Dhruval Kikani

Deep Learning for Toxic Comment Detection in Online Platforms

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

8 May 2023

Abstract

Author:	Dhruval Kikani
Title:	Deep learning For Toxic Comment Detection In Online Platforms
Number of Pages:	42 pages + 1 appendix
Date:	8 May 2023
Degree:	Bachelor of Engineering Smart IoT Systems
Degree Programme:	Information Technology
Professional Major:	Smart IoT Systems
Supervisors:	Vesa Ollikainen, Senior Lecturer

This research aimed to explore the use of Deep Learning Artificial Neural Networks (ANNs) for toxic comment classification on social media and online forums. The prevalence of toxic interactions on these platforms has reached an all-time high, resulting in a decline in digital civility. The research reviews various algorithms and techniques for building promising ANNs and compares the performance of three chosen models on the Kaggle competition dataset to determine the best-performing ANN for this data problem.

The study includes a detailed overview of the background and techniques used for building and achieving the targeted results using Python and its libraries. It also covers technical aspects such as the building process, training, results, and evaluation, comparing the datasets using graphs and similar methods with the help of Python and its libraries. Classifying the nature of hate comments will provide flexibility for platforms to deal with them and open doors for new discussions and solutions.

Keywords: Online discussion forums, social media platforms, Digital civility, Toxic interactions, Deep Learning, Artificial Neural Networks (ANNs), Toxic comment classification, Algorithms, Kaggle competition, Python, Libraries, building process, Training, Results, Evaluation, Hate comments, Flexibility, New discussions, Solutions.

Contents

List of Abbreviations

AI: Artificial Intelligence refers to the ability of machines or computer programs to perform tasks that typically require human intelligence, such as learning, problem-solving, perception, and decision-making.

ANN: Artificial Neural Network is a computational model inspired by the structure and function of the human brain, used for various machine learning tasks.

CNN: Convolutional Neural Network is a deep learning model commonly used for image and video recognition tasks.

LSTM: Long Short-Term Memory is a type of recurrent neural network designed to handle sequence data and overcome the vanishing gradient problem.

ML: Machine Learning is a field of artificial intelligence that uses statistical techniques to enable computer systems to learn from data, without being explicitly programmed.

RNN: Recurrent Neural Network is a type of neural network that can handle sequential data by processing input data and storing previous states to inform future predictions.

1 Introduction

Over the last few years, online discussion forums and social media platforms have risen in popularity. These platforms allow people to openly express their thoughts, feelings, and sentiments while also facilitating information sharing and assisting people in seeking solutions to personal and educational concerns.

The civility of online communication reached a four-year low in 2019, according to Microsoft's Digital Civility Index [1].

Toxic interactions abound on social media platforms and online forums, which are dominated by harsh political debates and attacks on people's perceived attributes. What should be a good social force, a platform to freely discuss and debate online, has devolved into a quagmire of hate. In this, context AI/Machine learning can be a humongous help, since what can take human force hours to work on, can complete in a few seconds.

The objective of this thesis is to review existing promising Deep learning Artificial neural networks for toxic comment classification using different algorithms and techniques and compare three chosen models' performance on the Kaggle competition to find out the best performing ANN for such data problems. Classifying the nature of the hate comments will provide all platforms flexibility in dealing with it and open doors for new discussions and solutions.

The thesis contains a detailed overview of the background and techniques used for building and achieving the targeted results using Python and its libraries. The research shares valuable information about the theoretical background required, it also discusses some of the previous studies and solutions. The study also covers technical aspects such as the building process, training, results, and evaluation and compares the results using graphs and similar methods with the help of Python and its libraries.

2 Theory background

2.1 Basics of Machine Learning

Machine learning is an automated method of constructing analytical models that facilitates data analysis. Machine learning is a subfield of artificial intelligence that is based on the premise that machines can acquire knowledge from data, identify regularities, and render decisions with minimal human intervention.

The contemporary field of machine learning is distinguished from its historical antecedent by virtue of advancements in computational technologies. The objective of the study conducted by artificial intelligence scholars was to investigate the capacity of computers to acquire knowledge from data. The concept of pattern recognition and the notion that computers can acquire knowledge without explicit programming were the driving forces behind this research. The iterative aspect of machine learning is essential due to the ability of models to autonomously adapt to new data. In order to ensure the reliability and replicability of outcomes, it is imperative to draw insights from prior computations. It's an old science, but it has recently garnered new momentum [2].

How a prediction-making algorithm learns to improve its accuracy and reduces its loss is a common way to classify traditional machine learning.

Accuracy is a performance indicator used in machine learning to assess the predictability of a model. It calculates the fraction of correct predictions made by the model out of all predictions made by the model [3].

Loss, on the other hand, quantifies how well a model predicts output values for a given input. The difference between the predicted and true output values for the training data is calculated by the loss function, and the purpose of training the model is to reduce this difference. The loss function utilized is determined on the individual problem being solved and the type of model being employed [3].

In the field of data science, researchers employ four primary approaches known as reinforcement learning, semi-supervised learning, unsupervised learning, and supervised learning. The selection of an appropriate algorithm by data scientists is contingent upon the nature of the data they aim to predict [4].

Figure 1. illustrates all the different steps of the typical machine-learning process.

The very first step of the process is to get data, to be successful in the field of machine learning, one must practice many dataset types. However, finding an appropriate dataset for each type of machine-learning project can be challenging. A vast amount of data is required

to work on machine learning projects because ML/AI models cannot be trained without the data. One of the most important steps in developing an ML/AI project is gathering and preparing the dataset. These datasets can be obtained from various sources like Kaggle, UCI Machine Learning Repository, etc.

After finding the appropriate data, the next step is to clean the data set and prepare it for the training process which is also called prepossessing. Often there are flaws in a dataset or it requires some manipulation, like data might contain missing values or there might be cases where categorical data need to be encoded, these flaws and manipulations are addressed before training since they can make a significant difference in expected results after the data is clean it is divided into train test and validation split, this is also called data pre-processing.

The subsequent step after pre-possessing is the training, the data is trained using a model, and the model is crafted depending on the type of the problem. Different libraries are used to craft and build these training models.

After the training the next step is to test it, different types of graphs and visual representations are used to evaluate the results, testing the data also includes the last step which improving, depending on the performance of the model it is

manipulated, factors like changes in pre-processing or increase in training might help in getting the appropriate results.

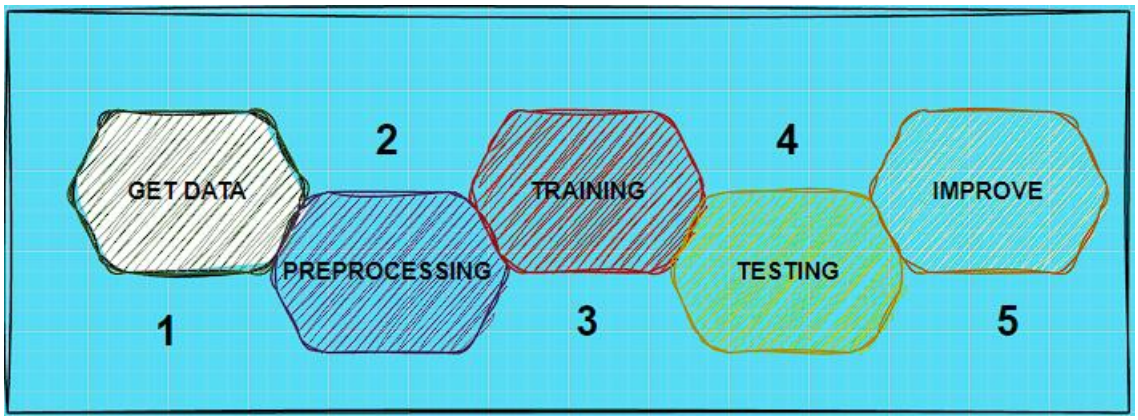


Figure 1: A workflow of a machine learning operation [5].

