Article

An Assessment of Deep Learning Models and Word Embeddings for Toxicity Detection within Online Textual Comments

Danilo Dessì [1,‡*] 0000-0003-3843-3285, Diego Reforgiato Recupero [2,‡] 0000-0001-8646-6183 and Harald Sack [1,‡] 0000-0001-7069-9804

1 FIZ Karlsruhe - Leibniz Institute for Information Infrastructure & Karlsruhe Institute of Technology, (Germany); {danilo.dessi, harald.sack}@fiz-karlsruhe.de

2 Department of Mathematics and Computer Science, University of Cagliari (Italy); diego.reforgiato@unica.it

* Correspondence: diego.reforgiato@unica.it

‡ These authors contributed equally to this work.

Version March 23, 2021 submitted to Electronics

Abstract: Today, due to the explosion of online communication, more and more people are interacting online and a lot of textual comments are being produced. However, a paramount inconvenience within online environments is that comments that are shared within digital platforms can hide hazards such as fake news, insults, harassment, and, more in general, comments that may hurt someone's feelings. In this scenario, the detection of this kind of toxicity has an important role to moderate online communication. Recently, deep learning technologies delivered impressive performance within Natural Language Processing applications encompassing Sentiment Analysis and emotion detection across numerous datasets. Such models do not need any pre-defined hand-picked features, but they learn sophisticated features from the input datasets by themselves. In such a domain, word embeddings have been widely used as a way of representing words in Sentiment Analysis tasks proving to be very effective. Therefore, in this paper, we investigated the use of deep learning and word embeddings to detect six different types of toxicity within online comments. In doing so, the most suitable deep learning layers and state-of-the-art word embeddings to identify toxicity are evaluated. The results suggest that Long-Short Term Memory layers in combination with mimicked word embeddings are a good choice for this task.

Keywords: Deep Learning; Word Embeddings; Toxicity Detection; Binary Classification

---

1. Introduction

In these years, short text information is continuously being created due to the explosion of online communication, social networks, and e-commerce platforms. Through these systems, people can interact with each others, express opinions, engage in discussions, and receive feedback about any topic. However, a paramount inconvenience within online environments is that text spread by digital platforms can hide hazards such as fake news, insults, harassment, and, more in general, comments that may hurt someone's feeling. These comments can be considered as the digital version of personal attacks (e.g., bullying behaviors) that can cause social problems (e.g., racism), and are felt as dangerous and critical by people who are struggling to prevent and avoid them. The risk of such a phenomenon has increased with the event of social networks and more in general within online communication platforms[1]. An attempt to deal with this issue is the introduction of crowdsourcing voting schemes

---

1 https://medium.com/analytics-vidhya/twitter-toxicity-detector-using-tensorflow-js-1140e5ab57ee

which give the possibility to denounce inappropriate comments in online environments to the users. Among many others, Facebook for example allows its users to report a post in terms of violence or hate speech [1]. This scheme allows Facebook to identify fake accounts, offensive comments, etc. However, these methodologies are often inefficient as they fail to detect toxic comments in real time [2], becoming a requirement within social network communities. A toxic post might have been published online much earlier than the time it is reported, and during the time it is online it might cause problems and offenses to several users which might have undesired behaviors (e.g., leaving the underlying social platform). Therefore, detecting toxicity within textual comments through novel technologies has great relevance in the prevention of adverse social effects in a timely and appropriate manner within online environments [3].

In the last years, the use of data for extracting meaningful information to interpret opinions and sentiments of people about various topics has taken hold. Today, textual online data is parsed to predict ratings about online courses [4], sentiments associated to companies and stocks within the financial domain [5] and, recently, healthcare [6], toxicity in online platforms [7]. All these approaches fall within the Sentiment Analysis research topic, which classifies data into positive or negative classes, and includes several subtasks such as emotion detection, aspect-based polarity detection [8], etc. To detect such knowledge, supervised Machine Learning-based systems are designed and provided by the research community to support and improve online services to mine and use the information. To employ supervised Machine Learning based tools, training data is required; however, the amount of labeled data might result insufficient, thus making challenging the design of these tools.

This is more stressed with the spread of Neural Networks and deep learning models, which can reproduce cognitive functions and mimic skills typically performed by the human brain, but need large amount of data to be trained. With the elapse of time, the interest in these technologies as well as their use for the identification of various kinds of toxicity within textual documents are grown [1].

Word embeddings are one of the cornerstones to represent textual data and feed Machine Learning tools. They are representations of words mapped to vectors of real numbers. The first word embedding model (Word2Vec) utilizing Neural Networks was published in 2013 [9] by researchers at Google. Since then, word embeddings are encountered in almost every Natural Language Processing (NLP) model used in practice today. The reason for such a mass adoption is their effectiveness. By translating a word to an embedding it becomes possible to model the semantic importance of a word in a numeric form and thus perform mathematical operations on it. In 2018, researchers at Google proposed the Bidirectional Encoder Representations from Transformers (*BERT*) [10], a deeply bidirectional, unsupervised language representation able to create word embeddings that represent the semantic of words in the context they are used. On the contrary, context-free models (e.g, Word2Vec) generate a single word embedding representation for each word in the vocabulary independently from the word context.

Within this scenario, in this paper various deep learning models fed by word embeddings are designed and evaluated to recognize toxicity levels within textual comments. In details, four deep learning models built by using the Keras[2] framework are designed, and four different types of word embeddings are analysed.

To this aim, the current state-of-the-art toxicity dataset released during the Kaggle challenge on toxic comments[3] is used.

The reader notices that this paper analyses the performances of deep learning and classical Machine Learning approaches (using tf-idf and word embeddings) when tackling the task of toxicity detection. Basically we want to assess whether the syntactic and semantic information lying within the text can provide hints on the presence of certain toxicity classes. In some domains and tasks this is

---

[2] https://keras.io/
[3] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview

not possible: for example, for the problem of identifying empathetic VS non empathetic discussion within answers of a therapist during motivational interviews it has been initially observed that syntactic and semantic information do not provide any clue for the classification task leading to very low accuracies [11]. Thus, for a fair analysis, it is important that the dataset does not contain any unbalanceness. Machine Learning classifiers fail to cope with imbalanced training datasets as they are sensitive to the proportions of the different classes [12]. As a consequence, these algorithms tend to favor the class with the largest proportion of observations, which may lead to misleading accuracies. That is why we preprocessed the mentioned dataset to make it balanced and then applied a 10-fold cross-validation to tackle the proposed task.

Thus, this paper provides the following contributions:

- We analysed four deep learning models based on Dense, Convolutional Neural Network (CNN), and Long-Short Term Memory (LSTM) layers to detect various levels of toxicity within online textual comments.
- We evaluate the use of four word embedding representations based on *Word2Vec* [13,14] and Bidirectional Encoder Representations from Transformer (*BERT*) [15] algorithms for the task of toxicity detection in online textual comments.
- We provide a comparison between deep learning models against common baselines used within classification tasks of textual resources.
- We release contextual word embeddings resource trained on a dataset including toxic comments.
- We also release mimicked word embeddings of tokens that are missing in the pre-trained Google Word2Vec[4] word embeddings.

The source code used for this study is freely available through a GitHub repository[5].

The remainder of this paper is organized as follows. Section 2 includes a literature review and discusses current methods for toxicity detection in textual resources. Section 3 formalizes the problem. Section 4 describes the word embeddings and deep learning models adopted in this research work. Research results and their discussion are reported in Section 5. Finally, Section 6 concludes the paper and illustrates future directions to further tackle the detection of toxic comments.

