School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Computer Science

Toxic Span Detection in Online Posts

Athanasios Katsiolis
Supervisors: John Pavlopoulos, Ion Androutsopoulos

November 2020

Abstract

Human interaction through the internet today is substantial, and it is constantly increasing. A major means of that interaction are comments in prominent websites, e.g. popular news portals or social media platforms. Sadly, the behaviour of users of these websites frequently becomes rude or disrespectful, preventing the regular operation of the websites. We refer to these behaviours, namely to the comments of these users, as "toxic". The aim of this thesis is to identify toxicity in comments, and in particular to identify parts of these comments, to which toxicity can be attributed to. We call these parts of comments toxic spans. To that end, we apply a supervised and two unsupervised methods and compare their e□ectiveness. We refer to the □rst case as supervised, because during the training of the model that implements the method, human annotated toxic spans are provided to the model, in contrast with the latter case. Finally, we used the GPT-2 language model to generate arti□cial comments with toxic spans also generated by the model. We used the arti□cial data with the most e□ective method, to analyse whether they improve the performance of the method.

Περίληψη

Ο αριθμός των ανθώπων που αλληλεπιδρά μέσω του Διαδικτύιου σήμερα είναι αρκετά μεγάλος και συνεχώς αυξάνεται.Ένα σημαντικό τμήμα αυτής της αλληλεπίδρασης γίνεται μέσω σχολίων σε πολύ δημοφιλείς ιστοσελίδες, όπως για παράδειγμα δημοφιλείς ειδησεογραφικές ιστοσελίδες ή δημοφιλή μέσα κοινωνικής δικτύωσης. Έχει παρατηρηθεί όμως ως φαινόμενο, η συμπεριφορά των χρηστών αυτών των ιστοσελίδων, δηλαδή τα σχόλια των χρηστών αυτών των ιστοσελίδων, να είναι αγενή ή να μην δίχνουν σεβασμό απέναντι σε άλλους χρήστες. Αναφερόμαστε συνολικά σε τέτοιου είδους συμπεριφόρες με τον όρο “τοξικές”. Ο σκοπός αυτής της διπλωματικής εργασίας είναι η ανίχνευση τοξικότητας σε σχόλια ιστοσελίδων, και συγκεκριμένα η ανίχνευση των τμημάτων των σχολίων τα οποία ευθύνονται για την τοξικότητα του σχολίου. Αποκαλούμε τα τμήματα αυτά με τον όρο "τοξικό τμήμα". Για την ανίχνευση τοξικών τμημάτων σε σχόλια εφαρμόσαμε δύο ειδών μεθόδους, μια μέθοδο μάθησης με επίβλεψη και δύο μεθόδους μάθησης χωρίς επίβλεψη. Ο πρώτος τύπος μεθόδων που εφαρμόσαμε χαρακτηρίζεται ως μάθηση με επίβλεψη, επειδή κατά τη διάρκεια της εκπαίδευσης του μοντέλου που υλοποιεί την μέθοδο, δίνονται στο μοντέλο τα τοξικά τμήματα του κάθε σχολίου, τα οποία έχουν ανιχνεύσει άνθρωποι επισημειωτές. Αντίθετα ο δεύτερος τύπος μεθόδων δεν χρησιμοποιεί τοξικά τμήματα. Τέλος, με την χρήση του γλωσσικού μοντέλου GPT-2, δημιουργήσαμε τεχνητά δεδομένα με επισημειώσεις, τις οποίες επίσης παρήγαγε το μοντέλο. Στη συνέχεια, χρησιμοποιήσαμε τα τεχνητά δεδομένα με την πιο αποτελεσματική μέθοδο ανίχνευσης τοξικών τμημάτων από τις τρεις μεθόδους που εφαρμόσαμε, για να διαπιστώσουμε εαν η χρήση τεχνητών δεδομένων βοηθά την μέθοδο για καλύτερη ανίχνευση τοξικών τμημάτων.

Acknowledgements

I would like to express my gratitude to my supervisor Professor Ion Androutsopoulos, for his continuous guidance throughout the past year on carrying out research, in the rapidly evolving and compelling area of Natural Language Processing.
I would also like to express my special thanks to my advisor John Pavlopoulos, for his immediate assistance in overcoming di□culties throughout this research.
Finally, I would also like to thank Christos Baziotis for joining the meetings, monitoring the progress of the research and providing essential advises.

Contents

1 Introduction

Currently, Internet is the major means of communication globally. Its intrinsic characteristic of high and immediate dissemination of information has attributed Internet this role. People access daily the Internet primarily to be informed about the latest news, socialise or entertain themselves. Leading news portals, social networks and popular entertainment platforms account for a signi□cant proportion of Internet's tra□c, where the interaction between people is high. People also use the Internet extensively to be informed about what interests them. Blogs and forums gather people of similar interests where the interaction between people is again of central importance.

Often, it has been observed, that the interaction of users is not the anticipated one. They post abusive or 'toxic' comments, or more generally user-generated content, that prevents the regular and intended functionality of the website or platform. We refer to these behaviours with the term toxic, and more formally, as in [Pav+20], we de□ne a toxic comment as: They post abusive or 'toxic' comments, or more generally user-generated content, that prevents

" A rude, disrespectful, unreasonable comment or otherwise somewhat likely to make a user leave a discussion or give up on sharing their perspective "

Moderation is a counter measure to prevent those kinds of behaviours, but usually the volume of comments is such, that it is not possible to be tracked by humans. Assistance from systems that detect toxicity can greatly bene□t moderators and subsequently, the regular functionality of websites that host user comments, or more generally user-generated content. In this thesis we focus on user comments on news portals. We aim to detect toxic spans in user comments, a new problem discussed below.

1.1 Problem Statement

Toxic Span Detection is the task of □nding parts of a text that make the text toxic. Note that this is not always possible. Examples of toxic comments along with their annotated toxic spans follow. The highlighted parts of the comments are the parts that humans annotated as toxic.

Caution: The following examples contain real-life toxic comments, which the reader may find disturbing.

- "Only an idiot would use and believe anything this This Republican propaganda machine publishes: www.realclearpolitics.com"
- "Because driving under Ontario laws is stupid enough in Ontario."
- "People insist on being dumb. No other explanation."
- "Of course you cannot defend your own hypocrisy...I see you are starting of 2017 on the same losing track as always...oh, well, no cure for stupidity."
- "Well Dave, maybe if my ancestors killed off most of yours,then we would be providing health care for you."
- "Nice propaganda piece, the protestors were aggressive, but we nicely shot at them and tried to freeze them do death."

In the examples, the toxic spans are mainly offensive words, while the last two examples do not have any annotation. Though the last two comments are sarcastic and hence toxic, it was not possible to attribute that to any particular span. As mentioned earlier, it is not always possible to identify toxic spans within comments, even when the comments are considered overall toxic.

Recently, there have been various attempts to develop systems for toxicity detection [PMA17], [Nob+16], [PF17] and in [Dav+17] toxicity was further classified as offensive or hate speech. In [Zam+19] except for toxicity identification, they also tried to identify the category and the target of offensive language, while in [Pav+20] context was exploit for identifying toxicity more efficiently.

Although systems that detect toxicity already exist, they usually classify whole texts without providing any other information than just a classifier's score, thus moderators have to examine the entire comment that was classified as toxic. Providing human annotators with toxic spans, facilitates moderation considerably, in that they more easily form an opinion about the comment, especially in cases where texts are large. Considering also that moderators often deal with substantially large numbers of comments, which is a complex and error-prone task, automatically highlighting toxic spans is even more crucial.

1.2 Thesis Structure

In this thesis we develop and evaluate systems for detecting toxic spans in comments. Additionally, we generate artificial data with toxic span annotations to study if the use of the artificial data can benefit systems that detect toxic spans. Below we present the structure of the thesis with a brief explanation of each chapter.

Chapter 2

In this chapter we discuss the three methods we develop for detecting toxic spans and explain how they function. We also present the architecture of the classifiers that the methods adopt.

Chapter 3

In chapter 3, we present the datasets we apply our methods on, introduce the measures that we use to evaluate our methods and report the experimental results we get. We also perform qualitative analysis, namely we examine individual outcomes of the most effective method, to better understand the capabilities and limitations of the method.

Chapter 4

In this chapter we discuss the model that we selected for generating additional artificial training data and we demonstrate some examples of the artificial data. We also report results that the addition of artificial data as training examples yields.

Chapter 5

In the last chapter we outline the conclusions that we reached after executing our experiments and future work proposals.

2 Methods

For toxic spans detection in comments we apply supervised as well as unsupervised methods. We refer to the former case as supervised because, during model training, toxic span annotations are provided to the model, whereas in the latter case, no toxic spans are given to the models. Although in the unsupervised setting the models are not provided with toxic spans, they are trained with respect to whether a comment is toxic or not and thus the methods can be characterised as unsupervised depending on classi□ers trained with supervised learning for another task.

2.1 Unsupervised methods

In the unsupervised approach, we apply two methods for identifying toxic spans, the Input Erasure method [LMJ17] and the LIME [RSG16] algorithm, which we describe in the following sections in detail.

