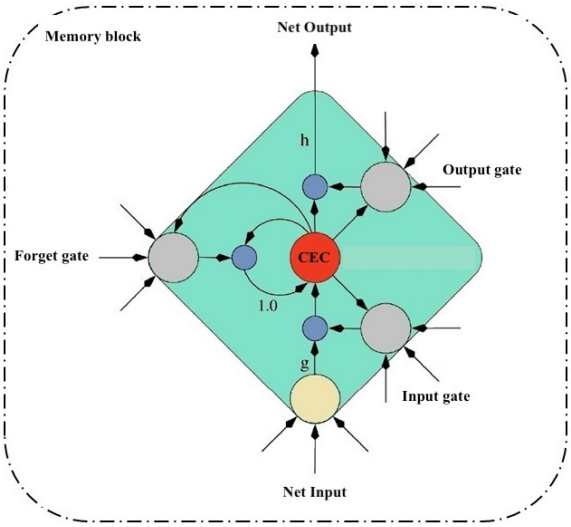

Figure 3. LSTM network memory block. The block has a recurrent connection with the weights set to 1.0. The three gateways collect input from the rest of the network and check the status of the memory block via three multiplicate units (marked in blue). The letters *g* and *h* depict the application of a nonlinear function [48].

Bidirectional Long Short-Term Memory network (BiLSTM) represents a specific type of LSTM network. BiLSTMs consists of two individual hidden layers. The first layer is used to process the input sequence forward, and on the other hand, the second hidden layer is used to process the sequence backwards. The hidden layers merge in the output layer, thanks to that the output layer can access to each point's past and the future context in the sequence. LSTM and their bidirectional variants proved to be very suitable. They can learn how and when they can forget certain information and also they can learn not to use some gateways in their architecture. Faster learning rate and better performance are the advantages of a BiLSTM network [49].

3.4. Gated Recurrent Unit

Cho et al. [50] also tried to solve the vanishing gradient problem described in Section 3.3 in the publication, where they presented the recurrent neural network called Gated Recurrent Unit (GRU). GRU can be considered as a variation of the LSTM network because both are designed in a similar fashion. GRU solves the vanishing gradient problem using an update and reset gates. The update gate helps the model to determine, how much of the previous information (from the previous time steps) needs to be used in the future, and the reset gate determines, how much of that information will be discarded. We also used a bidirectional variant of the GRU network (BiGRU) in our experiments.

3.5. Transformer Models

Transformer models are currently very popular methods used to solve various NLP tasks such as question answering, language understanding or summarization, but has been successfully used in text

classification tasks [51]. BERT (Bidirectional Encoder Representations from Transformers) is a language transformation model introduced by Google [52]. BERT is is "deeply bidirectional", which means, it learns the deep representation of texts by considering both, left and right contexts. It is a method used for training of general-purpose language models on very large corpuses and then using that model for the NLP tasks. So there are two steps involved in using BERT: pre-training and fine-tuning. During the pre-training phase, the BERT model is trained on unlabelled data. Then, the model is initialized with the pre-trained parameters and fine-tuned for specific NLP task. Fine-tuning of the BERT model is much less expensive on the computational resources. BERT uses the same architecture in different tasks. BERT is built using the Transformers [53]. The model comes in two variants, BERT-base and BERT-large. BERT-base consists of 12 Transformer blocks, hidden size of 768 and 12 self-attention heads, BERT-large consists of 24 Transformer blocks, hidden size of 1024 and 16 self-attention heads. There are several BERT variations currently introduced, such as DistilBERT which aims to reduce the size of the BERT model, while retaining the performance [54], RoBERTa, which optimizes BERT hyper-parameters to improve the performance [55] or XLNet learns the bidirectional contexts over all permutations of the factorization order [56].

4. Data Understanding and Preparation

4.1. Dataset Description

In the experiments, we used the Toxic Comment Classification Challenge (Available online: www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) competition dataset, as it presents an interesting challenge, widely used in training of the toxicity detection models. It consists of Wikipedia comments, which contain the comment (id) and textual content of the comment (comment_text feature). Each comment is marked with a specific type of toxicity. Each type of the toxicity is represented by a particular label: *toxic*, *severe_toxic*, *obscene*, *threat*, *insult* and *identity_hate*. The comments can be labelled with multiple types of toxicity. Figure 4 depicts a sample of the training data. As we can see from the figure, toxic comments are often written using an explicit language, written in all caps, using numerous punctuation marks.

comment_text	toxic	severe_toxi	obscene	threat	insult	identity_hate
shut the fuck up bastard	1	1	1	0	1	0
Fucking lying nigger, fes up you peice of shit211.28.54.73	1	1	1	0	1	1
YOU ADMINISTRATE LIKE SHIT ANYWAY, YOU FUCCIN ASSHOLE! ( )	1	1	1	0	1	0
Total Asshole bitches like you just need to get the fuck off. Motherfucking Shithole. Eat your dick.	1	1	1	0	1	0
Go fuck yourself this ain't any of your business and i fucking know what tor is. Asshole.	1	1	1	0	1	0
Who are YOU to tell ME??!! I'll do WHAT I like WHEN I like...cocksucker!!	1	1	1	0	1	0
i hope you feel like shit for deleting my post. go fuck yourself	1	1	1	0	1	0
DOOSH DOOSH DOOSH DOOSH DOOSH FUCKING COCK SUCKER QUEIR BATE ASS WIPE DONT TELL ME WHAT THE FUCK TO WRITE	1	1	1	0	1	0
SuPeRTR0LL WiLL LiVe FoReVeR! iF You DoN'T ReSPeCT THe SuPeRTR0LL You WiLL Die You PaTHeTiC Foo…	1	0	1	1	1	0
WHY THE FUCK DO U EDIT MY CONTRIB. TO WIKI ABOUT THE IRANIAN ARMY…	1	0	1	1	1	1
EVERYONE WANTS TO KILL BILL GOD DAM GATES HE IS A NERD WHY NOT.??	1	0	0	1	1	0
That's it. Remember last time you fucked with me... the dildo, yo…	1	0	1	1	0	0
Who gives a shit? Anthony Bourdain is a tool. I just wish he had …	1	0	1	1	0	0
What the fuck does a dumb Canadian polock like you know about Sta…	1	0	1	0	1	1
These hoe's lame on here on leaving this site yall gay and shit	1	0	1	0	1	1
MAYBE ITS BECAUSE YOU ARE A GAY	1	0	0	0	1	1
you need to go to Oz to get a fucking brain	1	0	1	0	1	1

Figure 4. Sample of training data Toxic Comment Challenge.

Dataset is divided into the training and testing set. Training data consists of 159,751 samples. The models were evaluated on the independent testing set consisting of totally 153,164 records. However, to prevent the effect of the hand-labelling of the dataset, the testing set contained comments

to be not included in scoring (labelled by −1). After removal of these record, testing set used for evaluation consisted of 63,978 records. Table 1 shows the number of samples in the training and test set for each class. Please note that most of the toxic comments are labelled with more than one type of toxicity. Non-toxic comments have not assigned a specific label, but are not assigned with any of the toxic labels ("clear" comments in the Table 1).

Table 1. Type of toxicity occurrences in training and testing dataset.

Toxicity	Training Set	Testing Set
toxic	15,294	6090
severe toxic	1595	367
obscene	8449	3691
threat	478	211
insult	7877	3427
identity hate	1405	712
"clear"	143,346	5773

The average comment consisted of 394 characters. Figure 5 depicts the number of comments depending on the length of the comment within each class. In our experiments, we worked with an average comment length of 200 characters.