2. Related Work

A few past works have already addressed the challenge of detecting toxicity within textual comments left by users within online environments. Generally, they rely on Sentiment Analysis methods [16–21] to detect and extract the subjective information and classify emotions and sentiments to determine if a toxicity facet is present or not. For doing so, NLP, Machine Learning, Text Mining, and Computational Linguistics are the most prominent technologies that are employed [22,23]. Sentiment Analysis methods, like many others within the Machine Learning domain, can be mainly split into two categories. i.e., supervised and unsupervised. Supervised techniques require the use of labeled data (training set) to train a model that can be applied to unseen data to predict a sentiment or an emotion [24–26]. These methods often are limited by the lack of labeled data, or by the fact that there are not either good or enough examples for certain categories (e.g., in case of dataset imbalance) [27]. On the other hand, unsupervised Sentiment Analysis approaches usually rely on semantic resources like lexicons, where words are assigned to scores for reflecting words relevance for target categories to infer sentiments and emotions of the input data [28–30]. Both supervised and unsupervised approaches are largely explored in literature for Sentiment Analysis tasks, which include Sentiment Analysis polarity detection (i.e., identifying whether a certain text is either positive or negative) [31], figurative-language uncovering (understanding if the input text if figurative or objective) [23,32], aspect-based polarity detection (e.g., assigning sentiment polarity to features of a certain topic such as the screen of an

4 https://code.google.com/archive/p/word2vec/
5 https://github.com/danilo-dessi/toxicity

iPhone) [33,34], sentiment scores prediction (e.g., identifying a continuous number in [-1,1] to a certain topic or text) [4], and so on.

However, only recently, these methodologies have been explored for toxicity detection [35], although the need to monitor online communications to identify toxicity and make the communications safe and respectful is an old and still open issue. Hence, the gap between the current methodologies and their potential use within toxicity detection remains an open challenge. Therefore, dealing with toxicity raises new challenges and research opportunities where deep learning-based approaches for Sentiment Analysis can have a relevant role in making advancements for the identification of toxicity levels.

Also, Semantic Web technologies are being used within Sentiment Analysis tasks. It has been proved that they bring several benefits leading to higher accuracy [36]. For example, the use of sentiment-based technologies to detect toxicity is investigated in [37]. However, the use of word embedding representation is not taken into account. A work worth noting is [23], where authors analysed the problem of figurative language detection in social media. More in detail, they focused on the use of semantic features extracted with Framester for identifying irony and sarcasm. Semantic features have been extracted to enrich the representation of input tweets with event information using frames and word senses in addition to lexical units. One more example of an unsupervised method that exploits Semantic Web technologies is represented by Sentilo [38,39]. Given a statement expressing an opinion, Sentilo recognizes its holder, detects its related topics and subtopics, links them to relevant situations and events referred to by it, and evaluates the sentiment expressed on each topic/subtopic. Moreover, Sentilo is domain-independent and relies on a novel lexical resource, which enables a proper propagation of the sentiment scores from topics to subtopics. Its output is represented as an RDF graph and, where applicable, it resolves holders' and topics' identity on Linked Data.

Recently, authors in [35] discussed the problem of toxicity detection and proved that context can both amplify or mitigate the perceived toxicity of posts. Besides, they found out no evidence that context actually improves the performance of toxicity classifiers. In another work [40] authors presented an interactive tool for auditing toxicity detection models by visualizing explanations for predictions and providing alternative wordings for detected toxic speech. In particular, they displayed the attention of toxicity detection models on user input, providing suggestions on how to replace sensitive text with less toxic words.

Others, [41], tackled the problem of identifying disguised offensive language, such as adversarial attacks that avoid known toxic patterns and lexicons. To do that, they proposed a framework to fortify existing toxic speech detectors without a large labeled corpus of veiled toxicity. In particular, they augmented the toxic speech detector's training data with new discovered offensive examples.

Deep learning technologies have been leveraged by authors in [42] to tackle the problem of toxic comments detection. More in details, the authors introduced two state-of-the-art neural network architectures and demonstrate how to employ a contextual language representation model.

One more work that deals with a sentiment toxicity detection problem is [7], where authors adopt both pre-trained word embeddings and close-domain word embeddings previously trained on a large dataset of users' comments [43]. However, their approach is based on a Logistic Regression (LR) classifier and does not use state-of-the-art deep learning technologies. Well established methodologies (e.g., k-nearest neighbors (kNN), Naive Bayes (NB), Support Vector Machines (SVM), etc.) are today outperformed for the same tasks by CNN-based models by [44].

One more work for toxicity detection is proposed by authors in [45] and it lies within the context of multiplayer online games. There, social interactions are an essential feature for a growing number of players worldwide. This interaction might bring undesired and unintended behavior especially if the game is designed to be highly competitive. They defined toxicity as the use of profane language by one player to insult or humiliate another player in the same team. Given the specific domain, the use of bad words is a necessary, but not sufficient condition for toxicity as they can be used to curse without the intent to offend anyone. Authors looked at the 100 most frequently used n-grams for n=1,2,3,4 and manually determined which of them are toxic or not. With such training data they use a SVM to predict the odds of winning for each team to observers based on their communication, while the match is still going.

Another work that embraces both deep learning and word embeddings for toxicity detection is reported in [1], where FastText[6] pre-trained embeddings are used to feed four different deep learning models based on CNN, Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) layers. However, the experiments show weak results probably due to the class imbalance of classes. Conversely, in this work deep learning models by using a balanced dataset are trained, considering one toxicity class at a time, and trying to better represent the input texts by using word embeddings tuned to the target domain. More precisely, in one set of experiments, domain generated word embeddings are created through mimicking techniques; this allows to face slang, misspellings, or obfuscated contents not represented within pre-trained word embedding representations [46,47]. Besides the Word2Vec embeddings, state-of-the-art word embeddings called BERT [15,48,49] are used to tune the vectors to the context where words are used.

3. Problem Formulation

The problem faced in this paper is a multi-class multi-label classification problem. We turned it into several binary single-label classification problems. More precisely, given a textual comment c and a toxicity facet t, the approach is aimed to build a deep learning model

$$\gamma : (c, t) \rightarrow l$$

where l is a binary label that can only assume values in $\{0, 1\}$ and indicates if the toxicity t is present in c (i.e., l takes the value 1) or not (i.e., l takes the value 0). Therefore, with such an approach, an independent binary classifier for each toxicity label is trained. Given an unseen sample, each binary classifier predicts whether that underlying toxicity is present or not in the sample. The combined model then predicts all the labels for this sample for which the respective classifier predicts a positive result. Although this method of dividing the task into multiple binary tasks may resemble superficially the one-vs-all and one-vs-rest methods for multi-class classification, it is essentially different from them because a single binary classifier deals with a single label without any regard to other labels whatsoever. This means that each binary classification task we formulated does not benefit from the information of the other labels at training time. However, this mapping is straightforward and does not change the semantic of the input problem [50]. By building these models for various t, the performances of the proposed solutions are evaluated with the goal of finding which combination of the deep learning layers and word embeddings can better capture the text peculiarities for toxicity detection.

4. The Proposed Approach

In this section we will describe the deep learning models and word embedding representations for representing the text expressing the various toxicity categories.