2.1.1 Input Erasure

As described in [LMJ17] the Input Erasure method erases parts of the model's inputs and observes the e□ects of these erasures on the classi□er. While in [LMJ17] various parts of the representations are erased, such as input word-vector dimensions, intermediate hidden units and input words, we just erase words from the model's input. Speci□cally, we implement the Input Erasure method as follows:

Every comment of the dataset is split on whitespace characters, so the comment breaks into parts originally separated by whitespace characters. Subsequently, we modify the comment each time omitting one of its parts produced from splitting and apply the model to the modi□ed comments. An example of the comment modi□cation used in Input Erasure method for the comment:

"Having a di□cult time believing this article."

is shown in table 2.1. The third column of the table shows the toxicity score reduction that each erased part produces to the toxicity score of the original comment. These scores

are hypothetical, and they have not been produced by applying the method. Finally, to determine the toxic span for the comment, we keep the span of each missing part that reduces the toxicity score that the model yields for the entire comment over a threshold. The choice of the threshold's value is described in chapter 3.

erased part	modi□ed comment	toxicity reduction
Having	~~Having~~ a di□cult time believing this article.	0.02
a	Having ~~a~~ di□cult time believing this article.	0.01
di□cult	Having a ~~dif□cult~~ time believing this article.	0.11
time	Having a di□cult ~~time~~ believing this article.	0.02
believing	Having a di□cult time ~~believing~~ this article.	0.07
this	Having a di□cult time believing ~~this~~ article.	0.003
article.	Having a di□cult time believing this ~~article~~.	0.04

Table 2.1: Example of comment modi□cation for the Input Erasure method.

For instance, if we specify the threshold to 0.05, then we select the words of the third and □fth row of the table as toxic, because the reduction they produce to the toxicity score is over 0.05. Below we have highlighted the identi□ed toxic spans of the example comment.

"Having a dif□cult time believing this article."

2.1.2 LIME

The second method we apply for detecting toxic spans within comments is the Local Interpretable Model-Agnostic Explanations (LIME) algorithm. As proposed in [RSG16] the LIME algorithm provides explanations of a model's predictions for each instance in order for users to trust the model. To achieve that, the explanations of the LIME algorithm provide an understanding of the relationship between the classi□er prediction and the components of the instance, which in our case are words.

The algorithm produces explanations for an instance by learning the local behaviour of the model being interpreted, in the vicinity of that instance. Speci□cally, LIME draws samples of that instance by modifying it randomly, and weights them according to their distance of the original instance. Formally, LIME de□nes the explanation for a comment, as a linear model which acts on the presence or absence of interpretable components, i.e. words. Finally, the explanation is produced by minimizing a locally weighted square loss function which receives as input, the weighted samples and their labels. The label for each sample is produced by applying the model being interpreted to that sample.

In our setting, we classify a comment as toxic or not, so the LIME algorithm yields explanations, i.e. words, which are the most relevant for making the classi□er determine

each class. For instance, □gure 2.1 illustrates the analysis that the LIME algorithm provides for the comment

"The truth hurts so you have to resort to idiotic comments you just don't understand how ignorant you are."

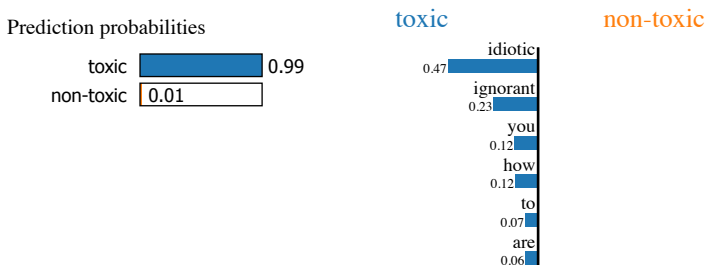


Figure 2.1: LIME algorithm analysis for a comment.

The analysis in □gure 2.1 shows that the classi□er predicts with probability 99% that this comment is toxic, and provides an explanation for the comment as well, i.e. words, that contribute to that prediction. The explanation is presented as a list of weighted features. As mentioned before, an explanation is a linear model which acts on words, so the features of the explanation model are words. In this example, the two most essential words for the classi□er prediction is "idiotic" and "ignorant" which contribute to the toxic category 47% and 23% respectively. Removing those words from the comment, would move the classi□er's prediction towards the opposite class by about 70%, which is the sum of the weights of both features. The use of the LIME algorithm and the analysis of □gure 2.1 is done with the lime package [1], provided by the authors of [RSG16].

Using the analysis of the LIME algorithm, the method detects as the comment's toxic span, every word of the comment that its in□uence score for the toxic category is over a threshold. The choice of the threshold's value is described in chapter 3. For the comment of □gure 2.1, if we specify the threshold to 0.2, the identi□ed toxic span of the comment would be the spans of the words "idiotic" and "ignorant", because the contribution of these words to the toxic category is 47% and 23% respectively, which are greater than 0.2. Below we show the comment having highlighted the identi□ed toxic spans.

"The truth hurts so you have to resort to idiotic comments you just don't understand how ignorant you are."

[1] LIME package https://github.com/marcotcr/lime

2.1.3 Bi-LSTM binary classifier

For the implementation of the unsupervised methods that were described in the previous sections we trained a bidirectional long short-term memory (Bi-LSTM) binary classifier, and a binary classifier based on a BERT [Dev+19] model. The classifiers are described in the current and the following sections.

Bi-LSTMs are recurrent neural networks for processing sequential data. The output of the network after processing each element of the sequence depends on the current element as well as the previous elements of the sequence. When the size of the sequence is large though, recurrent neural networks face the problem of forgetting information processed in the beginning of the sequence. Bi-LSTM models have a mechanism for maintaining important information from the entire sequence during learning.

Bidirectional LSTMs process the sequence from both directions. In particular, there is an LSTM [HS97] model that processes the sequence starting from the first element of the sequence and another LSTM model that process the sequence starting from the last element. The output of a Bi-LSTM model for every element of the sequence is the combined output of the LSTMs of both directions for that element. Consequently, each element output combines information from both sides of the sequence. The architecture of a Bi-LSTM model is depicted in figure 2.2 [2].

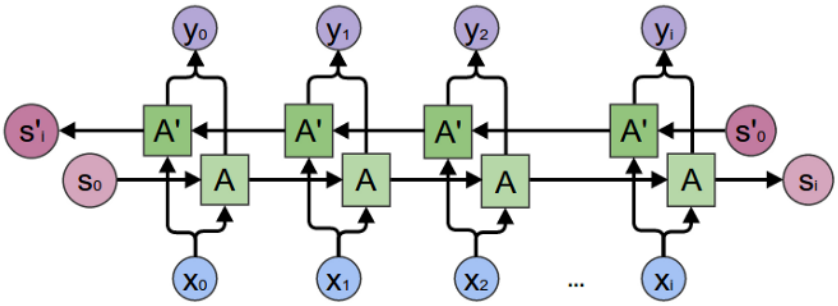


Figure 2.2: Architecture of a Bidirectional long short-term memory model.

Since the model requires numbers as input instead of text, our data must be turned into the form that the model requires. Firstly, every comment of the training dataset is tokenised, and from the generated tokens of every comment we build a vocabulary. The vocabulary is just an assignment of a number to each unique token, and is built on the training dataset.

In order to enhance the performance of the classifier, instead of just providing it with the index of each token of the comment, word embeddings can also be used. Word embeddings are dense vectors trained in order to capture essential properties of words and are provided

[2] Figure 2.2 is taken from Towards Data Science https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

in different dimensions. In our model we used the GloVe [PSM14] word embeddings of dimensionality 200, trained on the Wikipedia14 and Gigaword5 corpora.

The architecture of our classifier begins with an Embedding layer followed by the Bi-LSTM layer. The output features of the last elements of both directions, which in figure 2.2 are denoted as S_i and S'_i, are concatenated and go through a Dropout layer followed by a Linear layer with a tahn activation function. Finally, the last layer is the classification layer, which is a Linear Layer with a sigmoid activation function. Figure 2.3 displays the architecture of the classifier and table 2.2 contains its hyper-parameters. Data preprocessing and the selection of hyper-parameters are explained in chapter 3.

For computing the loss of the model we used the Binary Cross Entropy and for updating weights we used the Adam [KB15] optimizer with learning rate 0.001. The framework we used for developing, training and evaluating the classifier is Pytorch [3].