2.2 Dataset

A dataset in machine learning is a group of data points that can be analysed and predicted by a computer as a single entity. This means that since machines don't see data as humans do, the data collected should be standard and intelligible. There are 3 types of datasets: training, testing, and validation data test [6]. All types of datasets are explained in subsection 2.3.

Kaggle is an internet-based platform that caters to individuals who specialize in data science and machine learning, providing them with a community-oriented service. The platform enables users to engage in collaborative efforts with other users, locate and distribute datasets, utilize GPU-integrated notebooks, and participate in data science competitions through Kaggle. The datasets accessible on this platform, originate from various competitions or discussions.

The Kaggle dataset used here can have 2 types of sentences, figure 2 provides a visual illustration of one instance from each type of sentence present in the dataset, showcasing the distinction between the toxic and normal data categories.

```
print ("Toxic Sentence = ",training_data.comment_text.iloc[6])
print("Normal Sentence = ",training_data.comment_text.iloc[9])
```

```
Toxic Sentence =  COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK
Normal Sentence =  alignment on this subject and which are contrary to those of DuLithgow
```

Figure 2. Represents one example of both type sentences present in the dataset.

The types of sentences are two, but they can be categorised into different classes of the toxicity. Figure 3 shows the distribution of toxic comments into different classes of the toxicity with respect to the number of counts.

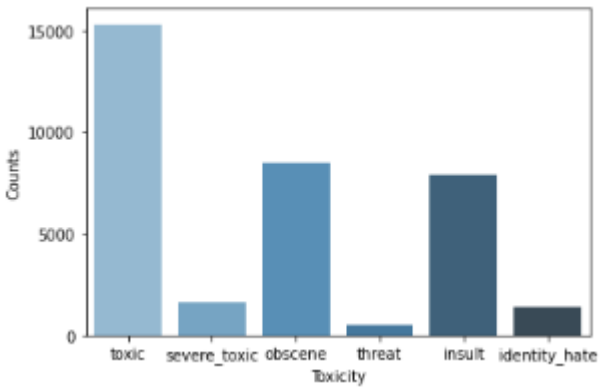


Figure 3. Data visualisation of Kaggle data

2.3 Training, testing, and validation data test

The dataset is divided into 3 sections called training, testing and validation, each of the data sections has a different use case from the other and has a significant role in the machine learning process.

Figure 4 below shows the percentage in which the data is split from the original dataset.

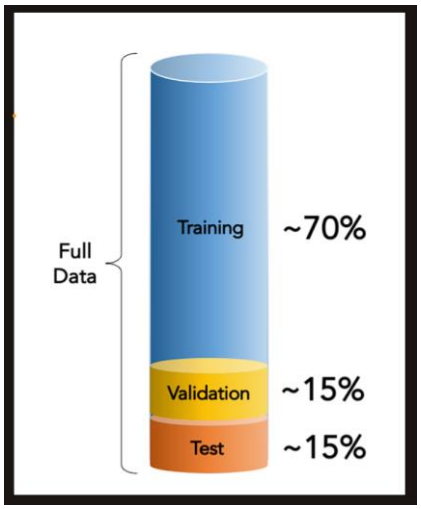


Figure 4. Displays the training, testing, and validation split [7].

Training data is a form of data that is utilized to construct the machine learning algorithm. The algorithm is supplied with input data that conforms to the intended outcome by the data scientist. The model engages in ongoing evaluation of the data in order to gain further insights into the data's patterns, subsequently adapting itself in order to attain its objective [8].

The concept of validation data pertains to a dataset that incorporates novel data points that have not been previously evaluated by the model during the training phase. Validation data is utilized by data scientists to evaluate the efficacy of the model's predictions on new data. This serves as an initial test to gauge the model's performance on unobserved data. Validation data is not often used by data scientists, but it might offer some useful information for optimizing the hyperparameters that affect how the model evaluates data [8].

The testing data refers to a dataset that is utilized after the development of a model, with the aim of validating the model's ability to generate accurate predictions. It is important to ensure that the testing dataset is unlabelled in cases where the training and validation datasets are labelled, as this enables effective monitoring of the model's performance metrics. The utilization of test data provides a prompt and pragmatic means of validating an unfamiliar dataset, thereby demonstrating the efficacy of the machine learning algorithm that was employed [8].

2.4 Machine learning techniques

2.4.1 Supervised and unsupervised learning

Supervised and unsupervised learning are machine learning techniques or strategies which are unique to their application depending on the expected output.

The major distinction between the two methods is, unsupervised machine learning processes unlabelled or raw data, whereas supervised machine learning uses labelled input and output training data [9].

Supervised learning is a subset of machine learning and artificial intelligence which is characterized by the way it trains algorithms using labelled datasets to produce and predict accurate outcomes [10].

The learning process in supervised learning can be categorised into four parts, figure 5 shows all the learning phases in order which are to learn and recognize pattern in data.

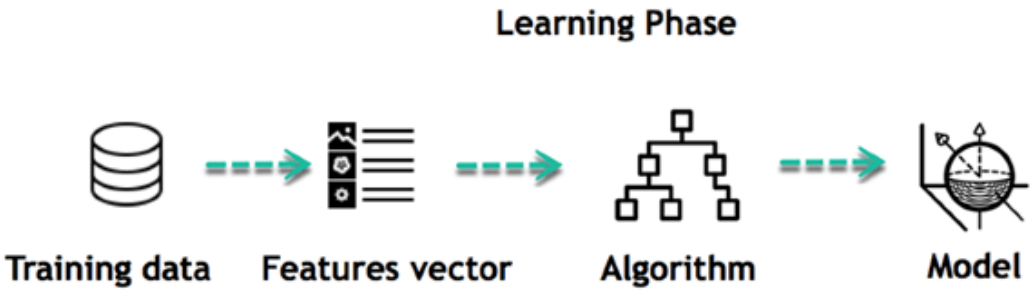


Figure 5. Represents the learning process of supervised learning [11].

It starts by collecting or gathering appropriate labelled data. After getting the data, it is pre-processed and split into a training, test, and validation dataset.

After the pre-processing the next step is to identify the training dataset's input features which are also called features vectors, the vectors should have sufficient knowledge so that the model can accurately predict the output.

After determining the features, the following step is to identify a suitable algorithm for the creation of a training model which is the following step in the process. The process is finished by executing the model and evaluating its results or outcomes by checking its accuracy through various methods.

The supervised learning algorithm can be further categorized into two types of problems classification and regression.

Classification refers to the process of organizing a particular dataset into distinct categories or classes. The first phase involves the prediction of the class of a given set of data points. Classes can be alternatively denoted as targets, labels, or categories. Classification predictive modelling refers to the process of estimating the mapping function that relates discrete input variables to output variables. The primary aim is to determine the classification or grouping to which the novel data will be assigned. Several classification algorithms include random forests, decision trees, and SVM (Support Vector Machine) [10].