4.1. Preprocessing

Text preprocessing techniques such as stop words and punctuation removal, lemmatization, stemming, matching words with a dictionary to correct grammar, removing words containing alpha-numeric characters, and so on, are common practices when Machine Learning algorithms are applied [51,52], and text representation is generated as a result of different feature engineering processes. However, with the introduction of deep learning approaches, these techniques have not

[6] https://fasttext.cc/docs/en/english-vectors.html

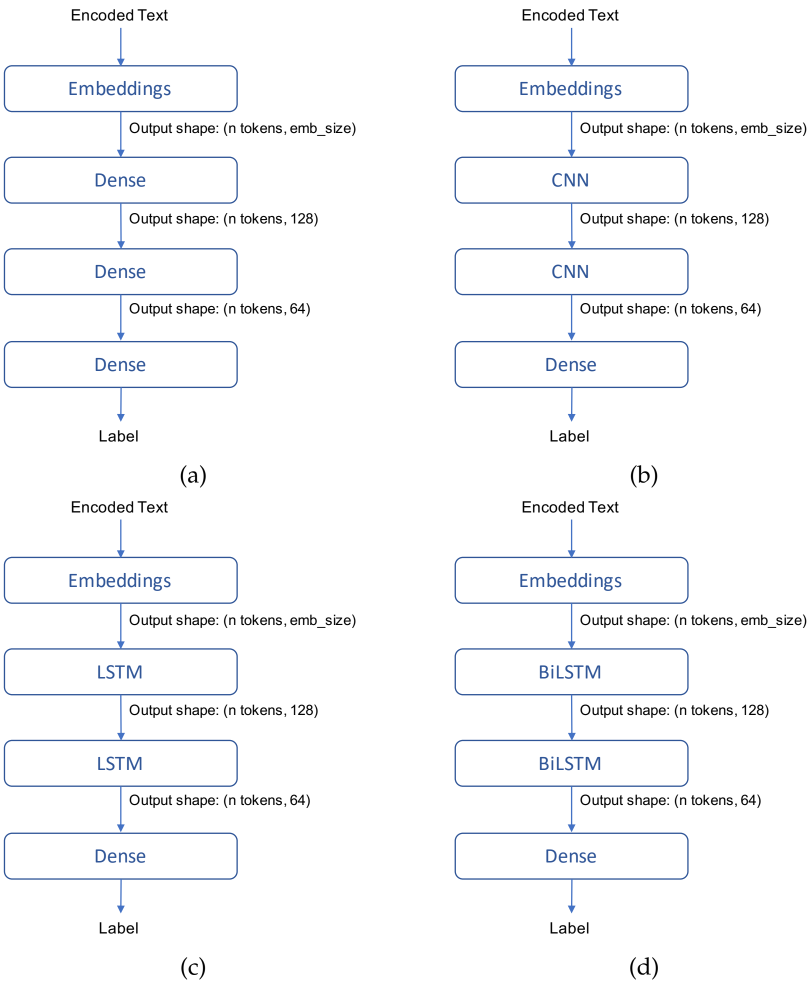

Figure 1. The deep learning models. (a) Dense (b) CNN (c) LSTM (d) Bidirectional LSTM. The output shape of the employed layers is indicated within the parenthesis.

shown promising results. The reason is that neural networks learn from any element found within the text because each token contributes to the sentence semantics. Therefore, although certain terms might be included in existing stop word lists, they are maintained because they can enrich the semantics of text content and improve the performance of the deep learning model [1]. Hence, as suggested by authors in [1], all the above mentioned preprocessing steps are ignored; only the conversion of texts in lower case is performed. Afterward, the whole set of input text is ready to feed a deep learning model. More precisely, imagine to have a toxicity target class t and a set of pairs $P = \{(c_0, l_0), \ldots, (c_n, l_n)\}$, where c_i is a textual comment and l_0 is a binary label that can only take either the value 0 if the comment c_i does not include the toxicity t or 1 if the comment c_i expresses some level of toxicity t. From the set P, the set $P' = \{(c'_0, l_0), \ldots, (c'_n, l_n)\}$ is derived, where each comment c'_i is an integer-encoded comment of the original c_i. In details, let W be the list of all the words belonging to all textual comments, and WS the set of all the words in W without duplicates (i.e., WS has only one occurrence for each input word, whereas W can contain multiple occurrences of the same element). Then, two functions θ and ϕ, which map the elements in W and WS to unique integer values, respectively, are built. For example, consider the sentence *you both shut up or you both die* and imagine to have the toy functions θ_{toy} and ϕ_{toy}. The function θ_{toy} maps "you" to "7", "both" to "43", "shut" to "22", "up" to "76", "or" to "10", "you" to "3", "both" to "41", and "die" to "50". The function ϕ_{toy} maps "you" to "7", "both" to "43", "shut" to "22", "up" to "76", "or" to "10", and "die" to "50". Then the integer-encoded sentence is $[7, 43, 22, 76, 10, 3, 41, 50]$ by applying θ_{toy}, and $[7, 43, 22, 76, 7, 43, 10, 50]$ by using ϕ_{toy}. The reader notices that by using θ_{toy} the words "you" and "both" are mapped to different integers. Within our approach, the function θ is used for *BERT* word embeddings, whereas the function ϕ is used to encode the input text when *Word2Vec* word embeddings are employed.

4.2. Deep Learning Models

The designed deep learning model schemes are shown in Figure 1. In particular, we illustrate four deep learning models based on *Dense*, *CNN*, and *LSTM* layers available within the Keras framework[7]. All the models present the same number of layers. It is worth to note that the input and the output layers among the models are the same to better compare their performances considering only the type of neural network that they adopt. More precisely, the input layer is an *Embedding* layer, which has the goal of mapping the words of the input text to the underlying word embeddings. The last layer is a *Dense* layer that maps the intermediate results of the models in a single label that can only take the values 0 and 1. For doing so, it uses the *sigmoid* activation function to compute a probability that can be easily used to obtain the correct label value. In the next paragraphs we will give more details about the deep learning layers.

The literature already showed [53] that deep learning methods trained with word embeddings outperform those trained with tf-idf features. Therefore, we did not include the latter in our analysis as we believe that they would not add additional value to the current evaluation.

4.2.1. Dense Model

The first model is depicted in Figure 1(a). It is composed of two inner *Dense* layers with 128 and 64 neurons. They are densely-connected layers able to reduce the input size of hundred and thousands of nodes to a few nodes whose weights can be used to predict the final class of the input.

4.2.2. CNN Model

The CNN model depicted in Figure 1(b) is based on inner CNN layers. These layers perform filtering operations to detect meaningful features of textual input for the target toxicity facet. Filters

[7] https://keras.io/

can be envisioned as kernels that slide on the vector representation and perform the same operations on each element until all the vectors have been covered. Two kernels of size 10 for the first layer, and size 5 for the second layer are used. For these layers, the same number of neurons previously introduced for the *Dense* layers is used to better compare the model performances.

4.2.3. LSTM Model

The model depicted in Figure 1(c) exploits the LSTM layers to perform a binary classification of the input text. LSTMs are an extended version of Recurrent Neural Networks (RNN) and are designed to work on sequences. They use memory blocks to hold the state of the computation which makes it possible to learn temporal dependencies of data, binding the chunks of data that are currently being processed with the chunks of data already processed. This allows to infer semantic patterns that describe the history of the input data, solving the problem of common RNN whose results mostly depend on the last seen data fed into the model, smoothing the relevance of data previously processed.

4.2.4. Bidirectional LSTM

The last model, shown in Figure 1(d), is an evolution of the LSTM model. It uses bidirectional LSTM layers to find patterns that can be discovered by exploring the history of the input data in both forward and backward directions. The idea of this kind of network consists of presenting the training data forwards and backward to the two bidirectional LSTM hidden layers whose results are then combined by a common output layer.

4.3. Word Embeddings Representations

In this section, the word embedding representations used to model the syntactic and semantic properties of the words in vectors of real numbers are introduced. Within this work, the employed word embedding representations are *Word2Vec* [13,14] and BERT [15]. We chose the most common sizes for the embeddings, i.e., 300 for *Word2Vec* embeddings and 1024 for BERT word embeddings.