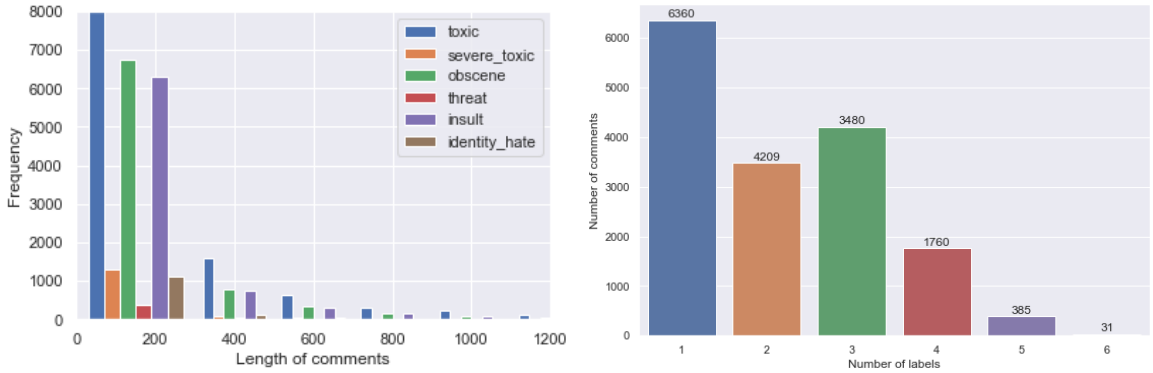

Figure 5. The first graph depicts the length of the comments in respective categories. The second one depicts the number of the comments belonging with multiple categories.

Comments in the data may belong to multiple classes (e.g., toxic comments may contain various types of toxicity). On the contrary, the initial data exploration showed, that certain comments labelled as an obscene, threat, insult, or identity hate, may not be considered as toxic (see Table 2). We decided to keep such comments in the dataset, even though they are not considered as toxic, they could be considered as anti-social from a different perspective.

Table 2. A comparison of the occurrence and frequency of individual categories with the toxic category.

Category	Frequency
severe toxic AND toxic	1595
obscene AND toxic	7926
threat AND toxic	449
insult AND toxic	7344
identity hate AND toxic	1302
severe toxic but NOT toxic	0
obscene but NOT toxic	523
threat but NOT toxic	29
insult but NOT toxic	533
identity hate but NOT toxic	103

Figure 6 shows the correlation between the target variables. The highest levels of correlation can be observed between the obscene, toxic its and insult classes.

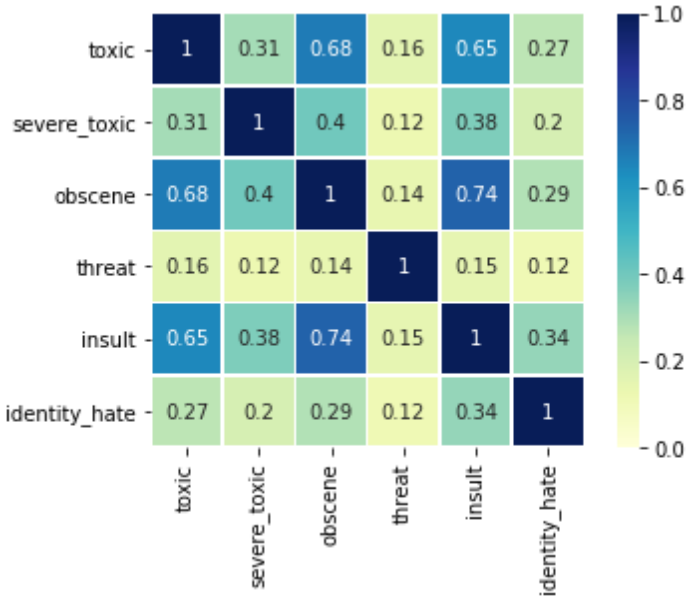

Figure 6. Correlation of features and targets.

4.2. Data Pre-Processing

The initial step after the data understanding is the preparation of the data to the form suitable for the classifier training. When working with the textual documents, data pre-processing usually involves a series of tasks of text cleaning, formatting and creation of its representation used in the model training. Notably, in this case, the texts contain slang expressions, emoticons, incomplete words, typos, etc.; which are usually addressed in the pre-processing. However, in case of abusive language/toxic comments detection from the social media (as well as other similar tasks), it should be noticed, that abusive/toxic content is often associated with non-standard textual content (e.g., slang, upper case letters, emoticons, etc.). Authors in [6] point out, that in a domain such as antisocial behaviour detection on the Internet, it is essential and useful to keep the data in their original form as much as possible. Application of standard pre-processing methods such as lowercasing, stopwords removal or stemming could lead to the loss of the individuality and specific features of both, the content of the message that the author wants to submit and also for the author himself. By using the standard pre-processing methods, in this case, we may lose important traits, crucial in the process of extracting features. In our experiments, one of our primary motivation was to explore this hypothesis.

We decided to compare the models performance also from the aspect of standard pre-processing techniques used in the process and compared on the data pre-processed in a standard manner and on the data with no traditional pre-processing at all.

During the experiments, where we used the data pre-processing, we studied the effect of the standard techniques commonly used in the textual data preparation:

- Tokenization—splitting of strings into a tokens, representing the lexical units (e.g., words);
- Lowercasing—conversion of the entire text to lowercase (words with different cases map to the same lowercase form);
- Punctuation removal—removal of all punctuation marks within the sentence;
- Stop words removal (stop words—words with minimum information value, e.g., conjunctions, prepositions, confusions, etc.).

4.3. Features Representation

We also used the different text representation methods. We used the standard vector space representation using TF-IDF (Term Frequency-Inverse Document Frequency) weighting [57], special tokenizers of popular transformers models (BERT, DistilBERT and XLNet tokenizer) and the representation of the text documents using the following word embeddings:

- word2vec— provides direct access to vector representations of words. It is a combination of two techniques, two neural networks—Continuous bag of words (CBOW) and Skip-gram model [58].
- GloVe—is one of the newest methods for calculating the vector representation of words. However, this approach does not use the whole corpus. It is learned only based on global statistics on the occurrence of words in a current context. The method captures various linguistic patterns and can successfully solve problems based on the principle of analogy [59].
- fastText—is a library created by Facebook's research team to learn and calculate the word representation and sentence classification. Its principle is to assign a vector representation to n-grams of characters that contain individual words [60].

We used GloVe embeddings pre-trained on Common Crawl (300 dimensions) and Twitter (200 dimensions), for the fastText we used pre-trained word vectors for 157 languages, trained on Common Crawl and Wikipedia (also in dimension 300).

Compared to word embeddings, BERT model includes an attention mechanism which is able to learn contextual relations between words in text. BERT consists of an encoder which processes the text input and decoder used to perform the prediction. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. BERT, which uses the Transformer encoder is able to learn the context of a word based on its entire surrounding (both left and right context of the words). When comparing to other directional models, that read the text input sequentially, transformer models read it as an entire sequence at once (from that point of view, it can be considered as non-directional). We used following tokenizers from Transformers (https://huggingface.co/transformers/:

- BERT tokenizer —bert-base and bert-large cased/uncased tokenizer—based on word piece;
- DistilBERT Tokenizer—distilbert-base-cased tokenizer—is identical to BertTokenizer and runs end-to-end tokenization: punctuation splitting and wordpiece;
- XLNet tokenizer—xlnet-base-cased tokenizer—based on sentence piece.

5. Performance Metrics

To evaluate the models, we decided to use the standard metrics used in classification, e.g., accuracy, precision, recall and F1 score. Such metrics are easy and straightforward to obtain for a binary classification problems and can be computed as:

- Accuracy = TP + TN/TP + FP + FN + FP
- Precision = TP/TP + FP
- Recall = TP/TP + FN
- F1 score = 2 $*$ (Precision $*$ Recall) / Precision + Recall,

where:

- TP—True Positive examples are predicted to be positive and are positive;
- TN—True Negative examples are predicted to be negative and are negative;
- FP—False Positive examples are predicted to be positive but are negative;
- FN—False Negative examples are predicted to be negative but are positive.