There can be different types of classification depending on the expected output of the problem, it can be categorized into Binary classification, multi-class classification, and multi-label classification**.**

Binary classification is where the outcome depends on two classes. Example: Male or female, Spam email or not.

Multi-class classification is where the outcome depends on more than two classes.

Multi-label classification is a type of classification where each sample is assigned to a set of labels or targets, this thesis is a part of multi-label classification since the output of the toxic comments can be more than 1

meaning some comments may be a part of the threat category as well can be part of insult category.

Regression is a type of supervised learning which is used to comprehend the relationship between dependent and independent variables, it is frequently used to produce projections, such as for a company's sales revenue for a given business. Some of the regression algorithms include linear regression, logistical regression, and polynomial regression [10].

unsupervised learning is a machine learning technique that allows models to be trained on unlabelled data and then allows them to operate independently. In other words, models are not supervised using training datasets, instead, models themselves decipher the provided data to reveal hidden patterns and insights [12].

The unsupervised learning algorithm can be further categorized into two types of problems clustering and association.

Clustering is an unsupervised learning algorithm where data points that have similar properties are divided into groups or clustered together so that the data points within each group are more similar to one another and different from the data points within the other groups.

Association is an unsupervised learning methodology that utilizes a rule to identify the relationships between variables in a large database. The collection is characterized by a set of items that exhibit co-occurrence, as stated in reference [12].

2.4.2 Deep learning and neural networks

Deep learning is a branch of machine learning that prioritizes learning. Deep learning in the present era frequently comprises numerous consecutive representation layers that are exposed to training-based automated learning

data. The term "neural networks" refers to these model-based representations that are successively overlaid [13].

A network of neurons is called a neural network which has three or more levels, including a hidden layer, an output layer, and an input layer. The origin of the phrase "neural network" motivated by our growing understanding of the human brain, is neurobiology. It is intended to replicate, how the brain functions in humans, so that computers may be programmed to handle abstractions, issues that are not clearly defined [13].

2.4.3 Architecture of neural networks

Artificial neural networks (ANNs) are computational models based on the structure and operation of biological neurons. ANNs come in a variety of classes, each with a distinct architecture and set of uses [14].

The data flow and basic architecture of ANN can be seen from figure 6.

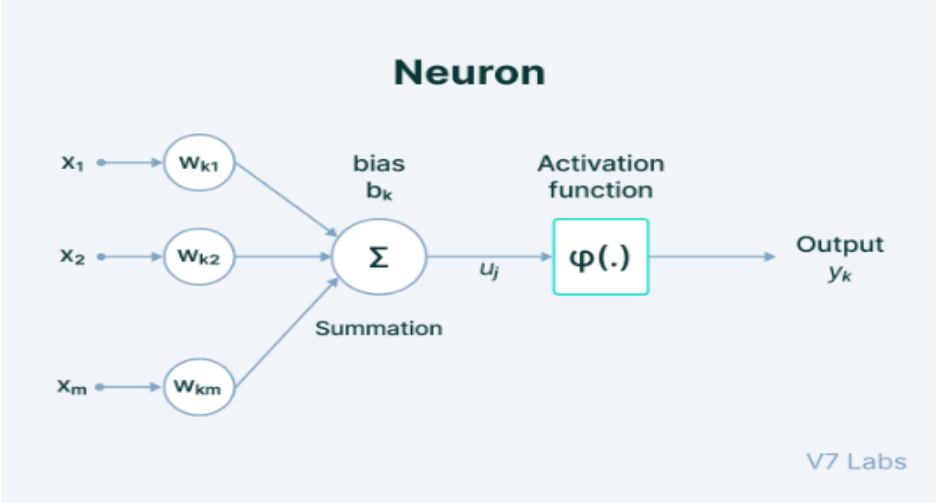


Figure 6. Represents the neurons in Artificial Neural Network [15].

Neurons and layers make up the neural network. The core component of neural networks are neurons which are represented by the filled small circles in Figure 6, these are networks that receive information and send the result to other cells

in the layer. The synthetic neuron receives some input values, such as x = [x1, x2, ... xm], each of which is multiplied by a particular weight [wk1, wk2, ... wkm].

These weighted inputs are added together to create the neurons' logits, z.

Then, a function f, also referred to as an activation function, is fed through the neuron's logit. The output of the activation function, y = f (z+b), where b is the bias factor, indicates the presence of activated neurons. This output y from the function can be transmitted to another neuron. Different activation functions exist, namely Sigmoid, Tanh, ReLU, and softmax [15].

The multi-layer neural network in figure 7 demonstrates the complex hierarchical structure that allows the network to learn and model intricate relationships in the data. The specific architecture, including the number of hidden layers, the number of neurons in each layer, and the activation functions used, can vary depending on the problem being solved.

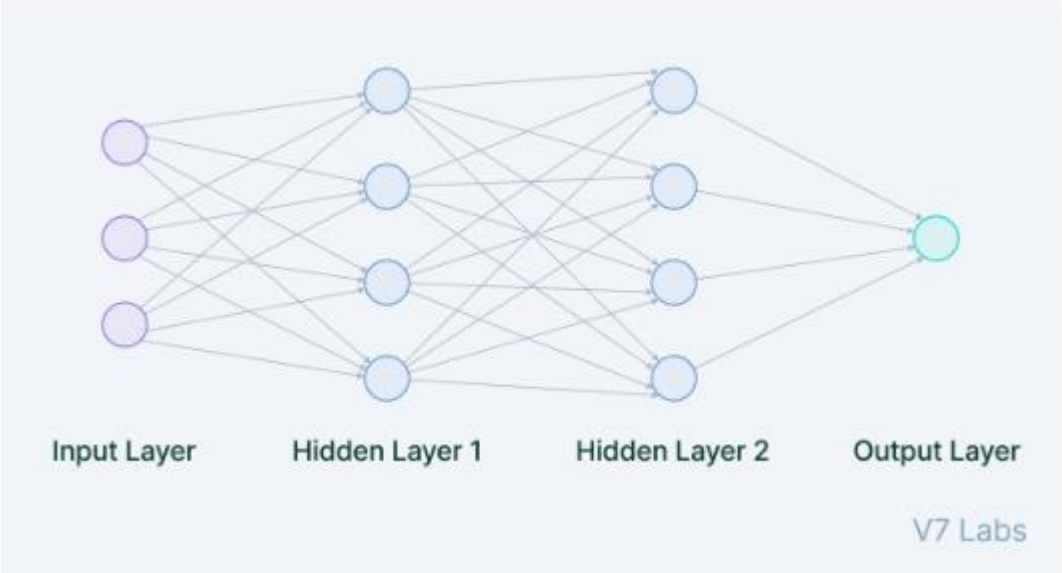


Figure 7. Demonstrates a multi-layer neural network [15].

The input layer is the initial layer in the neural network, and its neurons called Input neurons are in this layer. The output neurons are in the right output layer which can contain one single output neuron or multiple neurons. The layers in the middle of the input and output layers are called hidden layers, through the input layer, the initial input from the input data is absorbed and stored in input neurons. Then, the hidden layer modifies the data, and the nodes are subjected to the output layers based on the weights. Numerous simple processing nodes that are intricately coupled may number in the thousands or millions in a typical neural network. For deep learning, the number of hidden layers is generally high than in the standard neural network, the number of hidden layers depends on the complexity of the problem.