4.3.1. Word2Vec

The *Word2Vec* [13,14] word embedding generator aims to detect the meaning and semantic relations among the words by investigating the co-occurrence of words in documents within a given corpus. The idea behind this algorithm is to model the context of words by exploiting Machine Learning and statistics and come up with a vector representation for each word within the corpus. The resulting word vector representations allow the recognition of relatedness between words. For example, the verbs *capture* and *catch*, which are syntactically different but share common meaning and present analogous co-occurring words, are associated to similar vectors. A *Word2Vec* model can be trained by using either the Continuous Bag-Of-Words (CBOW) or the Skip-gram algorithm. Within our work, the Skip-gram algorithm is adopted because from a preliminary evaluation it obtained higher performances. In details, the following *Word2Vec* word embeddings are used:

- *Pre-trained*. Pre-trained word embeddings released by Google and available online[8]. They are trained on the Google news dataset and contain more than 1 billion words. However, their use can be limited by words that could be misspelled (e.g., words with orthographic errors) or domain-dependent words within the input data. These words are commonly referred to as Out Of Vocabulary (OOV) words.
- *Domain-trained*. Domain-trained word embeddings are trained on the original unbalanced dataset (we merged the training and the test set) provided by the Kaggle challenge. The reader notices that we computed the domain-trained embeddings on the new training sets only (at each iteration

[8] https://code.google.com/archive/p/word2vec/

of the 10-fold cross-validation procedure) of our evaluation strategy. Training the embeddings on the domain data solves the problem of OOV words because for each word it is possible to associate a vector. However, words that are not frequent within our data might have a vector that does not fully and correctly represent words' semantics. The Skip-gram *Word2Vec* algorithm available within the *gensim*[9] library is used. The model is trained using 20 epochs.

- *Mimicked*. Mimicked word embeddings are embeddings of OOV words that are not present within the original model used to represent the text data, but are inferred by exploiting syntactic similarities of words that are in the originally considered vocabulary. More in details, we used the algorithm proposed by [47], which is based on an RNN and works at character level. Words within an original vector model representation are firstly encoded by sequences of characters, and characters are associated with new vector representations. Then, by using a BiLSTM network, an OOV word w is associated to a new word embedding e. To create word embeddings for the OOV words we used the default input dataset, the hyperparameters mentioned in [47] and the pre-trained Word2Vec Google embeddings.

4.3.2. BERT

The BERT word embeddings model was introduced in late 2018 by authors in [15]. It is a novel model of pre-trained language representations that allows the tuning of word vector representations to the meaning that the word has in a given context, overcoming ambiguity issues of words. One of the famous examples is usually reported with the word *bank*. Consider the two sentences "*The man was accused of robbing a bank*" and "*The man went fishing by the bank of the river*". The introduced word embedding models describe the word *bank* with the same word embedding, i.e., they express all the possible meanings with the same vector, and, therefore, cannot disambiguate the word senses based on the surrounding context. On the other hand, BERT produces two different word embeddings, coming up with more accurate representations for the two different meanings. For doing so, BERT computes context-tuned word embeddings resulting in more accurate representations which might lead to better model performances. In this work, the *bert_24_1024_16 BERT* model trained on *book_corpus_wiki_en_cased* is employed and fine-tuned by using the *bert_embedding*[10] library.

4.3.3. Word Embeddings Preparation

To load word embeddings into a deep learning model, they have to be organized into a matrix M. For *Word2Vec* word embeddings, the set WS of words in the input data is used to build M as a matrix of size $(|WS|, 300)$, where each row with index $\phi(w) \mid w \in WS$ (i.e., $row_{\phi(w)}$) contains the word embedding of the word w. If a word w is not present in the *Word2Vec* selected resource (e.g., when only pre-trained word embeddings are used), then $row_{\phi(w)}$ is a row with all its entries set to 0. Similarly, when the *BERT* embeddings are employed, the matrix M size is $(|W|, 1024)$, where each row with index $\theta(w) \mid w \in W$ (i.e., $row_{\theta(w)}$) contains the word embedding of the word w. The generated matrix M is loaded into the *Embedding* layer of the employed deep learning model to map the encoded textual comments to the correct word embeddings.

5. Experimental study

In this section we describe the dataset used to perform our experiments, the obtained results, and the related discussion. All the experiments are run by using a 10-fold cross-validation setup. Each model is trained with batches of size 128. The model is configured to train at most with 20 epochs. However, an early stopping method with patience of 5 epochs and a delta of 0.05 that monitors the accuracy of the model are embedded within the training stage. The loss function used to train the

---

[9] https://radimrehurek.com/gensim/
[10] https://pypi.org/project/bert-embedding/

Table 1. Number of textual comments for each class.

Toxicity class	Number of comments	Percentage	Balanced dataset size
No toxic	201,081	89.95%	-
toxic	21,384	9.57%	42,768
severe toxic	1,962	0.88%	3,924
obscene	12,140	5.43%	24,280
threat	689	0.31%	1,378
insult	11,304	5.06%	22,608
identity hate	2,117	0.95%	4,234

models is the *binary crossentropy* and the used optimizer is *rmsprop* with the default learning rate 0.001 provided by the used library. The domain-trained word embeddings have been computed on the training sets only at each iteration of the 10-fold cross-validation procedure. All the other parameters have been empirically set on the basis of the models performance and previous experiences in past works [4,46]. The experiments have been carried out on a Titan X GPU mounted on a server with 16 GB of RAM memory.

5.1. The Dataset

To perform our analysis we employed the dataset released by a Kaggle competition[11]. The dataset is collected from Wikipedia comments that have been manually labeled into 6 different toxicity classes. It consists of training and test files. However, the original split is not kept in order to apply the proposed approach and balance the data. The dataset is composed of more than 200k comments and presents annotations for six different toxicity classes and one more class when no toxicity is present. Table 1 reports the number of comments and the related percentage concerning the original dataset (second and third columns) belonging to each of the seven resulting classes. The first row includes the comments that do not present toxicity, then from the second row on, the number of comments for each toxicity class ($toxic, severetoxic, obscene, threat, insult, identityhate$) are reported. Besides, from Table 1 it is worth to note that the dataset is strongly unbalanced as nearly 90% of the overall comments do not present toxicity. Therefore, as mentioned early in the paper, the training of a model is biased because the model does not have a sufficient number of examples of the minority class to correctly identify a pattern. A random model that always predicts the majority class can obtain better performances although it is not be able to recognize elements that should belong to the minority class. Hence, having a balanced dataset is a common procedure in several classification tasks [54] and allows understanding better the performances of a model [12]. It follows that for each toxicity class we built a dataset where the number of positive examples (i.e., comments that present the target toxicity class) and the number of negative examples (i.e., comments that do not present that target toxicity class) are the same. The size of the created datasets for each class are reported in Table 1 under the *Balanced dataset size* column. The reader notices that, for a certain toxicity class, the negative examples are chosen among all the other classes including the *No toxic* comments.