To apply such metrics in the multi-label classification, those metrics could be computed for each class (one-vs-rest approach). Usually, we need to compute the confusion matrix (see Figure 7) for each

class $c_i \in C = \{1, ..., K\}$. For each class c_i, the i-th class is considered as positive, while the rest of other classes as a negative class. Then, to summarize the performance of the classifier on all classes, metrics can be micro or macro averaged [61]. The use of micro or macro averaging is dependent on the particular use case. In the following formulas, we will use TP_i, FP_i, and FN_i as the true positive, false positive, and false-negative rates associated with the class i.

Micro-averaging at first computes the confusion matrix for all classes and then calculates the overall metrics. Micro-averaging may be preferred in case of class imbalance present in the data. Micro-averaged precision and recall metrics are computed as:

- $\text{Precision}_{\text{micro}} = \frac{\sum_{i=1}^{|C|} \text{TP}_i}{\sum_{i=1}^{|C|} \text{TP}_i + \text{FP}_i}$
- $\text{Recall}_{\text{micro}} = \frac{\sum_{i=1}^{|C|} \text{TP}_i}{\sum_{i=1}^{|C|} \text{TP}_i + \text{FN}_i}$
- $\text{F1 score}_{\text{micro}} = 2 * \frac{\text{Precision}_{\text{micro}} * \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}}$

On the other hand, macro-averaging is based on the computation of precision and recall for each class and then averaging the overall metrics:

- $\text{Precision}_{\text{macro}} = \frac{\sum_{i=1}^{|C|} \text{Precision}_i}{|C|}$
- $\text{Recall}_{\text{macro}} = \frac{\sum_{i=1}^{|C|} \text{Recall}_i}{|C|}$
- $\text{F1 score}_{\text{macro}} = 2 * \frac{\text{Precision}_{\text{macro}} * \text{Recall}_{\text{macro}}}{\text{Precision}_{\text{macro}} + \text{Recall}_{\text{macro}}}$

To compare the models, we also used the Area Under Curve (AUC) score to evaluate the models. AUC score computes the area under the Receiver Operating Characteristic (ROC) curve. Although the AUC score is not an ideal metric to compare the models trained on highly-imbalanced data, we used it to compare the models with other models from the relevant literature. The reason behind this is the fact, that the most studies use the AUC score, as it was specified as a criterion in the Toxic Comments Classification Challenge competition.

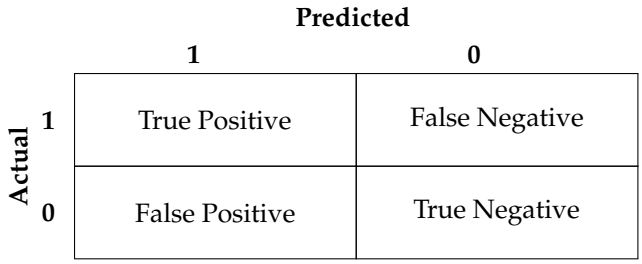

Figure 7. Confusion matrix.

6. Models and Settings

6.1. Deep Learning Models

To choose the most suitable model, we performed an initial set of experiments to evaluate the different neural network architectures. We have considered multiple architectures, from simple feedforward to composed models and implemented selected deep neural network architectures-FFNN, CNN, GRU, LSTM, and a combination of bidirectional GRU/LSTM and convolutional layer in order to choose the best performing model for the following experiments with pre-processing techniques.

We used the following deep learning models:

- FFNN with three fully connected layers with 32, 64, 128 neurons, and a 20% dropout regularization

- CNN with one-dimensional convolutional layer with 64 filters, kernel size 3, max-pooling layer with window size 2, a flatten layer and fully-connected layer of 128 neurons, 20% dropout regularization
- GRU, in which we replaced the convolution block with a GRU layer with 128 units
- LSTM, in which we replaced the convolution block with a LSTM layer with 128 units.

Besides the mentioned neural network architectures, we implemented two composed architectures:

- The first architecture includes an embedding layer, a following bidirectional LSTM layer with 128 units, a 1D convolution layer with kernel size 3, and global max and average pooling layers. These pooling layers are concatenated, following with the fully connected layer with 64 neurons. In the bidirectional layer, we used recurrent dropout 10%, and other 20% in a separate layer.
- The second composed architecture, we replaced the LSTM layer with a GRU layer with the same parameters. Figure 8 shows the architecture of the LSTM layer.

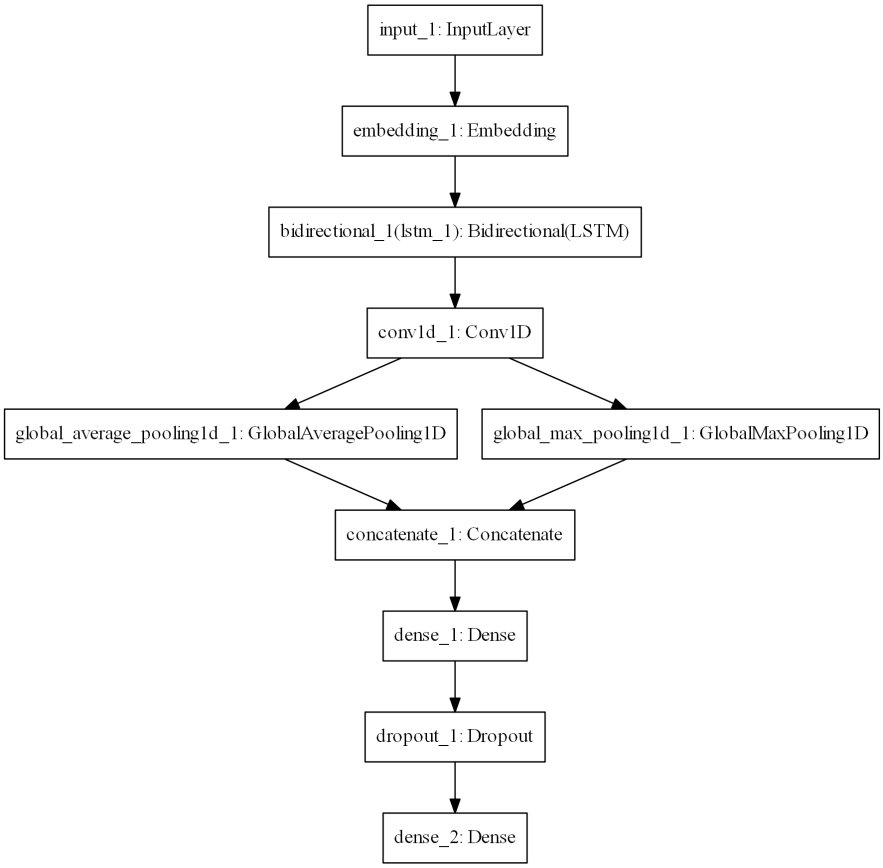

Figure 8. Architecture of the composed BiLSTM + CNN model for multi-label classification.

6.2. Transformer Models

We compared the composed architecture with some recent popular language models. We used a dedicated tokenizer belonging to each of the models. To obtain the desired classification result, we connected each transformer model's output to a feedforward neural network—a fully connected layer with 128 neurons and regularisation layer dropout (15%). The last fully-connected layer represents the output layer of the neural network and consists of six output class (toxicity type in the input text). The architecture of this setup is depicted in the Table 3.

Table 3. Architecture of the pre-trained transformer models.