2.5 Overfitting and Underfitting in Machine Learning

Overfitting is a concept where the model learns the training data so well that it fails to perform with the unseen testing data. In overfitting when a model is trained, it starts learning from the noise and inaccurate data entries in our data set causing high variance in results when used with the testing dataset. Overfitting can be avoided or reduced using techniques like early stopping, regularization, etc [16].

Underfitting is a concept where the model is unable to learn the patterns from the data, in a situation where the model is underfitting it still performs decently with the training data but does fall short when checked with testing data. It generally occurs when less data is used to build an accurate model. Underfitting can be avoided or reduced by increasing the complexity of the model or increasing the features so that it can learn better [16].

Figure 8 visually demonstrates the differences between these three scenarios, allowing us to compare their characteristics. It provides valuable insights into the impact of model complexity on performance and serves as a useful reference for understanding the trade-off between underfitting, appropriate fitting, and overfitting in machine learning.

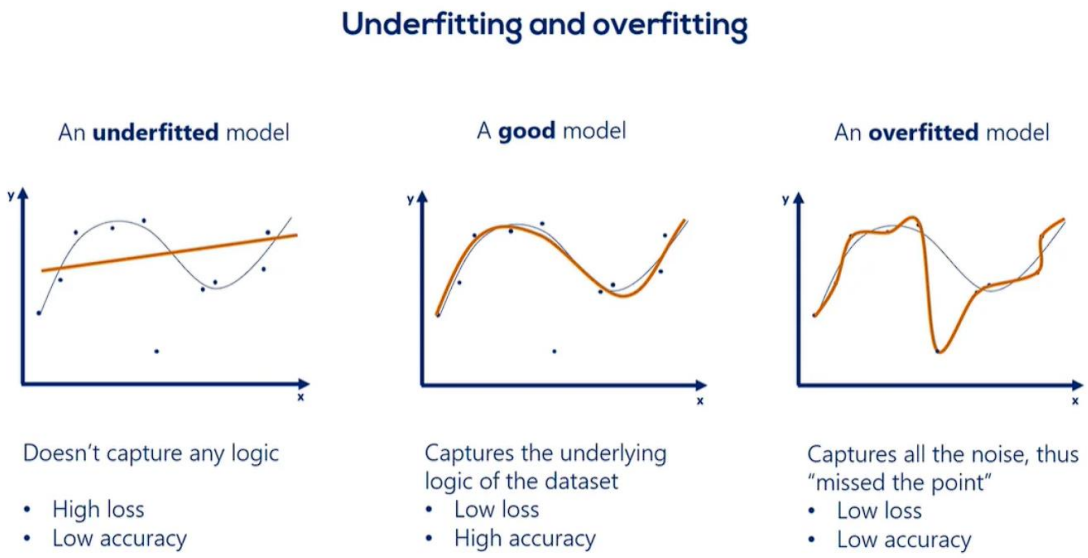


Figure 8. Illustrates a side-by-side representation of underfitting, Appropriate-fitting, and overfitting [17].

As seen from the above image a good way to identify underfitting and overfitting is to look at its loss and accuracy curves, a model with low loss and high accuracy is appropriate and a good model.

2.6 Previous Studies

There have been several past studies on classifying toxic comments. The number of different approaches to this problem is very high, different ML engineers practice different techniques to find an appropriate solution to this and help the online community.

The most basic solution to it is to classify whether the comment is toxic or not, which means a binary classification problem. Research from Kevin Khieu and Neha Narwal from Stanford share results stating LSTM (Long Short-Term Memory Networks) showed good results for binary classification with an accuracy of 0.889 and precision of 0.925, the report also mentions good performance using the CNN models as well [18].

Furthermore, there have been advanced studies made on the given problem, and there have been competitions for deciding the best models and approaches on Kaggle which consist of multiclass classification solutions, which is classifying the comments into different toxic categories like toxic severe toxic, obscene, threat, insult, or identity hate. The first winner's Toxic Crusaders used techniques like Diverse pre-trained embeddings, TTA, Rough-bore pseudo-labelling (PL), and Robust CV + stacking framework to achieve a leaderboard score of 0.98856 [16]. One of the participants Alexander Burmistrov had made a standard single model using RNN and word embeddings and it was implemented using Keras. The model had 6 layers and was able to achieve a 0.9872 leaderboard score [19].

A study, Machine learning methods for toxic comment classification: a systematic review by Darko Androcec is about the different ML models used in research papers. According to the report, the most effective and used methods were different deep neural network techniques, but simpler supervised learning methods like logistic regression were also used for baseline approaches [20].

There are many unique approaches to the classification of Kaggle and one can be very invested in those different yet useful projects.

3 Project Setup

This section focuses on the different tools like software and computer languages used in the research.

3.1 Software Tools

Jupyter Notebook also known as IPython Notebook is a web-based interactive development environment that serves as a platform for generating notebook documents [21]. This tool offers a versatile interface, empowering users to customize and organize workflows in various domains such as data science, scientific computing, computational journalism, and machine learning.

3.2 Programming languages

Python is a high-level, all-purpose programming language. The design philosophy of the code prioritizes readability and relies on indentation extensively.

Python employs automatic memory management through the process of garbage collection and incorporates dynamic typing. It provides support for a diverse range of programming paradigms, encompassing procedural, object-oriented, functional programming, as well as structured programming. Its wide-ranging standard library contributes to its reputation as a language that includes a comprehensive set of functionalities, often described as a "batteries included" language [22].

In the early day’s machine learning tasks were very time-consuming since all the job was done manually with the help of mathematical and statistical formulas. This made the whole process very tedious, and inefficient. But in modern days it became easy and efficient with the help of different libraries, frameworks, and modules Python has to offer, it makes the whole process a lot quicker since all the calculations and statistical analysis are done directly by the computer resulting in better results. Python has a lot of libraries to offer, and each provides different functionality, some of which are Numpy, Scikit-learn, TensorFlow, Keras, pandas, etc.

4 Exploratory Data Analysis (EDA)

This is the stage at which data is analysed using the creation of plots to find out trends, anomalies, and unseen patterns in the data, EDA provides necessary steps for future steps in building and training.

4.1 Pre-processing

Data pre-processing is a critical stage in machine learning that improves the quality of the data to encourage the extraction of valuable insights from the data. Data pre-processing in machine learning is, to put it simply, a data mining approach that converts raw data into a format that is readable and intelligible [23].

Preparing raw data to make it acceptable for creation and training may include many steps like cleaning data and handling missing values. The number of steps in the pre-processing may vary depending on the number of flaws the data has or what type (image, text, etc) of data is used.

The sections below (4.1.1 – 4.1.4) represent all the pre-processing steps taken to make the data ready for training.

4.1.1 Cleaning Data

This section focuses on the approaches used for cleaning text data with the help of Python and its libraries.