5.2. Baselines

For evaluation purposes, the deep learning models have been compared to a certain number of baselines. These are classical Machine Learning classifiers that are usually employed with the *tf-idf* to

[11] https://www.kaggle.com/

represent textual resources [51]. More precisely, the deep learning models are compared against the following classifiers:

- **Decision Tree (DT)**. The Decision Tree algorithm builds a model by learning decision rules that when applied to the input features can correctly predict the target class. The model has a root node that represents the whole set of input data. This node is subsequently split into its children by applying a given rule. The process is then applied to its children recursively as long as there are nodes that can be split.
- **Random Forest (RF)**. This method adopts more DTs applied on different samples of the input data and uses a majority voting strategy to predict the output classes. The strength of this algorithm is that each DT is individually trained; therefore, overfitting and errors due to biases are limited. We adopted a classifier that made use of 100 DTs estimators.
- **Multi-Layer Perceptron (MLP)**. This is a neural network that is composed of a single layer of nodes. In our experiment, we used a layer with 100 nodes.

For these classical Machine Learning methods employed as baselines the adoption of just word embeddings is not promising and this has already been shown in literature [55]. In particular, when employing word embeddings for classical Machine Learning methods, they should be processed by operations such as the average or the sum before being fed to a given classifier. This causes loss of syntactic and semantic information expressed by the embeddings of each word.

To develop the algorithms above we employed the *scikit-learn*[12] library.

Additionally, the area under the ROC (Receiver Operating Characteristic) curve (ROC-AUC) is also reported in Table 2 in order to understand the performance of our model with respect to the best models proposed for the challenge's task.

Table 2. ROC-AUC values of our deep learning models on each binary classification and average for each model.

Learning Model	Word Embeddings	Toxic	Severe Toxic	Obscene	Threat	Identity Hate	Insult	Average
Deep Model Dense	pre-trained	0.921	0.968	0.936	0.977	0.944	0.933	0.947
	domain-trained	0.915	0.959	0.928	0.968	0.934	0.924	0.938
	mimicked	0.922	0.969	0.938	0.981	0.941	0.931	0.947
	bert	0.898	0.964	0.904	0.945	0.924	0.906	0.924
Deep Model CNN	pre-trained	0.905	0.964	0.924	0.969	0.934	0.915	0.935
	domain-trained	0.895	0.950	0.857	0.957	0.909	0.903	0.912
	mimicked	0.906	0.961	0.923	0.974	0.935	0.914	0.936
	bert	0.881	0.952	0.894	0.909	0.892	0.895	0.904
Deep Model LSTM	pre-trained	0.970	0.982	0.980	0.983	0.968	0.976	0.977
	domain-trained	0.963	0.980	0.977	0.983	0.968	0.970	0.974
	mimicked	0.971	0.983	0.977	0.985	0.970	0.977	0.977
	bert	0.930	0.974	0.940	0.956	0.950	0.940	0.948
Deep Model Bidirectional LSTM	pre-trained	0.969	0.981	0.973	0.984	0.967	0.975	0.975
	domain-trained	0.963	0.980	0.977	0.984	0.964	0.970	0.973
	mimicked	0.969	0.963	0.980	0.988	0.970	0.976	0.974
	bert	0.930	0.970	0.939	0.951	0.947	0.941	0.946

[12] https://scikit-learn.org/stable/index.html

Table 3. Precision (p), recall (r), and f-measure (f) related to the binary classification for each toxicity class using the balanced dataset.

Learning Model	Word Embeddings	Toxic			Severe Toxic			Obscene		
		p	r	f	p	r	f	p	r	f
Decision Trees	tf-idf	0.859	0.855	0.857	0.847	**0.947**	0.894	0.926	**0.929**	**0.928**
Random Forests	tf-idf	**0.860**	0.856	**0.858**	0.888	0.940	0.913	**0.945**	0.834	0.913
MLP	tf-idf	0.849	**0.857**	0.853	**0.913**	0.918	**0.915**	0.884	0.895	0.889
Deep Model Dense	pre-trained	0.863	**0.856**	**0.858**	0.923	0.910	0.916	**0.886**	0.867	0.876
	domain-trained	0.855	0.848	0.851	0.893	0.910	0.899	0.874	0.863	0.867
	mimicked	**0.868**	0.844	0.855	**0.926**	0.914	**0.919**	0.880	**0.877**	**0.878**
	bert	0.828	0.817	0.822	0.912	**0.917**	0.913	0.844	0.821	0.832
Deep Model CNN	pre-trained	**0.848**	0.849	0.848	**0.910**	0.911	**0.909**	**0.863**	0.861	0.861
	domain-trained	0.846	0.841	0.842	0.903	0.875	0.888	0.858	0.849	0.853
	mimicked	0.836	**0.865**	**0.850**	0.886	**0.919**	0.901	0.856	**0.870**	**0.862**
	bert	0.801	0.812	0.805	0.899	0.911	0.904	0.819	0.832	0.825
Deep Model LSTM	pre-trained	**0.914**	0.915	0.914	0.944	0.962	**0.953**	0.927	**0.949**	**0.938**
	domain-trained	0.903	0.916	0.909	**0.947**	0.948	0.947	**0.929**	0.944	0.936
	mimicked	0.895	**0.938**	**0.916**	0.941	**0.966**	**0.953**	0.928	0.938	0.932
	bert	0.866	0.851	0.858	0.927	0.932	0.929	0.889	0.861	0.875
Deep Model Bidirectional LSTM	pre-trained	0.906	**0.923**	0.914	0.936	0.959	0.947	**0.963**	0.854	0.905
	domain-trained	0.905	0.915	0.910	**0.948**	0.962	**0.955**	0.941	0.933	**0.937**
	mimicked	**0.910**	0.921	**0.915**	0.939	**0.963**	0.951	0.929	**0.945**	**0.937**
	bert	0.875	0.841	0.856	0.933	0.941	0.937	0.892	0.852	0.871

Learning Model	Word Embeddings	Threat			Identity Hate			Insult		
		p	r	f	p	r	f	p	r	f
Decision Trees	tf-idf	0.917	0.891	0.903	0.819	**0.927**	0.869	0.887	**0.891**	**0.889**
Random Forests	tf-idf	**0.954**	0.897	**0.924**	0.847	0.911	0.877	**0.929**	0.851	0.888
MLP	tf-idf	0.914	**0.916**	0.913	**0.889**	0.897	**0.893**	0.871	0.880	0.876
Deep Model Dense	pre-trained	**0.934**	0.930	**0.931**	**0.897**	0.865	0.879	0.872	0.865	**0.869**
	domain-trained	0.913	0.918	0.914	0.858	0.877	0.866	**0.876**	0.846	0.860
	mimicked	0.933	**0.932**	**0.931**	0.881	**0.882**	**0.880**	0.873	**0.857**	0.863
	bert	0.867	0.891	0.877	0.874	0.865	0.855	0.841	0.827	0.834
Deep Model CNN	pre-trained	**0.932**	0.870	0.891	**0.872**	0.863	0.867	0.842	**0.862**	**0.851**
	domain-trained	0.898	0.899	0.898	0.823	0.868	0.842	**0.874**	0.816	0.843
	mimicked	0.927	**0.918**	**0.922**	0.860	**0.879**	**0.869**	0.847	0.849	0.847
	bert	0.842	0.872	0.849	0.824	0.842	0.832	0.831	0.821	0.826
Deep Model LSTM	pre-trained	0.932	**0.967**	0.948	0.907	0.909	0.906	0.918	0.939	0.928
	domain-trained	0.949	0.951	0.950	**0.913**	0.925	**0.918**	**0.919**	0.930	0.924
	mimicked	**0.953**	0.962	**0.957**	0.887	**0.946**	0.914	0.916	**0.948**	**0.931**
	bert	0.916	0.899	0.907	0.880	0.895	0.886	0.874	0.870	0.872
Deep Model Bidirectional LSTM	pre-trained	0.946	**0.961**	**0.952**	**0.905**	0.921	0.912	0.918	0.931	0.924
	domain-trained	**0.949**	0.949	0.949	0.904	**0.935**	**0.919**	0.918	**0.938**	**0.927**
	mimicked	0.941	0.944	0.940	0.902	0.934	0.916	**0.920**	0.935	**0.927**
	bert	0.913	0.900	0.905	0.900	0.857	0.874	0.889	0.866	0.877