Layer	Parameters
Input 1	input_shape = (,128)
Input 2	input_shape = (,128)
Transformer model	training = False
Dense	128 neurons, activation ReLU
Dropout	15%
Dense	6 neurons, activation Sigmoid

- BERT : we used BERT tokenizer, and following pre-trained models:
  - BERT model (Cased/Uncased): we used a pre-trained BERT model for sequence classification, which contains a sequence classification/regression head on top (a linear layer on top of the pooled output). In this model, the transformer is pre-trained, and the sequence classification head is only initialized and has to be trained. We used only BERT-base pre-trained model, with BERT-large, we experienced the stability issues with its training. Output of the classification head was used and fed into the FF network for classification.
  - Bare BERT (Cased/Uncased): in this case, we used pre-trained BERT model transformer, which is outputting raw hidden-states without any specific head on top. We used BERT-base and BERT-large versions of the model, both cased and uncased versions. We used pooled output of the model for the classification.
- DistilBERT: we used distilbert-base tokenizer, and we created DistilBERT model (Cased/Uncased) based on the architecture shown in Table 3. We used model with a sequence classification/ regression head on top (a linear layer on top of the pooled output).
- XLNet: we used xlnet-base-cased tokenizer, and we created XLNet model based on the architecture shown in Table 3. We used model with a sequence classification/regression head on top (a linear layer on top of the pooled output). XLNet model doesn't provide a pre-trained and uncased version.

7. Experiments

During the experiments, we aimed to compare the effect of different pre-processing techniques on the classification of the toxic comments. In comparison, we used the composed architecture model with different pre-processing methods applied to the data. We aimed to compare the models' performance using:

- TFIDF representation with standard pre-processing;
- TFIDF representation without standard pre-processing;
- Pre-trained embeddings with standard pre-processing (GloVe, fastText);
- Pre-trained embeddings without standard pre-processing (GloVe, fastText);
- Custom-trained embeddings with standard pre-processing (word2vec);
- Custom-trained embeddings without standard pre-processing (word2vec);
- Pre-trained BERT language representations;
- Fine-tuning BERT language representations;
- Pre-trained DistilBERT language representations;
- Pre-trained XLNet language representations.

Figure 9 depicts the workflow of the experiments. It is important to note that due to extreme computational intensiveness of the models training, not every possible combination of the pre-processing and model was explored. Instead, we followed a methodology of the initial evaluation of the models using default settings to choose the best-performing model. Then, we followed with the

optimization of hyper-parameters of the best-performing model using grid-search and cross-validation. And finally, we evaluated the fine-tuned model using different combinations of pre-processing and text representations techniques. A more detailed description of the particular steps will be described in the following subsections.

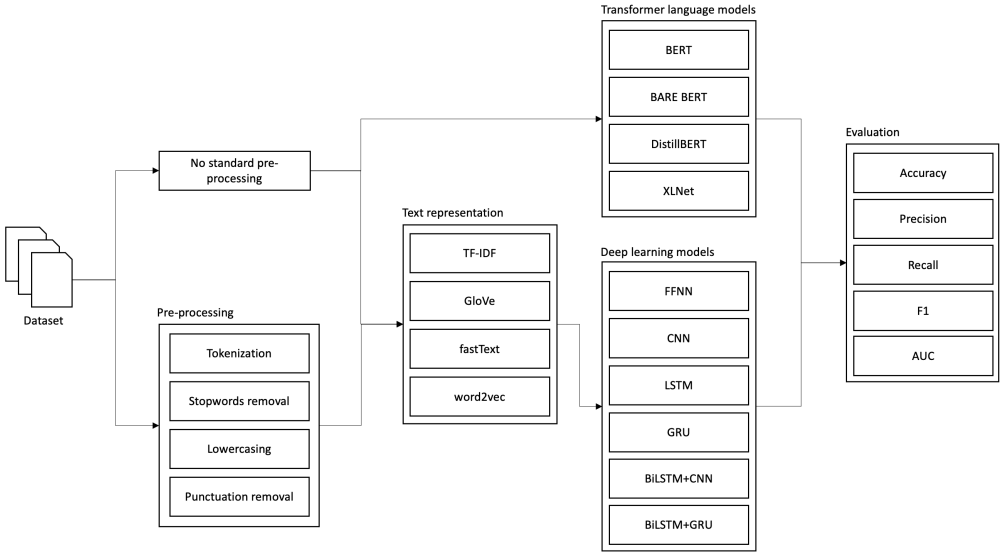

Figure 9. Overall schema of the experiments setup.

7.1. Selection of Best Deep Learning Model

Initial experiments were aimed to select the most suitable method to explore the pre-processing impact. To do so, we compared the described NN architectures on commonly used embeddings (GloVe). When comparing the particular architectures, we obtained the accuracy performance of each model, cross-validated on the training set and evaluated on the testing set; the results are shown in Table 4.

To compute the metrics (accuracy and loss), we transformed the class probabilities (output of the neural networks) into the crisp class predictions using a simple rule, which assigned the sample to a class if a probability of a given category was higher than 0.5. During this phase, we worked with this simplistic approach, in further evaluations of the best performing model, we also adopted a more advanced technique to identify the optimal threshold for each class.

Table 4. Initial evaluation of the models.

	FFNN	CNN	GRU	LSTM	BiLSTM + CNN	BiGRU + CNN
Accuracy	0.878	0.886	0.872	0.873	0.890	0.884
F1 score—micro avg	0.63	0.64	0.66	0.65	0.67	0.67
F1 score—macro avg	0.34	0.42	0.56	0.55	0.53	0.47

The training phase of the deep learning models on the used dataset is very demanding on computational resources. The training process is very time-consuming, even on recent GPUs. We decided to optimize the hyper-parameters of the best model on 10% stratified sample of the dataset. We performed the fine-tuning of the hyper-parameters using a grid search with cross-validation. We tuned the best-performing model (biLSTM-CNN), which we used with the obtained parameters as the starting architecture for further experiments. Considered parameters used for the grid search are shown in Table 5.

Table 5. Hyper-parameters used in grid search for the BiLSTM + CNN model.

Hyper-Parameters	Values
Activation	`'relu', 'than'`
Batch Size	`[16, 32, 64]`
Optimizer	`['SGD', 'RMSProp', 'Adam'`
Dropout rate	`[0.1, 0.2]`

We achieved the best results using the regularisation dropout 0.2, activation function ReLU, Adam optimizer and batch size 32. The complete results of this experiment are stored on a GitHub (https://github.com/VieraMaslej/toxic_comments_classification/blob/main/result_gridsearch.txt).

To gain a better understanding of the learning process and more importantly, to estimate the learning variance, we performed 10-fold cross-validation of the best performing model on the training data. Table 6 summarizes the results of the particular folds during the cross-validation of the BiLSTM + CNN model. This step was important to estimate the learning variance. As we can observe that the overall variance of the learning is acceptable, we will not use the cross-validation during further experiments with pre-processing. This enabled us to reduce the total time needed to train and test all evaluated combinations of the pre-processing and text representation methods.

Table 6. Cross-validation of the BiLSTM + CNN model.

Model	1	2	3	4	5	6	7	8	9	10	avg
Loss	0.047	0.045	0.045	0.045	0.047	0.047	0.045	0.046	0.045	0.043	0.046
Accuracy	0.991	0.994	0.994	0.971	0.991	0.991	0.994	0.994	0.994	0.994	0.991

All models were implemented in Python language using Tensorflow [62] and Keras [63] libraries. The source codes are available on GitHub (https://github.com/VieraMaslej/toxic_comments_classification). The experiments were conducted on a PC equipped with a 4-core Intel Xeon processor clocked at 4 GHz and NVIDIA Tesla K40c GPU with 12 GB memory.

7.2. Analysis of Text Representation and Pre-Processing Influence on Deep Learning Models

From the initial set of experiments, we selected a composed BiLSTM network architecture in combination with a convolution layer to be the most suitable to explore the effects of different pre-processing. During the following experiments, we focused on using different text representation and pre-processing settings. We computed commonly used metrics in classification, including accuracy, AUC score, precision, recall and F1. Interesting is an F1 score as it expresses the harmonic mean of precision and recall and describes the overall performance of the model better. We compared the performance of the model with a standard pre-processed text corpus and without pre-processing (only using simple tokenization). We also decided to compare different text representations, which we described in Section 4.