Figure 9 represents the function used to clean the Kaggle data with all steps taken into consideration for cleaning the data.

```
def  clean_text(text):
    text =  text.lower()                                                    # Step 1
    text = contractions.fix(text)                                           # Step 2
    text = re.sub(r"https?://\S+|www\.\S+", "", text)                       # Step 3
    html = re.compile(r"<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});")  # Step 4
    text = re.sub(html, "", text)                                           # Step 5
    text = text.translate(str.maketrans('', '', string.punctuation))        # Step 6
    emoji_pattern = re.compile(
        '['
        u'\U0001F600-\U0001F64F'  # emoticons
        u'\U0001F300-\U0001F5FF'  # symbols & pictographs
        u'\U0001F680-\U0001F6FF'  # transport & map symbols
        u'\U0001F1E0-\U0001F1FF'  # flags (iOS)
        u'\U00002702-\U000027B0'
        u'\U000024C2-\U0001F251'
        ']+',
        flags=re.UNICODE)
    text = emoji_pattern.sub(r'', text)                                     # Step 7
    text = re.sub("(\\W)"," ",text)                                         # Step 8
    text = re.sub('\S*\d\S*\s*','', text)                                   # Step 9

    return text
```

Figure 9. Demonstrates the Python code for cleaning data.

The most common technique in text cleaning is capitalization or lowercase due to the diversity of capitalization to form a sentence. This technique will project all words in text and document into the same feature space. However, it would also cause problems with exceptional cases such as the USA or UK, which could be solved by replacing typos, slang, acronyms, or informal abbreviations technique.

After lowering all the text next step taken is using the contraction package. The contraction package has a function called “fix” which is used to expand the contraction in English for example we'll -> we will, or we shouldn’t -> we should not have. The next step important step is the removal of noise from the data, this implies removing various unnecessary characters like URLs, non-ASCII characters, etc. Steps 4 and 5 are used to remove the HTML tags present in the data. The next step is to remove punctuations since these texts are taken from online discussion panels there is a high chance that they might have unnecessary punctuations like before starting the sentence or more than one punctuation after the sentence. Step 7 represents the method used to eliminate emojis or other symbols from the data, since this data is taken from online data

there might be cases where emoticons or flags might be used to describe more to the given sentence more. Steps 8 and 9 are additional steps taken just to make the data purer.

4.1.2 Tokenization, Indexing, and Index Representation.

1. Tokenization - Tokenization is an essential method of Natural Language Processing, in which text is broken into smaller pieces called tokens. These tokens may consist of any single word, phrase, or entire sentence and are used as input data for machine learning models. The process of tokenization involves removing all syntax, special characters, and white space from the text before splitting it into a set of tokens, as this will allow the ML model to analyse text in a more detailed way and then determine patterns and relationships between tokens [24].

2. Indexing - The technique of assigning numerical indices to categorical variables or features in a dataset is known as indexing in machine learning (ML). Since most ML algorithms require numerical data as input, this technique is crucial. It is easier to evaluate and analyse categorical variables when they are converted to numerical values using indexing, a pre-processing step [25].

3. Index Representation - In machine learning (ML), the term "index representation" refers to the use of indices to represent the values of categorical features or variables in a dataset. In NLP applications, this method is frequently applied, where words are represented by their index in a lexicon or dictionary.

All the steps mentioned above can be easily achieved by utilizing the "Tokenizer" class, which is part of the "Keras" library, plays a crucial role in facilitating the process of transforming a textual corpus into a vectorized representation. This involves converting each piece of text into either a sequence of integers, where each integer corresponds to the index of a token in

a predefined dictionary, or a vector where each token is represented by a binary coefficient derived from its word count.

Figure below, illustrates the conversion of text data into sequence vectors.

```
from tensorflow.keras.preprocessing.text import Tokenizer
max_features=100000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(Clean_data))
list_tokenized_train = tokenizer.texts_to_sequences(Clean_data)
```

Figure 10. Python code to convert text data into sequence vectors.

4.1.3 Lemmatization

Lemmatization is a process in which words are lemmatized meaning it is reduced to their lemma, or root form while keeping their meaning in the context of the phrase. Natural language processing (NLP) applications like text classification, sentiment analysis, and information retrieval frequently employ this method. Lemmatization is frequently contrasted with stemming, another method for breaking down words into their simplest forms, although it is less harsh and keeps the text's original meaning intact [26].

Python supports different kinds of libraries for lemmatization, to implement lemmatization in this project “WordNetLemmatizer” was used from the “ntkl” library.

WordNet Lemmatizer operates by determining a word's part of speech before using the WordNet lexical database to locate the base form or lemma. This tool is frequently used in NLP research and business and is offered as a component of well-known NLP libraries like NLTK and spaCy [26].

Figure 11 represents the function used for lemmatization, with use of “WordNetLemmatizer”.

```
# Init the Wordnet Lemmatizer
lemmatizer = WordNetLemmatizer()

def lemma(text, lemmatization=True):
  output=""
  if lemmatization:
    text=text.split(" ")
    for word in text:
       word1 = lemmatizer.lemmatize(word, pos = "n")
       word2 = lemmatizer.lemmatize(word1, pos = "v")
       word3 = lemmatizer.lemmatize(word2, pos = "a")
       word4 = lemmatizer.lemmatize(word3, pos = "r")
       output=output + " " + word4
  else:
    output=text

  return str(output.strip())
```

Figure 11. Represents the Python code for lemmatization with help of the “NTKL” Library.

4.1.4 Stop words.

In machine learning (ML) applications, stop words are common natural language words that are frequently removed from text data during pre-processing. These terms lack significance and have the potential to introduce noise into the analysis, resulting in inaccurate models.

Words like "the," "and" "an," and "in" are all examples of stop words. In NLP applications like text classification, topic modelling, and sentiment analysis, removing stop words is a common pre-processing step.

The evacuation of stop words can work on the proficiency and exactness of the ML model by decreasing the dimensionality of the information and zeroing in on the more significant words. However, it is essential to keep in mind that removing too many stop words may cause some contextual information to be lost, resulting in less accurate models [27].

There are several libraries such as NLTK, spaCy, and Scikit-learn that provide built-in stop words lists for multiple languages that can be used to make the data cleaner and more stable.

This project uses the spacy library since it has an extensive list of stopwords, after careful examination of the data, it was noticed the data contained unnecessary instances where single-letter or two-letter words existed without any context (e.g., “He is such a happy guy bb”.). To overcome this problem a custom stop words list was added to the default list provided by the spacy library, the custom list contained single and double letters from a to z (e.g., aa, bc).

Note – Certain letters were removed from the custom list like me, am, as, or letters like I and a, since they provide significant meaning in a sentence.

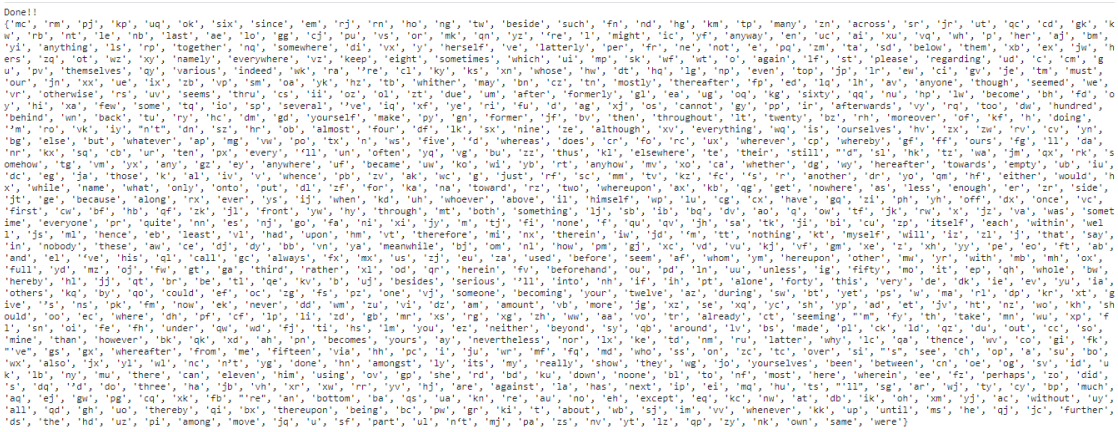


Figure 12. Represents the list of stop words (spacy + custom).