5.3. Results and Discussion

In this section, we discuss the results of the experiments we have carried. They are reported in Tables 2 and 3 in terms of ROC-AUC, precision, recall, and f-measure scores (for computing the ROC-AUC, the true positive rates and false positive rates are computed accordingly to Equations (1) and (2); precision, recall and f-measure are computed according to Equations (3), (4), and (5)). In the equations, TP (true positives) is the number of comments with the target toxicity class correctly guessed by the model, FP (false positives) is the number of comments erroneously associated to a target toxicity class, TN (true negatives) is the number of comments that the classifier correctly does not classify for a target class, and FN (false negatives) is the number of comments erroneously classified with a class different than the target class.

$$True\ positive\ rate = \frac{TN}{TN + FP} \tag{1}$$

$$False\ positive\ rate = \frac{FP}{FP + TN} \tag{2}$$

$$Precision\ (p) = \frac{TP}{TP + FP} \tag{3}$$

$$Recall\ (r) = \frac{TP}{TP + FN} \tag{4}$$

$$F - measure\ (f) = 2 \cdot \frac{P \cdot R}{P + R} \tag{5}$$

Results depicted in Table 3 show how the deep learning models perform against the baselines (classical Machine Learning approaches). For each deep learning model, the performance of the model in combination with the embedding representations is illustrated as well.

5.4. Comparison with the Kaggle Challenge

The results indicated in Table 2 report the ROC-AUC values of our deep learning approaches for each toxicity class and the average over all the classes. The reader notices that it is not the purpose of this paper to compete with the other participants of the Kaggle challenge where the data have been extracted and the evaluation has been reported using the ROC-AUC. The best three approaches of the challenge were *Toxic Crusaders*, *neongen & Computer says no*, and *Adversarial Autoencoder*, which reported a ROC-AUC value of 0.989, 0.988, and 0.988, respectively. The challenge task was to test any proposed approach on a highly unbalanced dataset. In this paper we wanted to study how deep learning methods and classical Machine Learning approaches (using tf-idf and word embeddings) perform on the toxicity problem without any bias (unbalanceness of the data). Moreover, it has been proved that optimizing a method for the ROC-AUC does not guarantee the optimization on the precision-recall curve [56]. This is why we included Table 3 with precision, recall and f-measure metrics computed on the preprocessed balanced dataset. There are several heuristics and tuning that can be done in presence of unbalanced datasets to help achieving high values of ROC-AUC. Those could not be performed by us since we used a balanced version of the original dataset.

5.4.1. Baseline Comparison

The results indicate that *Dense*- and *CNN*-based models are not much better than the baseline methods. Actually, in some cases, they are outperformed. For example, considering the toxicity classes *obscene* and *insult*, it is possible to observe that the f-measure computed on the baseline predictions is higher than the one obtained by *Dense*- and *CNN*-based models. On the other hand, *LSTM*-based models are able to outperform the baseline methods with a minimum improvement in terms of f-measure of 0.01, i.e., in percentage 1% (see *obscene* class), and a maximum of 0.058, i.e., in percentage 5.8% (see *toxic* class). These results are similar, and sometimes still more noticeable when the *Bidirectional LSTM* layers are employed. Moreover, considering that by using the balanced dataset every classifier is able to obtain a f-measure always higher than 0.8, the improvements can be considered remarkable. The only drawback is related to the computational time needed to train the deep learning model. Nevertheless, the training time is not reported since i) it is out of the scope of this study ii) with modern GPUs it is feasible to train complex deep learning models iii) the training step must be executed only once, and iv) the computational time needed for the prediction step does not depend on the underlying model used for the training step.

5.4.2. Dense-based Model

For the task of toxicity detection the *Dense*-based model never obtains the best performances. In most of the cases, the best results with this model are obtained with the *mimicked* word embeddings where for four out of six classes the achieved f-measure score is the highest. The *pre-trained* word embeddings obtain high performances too, especially for classes such as *Toxic*, *Threat* (in this case the f-measure is very close to the case when using *mimicked*), and *Insult*. The use of *domain-trained* word embeddings never meets high scores, except when the *precision* is considered for the *Insult* class. Similarly, *BERT* word embeddings performances are the worst.

5.4.3. CNN-based Model

Using the *CNN*-based model the results do not improve further with respect to the *Dense*-based model. In some cases, the performances of the model are even lower. With this model, the best results are obtained by employing the *mimicked* word embeddings for the toxicity classes *Toxic*, *Obscene*, *Threat*, and *Identity Hate*. For the other toxicity classes, the best results are obtained using the *pre-trained* word embeddings. *Domain-trained* and *BERT* embeddings are not able to properly represent the domain knowledge for the *CNN* model, thus the results are poor.

5.4.4. LSTM-based Model

The *LSTM* model outperforms both *Dense* and *CNN*-based models, proving its suitability to detect patterns for toxic detection. As previously mentioned, *mimicked* word embeddings are employed for the deep learning model to learn and uncover toxicity from the text comments. *Pre-trained* and *Domain-trained* word embeddings obtain good performances, and their results are not far from the model using the *mimicked* word embeddings. On the other hand, once again *BERT* is not a good representation for the *LSTM* model. Except for BERT, the three other word embeddings adopted with the *LSTM* model outperform the baseline methods for almost each toxicity level.

5.4.5. BiLSTM-based Model

Although the higher complexity of the employed layers, the results of the *BiLSTM* (Bidirectional LSTM) model are similar to those obtained by the *LSTM* model. In some cases, the *BiLSTM* is able to outperform the *LSTM*, in others it is not. Moreover, it differs from the other models because its best performances for many classes are obtained using the *domain-trained* word embeddings. The *pre-trained* and *mimicked* word embeddings continued to show good ability to represent domain knowledge, and *BERT* embeddings confirm to be the last choice for the task of toxicity detection. Similarly to the *LSTM* model, except using BERT, the model outperforms the baselines in almost each toxicity class.

5.4.6. Overall evaluation of the deep learning models

The use of deep learning for the task of toxicity detection has shown good performances in all the toxicity classes. Also, it turns out that although the small size of datasets employed for certain classes, they are able to detect patterns that allow to correctly perform the classification. More in details, the results suggest that the *Dense* and *CNN* models perform well since their f-measure is always higher than 0.8 but, for the toxicity detection task, they are outperformed by the *LSTM* and *BiLSTM* models, which obtain a f-measure higher than 0.9 in most of the cases. Results are comparable among the *LSTM* and *BiLSTM* models. However, because *BiLSTM*-based models need higher computational time to be trained than *LSTM* models, the latter are slightly preferred. It is worth to mention that the current models are trained without the context that was surrounding the comments in the Wikipedia pages (where the dataset has been originally collected) and, therefore, they might lack the necessary information to predict the correct class. One more obstacle might be also due to the presence of figurative language within the comments, which might change the meaning of the sentences, thus misleading the models. For example, a frequent sentence like *I am going to kill you* pronounced after a mistake or an undesired change in the Wikipedia pages does not necessarily convey a threat or hate emotion but it may be simply a joke.