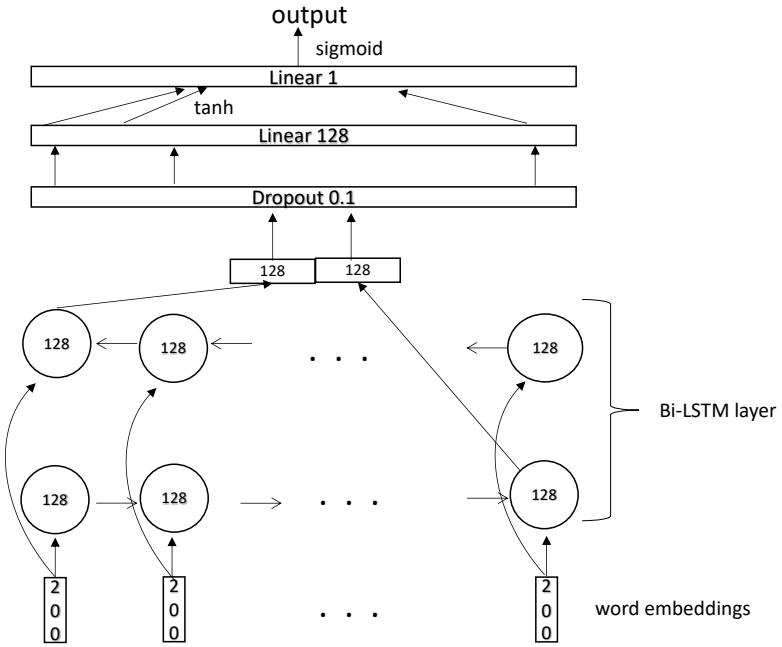


Figure 2.3: Bi-LSTM binary classifier.

layer	parameter	value
Embedding	dimensionality	200
Bi-LSTM	hidden units	128
Dropout	rate	0.1
Linear	output size	128
Linear	output size	1
Optimizer	learning rate	0.001

Table 2.2: Bi-LSTM binary classifier parameters.

[3] Pytorch framework https://pytorch.org/

2.1.4 BERT Binary classifier

BERT [Dev+19] is multi-layer Transformer [Vas+17] model designed for several natural language processing (NLP) tasks. It is a pre-trained deep bidirectional encoder which can produce state-of-the-art results for a speci□c task by adding one output layer and □ne-tuned on that task.

The pre-training of the BERT model was in two unsupervised tasks, Masked Language Model (MLM) and Next Sentence Prediction (NSP). In the □rst task some percentage of the input tokens were masked randomly and the objective of the model was to predict them. In the NSP task the model is given two sentences and predicted if the second sentence follows the □rst. BERT was pre-trained on the BookCorpus and English Wikipedia. A notable point is that in Wikipedia they extracted only text passages and ignored everything else such as lists, headers or tables.

As already mentioned, the BERT model is a multi-layer Transformer based on the implementation of [Vas+17]. It was released in two versions, Base and Large. We experiment with the base version which has 12 layers of Transformer encoder blocks, hidden size 768 and 12 attention heads. Figure 2.4 is taken from Jay Alammar's blog 'The Illustrated Transformer' [4], and illustrates a Transformer encoder layer. For the pre-trained weights as well as the model architecture we used the Hugging Face [5] library, a library concerned with NLP tasks.

The architecture of our BERT classi□er consists of the BERT base model followed by a dropout layer of rate 0.1 and a □nal Linear layer with output size 1 and a sigmoid activation function. If the size of the input sequence is I, the size of the BERT model's output is [I, 768], namely a representation of size 768 for every sequence element. The BERT model adds also a special token in the beginning of a sentence, the [CLS] token, as well as another special token at the end of it, the [SEP] token.

As mentioned in [Dev+19], the □nal hidden state of [CLS] token is used as the aggregate sequence representation for classi□cation tasks, so for the prediction of our classi□er we use the □nal hidden state of the [CLS] token. Figure 2.5 [6], demonstrates the classi□er.

Comment tokenization was made by the tokenizer supplied by the Hugging Face library which accompanies the model. In training, as loss function we used binary cross entropy for this model too, for updating the weights we used the Adam optimizer, but with learning rate 0.00003, as proposed in [Dev+19].

[4] Jay Alammar's blog 'The Illustrated Transformer' http://jalammar.github.io/illustrated-transformer/
[5] Hugging Face library https://huggingface.co/
[6] Figure 2.5 taken from the Towards Data Science website https://towardsdatascience.com/sentence-correctness-classi□er-using-transfer-learning-with-huggingface-bert-8884795ba5ca

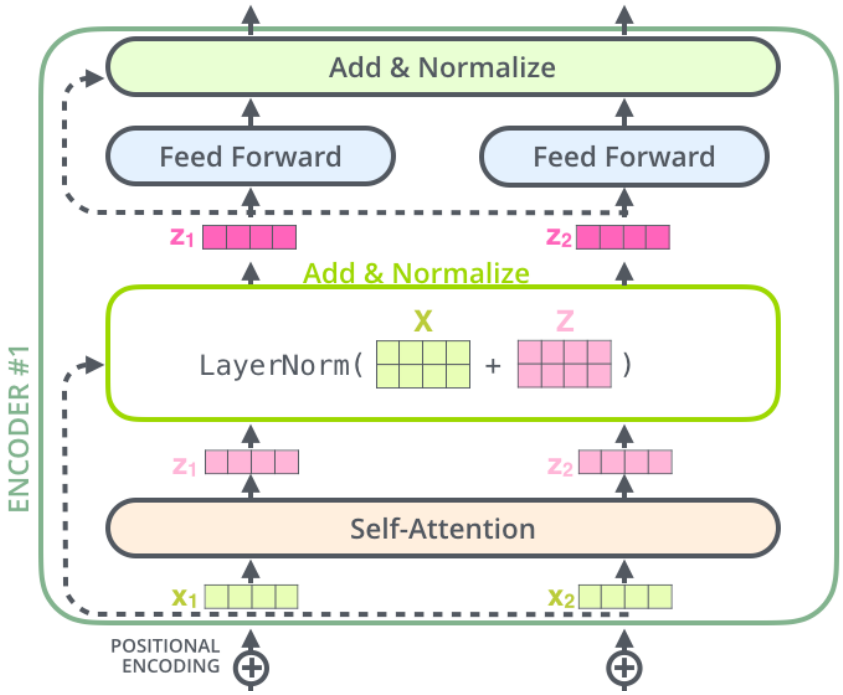


Figure 2.4: Transformer encoder layer.

2.2 Supervised method

As already mentioned, for toxic span detection we applied unsupervised methods as well as supervised and in particular, we use Sequence Labeling. Sequence Labeling is a general class of supervised learning algorithms that can be used to tag elements of a sequence, as in part of speech tagging (POS) or name entity recognition (NER). A detailed explanation of the Sequence Labeling algorithms can be found in chapter 7 of the [Eis19] book. Applications of common Sequence Labeling tasks such as NER and POS are in [MH16], and in [CN16] they also perform NER. In this setting, during training, toxic span annotations are provided to the model. At the end of the section we present the classifier that we train for the Sequence Labeling task.

While we present the datasets in detail in section 3.1, we discuss here briefly the dataset with its ground truth annotations, in order to explain how the supervised method is implemented. The annotation of a comment is the set with the character positions of the annotated parts of the comment. An example of a comment along with its ground truth annotation is shown in table 2.3. The left column of the table contains the annotated spans and the right column contains the comment with the toxic spans highlighted. In this comment the character positions from 7 to 15 and 23 to 31 have been annotated as toxic spans.

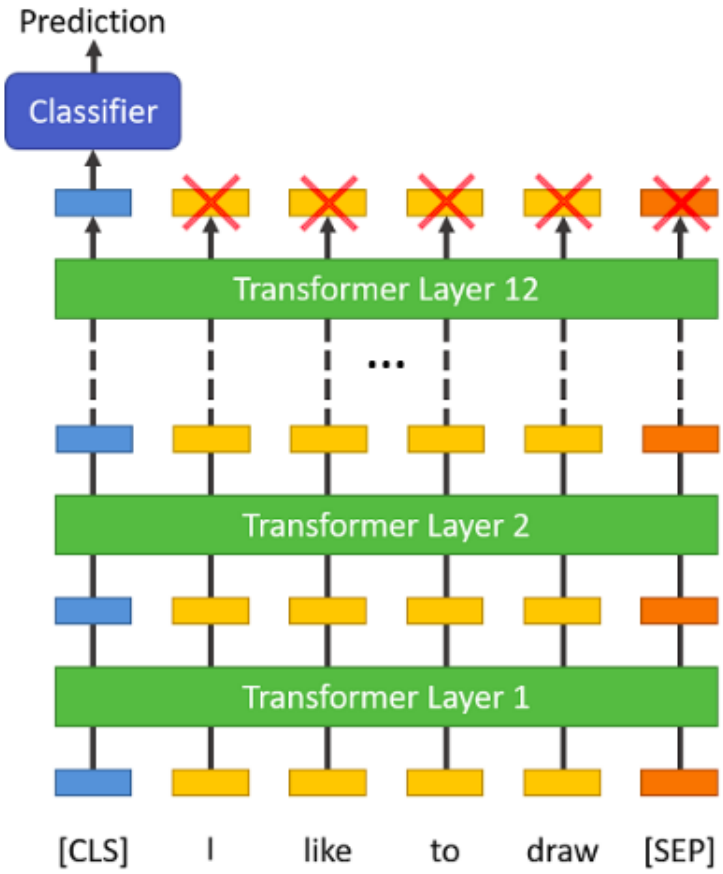


Figure 2.5: BERT binary classi□er.

annotated toxic span	comment
{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27, 28, 29, 30, 31}	What a sociopathic and parasitic leader we have.