4.1.5 Padding

The comments from the online conversation are in variable lengths some may be one-word answers while the others may be paragraph-long, machine-learning algorithms cannot pass inconsistent lengths of data to the model, To circumvent this issue padding is introduced.

In machine learning (ML), padding is the process of padding sequences with zeros or other values to make sure they are of the same length. Padding is frequently used in NLP applications, where ML algorithms are fed word or character sequences as input.

Padding is required since most ML algorithms demand input data to be a specific size. Without padding, sequences of various lengths would need to be stretched or shortened, potentially causing the loss of crucial data.

Based on the ML algorithm, padding can be introduced to the beginning or end of sequences. Zero padding is the most typical type of padding; zeros are added to the sequence until it meets the required length [28].

To implement padding in this thesis “pad_sequences” was used from the “Keras” library. The length of sentences was fixed to 200 words and post padding was implemented i.e., for all the sentences shorter than 200 words 0’s will be added at the end of the sequence vector.

The figure 13 shows the python code and steps taken to implement padding with the help of keras.

```
from keras.preprocessing.sequence import pad_sequences
maxpadlen = 200
X_padding=pad_sequences(list_tokenized_train, maxlen=maxpadlen, padding = 'post')
```

Figure 13. Represents the implementation of padding.

4.2 Training/Test/validation Split

The data after going through the pre-processing stages is now more stable and consistent which brings the next step which splitting the text data into training and validation sets.

To achieve the data split, “train_test_split” was used from “sklearn”, as represented in figure 14.

```
toxic_comments_labels = train_data[["toxic", "severe_toxic", "obscene", "threat", "insult", "identity_hate"]
y = toxic_comments_labels.values
x_train, x_val, y_train, y_val = train_test_split(X_t, y, test_size=0.20, random_state=42)
```

Figure 14. Illustrates splitting the data into training and validation sets.

5 Transfer Learning

Transfer learning is a technique that allows pre-trained models to be used as a starting point for new text classification tasks. These pre-trained models are trained on large datasets and can capture complex patterns and relationships in language.

There are various advantages to transfer learning in text classification. The first benefit is that it enables quicker training and improved performance on smaller datasets. Second, by learning more generalizable features from the trained model, models can improve. Thirdly, it lessens the requirement for significant amounts of labelled data [29].

In this thesis, pre-trained word embeddings from” fastText” were used to harness the power of transfer learning.

fastText is an open-source, lightweight library that enables users to learn text representations and perform text classification tasks. It is based on the skip-gram word embedding model and was created by the Facebook AI Research (FAIR) team [30].

The capability of fastText to learn sub-word embeddings is one of its important attributes. This means that in addition to learning word embeddings, it may represent a word as a collection of character n-grams (sub-words) and learn

embeddings for these n-grams. This is especially helpful for morphologically complicated languages where words can take on a wide variety of forms [30].

A code snippet that showcases the practical implementation of fastText for generating word embeddings is represented in figure 15.

```
word_index = tokenizer.word_index
embedding_dim_fasttext = 300
embeddings_index_fasttext = {}
f = open('/kaggle/input/sarcasm-with-wiki-and-golve-embeddings/wiki-news-300d-1M.vec/wiki-news-300d-1M.vec'
for line in f:
    values = line.split()
    word = values[0]
    embeddings_index_fasttext[word] = np.asarray(values[1:], dtype='float32')
f.close()

embedding_matrix_fasttext = np.random.random((len(word_index) + 1, embedding_dim_fasttext))
for word, i in word_index.items():
    embedding_vector = embeddings_index_fasttext.get(word)
    if embedding_vector is not None:
        embedding_matrix_fasttext[i] = embedding_vector
print(" Completed!")
```

Figure 15. Implementation of “FastText” word embeddings.

To implement FastText word embeddings the following steps were performed.

1) FastText word embeddings were loaded into the environment.
2) The vocabulary was assigned to the pre-trained word embeddings.
3) Generate an embedding matrix.

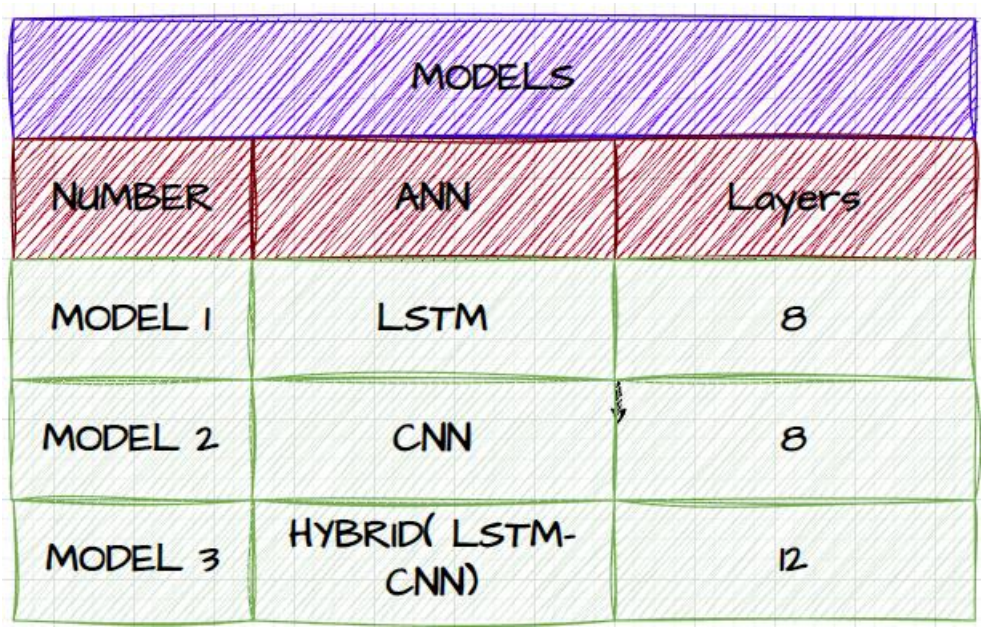
6 Training and Development Testing

After the completion of the prepossessing and feature engineering stage, the next step is model creation and model assessment.

This section covers the 3 models created for the training and shares relevant important information about the different parameters that can be considered while creating a machine learning model.

All the 3 models will have different ANNs and architecture respectively, the table 1 below gives information about the models.

Table 1: Displays information about all the 3 models along with the respective ANN used and the number of layers.



MODELS		
NUMBER	ANN	Layers
MODEL 1	LSTM	8
MODEL 2	CNN	8
MODEL 3	HYBRID(LSTM-CNN)	12

6.1 Classes of artificial neural networks

There are several artificial neural networks, this section focuses on the neural networks generally used for text classification problems.

CNN

Convolutional Neural Networks (CNNs) are a type of neural network that has proven to be extremely effective in image recognition applications. They have, however, been deployed successfully in natural language processing (NLP) applications like text classification.