In the first step, we explored how the model performs when using the TF-IDF data representation. Table 7 summarizes the results of the experiments. Basic document vector representation using TF-IDF did not prove to be very suitable for this task. The performance of the model using this representation suffered from poor recall. Although accuracy and AUC values gain reasonable values, those metrics are not very useful in imbalanced classes. To better understand the classifier performance, precision and recall provide better insight. In this case, it is clear that the minor classes failed to learn completely. On the other hand, we can observe that the standard pre-processing improves the classification (contrary to the expectations). In TF-IDF, the pre-processing may improve the created vector representation, as it is created from the corpus itself (not from pre-trained vectors, such embeddings).

Table 8 depicts the BiLSTM + CNN model performance using the word2vec embeddings. In both cases (with and without pre-processing), word2vec representations were trained from the

dataset. word2vec representation brings massive improvement in comparison to TF-IDF, rapidly improving the performance metrics (both, micro and macro averaged). The results also demonstrate the influence of the pre-processing techniques applied in text preparation. The model gained slightly better performance on the not processed text, improving recall values (most importantly, macro-averaged recall).

Table 7. Performance of the composed BiLSTM + CNN model with TF-IDF text documents representation.

	Accuracy	AUC Score		Precision	Recall	F1 Score
TF-IDF	0.8971	0.8437	micro avg.:	0.60	0.05	0.09
			macro avg.:	0.33	0.02	0.03
TF-IDF + PP	0.9038	0.8533	micro avg.:	0.70	0.23	0.35
			macro avg.:	0.35	0.13	0.19

Table 8. Performance of the composed BiLSTM + CNN model with gensim word2vec embeddings.

	Accuracy	AUC Score		Precision	Recall	F1 Score
word2vec	0.8835	0.9791	micro avg.:	0.62	0.74	**0.67**
			macro avg.:	0.58	0.53	**0.51**
word2vec + PP	0.8787	0.9769	micro avg.:	0.61	0.71	0.65
			macro avg.:	0.59	0.45	0.46

Table 9 summarizes the model performance using pre-trained word embeddings, both with standard text pre-processing and a model with no pre-processing. We used two different GloVe pre-trained embeddings, Common Crawl (840 B tokens, 2.2 M vocab) and Twitter (2 B tweets, 27 B tokens). The model performer very similar using different GloVe and fastText embeddings. Although the averaged F1 metrics are very similar, we can observe some differences, how the models perform on precision and recall metrics. Skipping of the pre-processing in case of the CC GloVe embeddings causes recall drop and improvement of the precision, while in case of the Twitter GloVe embeddings it is otherwise. The difference may be caused by either size of the tokes, or how the embeddings were trained. It is possible that the Twitter embeddings are built using the data closer to the domain (as tweets may be similar to the comments). We used F1 metric to select the best performing model, Twitter GloVe embeddings without pre-processing was the best method from that point of view.

Table 9. Performance of the composed BiLSTM + CNN model with pre-trained GloVe and fastText embeddings.

	Accuracy	AUC Score		Precision	Recall	F1 Score
Common Crawl GloVe	0.8904	0.9796	micro avg.:	0.63	0.72	0.67
			macro avg.:	0.60	0.52	0.53
Common Crawl GloVe + PP	0.8621	0.9766	micro avg.:	0.54	0.79	0.64
			macro avg.:	0.47	0.55	0.46
Twitter GloVe	0.8787	0.9798	micro avg.:	0.59	0.77	**0.67**
			macro avg.:	0.57	0.59	**0.56**
Twitter GloVe + PP	0.8903	0.9771	micro avg.:	0.65	0.67	0.66
			macro avg.:	0.63	0.51	0.54
fastText	0.8839	0.9787	micro avg.:	0.61	0.73	0.66
			macro avg.:	0.58	0.52	0.51
fastText + PP	0.8947	0.9749	micro avg.:	0.63	0.70	0.67
			macro avg.:	0.49	0.44	0.43

7.3. Selection of Best Transformer Model

Table 10 compares the performance of BERT model, bare BERT and its DistilBERT and XLNet variants. We compared the performance of these models with BiLSTM + CNN architecture. Regarding the pre-processing, it was a little bit different in this case. As the transformer models are available pre-trained on the text corpora in two different versions-cased and uncased (except the XLNet model, that doesn't come with the uncased version). Furthermore, we used the BERT tokenizer, in which we used lowercasing of the input text (in cased versions) or did not use it (in uncased versions). As described in Section 6.2, we trained two BERT-base for sequence classification models, two bare BERT-base models, two bare BERT-large models, two DistilBERT models and a single XLNet model. We used default hyper-parameters as depicted in Table 11.

Table 10. Performance of the transformer models.

	Accuracy	AUC Score		Precision	Recall	F1 Score
BERT-base (cased)	0.8884	0.9624	micro avg.:	0.66	0.64	0.65
			macro avg.:	0.34	0.34	0.34
BERT-base (uncased)	0.8798	0.9700	micro avg.:	0.61	0.71	0.65
			macro avg.:	0.31	0.38	0.34
bare BERT-base (cased)	0.8998	0.9802	micro avg.:	0.69	0.68	0.69
			macro avg.:	0.63	0.51	0.51
bare BERT-base (uncased)	0.8841	0.9839	micro avg.:	0.60	0.78	**0.68**
			macro avg.:	0.60	0.58	**0.57**
bare BERT-large (cased)	0.8795	0.9801	micro avg.:	0.62	0.73	0.67
			macro avg.:	0.53	0.51	0.51
bare BERT-large (uncased)	0.8921	0.9806	micro avg.:	0.67	0.68	0.67
			macro avg.:	0.60	0.41	0.42
DistilBERT (cased)	0.8866	0.9649	micro avg.:	0.63	0.68	0.66
			macro avg.:	0.34	0.37	0.34
DistilBERT (uncased)	0.8649	0.9781	micro avg.:	0.59	0.73	0.65
			macro avg.:	0.48	0.39	0.34
XLNet (cased)	0.9630	0.9530	micro avg.:	0.60	0.70	0.65
			macro avg.:	0.30	0.37	0.33

Table 10 summarizes the model's performances using the cased and uncased version. We also tried to use a BERT-large version of the BERT sequence classification models, but it is probable that those models were over-fitting in the first epoch and the results were worse than in BERT-base version.

Based on the previous experiment, the BERT-base uncased model provided the best results among the transformer models. Following the initial experiments, we proceed with the fine-tuning of the model. We optimized the values of the hyper-parameters summarized in Table 12. Optimization of the hyper-parameters did not lead to a significantly improved performance, however, for the combination of the accuracy and AUC metrics, the best combination of hyper-parameters turned out to be the settings: learning rate = 0.00002, batch size = 16, dropout = 0.15. The results of the fine-tuned model are shown in Table 13. This model also achieved the best micro-averaged F1 metrics.

Table 11. Hyper-parameters used for transformer models from Table 10.

Hyper-Parameters	Values
Batch size	32
Learning rate	0.00003
Dropout rate	0.15

Table 12. Explored combination of the bare BERT-base hyper-parameters during the fine-tuning.

Hyper-Parameters	Values
Dropout rate	`[0.1, 0.15]`
Learning rate	`[0.00002, 0.00003]`
Batch Size	`[16, 32]`

Table 13. Bare BERT-base uncased model performance after the fine-tuning.