Table 2.3: A comment with its annotation.

For detecting toxic spans in the supervised case, we trained a BERT [Dev+19] model for Sequence Labeling. Although the classi□er is explained at the end of the section, we brie□y mention part of its functionality here which is necessary for demonstrating how we implemented Sequence Labeling.

Every comment, before it is supplied as input to the BERT model should be tokenised. In BERT, tokens may be single characters, frequently occurring sequences of characters, or even entire frequent words. Each token corresponds to some character positions in the original comment. An example of a tokenised comment along with the character positions of its tokens is shown in table 2.4. The □rst row of the table has the original comment, the second row has the comment split in tokens, and the third the character positions of these tokens in the original comment. For instance, the second token 'those', is in character positions 5 to 10 in the original comment.

comment	Ride those trailer brakes, you morons!
tokenised comment	['Ride', 'those', 'trailer', 'brakes', ',', 'you', 'm', '##oro', '##ns', '!']
character positions of the tokens	[(0, 4), (5, 10), (11, 18), (19, 25), (25, 26), (27, 30), (31, 32), (32, 35), (35, 37), (37, 38)]

Table 2.4: A tokenised comment along with character positions of its tokens.

When a model is trained in a Sequence Labeling setting, there is a label for each element of the sequence. We formulate our data in that setting firstly by tokenising the comment, which yields the sequence, and secondly we assign a label for each token which can be toxic or non-toxic. A token is assigned the toxic ground truth label if at least a character position of its span is included in the ground truth annotated toxic span, otherwise it is assigned the non-toxic ground truth label. Table 2.5 shows the ground truth labels assigned to the comment of table 2.4, with label 1 meaning that the token is toxic and label 0 otherwise. Note that only the seventh, eighth and ninth tokens have been assigned toxic labels, since only the spans of those tokens include character positions that belong to the ground truth annotation set of this comment.

comment	Ride those trailer brakes, you morons!
tokenised comment	['Ride', 'those', 'trailer', 'brakes', ',', 'you', 'm', '##oro', '##ns', '!']
character positions of tokens	[(0, 4), (5, 10), (11, 18), (19, 25), (25, 26), (27, 30), (31, 32), (32, 35), (35, 37), (37, 38)]
ground truth annotation of spans	[31, 32, 33, 34, 35, 36]
ground truth labels	[0, 0, 0, 0, 0, 0, 1, 1, 1, 0]

Table 2.5: A comment formulated in the Sequence Labelling setting.

Finally, the way we compute the toxic span for a comment is straightforward. Since we are in a Sequence Labeling task, the classifier assigns a label to every token of the sequence. Thus, the toxic spans that the classifier predicts for a comment is represented by a set of the character offsets (union of spans) of those tokens that the classifier classified as toxic.

For the supervised case we deployed the model described in section 2.1.4, changing only the final layer of the model in order to suit the Sequence Labeling setting. Instead of using the final hidden state of the $[CLS]$ token, we used the final hidden states of every sequence element so that we classify each one of them. The final hidden state of every token, after going through the dropout layer, passes through the Linear layer with a sigmoid activation

function which yields the classi□cation score for the token. Figure 2.6 [7] illustrates the model used for Sequence Labeling.

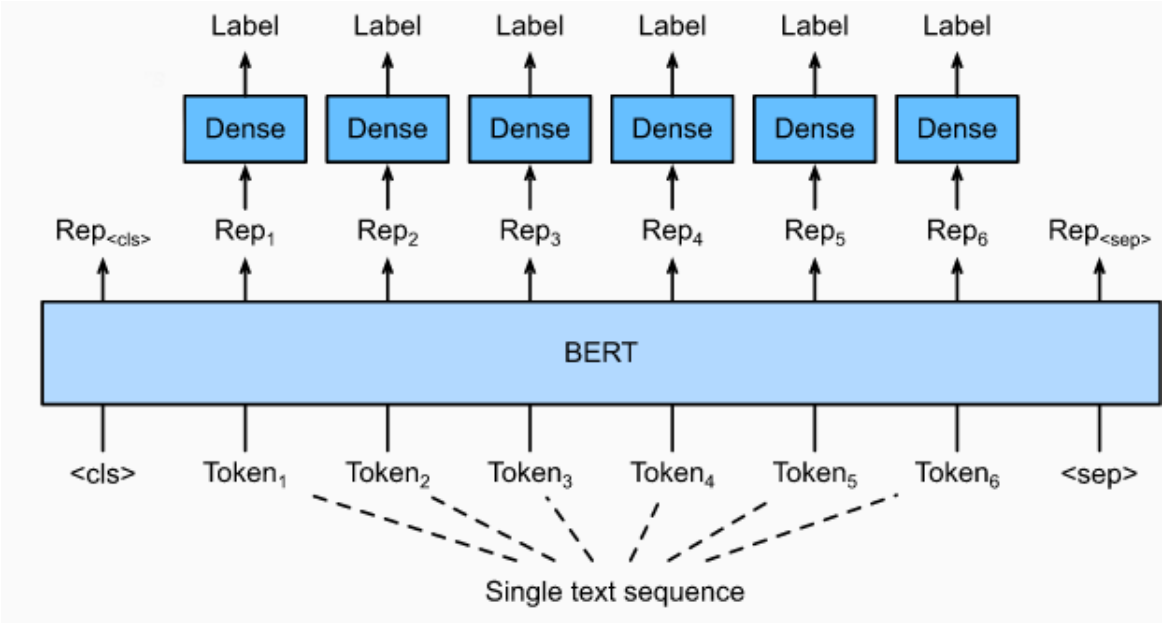


Figure 2.6: BERT for Sequence Labeling.

[7] Figure 2.6 is taken from Dive into Deep Learning website https://d2l.ai/chapter_natural-language-processing-applications/□netuning-bert.html

3 Experiments and Results

In this section we will introduce the datasets that we applied our methods on, the measures we use to evaluate the methods and the experimental results we got. At the end of the section we also conduct qualitative analysis of the methods that accomplish the highest results.

3.1 Datasets

As mentioned in chapter 2, for detecting toxic spans in comments we apply unsupervised and supervised methods. In order to implement those methods, we trained our models on a dataset without toxic span annotations for the unsupervised setting and on a dataset with toxic spans from human annotators for the supervised setting. We refer to the former dataset as CivilCom and to the latter as CivilAnn. Below we present those datasets.

3.1.1 CivilCom

The CivilCom dataset consists of comments from the Civil Comments platform. When Civil Comments shut down, it made their platform comments publicly available for researchers to improve civility in online conversations. The dataset comprise of approximately 2 million public comments from independent news sites. Furthermore, this dataset was at a kaggle competition [1], where each comment was attributed a toxicity score. Every comment was shown to up to ten human raters and the toxicity score for each comment is the fraction of raters that found that comment toxic. An example of a toxic and a non-toxic comment of the dataset along with their toxicity scores, is shown in table 3.1.

comment	toxicity
Thank you!! This would make my life a lot less anxiety-inducing. Keep it up, and don't let anyone get in your way!	0.0
haha you guys are a bunch of losers.	0.9

Table 3.1: Sample from CivilCom dataset.

[1] Kaggle competition https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classi□cation/data

In gure 3.1 there is a histogram of the dataset's comment toxicity. The horizontal axis shows the toxicity score that was attributed to a comment and the vertical axis shows the number of comments that have such a score. For instance, the number of comments with toxicity score less than or equal to 0.1 is approximately 1.3 million. It is evident that the dataset is highly imbalanced, since the vast majority of comments have toxicity scores less than or equal to 0.1 and very few have toxicity scores higher than 0.6.

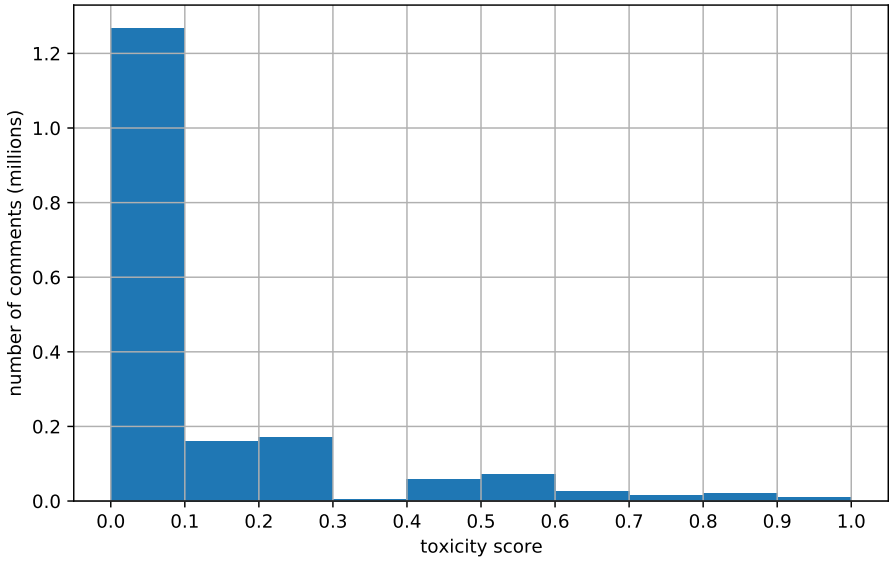


Figure 3.1: Histogram of comment toxicity. The CivilCom dataset.