CNNs are highly suited for text categorization because they can recognize essential text properties such as n-grams and phrases without requiring complex feature engineering. In a CNN, the input data is commonly represented

as a matrix of word embeddings, with each row representing a single word and each column representing an embedding characteristic or dimension.

The CNN architecture is made up of a succession of convolutional layers that scan the input matrix for patterns or features using filters. The filters are applied to each possible word window in the input, allowing the network to learn patterns at various scales. The convolutional layer's output is subsequently passed via a pooling layer, which reduces dimensionality and collects the most significant information. Finally, before making the final classification judgment, the output of the pooling layer is sent through one or more fully linked layers [31].

An example of CNN architecture is represented below in figure 16.

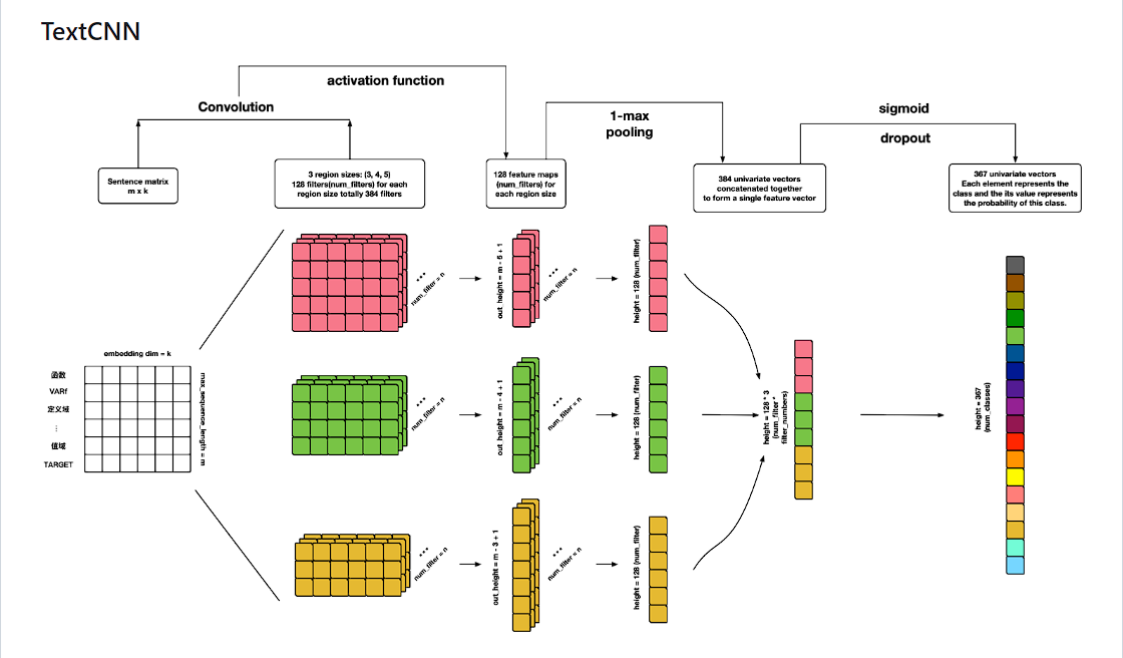


Figure 16. Represents an example of CNN in text classification [32].

RNN

Recurrent Neural Networks (RNNs) are a type of artificial neural network that works well with sequential data, such as text. RNNs have seen a lot of success in natural language processing (NLP) applications including text classification.

The input data to an RNN is commonly represented as a sequence of word embeddings, with each word in the sequence represented as a vector. The RNN analyses the input sequence one word at a time while keeping a hidden state that stores information about the words before it. At each time step, this concealed state is updated and used to inform the processing of the following word in the sequence.

The most common type of RNN is the Long Short-Term Memory (LSTM) network. The ANN used in this thesis is also LSTM.

LSTMs were created to address the problem of vanishing gradients in RNNs that can occur when training on extended sequences.

To do this, LSTMs employ system comprises a memory cell and three distinct gates, namely the input gate, output gate, and forget gate. The input gate regulates the extent to which the new input is incorporated into the memory cell, while the forget gate determines the degree to which the previous memory cell state is retained. Furthermore, the output gate governs the amount of the current memory cell state that is emitted as output [33].

The visualisation of the gates can be seen in the Figure 17 below.

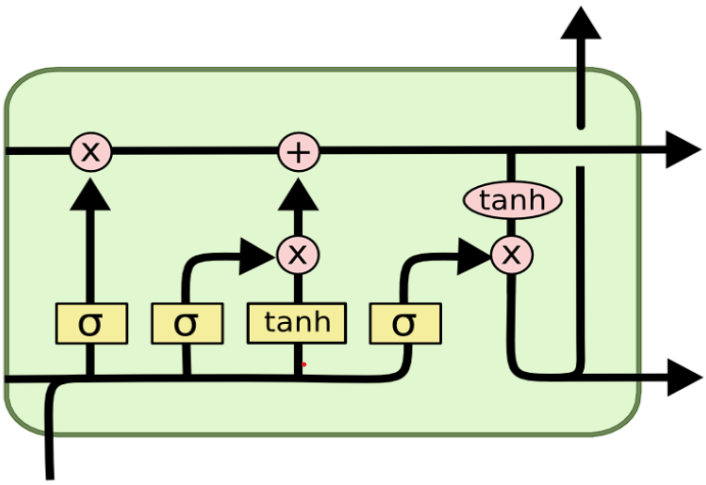


Figure 17. Internal Structure of an LSTM node [34].

6.2 Hyperparameters

Hyperparameters in machine learning are parameters that are set prior to train a model and are not learned from the data during training. They are frequently set by the user or determined through trial and error.

Learning rate, batch size, number of hidden layers, number of neurons in each hidden layer, regularization parameter, and activation function are examples of hyperparameters.

The performance of a machine learning model can be significantly influenced by hyperparameters. If the hyperparameters are not effectively set, the model may underfit or overfit the training data, resulting in poor test data performance.

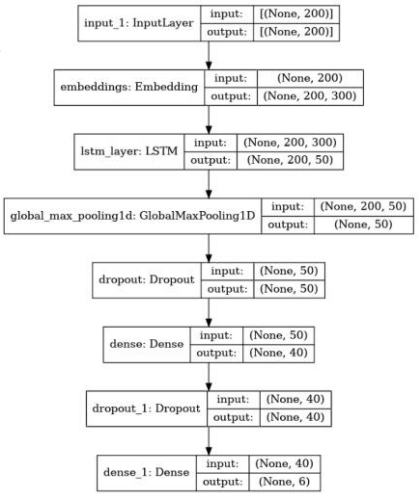
The practice of determining the best hyperparameters for a specific task is known as hyperparameter tuning. This is usually accomplished using a combination of hand adjustment and automated techniques [35].