	Accuracy	AUC Score		Precision	Recall	F1 Score
bare BERT-BASE uncased	0.8971	**0.9842**	micro avg.:	0.67	0.72	**0.69**
			macro avg.:	0.69	0.51	**0.55**

7.4. Evaluation

From the experiment results with deep learning models, we can consider the GloVe pre-trained embeddings without standard pre-processing as the most suitable representation. The results proved that omitting the traditional pre-processing techniques improve the classification results. This is especially important in the case of macro-averaged metrics, which are more informative in classification tasks with highly imbalanced data. Another important aspect (besides the improvement of the performance metrics) is the demand on resources and computational intensiveness-the pre-processing techniques represent a step in the overall data analysis process and skipping them can reduce the time of the total data preparation phase. On the other hand, pre-processing usually leads to the reduction of the training data dimensionality. When we leave out such a step, we could expect the more resource-demanding training of the models. Another crucial aspect is the deployment of the models in real-world scenarios, where the training time of the model is not essential. In such a case, the ability to process the data and prediction time is essential. Without pre-processing, it is sufficient to create word tokens from the text and apply a trained model to obtain the prediction.

Tables 14–16 depict the BiLSTM + CNN model (with GloVe Twitter and word2vec embeddings) and bare BERT-base uncased model performance on particular classes. In this task, the class imbalance is present and heavily influences the classification. Minor classes (e.g., *severe_toxic* or *threat*) presented a real challenge to learn from the training data. Much better picture about the real quality of the classification into the particular classes is given by the Matthews Correlation Coefficient (MCC) [64]. When considering this metric, both models perform in a similar fashion. Both models struggle with minor classes, with a model trained using GloVe Twitter embeddings performing better on a *severe_toxic* class, while bare BERT handling better the *threat* category. For some models (e.g., for BERT), the lack of training samples from minor classes may present a problem, as some of the BERT modifications were not able to learn some of the minor classes at all. Composed architecture with embeddings was able to learn minor classes; however, in both cases, with at least one metric severely lacking. There may be more reasons why most of the models fail to perform well, even in minor classes. For example analysis of the misclassifications revealed possible problems in the labelling of the data, where numerous comments labelled as *toxic* did not fall into the proper definition of the toxic comments [11]. Besides the questionable labelling, which may have influenced the evaluation of the trained patterns, several NLP-related phenomena may influence the classification, e.g., toxic comments written without any explicit language or written in ordinary style, comments containing sarcasm, irony or metaphors which require the deeper understanding of the content.

To further improve the best performing models, we fine-tuned the thresholds used to convert class probabilities to crisp values. The unbalanced number of samples in individual classes can have an impact on the resulting metrics when transforming the probabilities for each class in a similar manner. Therefore, to improve the model, we used optimization to find the best threshold for each class we trained a separate classifier, to find the optimal set of thresholds for the probabilities, specific for each class. We used the optimization implemented in scipy library, selected F1 as an optimization criterion.

After then, we computed the overall metrics and metrics for particular classes. Tables 17 and 18 show how the performance metrics improved after fine-tuning of the probability thresholds.

Table 14. Results of the individual toxicity classes of biLSTM + CNN model with custom-trained word2vec embeddings without standard pre-processing.

Class	Precision	Recall	F1 Score	MCC	Support
toxic	0.58	0.85	0.69	0.66	6090
severe_toxic	0.35	0.50	0.41	0.42	367
obscene	0.66	0.75	0.71	0.69	3691
threat	0.48	0.07	0.12	0.18	211
insult	0.67	0.68	0.68	0.66	3427
identity_hate	0.73	0.33	0.46	0.49	712

Table 15. Results of the individual toxicity classes of biLSTM + CNN model with pre-trained GloVe Twitter embeddings without standard pre-processing.

Class	Precision	Recall	F1 Score	MCC	Support
toxic	0.56	0.88	0.68	0.66	6090
severe_toxic	0.37	0.46	0.41	0.41	367
obscene	0.61	0.79	0.69	0.67	3691
threat	0.47	0.26	0.34	0.35	211
insult	0.65	0.71	0.68	0.66	3427
identity_hate	0.76	0.42	0.54	0.56	712

Regarding the interpretability of the trained models, we further explored, how the classifiers performed when predicting the actual class labels. We focused on examination of particular comments, especially those, which were classified correctly in multiple categories and on the other hand, the comments that were misclassified. Many comments were categorized correctly, based on the grammar and language used, e.g., the comment "u are a gigantic faggot" was correctly predicted as toxic, obscene, insult, and identity hate. On the other hand, we observed an inconsistency in comments labelling, which is a frequent issue in many hand-labelled datasets. For example, the comment "Bull... Your mom kicking your ass for not studying is influence of Custom. but then again, without all that comic mischief (such as a dead guy and a crying ilha), life would be pretty fukin boring ..." was predicted by the models as toxic and obscene, because of using an explicit language. However, in the testing data, the comment was assigned just with the toxic label. In similar fashion, a comment "oh, shutup, you douchey Wikipedia rent-a-cop" is clearly an insult and was correctly predicted as toxic and insult, but in the testing data was assigned only with the first label.

Table 16. Results of the individual toxicity classes of bare BERT uncased model after fine-tuning.

Class	Precision	Recall	F1 Score	MCC	Support
toxic	0.58	0.87	0.70	0.67	6090
severe_toxic	0.40	0.23	0.29	0.30	367
obscene	0.60	0.80	0.69	0.67	3691
threat	0.63	0.37	0.47	0.48	211
insult	0.66	0.73	0.69	0.67	3427
identity_hate	0.73	0.51	0.60	0.61	712

Table 17. Results of the individual toxicity classes of **optimized** biLSTM + CNN model with pre-trained GloVe Twitter embeddings without standard pre-processing.

Class	Precision	Recall	F1 Score	MCC	Support
toxic	0.59	0.85	0.70	0.67	6090
severe_toxic	0.33	0.66	0.44	0.46	367
obscene	0.61	0.78	0.69	0.67	3691
threat	0.47	0.26	0.34	0.35	211
insult	0.69	0.68	0.68	0.66	3427
identity_hate	0.76	0.43	0.55	0.56	712

Table 18. Results of the individual toxicity classes of optimized and fine-tuned bare BERT uncased model.

Class	Precision	Recall	F1 Score	MCC	Support
toxic	0.64	0.81	0.71	0.69	6090
severe_toxic	0.30	0.64	0.41	0.43	367
obscene	0.67	0.75	0.71	0.69	3691
threat	0.60	0.39	0.47	0.48	211
insult	0.64	0.75	0.69	0.67	3427
identity_hate	0.60	0.71	0.65	0.64	712

7.5. Comparison with Other Models from State-of-the-Art

To compare the models' performance with other similar approaches presented in the literature, we have to consider several aspects. There are numerous studies, in which authors transformed the Kaggle toxic comments competition dataset labels to the binary values (representing toxic and non-toxic values) and solved the binary classification task of toxicity detection [35,65]. On the other hand, when solving the task as a multi-label classification, comparison of these approaches could still be inconsistent, as the evaluation of the models in the literature is not coherent, as multiple studies using different data for evaluation, e.g., a subset of training data (split to 80/20 train/test ratio), or separate labelled testing data, which became available later. Therefore, to compare with the different models, different evaluation of the models had to be implemented. To compare with the results presented in [11,36], we needed to compute the averaged precision and recall values for each class (as a separate classification problems, contrary to multi-label evaluation in previous experiments), and then compute the F1 metric. Overall performance of our approach is compared with several similar approaches using the accuracy, precision, recall, F1 score. Table 19 summarizes the evaluation metrics of the models.

Table 19. Comparison of the performance metrics of the proposed approaches with similar models.