Finally, as in the kaggle competition, we turned the toxicity score of each comment to a binary label, in other words, every comment turns into a toxic or a non-toxic comment. We consider a comment toxic if its toxicity score is equal or higher than 0.5 and non-toxic otherwise. After the binary modi cation, 95% of the dataset contains non-toxic comments while toxic comments are just 5% of the dataset, which is indicative of the dataset's high imbalance.

3.1.2 CivilAnn

The CivilAnn dataset contains also comments from the Civil Comments platform along with human annotated toxic spans. It comprise a small subset of Civil Comments platform's toxic comments, with size of approximately 8500 comments. As already mentioned in section 2.2, the annotation of a comment is the character positions of the annotated toxic span. A sample of the annotated dataset is shown in table 3.2. For instance, in the rst comment of the table, the annotators have identi ed positions 12 to 21 as the comment's toxic span, while in the third and forth examples they have annotated more than one individual spans. These positions are highlighted in the comment on the left side of the table.

comment	annotation
Lol. What a ridiculous insult	[12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
What if his opinion is that most other commenters are idiots? :-)	[55, 56, 57,58,59,60]
Of course you cannot defend your own hypocrisy...I see you are starting of 2017 on the same losing track as always...oh, well, no cure for stupidity.	[37, 38, 39, 40, 41, 42, 43, 44, 45, 139, 140, 141, 142, 143, 144, 145, 146, 147]
If you're dumb enough to believe him on this, then I've got a bridge in Brooklyn I will sell you... The guy is a pathological liar. Don't believe anything he says. It's all lies.	[10, 11, 12, 13, 128, 129, 130, 131]
So tired of all these Portlanders moving to California and ruining the culture. When will it stop?!?	[]

Table 3.2: Sample from the CivilAnn dataset.

The dataset is available as a Codalab provided by the the organizers of SemEval 2012 Task 5: Toxic Span Detection [2], a competition for detecting toxic spans within comments. Each comment was shown to 3 annotators who were instructed to extract the toxic spans of the comments, if they identi□ed any. As a result, there are comments that do not have any toxic span annotation. The number of comments in the dataset that do not have an annotation is 528.

Notice that, as mentioned in the Introduction, there are cases where toxic spans cannot be detected within a comment. An example of this case is the comment in the last row of table 3.2, where even if a group of people is criticised about the place of their origin, and hence the comment is toxic, it is di□cult to annotate a particular span.

3.2 Evaluation measures

Having introduced our models and the datasets we trained our models on, we now describe the evaluation measures we use to assess our methods.

[2]SemEval 2012 Task 5: Toxic Span Detection competition https://competitions.codalab.org/competitions/25623#learn_the_details-data

3.2.1 Evaluation measures for binary toxicity classifiers

We begin with the evaluation measures for our binary classi□ers Bi-LSTM and BERT described in section 2. We compute Precision, Recall and F1 scores, measures that are common for evaluating classi□ers.

If we denote with TP, FP and FN the number of true positives, false positives and false negatives respectively, where

- true positives are the comments that the classi□er predicted as toxic and are truly toxic
- false positives the comments that the classi□er predicted toxic but they are non-toxic
- false negatives the comments that the classi□er predicted as non-toxic but they are toxic

we de□ne Precision, Recall and F1 as follows:

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN},$$

$$F1 = \frac{2 \quad P \quad R}{P + R}.$$

In addition to those measures we also report Precision-Recall Area Under the curve (AUC) score. Even though a common measure to report for classi□ers is the ROC AUC score, in cases where the data are highly imbalanced and the category of interest is the minority class, Precision-Recall AUC score is more indicative about the performance of the model. ROC AUC score yields high values due to the e□cient performance of the model on the majority class. Since we are interested in the toxic class, and the toxic comments were the minority class with just the 5% of the dataset, we report Precision-Recall AUC score.

3.2.2 Evaluation measures for toxic span detection methods

We now de□ne the measures that we use to evaluate evaluate the methods we applied for detecting toxic spans in a comment, both supervised and unsupervised. Since we detect parts of a comment, the measures we adopt are at the character level and are computed as in [Da +19].

Let S_t be the set of characters of the ground truth annotated toxic span and S_a the set of characters that a method detects as toxic span in a comment, we define Precision, Recall and F1 scores at the character level in a similar manner that we defined Precision, Recall and F1 before. Specifically, if we denote with $|S|$ the cardinality of a set, we have:

$$P = \frac{|S_t \cap S_a|}{|S_a|}, \qquad R = \frac{|S_t \cap S_a|}{|S_t|},$$

$$F1 = \frac{2 \quad P \quad R}{P + R}.$$

For each method we report micro-averaged scores, so if c denotes a comment, S_t^c denotes the annotated toxic span of comment c and S_a^c is the predicted toxic span of comment c, then we have:

$$mP = \frac{\sum_c |S_t^c \cap S_a^c|}{\sum_c |S_a^c|}, \qquad mR = \frac{\sum_c |S_t^c \cap S_a^c|}{\sum_c |S_t^c|},$$

$$mF1 = \frac{2 \quad mP \quad mR}{mP + mR},$$

where mP is micro-Precision, mR is micro-Recall and $mF1$ is micro-F1.

Finally, since we defined Precision and Recall at the character level for evaluating comment toxic spans, we also report Precision-Recall AUC score for each toxic span detection method using Precision and Recall defined at the character level.

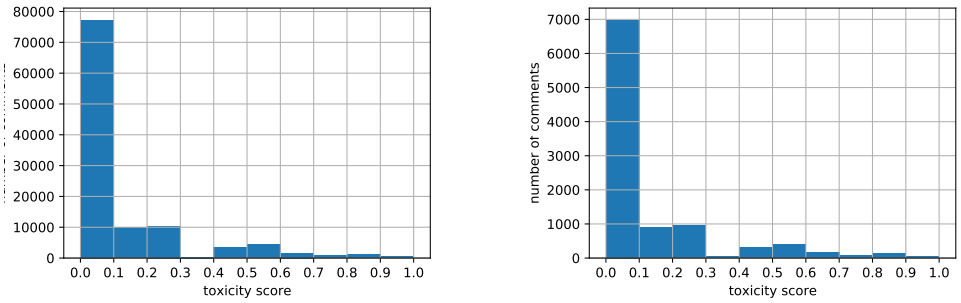
3.3 Experimental results

Before we present the results of the toxic span detection methods we experimented with, since the two unsupervised methods employ supervised classifiers trained to classify entire comments as toxic or non-toxic, we firstly report binary toxicity classification results from those classifiers.

3.3.1 Bi-LSTM and BERT binary classifiers

In the beginning of this section we mentioned that the classifiers of the unsupervised methods were trained on CivilCom, though we used just a fraction of the dataset for

training and testing our models. In particular, we sampled 110 thousand comments for training and developing our models and 10 thousand additional comments for testing, retaining the distribution of toxic comments of the original dataset. Speci□cally, the toxic comments constitute 5% and 7% of the training and testing datasets respectively. In □gure 3.2 we plot the histograms of comments' toxicity for the train and test subsets and we observe the similarity with the corresponding histogram of the original dataset in □gure 3.1.



(a) Train dataset.

(b) Test dataset.

Figure 3.2: Histograms of comments' toxicity score.

Note that CivilAnn contains comments from the CivilCom dataset, and since we use part of CivilAnn as our test dataset for detecting toxic spans, before sampling from CivilCom to construct train, development and test datasets to train the binary toxicity classi□ers, we omitted the common comments.

The data preprocessing that we performed for both models before training was to lowercase all comments in the datasets. Additionally, for the Bi-LSTM model, we tokenized the comments with the spaCy tokenizer for the English language and kept the tokens that appear at least twice in the dataset.

For the Bi-LSTM binary classi□er we □rstly experimented with di□erent hidden sizes, namely, we trained the classi□er with two times and four times the current size with di□erent dropout rates. Secondly, we added another Bi-LSTM layer too and the results remained essentially unchanged. Therefore, we kept the current values of the Bi-LSTM classi□er's hyper-parameters due to the smaller size of the classi□er.

Initially, due to limited computational resources and the large size of the BERT model, we trained (□ne-tuned) the model freezing the □rst 9 Transformer layers. Subsequently, gaining access to higher computational resources, we varied the number of the layers we froze. Firstly, we observed that freezing or unfreezing some additional layers does not improve the measures of the model considerably, and secondly that freezing the □rst 7 layers yields good results.