5.4.7. Overall evaluation of word embeddings

From the results, it is noticeable that the *Word2Vec* algorithm is a good choice to represent textual resources to be parsed with deep learning models. Results suggest that *mimicked* word embeddings are the best choice because they enclose the knowledge of *pre-trained* word embeddings that have been built on a large dataset and do not suffer from the OOV words problem [46]. *Domain-trained* word embeddings obtain good results but, for most of the cases, they are outperformed. This may depend on the fact that the resources employed to train these embeddings are not very large and, besides, there are not a sufficient number of examples of toxicity due to the unbalanced number of toxic comments in the input dataset (i.e., more than 200*k* comments do not present toxicity, the reader can see Table 1).

Surprisingly, *BERT* embeddings perform badly for the task of toxicity detection although they are currently the state-of-the-art word embedding representations. A possible motivation behind this finding is that assigning a different embedding to the same word is somehow misleading to the training of the deep learning models. More precisely, the tuning step performed to generate the BERT embeddings on our data is not able to capture the context of the words due to the length of some input textual comments and to the typos and incorrect grammar often present within them, thus transferring possible erroneous information to our deep learning models. One more reason might be due to the lack of the surrounding context of the comments; it might have limited the fine-tuning of the model, therefore leading the semantics of words to be captured badly. This fact is worth to be investigated, and a close analysis to this problem is required.

6. Conclusion and Future Work

In this paper, we presented an assessment of various deep learning models fed by various word embedding representations to detect toxicity within textual comments. From the obtained results we can definitely state that toxicity can be identified by machine and deep learning approaches fed with syntactic and semantic information extracted from the text. We show how *LSTM*-based model is the first choice among the experimented models to detect toxicity. We also show how various word embeddings may represent the domain knowledge in a variety of ways, and an unique model for all cases might be insufficient. In particular, the results are encouraging when using mimicking techniques to deal with OOV words where there are not many examples to build significant domain-dependent word embeddings. As future works, we plan to perform a deeper assessment of deep learning models by using and combining different layers, to better detect patterns and on real scenarios where classes may be unbalanced as well. Moreover, we would like to investigate other contextualized word embedding representations such as *ELMO* [57] for the toxicity detection task. An analysis of the proposed approaches on which configuration, parameter settings and heuristic may be added to tackle the same problem but in presence of highly unbalanced datasets is definitely a research direction we would like to investigate as well. Finally, we would like to investigate the impact of using different embeddings for the same word since it might be the cause of failure of *BERT* embeddings in our experiments. We also think that an ensemble strategy of the proposed approaches should result in better overall performances and are then investigating this direction as well.

Acknowledgments: We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

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

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area under the curve
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long-Short Term Memory
CBOW	Continuous Bag-Of-Words
CNN	Convolutional Neural Network
DT	Decision Tree
ELMO	Embeddings from Language Models
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
kNN	k-Nearest Neighbors
LR	Logistic Regression
LSTM	Long-Short Term Memory
MLP	Multi-Layer Perceptron
NB	Naive Bayes
NLP	Natural Language Processing
OOV	Out Of Vocabulary
RF	Random Forest
ROC	Receiver Operating Characteristic
RNN	Recurrent Neural Network
TF-IDF	Term Frequency–Inverse Document Frequency
SVM	Support Vector Machine

References

1. Saeed, H.H.; Shahzad, K.; Kamiran, F. Overlapping Toxic Sentiment Classification Using Deep Neural Architectures. 2018 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, 2018, pp. 1361–1366.
2. Hosseini, H.; Kannan, S.; Zhang, B.; Poovendran, R. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138* **2017**.
3. Srivastava, S.; Khurana, P.; Tewari, V. Identifying aggression and toxicity in comments using capsule network. Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), 2018, pp. 98–105.
4. Dessì, D.; Dragoni, M.; Fenu, G.; Marras, M.; Recupero, D.R. Evaluating neural word embeddings created from online course reviews for sentiment analysis. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. ACM, 2019, pp. 2124–2127.
5. Dridi, A.; Atzeni, M.; Recupero, D.R. FineNews: fine-grained semantic sentiment analysis on financial microblogs and news. *International Journal of Machine Learning and Cybernetics* **2019**, *10*, 2199–2207.
6. Consoli, S.; Dessì, D.; Fenu, G.; Marras, M. Deep Attention-based Model for Helpfulness Prediction of Healthcare Online Reviews. Proceedings of the First Workshop on Smart Personal Health Interfaces co-located with 25th International Conference on Intelligent User Interfaces, SmartPhil@IUI 2020, Cagliari, Italy, March 17, 2020, 2020, pp. 33–49.
7. Carta, S.; Corriga, A.; Mulas, R.; Recupero, D.R.; Saia, R. A Supervised Multi-class Multi-label Word Embeddings Approach for Toxic Comment Classification. Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Vienna, Austria, 2019, pp. 17–19.

8. Schouten, K.; Frasincar, F. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering* **2015**, *28*, 813–830.
9. Mikolov, T.; Corrado, G.; Chen, K.; Dean, J. Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 2013, pp. 1–12.
10. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers); Association for Computational Linguistics: Minneapolis, Minnesota, 2019; pp. 4171–4186. doi:10.18653/v1/N19-1423.
11. Wu, Z.; Helaoui, R.; Kumar, V.; Recupero, D.R.; Riboni, D. Towards Detecting Need for Empathetic Response in Motivational Interviewing. Companion Publication of the 2020 International Conference on Multimodal Interaction, ICMI Companion 2020, Virtual Event, The Netherlands, October, 2020; Truong, K.P.; Heylen, D.; Czerwinski, M.; Berthouze, N.; Chetouani, M.; Nakano, M., Eds. ACM, 2020, pp. 497–502. doi:10.1145/3395035.3425228.
12. Dragoni, M.; Petrucci, G. A neural word embeddings approach for multi-domain sentiment analysis. *IEEE Transactions on Affective Computing* **2017**, *8*, 457–470.
13. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* **2013**.
14. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 2013, pp. 3111–3119.
15. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* **2018**.
16. Reforgiato Recupero, D.; Cambria, E. ESWC 14 challenge on Concept-Level Sentiment Analysis. *Communications in Computer and Information Science* **2014**, *475*, 3–20.
17. Recupero, D.R.; Dragoni, M.; Presutti, V. ESWC 15 Challenge on Concept-Level Sentiment Analysis. Semantic Web Evaluation Challenges - Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers, 2015, pp. 211–222. doi:10.1007/978-3-319-25518-7_18.
18. Recupero, D.R.; Consoli, S.; Gangemi, A.; Nuzzolese, A.G.; Spampinato, D. A Semantic Web Based Core Engine to Efficiently Perform Sentiment Analysis. The Semantic Web: ESWC 2014 Satellite Events; Presutti, V.; Blomqvist, E.; Troncy, R.; Sack, H.; Papadakis, I.; Tordai, A., Eds.; Springer International Publishing: Cham, 2014; pp. 245–248.
19. Dragoni, M.; Reforgiato Recupero, D. Challenge on fine-grained sentiment analysis within ESWC2016. *Communications in Computer and Information Science* **2016**, *641*, 79–94.
20. Reforgiato Recupero, D.; Cambria, E.; Di Rosa, E. Semantic sentiment analysis challenge at ESWC2017. *Communications in Computer and Information Science* **2017**, *769*, 109–123.
21. Kumar, V.; Recupero, D.R.; Riboni, D.; Helaoui, R. Ensembling Classical Machine Learning and Deep Learning Approaches for Morbidity Identification From Clinical Notes. *IEEE Access* **2021**, *9*, 7107–7126. doi:10.1109/ACCESS.2020.3043221.
22. Dridi, A.; Recupero, D.R. Leveraging semantics for sentiment polarity detection in social media. *International Journal of Machine Learning and Cybernetics* **2019**, *10*, 2045–2055.
23. Recupero, D.R.; Alam, M.; Buscaldi, D.; Grezka, A.; Tavazoee, F. Frame-Based Detection of Figurative Language in Tweets [Application Notes]. *IEEE Computational Intelligence Magazine* **2019**, *14*, 77–88.
24. Poria, S.; Cambria, E.; Gelbukh, A. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. Proceedings of the 2015 conference on empirical methods in natural language processing, 2015, pp. 2539–2544.
25. Tang, D.; Qin, B.; Liu, T. Document modeling with gated recurrent neural network for sentiment classification. Proceedings of the 2015 conference on empirical methods in natural language processing, 2015, pp. 1422–1432.
26. Atzeni, M.; Recupero, D.R. Multi-domain sentiment analysis with mimicked and polarized word embeddings for human-robot interaction. *Future Gener. Comput. Syst.* **2020**, *110*, 984–999. doi:10.1016/j.future.2019.10.012.
27. Yin, H.; Gai, K. An empirical study on preprocessing high-dimensional class-imbalanced data for classification. 2015 IEEE 17th International Conference on High Performance Computing and

Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. IEEE, 2015, pp. 1314–1319.
28. Momtazi, S. Fine-grained German Sentiment Analysis on Social Media. Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012, 2012, pp. 1215–1220.
29. Rothfels, J.; Tibshirani, J. Unsupervised sentiment classication of English movie reviews using automatic selection of positive and negative sentiment items. Technical Report, Stanford University, 2010.
30. Cheng, K.; Li, J.; Tang, J.; Liu, H. Unsupervised sentiment analysis with signed social networks. Thirty-First AAAI Conference on Artificial Intelligence, 2017.
31. Tripathy, A.; Agrawal, A.; Rath, S.K. Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications* **2016**, *57*, 117–126.
32. Reyes, A.; Rosso, P.; Buscaldi, D. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering* **2012**, *74*, 1 – 12. Applications of Natural Language to Information Systems.
33. Hamdan, H.; Bellot, P.; Bechet, F. Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis. Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015), 2015, pp. 753–758.
34. Dragoni, M.; da Costa Pereira, C.; Tettamanzi, A.G.; Villata, S. Combining argumentation and aspect-based opinion mining: the smack system. *AI Communications* **2018**, *31*, 75–95.
35. Pavlopoulos, J.; Sorensen, J.; Dixon, L.; Thain, N.; Androutsopoulos, I. Toxicity Detection: Does Context Really Matter? Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; Association for Computational Linguistics: Online, 2020; pp. 4296–4305. doi:10.18653/v1/2020.acl-main.396.
36. Saif, H.; He, Y.; Alani, H. Semantic Sentiment Analysis of Twitter. Proceedings of the 11th International Conference on The Semantic Web - Volume Part I; Springer-Verlag: Berlin, Heidelberg, 2012; ISWC12, pp. 508–524.
37. Brassard-Gourdeau, E.; Khoury, R. Subversive toxicity detection using sentiment information. Proceedings of the Third Workshop on Abusive Language Online, 2019, pp. 1–10.
38. Gangemi, A.; Presutti, V.; Recupero, D.R. Frame-Based Detection of Opinion Holders and Topics: A Model and a Tool. *IEEE Comp. Int. Mag.* **2014**, *9*, 20–30. doi:10.1109/MCI.2013.2291688.
39. Recupero, D.R.; Presutti, V.; Consoli, S.; Gangemi, A.; Nuzzolese, A.G. Sentilo: Frame-Based Sentiment Analysis. *Cognitive Computation* **2015**, *7*, 211–225. doi:10.1007/s12559-014-9302-z.
40. Wright, A.; Shaikh, O.; Park, H.; Epperson, W.; Ahmed, M.; Pinel, S.; Yang, D.; Chau, D.H. RECAST: Interactive Auditing of Automatic Toxicity Detection Models. Conference: Chinese CHI 2020: The eighth International Workshop of Chinese CHI, 2020, pp. 80–82. doi:10.1145/3403676.3403691.
41. Han, X.; Tsvetkov, Y. Fortifying Toxic Speech Detectors Against Veiled Toxicity. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP); Association for Computational Linguistics: Online, 2020; pp. 7732–7739. doi:10.18653/v1/2020.emnlp-main.622.
42. Morzhov, S. Avoiding Unintended Bias in Toxicity Classification with Neural Networks. 2020 26th Conference of Open Innovations Association (FRUCT), 2020, pp. 314–320. doi:10.23919/FRUCT48808.2020.9087368.
43. Dessì, D.; Fenu, G.; Marras, M.; Recupero, D.R. COCO: Semantic-Enriched Collection of Online Courses at Scale with Experimental Use Cases. World Conference on Information Systems and Technologies. Springer, 2018, pp. 1386–1396.
44. Georgakopoulos, S.V.; Tasoulis, S.K.; Vrahatis, A.G.; Plagianakos, V.P. Convolutional neural networks for toxic comment classification. Proceedings of the 10th Hellenic Conference on Artificial Intelligence. ACM, 2018, p. 35.
45. Martens, M.; Shen, S.; Iosup, A.; Kuipers, F. Toxicity Detection in Multiplayer Online Games. The 14th International Workshop on Network and Systems Support for Games (NetGames), At Zagreb, Croatia, 2015. doi:10.1109/NetGames.2015.7382991.
46. Atzeni, M.; Recupero, D.R. Multi-domain sentiment analysis with mimicked and polarized word embeddings for human–robot interaction. *Future Generation Computer Systems* **2019**.

47. Pinter, Y.; Guthrie, R.; Eisenstein, J. Mimicking word embeddings using subword rnns. *arXiv preprint arXiv:1707.06961* **2017**.
48. Si, Y.; Wang, J.; Xu, H.; Roberts, K. Enhancing Clinical Concept Extraction with Contextual Embedding. *arXiv preprint arXiv:1902.08691* **2019**.
49. Reimers, N.; Schiller, B.; Beck, T.; Daxenberger, J.; Stab, C.; Gurevych, I. Classification and Clustering of Arguments with Contextualized Word Embeddings. *arXiv preprint arXiv:1906.09821* **2019**.
50. Read, J.; Pfahringer, B.; Holmes, G.; Frank, E. Classifier chains for multi-label classification. *Machine Learning* **2011**, *85*, 333. doi:10.1007/s10994-011-5256-5.
51. Dessì, D.; Fenu, G.; Marras, M.; Recupero, D.R. Bridging learning analytics and Cognitive Computing for Big Data classification in micro-learning video collections. *Computers in Human Behavior* **2019**, *92*, 468–477.
52. Dessì, D.; Recupero, D.R.; Fenu, G.; Consoli, S. A recommender system of medical reports leveraging cognitive computing and frame semantics. In *Machine Learning Paradigms*; Springer, 2019; pp. 7–30.
53. Dang, N.C.; Moreno-García, M.N.; De la Prieta, F. Sentiment Analysis Based on Deep Learning: A Comparative Study. *Electronics* **2020**, *9*, 483. doi:10.3390/electronics9030483.
54. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research* **2017**, *18*, 559–563.
55. Lilleberg, J.; Zhu, Y.; Zhang, Y. Support vector machines and Word2vec for text classification with semantic features. 2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC), 2015, pp. 136–140. doi:10.1109/ICCI-CC.2015.7259377.
56. Davis, J.; Goadrich, M. The Relationship between Precision-Recall and ROC Curves. Proceedings of the 23rd International Conference on Machine Learning; Association for Computing Machinery: New York, NY, USA, 2006; ICML '06, p. 233–240. doi:10.1145/1143844.1143874.
57. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. Proc. of NAACL, 2018.