Model	Precision	Recall	F1
BiGRU (GloVe) [11]	0.73	0.85	0.772
BiGRU + Attention (GloVe) [11]	0.73	0.87	0.779
BiGRU + Attention (fastText) [11]	0.74	0.87	0.783
Ensemble model [11]	0.74	0.88	0.791
BiLSTM + CNN + GloVe(Twitter) + No PP	0.78	0.79	0.789
BiLSTM + CNN + GloVe(CC) + No PP	0.75	0.84	0.796
bare BERT-BASE uncased	0.78	0.83	0.801

To analyze how well the models perform on a particular class, we compared the results with other similar approaches described in [66]. Table 20 summarizes and compares our best-performing model with the best performing related models. This time, to be able to compare with the models from the study, we used the accuracy score computed for the specific categories. Please note that the

table contains the results presented in one-vs-all approach (e.g., each class compared to the rest of the others), computed from individual confusion matrices for each specific category.

Table 20. Comparison of the accuracy of the BiLSTM + CNN and bare BERT with other similar models (using Intel300, fastText300, GloVe300, word2vec300 embeddings) from paper [66].

	BiLSTM + CNN	bare BERT	Intel300 [66]	fastText300 [66]	GloVe300 [66]	word2vec300 [66]
toxic	0.9238	0.9382	0.95	0.95	0.93	0.93
severe_toxic	0.9943	0.9894	0.96	0.95	0.96	0.96
obscene	0.9609	0.9644	0.95	0.96	0.96	0.95
threat	0.9970	0.9971	0.96	0.96	0.97	0.96
insult	0.9674	0.9641	0.96	0.96	0.96	0.96
identity_hate	0.9919	0.9914	0.97	0.96	0.96	0.96
averaged	0.9725	0.9741	0.96	0.96	0.96	0.95

8. Conclusions

In work presented in this paper, we aimed to compare and evaluate different current state-of-the-art models for multi-label toxic comments classification. We experimentally evaluated the performance of deep learning models, including composed architectures with different methods of text representation and pre-processing. On top of that, currently, popular transformer language models, such as BERT and its modifications were compared as well. We aimed to explore the assumption that in tasks such as detection of anti-social behaviour in the online environments, the application of traditional pre-processing techniques could lead to loss of particular specific information characteristic for such behaviour. We aimed to explore the influence of different pre-processing and representation methods on the deep learning and transformer models also in multi-label task aimed to detect the specific type of anti-social behaviour (in this case, toxic comments). We experimentally evaluated composed architecture of BiLSTM + CNN network with different text representations, pre-trained embeddings and compared it with BERT and its variants.

Further evaluation of the various data preparation techniques confirmed the assumption, that in this type of task, using standard pre-processing may lead to influence the classifier performance. When comparing the network performance using different word embeddings, the results showed, that the application of traditional text preparation techniques does not bring any significant benefit in terms of evaluation metrics. When comparing the standard classification performance metrics, we face the problem of the class imbalance. The models often struggle to perform well in the minority classes. If the collection of more samples from such classes is not straightforward, more advanced approaches or using data augmentation techniques may be required. One of the possible strategies could be represented by the hierarchical ensemble model. In such an approach, different classifiers could be combined to either address the specific classes, or even to perform the classification on a different level of target attribute generalization. The particular model then could be used to distinguish between the toxic and non-toxic comments and particular toxicity type prediction could be handled by separate models. The class imbalance could be addressed by various techniques, e.g., weighting schemes in the ensemble.

It is also important to note that the presented comparison was carried out on a single dataset. Another limitation of the study is the use of pre-trained word embeddings and pre-trained transformer language models (except word2vec embeddings). Usually, pre-trained embeddings are build using corpora consisting of standard, clean forms of the words. Such representations do not cover the entire vocabulary used in the abusive comments, which results in the loss of information. In future work, we would like to focus on embedding custom-training for this particular domain. A very similar situation is with transfer learning models. In most of the NLP tasks, those models outperform the deep learning models. Pre-trained models may represent not an ideal solution in this case, and their further re-training may lead to superior results. Another factor is the computational intensiveness of

the deep and transformer models training which practically prevents the ultimate comparison of all models, with all pre-processing settings, hyper-parameter fine-tuning using cross-validation technique. In our experiments, we decided to evaluate the models' performance using standard, commonly used settings, then fine-tune the hyper-parameters of the best-performing architecture using grid search on sampled data. Similarly, we assumed the learning variance during the initial experiment using 10-fold cross-validation and did not perform the cross-validation of an entire set of pre-processing experiments, which would result in extreme time and resource-consuming setup.

Author Contributions: Conceptualization, M.S. and V.M.-K.; methodology, M.S.; software, V.M.-K.; validation, V.M.-K.; formal analysis, P.B.; investigation, V.M.-K. and K.M.; resources, M.S.; writing—original draft preparation, V.M.-K. and M.S.; writing—review and editing, V.M.-K., M.S. and P.B.; supervision, P.B.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Slovak Research and Development Agency under the contracts No. APVV-16-0213 and No. APVV-17-0267.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
FFNN	FeedForward Neural Network
NLP	Natural Language Processing
CBOW	Continuous Bag of Words
BCE	Binary Cross Entropy
ReLU	Rectified Linear Unit
AUC	Area Under Curve
ROC	Receiver Operating Characteristic

References

1. Cheng, J.; Danescu-Niculescu-Mizil, C.; Leskovec, J. Antisocial behavior in online discussion communities. In Proceedings of the 9th International Conference on Web and Social Media, University of Oxford, Oxford, UK, 26–29 May 2015,.
2. Elizabeth, B. *Making People Behave: Anti-Social Behaviour, Politics and Policy,* 2nd ed.; Willan Publishing: Cullompton, UK, 2013. [CrossRef]
3. Risch, J.; Krestel, R. Toxic Comment Detection in Online Discussions. In *Deep Learning-Based Approaches for Sentiment Analysis*; Algorithms for Intelligent Systems; Springer: Singapore, 2020. [CrossRef]
4. Dixon, L.; Li, J.; Sorensen, J.; Thain, N.; Vasserman, L. Measuring and Mitigating Unintended Bias in Text Classification. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, New Orleans, LA, USA, 2–3 February 2018. [CrossRef]
5. Morzhov, S. Avoiding Unintended Bias in Toxicity Classification with Neural Networks. In Proceedings of the Conference of Open Innovation Association, Yaroslavl, Russia, 20–24 April 2020. [CrossRef]
6. Mohammad, F. Is preprocessing of text really worth your time for online comment classification? *arXiv* **2018**, arXiv:1806.02908.
7. Singh, T.; Kumari, M. Role of Text Pre-processing in Twitter Sentiment Analysis. *Procedia Comput. Sci.* 2016, *89*, 549–554. [CrossRef]
8. Shelar, A.; Huang, C.Y. Sentiment analysis of twitter data. In Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence, Las Vegas, NV, USA, 13–15 December 2018. [CrossRef]