In table 3.3 we computed the measures described in section 3.2 for the two binary classi□ers on the test dataset and in □gure 3.3 we plot Precision-Recall curves. The classi□cation thresholds have been tuned on the development dataset and they were 0.3 for the Bi-LSTM and 0.35 for the BERT model. The measures and the curves indicate that the BERT classi□er is a better classi□er than BiLSTM one, as expected, but the di□erence in performance is small.

	Bi-LSTM	BERT
Precision	0.596	0.633
Recall	0.672	0.708
F1	0.632	0.668
Precision-Recall AUC score	0.703	0.744

Table 3.3: Bi-LSTM and BERT binary classi□ers scores.

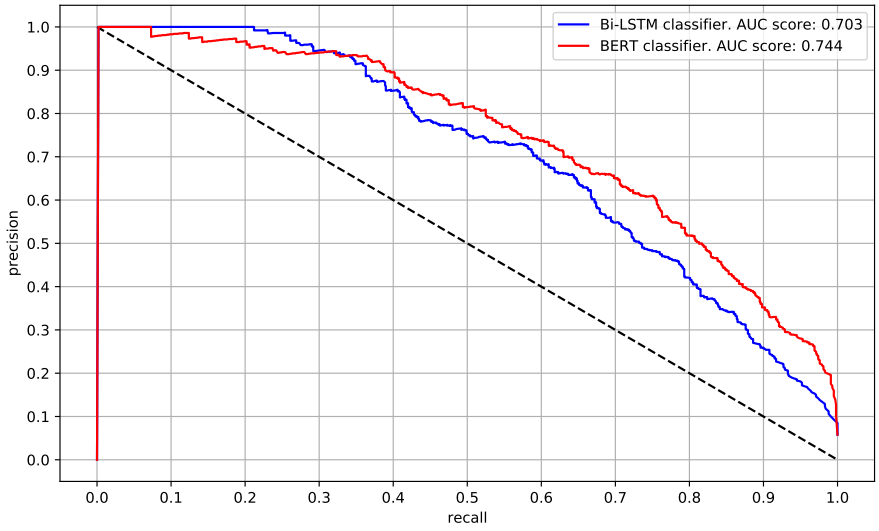


Figure 3.3: BERT and Bi-LSTM Precision-Recall curves.

3.3.2 Unsupervised methods

Having already presented the evaluation results for the classi□ers that the unsupervised toxic span detection methods depend on, we now present in table 3.4 the scores of the evaluation measures described in section 3.2.2 for these methods.

For evaluating the methods we sampled a subset of 690 comments from the CivilAnn dataset. In the unsupervised setting we sampled another 1000 comments to use it as development dataset for the toxic span detection methods.

As already mentioned in chapter 2, the Input Erasure method includes a missing part in the comment's toxic span if the removal of this part reduces the comment's toxicity score over a threshold. Likewise, the LIME method includes a word in the comment's

toxic span if the influence of this word for the toxic category is again over a threshold. To determine the value of these thresholds, we vary the thresholds and observe the effect on the evaluation scores on the development dataset. The results that we report in table 3.4 are scores computed with the threshold that produced the best F1 score in the development dataset.

The results indicate that the LIME method performs better than the Input Erasure method, and that the two classifiers yield similar results. The BERT-based classifier in combination with the LIME method achieve the highest results in the unsupervised setting.

	Precision	Recall	F1	PRAUC score
Bi-LSTM Input Erasure	0.553	0.383	0.453	0.411
BERT Input Erasure	0.53	0.4	0.458	0.413
Bi-LSTM LIME	0.575	0.408	0.477	0.424
BERT LIME	0.585	0.418	**0.487**	**0.435**

Table 3.4: Micro-averaging scores of unsupervised methods.

Before we introduce the results we received from the supervised method as well, we will mention some further experiments that we performed with the unsupervised methods. The first experiment is the following.

Earlier, when we described the dataset we trained the binary classifiers on, we mentioned that we excluded the subset of CivilAnn that we use as evaluation dataset. The reason we excluded these comments was to ensure that, when the unsupervised method would predict the toxic span of a comment, the binary toxicity classifier the unsupervised method relies on, would have not encountered the comment during its training.

In this experiment, we add the comments that we previously excluded to the training of the binary toxicity classifier and expect that the scores will increase. Due to the high computational demand to train the BERT model and to subsequently apply the LIME method, we conducted the experiment with just the Bi-LSTM classifier and the results are in table 3.5. The rows of the table mentioning the CivilAnn dataset are the micro-averaged scores having included in the training set of the binary toxicity classifier the comments from CivilAnn. In contrast with our expectation, the scores remained essentially unchanged. This shows that one does not need to make sure the binary toxicity classifier has not encountered during its training the comments the unsupervised method tries to extract toxic spans from.

The second experiment we conducted was with BERT using the Input Erasure method. Note that, the Input Erasure method initially splits each comment on the white space characters and subsequently, modifies the comment omitting each time one part created by

	Precision	Recall	F1	PRAUC score
Input Erasure	0.553	0.383	0.453	0.411
Input Erasure CivilAnn	0.521	0.403	0.454	0.408
LIME	0.575	0.408	0.477	0.424
LIME CivilAnn	0.58	0.4182	0.481	0.43

Table 3.5: Comparing micro-averaging scores having added previously excluded comments.

the split. In this experiment, instead of modifying the comment omitting each time a part of the white space split, we tokenize the comment with the BERT tokenizer and modify it omitting each time a token of BERT's tokenizer (which may be a single character, a frequent sequence of characters, or an entire word). Table 3.6 compares the micro-averaged scores of the two different comment modifications. We notice that if we modify the comment omitting BERT tokens, the scores drop significantly.

	BERT token	**white space**
micro-Precision	0.42	0.53
micro-Recall	0.331	0.4
micro-F1	0.367	0.458
precision-recall AUC score	0.33	0.413

Table 3.6: Comparison of micro-averaging scores of the Input Erasure modifying the comment with a different manner.

3.3.3 Supervised method

For completing the calculation of measures for all the methods we introduced, we compute the measures of section 3.2.2 for the supervised method too.

As evaluation dataset in the supervised setting we used the same subset of CivilAnn that we used for evaluation in the unsupervised setting. The rest of the dataset, which consists of approximately eight thousand comments was used for training and developing the BERT-based sequence labeling model.

Table 3.7 reports the scores for all methods collectively. It is evident from the table that the most effective method for toxic span detection overall, is the BERT-based sequence labeling model. It achieves the highest Precision-Recall AUC and F1 scores. A notable observation here, is that the supervised method achieves the highest scores even if the training dataset that is applied on is considerably smaller than those of the unsupervised ones. Apart from the difference between the methods, we also attribute the effectiveness of the supervised method to the pre-training of the BERT model, which is the reason why we did not implemented the supervised method with a Bi-LSTM as well.

	Precision	Recall	F1	PRAUC score
Bi-LSTM Input Erasure	0.553	0.383	0.453	0.411
BERT Input Erasure	0.53	0.4	0.458	0.413
Bi-LSTM LIME	0.575	0.408	0.477	0.424
BERT LIME	0.585	0.418	0.487	0.435
BERT Sequence Labeling	0.773	0.396	**0.522**	**0.55**

Table 3.7: Comparison of micro-averaging scores for all methods.

For further inspecting toxic span detection with Sequence Labeling, we computed Precision-Recall curves on the evaluation dataset from 200 to 8000 training comments as shown in gure 3.4. The PR-AUC score for each curve is in table 3.8. As we expected, adding extra training examples improves the PR-AUC score. Notice, that with 1000 training comments the supervised method achieves higher PR-AUC score than the best unsupervised method.

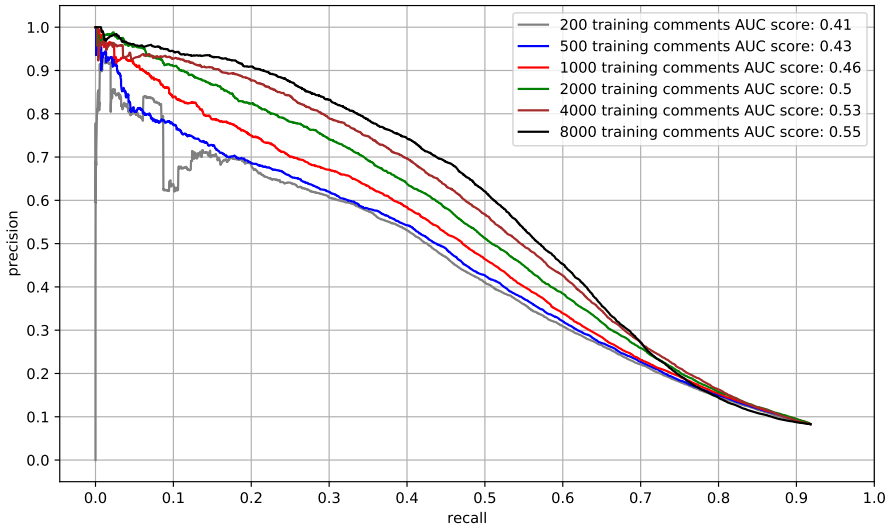


Figure 3.4: Precision-Recalls curves on evaluation CivilAnn subset applying the supervised method.