9. Wang, S.; Manning, C.D. Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 8–14 July 2012.
10. Feldman, R. Techniques and applications for sentiment analysis. *Commun. ACM* **2013**, *56*, 82–89. [CrossRef]
11. van Aken, B.; Risch, J.; Krestel, R.; Löser, A. *Challenges for Toxic Comment Classification: An In-Depth Error Analysis*; In Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Brussels, Belgium, 31 October 2018. [CrossRef]
12. Waseem, Z.; Davidson, T.; Warmsley, D.; Weber, I. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, 30 July–4 August 2017. [CrossRef]
13. Sarnovský, M.; Butka, P.; Bednár, P.; Babič, F.; Paralič, J. Analytical platform based on Jbowl library providing text-mining services in distributed environment. In Proceedings of the Information and Communication Technology, Daejeon, Korea, 4–7 October 2015. [CrossRef]
14. Davidson, T.; Warmsley, D.; Macy, M.; Weber, I. Automated hate speech detection and the problem of offensive language. In Proceedings of the 11th International Conference on Web and Social Media, Montréal, QC, Canada, 15–18 May 2017.
15. Salminen, J.; Almerekhi, H.; Milenković, M.; Jung, S.G.; An, J.; Kwak, H.; Jansen, B.J. Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media. In Proceedings of the 12th International AAAI Conference on Web and Social Media, Stanford, CA, USA, 25–28 June 2018.
16. Almerekhi, H.; Jansen, B.J.; Kwak, H.; Salminen, J. Detecting toxicity triggers in online discussions. In Proceedings of the 30th ACM Conference on Hypertext and Social Media, Hof, Germany, 17–20 September 2019. [CrossRef]
17. Fortuna, P.; Soler, J.; Wanner, L. Toxic, Hateful, Offensive or Abusive? What Are We Really Classifying? An Empirical Analysis of Hate Speech Datasets. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 6786–6794.
18. Shtovba, S.; Shtovba, O.; Petrychko, M. Detection of social network toxic comments with usage of syntactic dependencies in the sentences. In Proceedings of the Second International Workshop on Computer Modeling and Intelligent Systems, Zaporizhzhia, Ukraine, 15–19 April 2019.
19. Saif, M.A.; Medvedev, A.N.; Medvedev, M.A.; Atanasova, T. Classification of online toxic comments using the logistic regression and neural networks models. *AIP Conf. Proc.* **2018**, *2048*, 060011. [CrossRef]
20. Haralabopoulos, G.; Anagnostopoulos, I.; McAuley, D. Ensemble deep learning for multilabel binary classification of user-generated content. *Algorithms* **2020**, *13*, 83. [CrossRef]
21. Chowdhary, N.; Pandit, A.A. Fake Review Detection using Classification. *Int. J. Comput. Appl.* **2018**, *180*, 16–21.
22. Cardoso, E.F.; Silva, R.M.; Almeida, T.A. Towards automatic filtering of fake reviews. *Neurocomputing* **2018**, *309*, 106–116. [CrossRef]
23. Ventirozos, F.K.; Varlamis, I.; Tsatsaronis, G. Detecting aggressive behavior in discussion threads using text mining. In *Proceedings Computational Linguistics and Intelligent Text Processing*; Springer: Cham, Switzerland, 2018. [CrossRef]
24. Machová, K.; Mach, M.; Demková, G. Modelling of the Fake Posting Recognition in On-Line Media Using Machine Learning. *SOFSEM 2020: Theory and Practice of Computer Science*; Chatzigeorgiou, A., Dondi, R., Herodotou, H., Kapoutsis, C., Manolopoulos, Y., Papadopoulos, G.A., Sikora, F., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 667–675.
25. Machova, K.; Staronova, P. Selecting the Most Probable Author of Asocial Posting in Online Media. In Proceedings of the 17th International Conference on Emerging eLearning Technologies and Applications (ICETA), Starý Smokovec, Slovakia, 21–22 November 2019; pp. 480–485. [CrossRef]
26. Anindyati, L.; Purwarianti, A.; Nursanti, A. Optimizing Deep Learning for Detection Cyberbullying Text in Indonesian Language. In Proceedings of the 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, Yogyakarta, Indonesia, 20–22 September 2019. [CrossRef]
27. Al-Ajlan, M.A.; Ykhlef, M. Deep Learning Algorithm for Cyberbullying Detection. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*. [CrossRef]

28. Agrawal, S.; Awekar, A. Deep learning for detecting cyberbullying across multiple social media platforms. In *Advances in Information Retrieval*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018. [CrossRef]
29. Ranasinghe, T.; Zampieri, M.; Hettiarachchi, H. BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identificati on. In *FIRE (Working Notes)*; CEUR-WS: Kolkata, India, 12–15 December 2019.
30. Zimmerman, S.; Fox, C.; Kruschwitz, U. Improving hate speech detection with deep learning ensembles. In Proceedings of the LREC 2018—11th International Conference on Language Resources and Evaluation, Miyazaki, Japan, 7–12 May 2019.
31. Krešňáková, V.M.; Sarnovský, M.; Butka, P. Deep learning methods for Fake News detection. In Proceedings of the 2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo), Szeged, Hungary, 14–16 November 2019; pp. 000143–000148.
32. Mestry, S.; Singh, H.; Chauhan, R.; Bisht, V.; Tiwari, K. Automation in Social Networking Comments with the Help of Robust fastText and CNN. In Proceedings of the 1st International Conference on Innovations in Information and Communication Technology, India, 25–26 April 2019. [CrossRef]
33. Anand, M.; Eswari, R. Classification of abusive comments in social media using deep learning. In Proceedings of the 3rd International Conference on Computing Methodologies and Communication, Mettukadai, India, 27–29 March 2019. [CrossRef]
34. Srivastava, S.; Khurana, P.; Tewari, V. Identifying Aggression and Toxicity in Comments using Capsule Network. In Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), Santa Fe, NM, USA, 25 August 2018.
35. Georgakopoulos, S.V.; Vrahatis, A.G.; Tasoulis, S.K.; Plagianakos, V.P. Convolutional neural networks for toxic comment classification. In Proceedings of the ACM International Conference Proceeding Series, Patras, Greece, 9–12 July 2018. [CrossRef]
36. Ibrahim, M.; Torki, M.; El-Makky, N. Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning. In Proceedings of the 17th IEEE International Conference on Machine Learning and Applications, Orlando, FL, USA, 17–20 December 2018. [CrossRef]
37. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
38. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA,2016.
39. Leshno, M.; Lin, V.Y.; Pinkus, A.; Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* **1993**, *6*, 861–867. [CrossRef]
40. Nair, V.; Hinton, G.E. Rectified linear units improve Restricted Boltzmann machines. In Proceedings of the ICML 2010 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010.
41. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Neurocomputing: Foundations of Research*; Chapter Learning Representations by Back-Propagating Errors; MIT Press: Cambridge, MA, USA, 1988; pp. 696–699.
42. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
43. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
44. Polyak, B. Some methods of speeding up the convergence of iteration methods. *Ussr Comput. Math. Math. Phys.* **1964**, *4*, 1–17. [CrossRef]
45. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
46. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
47. Graves, A. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 5–13.
48. Tavcar, R.; Dedic, J.; Bokal, D.; Zemva, A. Transforming the LSTM Training Algorithm for Efficient FPGA-Based Adaptive Control of Nonlinear Dynamic Systems. *J. Microelectron. Electron. Compon. Mater.* **2013**, *43*, 131–138.
49. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]

50. Cho, K.; Merrienboer, B.; Gülçehre Ç.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014.
51. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to Fine-Tune BERT for Text Classification? In Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Cham, Switzerland, 13 October 2019. [CrossRef]
52. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL HLT 2019—2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MI, USA, 2–7 June 2019.
53. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
54. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2020**, arXiv:1910.01108.
55. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
56. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv* **2020**, arXiv:1906.08237.
57. Sammut, C.; Webb, G.I.; (Eds.) TF–IDF. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2010; pp. 986–987. [CrossRef]
58. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
59. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
60. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *arXiv* **2016**, arXiv:1607.04606.
61. Asch, V.V. *Macro-and Micro-Averaged Evaluation Measures*; CLiPS: Antwerpen, Belgium, 2013.
62. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
63. Gulli, A.; Pal, S. *Deep Learning with Keras*; Packt Publishing Ltd.: Birmingham, UK, 2017.
64. Matthews, B.W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta* **1975**, *405*, 442–451. [CrossRef]
65. Rastogi, C.; Mofid, N.; Hsiao, F.I. Can We Achieve More with Less? Exploring Data Augmentation for Toxic Comment Classification. *arXiv* **2020**, arXiv:2007.00875.
66. Saia, R.; Corriga, A.; Mulas, R.; Reforgiato Recupero, D.; Carta, S. A Supervised Multi-Class Multi-Label Word Embeddings Approach for Toxic Comment Classification. In Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019), Vienna, Austria, 17–19 September 2019. [CrossRef]