3.4 Hardware resources and computational demands

Initially, we implemented our experiments in Google Colab [3], a cloud environment provided by Google, which can be accessed through a web browser, based on the Jupyter project [4]. Google Colab is suitable for implementing deep learning experiments, however users

[3] Google Colab https://colab.research.google.com/notebooks/intro.ipynb
[4] Jupyter project https://jupyter.org/

	PR-AUC score
200 training comments	0.41
500 training comments	0.43
1000 training comments	0.46
2000 training comments	0.5
4000 training comments	0.53
8000 training comments	0.55

Table 3.8: Comparison of Precision-Recall AUC scores computed on the evaluation dataset.

are restricted by hardware and time limitations. Therefore, to carry out experiments with higher computational demands we subsequently gained access to a server of the University.

Considering hardware limitations, as already mentioned, due to the large size of the BERT model, it could not be trained in Google Colab, unless we froze several of its Transformer Layers. We overcame this limitation after gaining access to the University server.

The computational demands implementing each method in combination with the method's underlying model, varied considerably. An analysis of the time required for the implementation of each method follows.

The training of the Bi-LSTM model is by far faster than that of the BERT model. Specifically, in the case of the binary classifiers, the Bi-LSTM training on 100 thousand comments lasted a few minutes, while the training of the BERT model lasted 5 hours.

Furthermore, the Input Erasure method was also by far faster than the LIME algorithm, thus the implementation of the LIME algorithm depending on the BERT model was the method with the highest computational demands with a great difference from the second highest.

Finally, the BERT-based Sequence Labeling model, due to small size of the training dataset, it required approximately 40 minutes for training.

Table 3.9 displays the time that each method implementation required. Note that the times displayed in table 3.9 include the time needed for training the underlying model that each method is based on, and that they are approximate, rather than precise.

	hours
Bi-LSTM Input Erasure	0.3
Bi-LSTM LIME	1.2
BERT Input Erasure	6.2
BERT LIME	28
BERT Sequence Labeling	0.4

Table 3.9: Comparison of methods' computational demand regarding time.

3.5 Qualitative analysis

Considering that the BERT model for sequence labeling is the most e□cient model among those we examined, we also perform qualitative analysis, namely we inspect classi□ed comments to analyse better the behaviour of the model.

Below we present some examples. The highlighted text is the ground truth (human) toxic annotation and the text with red color is the classi□er's identi□ed toxic span.

Caution: The following examples contain real-life toxic comments, which the reader may nd disturbing.

The most frequent case was that the ground truth toxic annotation was an individual o□ensive word and the classi□er had identi□ed it.

- "It didn't take much rope for the village idiot to hang himself."
- "This clearly is why "Junior" is not quali□ed to be PM.....he's an idiot!"
- "Try the number 5 million, you idiots."
- "Really? Are you that stupid?"
- "WE have had alternative energy for 3 GENERATIONS, are you all so bloody Stupid? You think threatening BC is a good thing to do?"

There were also examples where the classi□er identi□ed more than one toxic spans.

- "Trump said, IN AS MANY WORDS, that Mexicans were stupidrapists and drug dealers."

- "Of course you cannot defend your own hypocrisy...I see you are starting of 2017 on the same losing track as always...oh, well, no cure for stupidity."

- "Bull hypocrisyshit. Did I spell that right? Or is it one word: bullshit. Je Bezos: do you tremble at the prospect of President Clinton?"

In several cases the classi er did not identify part of the toxic span. A reason that this happens at times is that the annotation is not an o ensive or disrespectful word, yet it is toxic due to the context. In the examples that follow, this is the case for the rst, the second and the last example.

- "Beijing millionaires and wanna be Beijing millionairesses who squeeze money from the rest of us! And stupid blind white men with blinders!"

- "Trump's comments are stupid, reckless and very dangerous. He is temperamentally and intellectually un t to be the person responsible for the lives of millions."

- "If you're dumb enough to believe him on this, then I've got a bridge in Brooklyn I will sell you...
  The guy is a pathological liar. Don't believe anything he says. It's all lies."

- "Stop with the silly suggestions. We all know what it is.
  Clowns... Killer clowns..."

- "Survival of the ttest would not have produced you. You are alive because your weak blood is supported by welfare and food stamps. Please don't reference Darwin in your icon. Loser."

- "Ignorant, sel sh, racist, misogynistic snow akes were THE major part of Trump gaining enough electoral votes. You can't x stupid."

Finally, the classi er occasionally identi ed toxic spans that were not included in the ground truth annotation, although maybe they should have.

- "Clinton should be the last person to say anything about anyone. She is such a lying scum bag. I can't handle just hearing her voice."

- "Trying the education route is best but you face a huge problem in that the average person today is dumber than a doughnut. Stupid, maybe you can do something but dumb, dumb is forever."

- "Trudeau and cabinet are the racist scum. Throw them all into the volcano, ashes to ashes."

- "How about disturbing the peace?? Hope he gets at least ten years without parole. Freaking idiots."

4 Artificial data

Language models in NLP are models that predict the next word in a sentence, as a result, these models are capable of producing artificial text. We examined if creating artificial data can benefit the models we deploy for detecting toxic spans. In this section we describe the model we adopted for generating text and the experimental results when the artificial data are including in the training set.

4.1 GPT-2

GPT-2 [Rad+19] is a state of the art language model developed by OpenAI. It is a large transformer-based language model trained on the task to predict the next word, given all of the previous words within some text. GPT-2 generates text samples when primed with an input and the text it generates is adapted to the style and context of the input.

As an initial experiment, I was provided with 432 artificial samples generated from a CivilAnn subset of 500 comments. These samples contain toxic span annotations indicated by the model as well. Below, we present some samples. The highlighted text denotes the toxic span. We also observed that within the model's synthetic text, there were repetitions of pieces of text.

Caution: The following examples contain real-life toxic comments, which the reader may find disturbing.

The most common generated toxic spans of the artificial text were individual offensive words. For instance,

- "You are ignorant. Pure and simple."
- "Or maybe you're just an idiot. I suppose that's possible too."

Another common toxic span category was inappropriate talking.

- "Stop with your "then you admit" crap. I admitted no such thing."

- "Pity you can only execute someone once. May he live a long life... in prison."

More rarely, toxic spans were racist or homophobic

- "And some people are so stupid that they will vote for a gay man because they are attracted to men. Or, they can be very stupid."
- "One anecdotal story does not make a fact. FACT - Planned Parenthood was started by Margaret Sanger to keep "Negros" from increasing in population."
- "The gay community is even more stupid than the average person."

or political. Note that, the toxic span of the second example does not consist of an abusive word, however it is toxic due to the context.

- "Microagression and liberal white guilt."
- "she's a perfect example as to why liberalism is a mental illness."

Other examples of toxic spans that do not consist of abusive words but are toxic because of the context are

- "I wouldn't be so crazy to expect our puppet legislators to be so reasonable....they are bought and paid for!"
- "Hey Craig, you do know that harassing any other blogger will get you 86 o this site don't you? The last time I checked, I have balls. How about you?"

The model produced also longer phrases as toxic spans, but it was very uncommon.

- "We don't know what this new group is or how it can be e ective. We don't know what useful equipment it can provide. Assholes are ruining this community by giving this crap a try."
- "I'd start by killing the LIO , Susitna Dam, Ambler Road, Knik Bridge, Juneau Rd. and the oil tax credits. Pain should be felt by all including the royalty occupying the LIO and oilies."
- "Quote from Governor Blagojevich : "No one is immune to be swept under the carpet, and the next door neighbor may be just as audacious and shameless as the clowns," he said."

4.2 Artificial data evaluation

To study the impact of adding the artificial data as training examples, we evaluate the model on the same subset of 690 CivilAnn annotated comments that we evaluated the toxic span detection methods on, computing the measures defined in section 3.2.2. Specifically, we compute learning curves for the F1 score in training, development and evaluation datasets, varying the size of the training dataset from 100 to 923 comments by 200 comments each time. For each training dataset size we also report the Precision-Recall AUC score, computed on the evaluation dataset.

We decided to carry out this experiment due to the relatively small number of artificial samples, thus the learning curves were calculated on these dataset sizes because, as mentioned earlier in this section, 500 annotated comments generated 432 artificial, so the total number of comments is 932, both artificial and the original. To form a clear understanding of the artificial data influence, we train the model initially with just the original comments and then we add artificial, to see if the scores improve. Therefore, we observe mainly the progress of scores for dataset sizes beyond 500.

Figures 4.1 and 4.2 illustrate F1 learning curves and Precision-Recall curves respectively for the different dataset sizes, and table 4.1 displays the Precision-Recall AUC scores. Since both the F1 score at the evaluation dataset as well as Precision-Recall AUC scores improve as the dataset size increases, we come to the conclusion that artificial data help the model detect toxic spans, though we see no improvement in test F1 beyond 700 training comments.

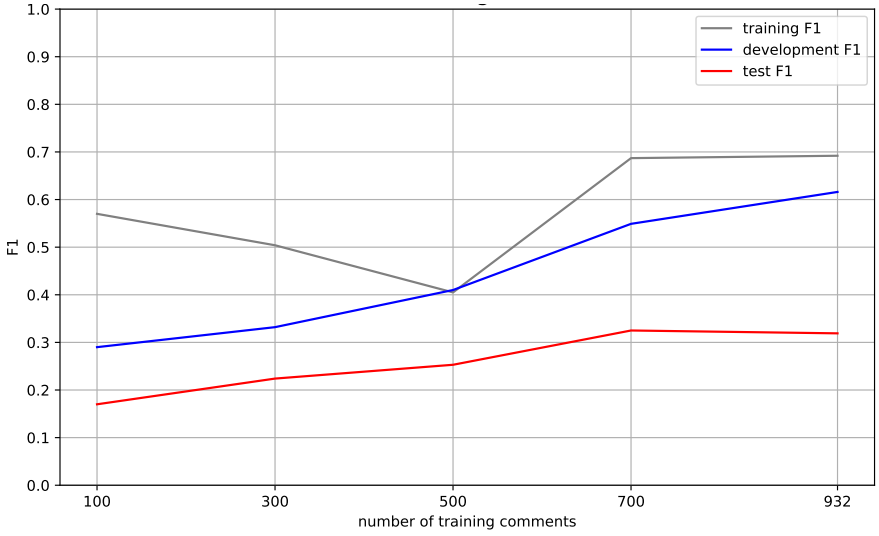


Figure 4.1: F1 learning curves at training, development and evaluation datasets.

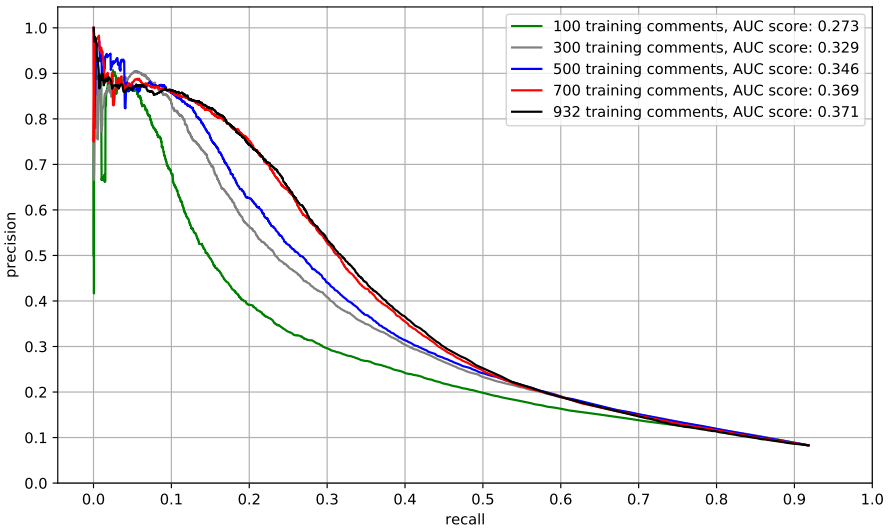


Figure 4.2: Precision-Recall curves computed on evaluation dataset.

	PR-AUC score
200 training comments	0.273
300 training comments	0.329
500 training comments	0.346
700 training comments	0.369
932 training comments	0.371

Table 4.1: Comparison of Precision-Recall AUC scores computed on the evaluation dataset.

5 Conclusions

Concluding, in this thesis we applied supervised as well as unsupervised methods for detecting toxic spans in comments. The unsupervised methods depend on supervised classifiers, and precisely binary classifiers that have been trained with respect to whether a comment is toxic or not. By contrast, the supervised span detection methods are directly trained on data that contain ground truth toxic spans. The measures that we use to evaluate our toxic span detection methods were the F1 score at character level as defined in section 3.2.2, and the Precision-Recall AUC score. The most effective method for identifying toxic spans based on those methods, was the supervised method implemented with a BERT base model for Sequence Labeling.

Subsequently, we augmented a subset of the CivilAnn dataset generating artificial data with the state-of-the-art language model GPT-2 and applied the BERT model for Sequence Labeling. We computed learning curves for the F1 score and the Precision-Recall AUC score. The results indicate that the artificial data can assist the sequence labeling model predict toxic spans though there were diminishing returns as we increased the size of the artificial dataset.

In terms of future work, one could investigate how self-attention mechanisms that have already been used in toxicity classifiers [PMA17] could also be used to detect toxic spans, comparing their performance to Input Erasure and LIME. Furthermore, since attention does not always provide reliable explanations about models' predictions, as suggested in [JW19], we aim to apply additional explanation methods. In particular, we plan to experiment with the input gradient-based method described in [RHD17] or apply the input reduction method of [Fen+18], where important words are removed iteratively from the input while the model's prediction is maintained.

We also intent to perform a larger scale experiment with artificial data to analyse more thoroughly the extend to which which artificial data can benefit in spans detection.

Finally, a long-term objective is, after identifying toxicity to modify the text and remove its toxicity, preserving the main text of the comment if possible. However, this is not always possible. To illustrate this better we present two examples. The comment

"People that do not live in Europe are inferior people"

cannot be modi□ed to a non-toxic comment, whereas the comment

" That was a stupid way to do it. "

can be modi□ed to

" You could have done it more e□ciently. "

which is not considered toxic anymore.

Bibliography

[CN16] Jason P.C. Chiu and Eric Nichols. “Named Entity Recognition with Bidirectional LSTM-CNNs”. In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 357–370 (cit. on p. 10).

[Da +19] Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. “Fine-Grained Analysis of Propaganda in News Article”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5636–5646 (cit. on p. 17).

[Dav+17] T. Davidson, Dana Warmsley, M. Macy, and Ingmar Weber. “Automated Hate Speech Detection and the Problem of O□ensive Language”. In: *ICWSM*. 2017 (cit. on p. 2).

[Dev+19] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL-HLT*. 2019 (cit. on pp. 7, 9, 11).

[Eis19] Jacob Eisenstein. *Introduction to Natural Language Processing*. MIT Press, 2019 (cit. on p. 10).

[Fen+18] Shi Feng, Eric Wallace, Alvin Grissom, et al. “Pathologies of Neural Models Make Interpretation Di□cult”. In: *EMNLP*. 2018 (cit. on p. 32).

[HS97] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (1997), pp. 1735–1780 (cit. on p. 7).

[JW19] Sarthak Jain and Byron C. Wallace. “Attention is not Explanation”. In: *NAACL-HLT*. 2019 (cit. on p. 32).

[KB15] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2015) (cit. on p. 8).

[LMJ17] Jiwei Li, Will Monroe, and Dan Jurafsky. *Understanding Neural Networks through Representation Erasure*. 2017. arXiv: 1612.08220 [cs.CL] (cit. on p. 4).

[MH16] Xuezhe Ma and Eduard Hovy. “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074 (cit. on p. 10).

[Nob+16] Chikashi Nobata, J. Tetreault, A. Thomas, Yashar Mehdad, and Yi Chang. “Abusive Language Detection in Online User Content”. In: *WWW*. 2016 (cit. on p. 2).

[Pav+20] John Pavlopoulos, Je□rey Scott Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. “Toxicity Detection: Does Context Really Matter?” In: *ACL*. 2020 (cit. on pp. 1, 2).

[PF17] J. Park and Pascale Fung. “One-step and Two-step Classi□cation for Abusive Language Detection on Twitter”. In: *ALW@ACL*. 2017 (cit. on p. 2).

[PMA17] John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. “Deeper Attention to Abusive User Content Moderation”. In: *EMNLP*. 2017 (cit. on pp. 2, 32).

[PSM14] Je□rey Pennington, Richard Socher, and Christopher Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543 (cit. on p. 8).

[Rad+19] A. Radford, Je□rey Wu, R. Child, et al. “Language Models are Unsupervised Multitask Learners”. In: 2019 (cit. on p. 28).

[RHD17] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. “Right for the Right Reasons: Training Di□erentiable Models by Constraining Their Explanations”. In: *Proceedings of the 26th International Joint Conference on Arti cial Intelligence*. IJCAI’17. Melbourne, Australia: AAAI Press, 2017, pp. 2662–2670 (cit. on p. 32).

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classi□er”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016) (cit. on pp. 4–6).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention is All you Need”. In: *NIPS*. 2017 (cit. on p. 9).

[Zam+19] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, et al. “Predicting the Type and Target of O□ensive Posts in Social Media”. In: *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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 1415–1420 (cit. on p. 2).

List of Acronyms

NLP Natural Language Processing

MLM Masked Language Model

NSP Nesxt Sentence Prediction

BERT Bidirectional Encoder Representations from Transformers

LIME Local Interpretable Model-Agnostic Explanations

AUC Area Under the Curve

NER Name Entity Recognition

POS Part of Speech Tagging

List of Figures

List of Tables