6.3 Models

6.3.1 LSTM

The first model used was the Long Short-Term Memory artificial neural network since it is most consistent with text data. A hierarchical display of the LSTM model and information regarding layers and neurons can be found in the figures 18 & 19.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 200)]             0
_________________________________________________________________
embeddings (Embedding)       (None, 200, 300)          56806500
_________________________________________________________________
lstm_layer (LSTM)            (None, 200, 50)           70200
_________________________________________________________________
global_max_pooling1d (Global (None, 50)                0
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 40)                2040
_________________________________________________________________
dropout_1 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 246
=================================================================
Total params: 56,878,986
Trainable params: 72,486
Non-trainable params: 56,806,500
_________________________________________________________________
```



Figures 18 & 19. The summary and architecture of the LSTM model.

The model consists of one input layer, one embedded layer followed by one LSTM layer with 50 neurons, the model has 2 dropout layers, 2 dense layers and one global max pooling layer in the order shown in the figure.
Dropout Layers: A regularization technique used in neural networks to prevent overfitting. During training, some neurons in dropout layers are randomly dropped out (i.e., set to zero), preventing the model from depending too heavily on any one feature or collection of features. This contributes to the model's generalization performance [36].

Dense Layers: The basic building elements of a neural network are dense layers. They are fully connected layers, with every neuron in one layer connected to every neuron in the next. Dense layers can be layered to construct deep neural networks and are used to learn complicated non-linear correlations between input and output data [37].

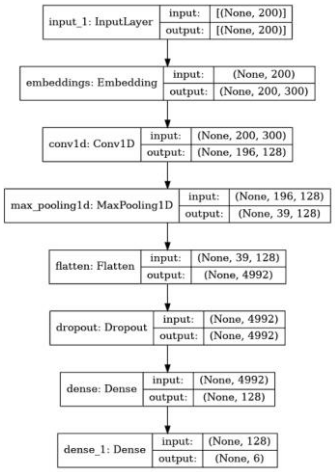
Global Max Pooling Layer: Global max pooling is a pooling operation that is performed to the whole feature map of a convolutional neural network (CNN) to produce a single output. The global max pooling layer concatenates the maximum value of each feature map, which captures the most relevant feature in each feature map, into a vector. This can be used as a feature vector for classification or other downstream operations [38].

The total number of trainable parameters achieved was 56,878,986 out of which 72,486 were trainable.

6.3.2 CNN

The second model created was the one-dimensional convolution neural network. A hierarchical display of the CNN model and information regarding layers and neurons can be found in the figures 20 & 21.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200)]             0

 embeddings (Embedding)      (None, 200, 300)          56806500

 conv1d (Conv1D)             (None, 196, 128)          192128

 max_pooling1d (MaxPooling1D  (None, 39, 128)          0
 )

 flatten (Flatten)           (None, 4992)              0

 dropout (Dropout)           (None, 4992)              0

 dense (Dense)               (None, 128)               639104

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 57,638,506
Trainable params: 832,006
Non-trainable params: 56,806,500
```



Figures 20 & 21. The summary and architecture of the CNN model.

The model consists of one input layer, and one embedded layer followed by one Conv1D layer, the model has 1 dropout layer, 2 dense layers and one 1D max pooling layer in the order shown in Figures 22 and 23.

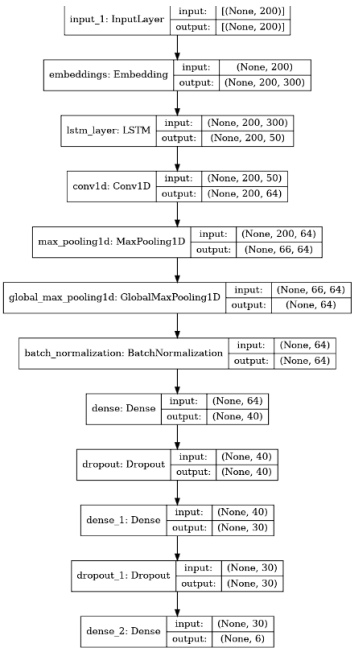
The total number of trainable parameters achieved was 57,638,506 out of which 832,006 were trainable.

6.3.3 LSTM-CNN

The third model created was the hybrid model of Long Short-Term Memory and a one-dimensional convolution neural network. A hierarchical display of the

LSTM-CNN model and information regarding layers and neurons can be found in the figures 22 & 23.

```
Model: "model_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_31 (InputLayer)       [(None, 200)]             0

 embeddings (Embedding)      (None, 200, 300)          56806500

 lstm_layer (LSTM)           (None, 200, 50)           70200

 conv1d_26 (Conv1D)          (None, 200, 64)           9664

 max_pooling1d (MaxPooling1D  (None, 66, 64)           0
 )

 global_max_pooling1d_20 (Gl  (None, 64)               0
 obalMaxPooling1D)

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_60 (Dense)            (None, 40)                2600

 dropout (Dropout)           (None, 40)                0

 dense_61 (Dense)            (None, 30)                1230

 dropout_1 (Dropout)         (None, 30)                0

 dense_62 (Dense)            (None, 6)                 186

=================================================================
Total params: 56,890,636
Trainable params: 84,008
Non-trainable params: 56,806,628
_________________________________________________________________
```



Figures 22 & 23. The summary and architecture of the LSTM-CNN model.

The model consists of one input layer, one embedded layer followed by one LSTM layer with 50 neurons and one Conv1D layer, the model has 2 dropout layers, 3 dense layers, one 1D max pooling layer, one global max pooling layer and one batch normalization layer in the order shown in figure 20 and 21.

The total number of trainable parameters achieved was 56,890,636 out of which 84,008 were trainable.

7 Results and Evaluation

This section focuses on the training and testing results of all the 3 models.

The training was simple, only 2 epochs were required, although 5 and 10 epochs were tried but there wasn't a significant difference, so it was decided to stick with 2.

An epoch is a single iteration of the whole training dataset during the training process of neural networks, the neural network is exposed to all the training data during each epoch, and the weights of the network are modified based on the mistakes made during forward and backward propagation.

7.1 Training Results

7.1.1 LSTM

The following results were achieved for the LSTM model, the training loss was decreasing from 0.0662 to 0.0530 while validation also showed the same trend, it decreased from 0.0518 to 0.0500, on the other hand, the training accuracy increased from 0.9212 to 0.9912 and the validation accuracy stayed consistent at 0.9941, the figures 24 & 25 helps visualise the model accuracy and loss for the LSTM model.

Figure 24 & 25. Loss and Accuracy values for the LSTM model.

As seen in the images above the epoch are in decimal number, the decimal number for an epoch refers to the fraction of the entire training dataset that is

used during that epoch. For example, if a training dataset has 1000 samples and the batch size is set to 100, each epoch will contain 10 iterations and the epoch's decimal number will range from 0.0 to 0.9.

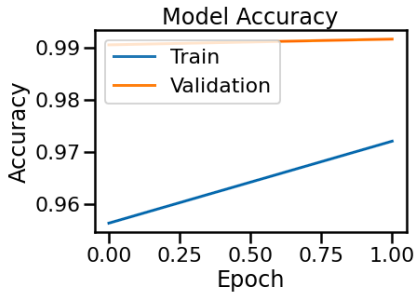
After trying various methods like tunning, the layers, filters, dropouts, or changing optimizers/loss functions to reduce the validation loss but the result was sometimes a smooth curve and sometimes flattening out, in the end, the introduction of one dimensional global max pooling layer and dropout layers showed the most consistent results.

7.1.2 CNN

The following results were achieved for the CNN model, the training loss was decreasing from 0.0707 to 0.0578 while validation also showed the same trend, it decreased from 0.0593 to 0.0529, on the other hand, the training accuracy increased from 0.9563 to 0.9720 and the validation accuracy followed the same trend and went from 0.9905 to 0.9916, the figures 26 & 27 helps visualise the model accuracy and loss for the CNN model.

```
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'], loc='upper left')
plt.show()
```

```
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'], loc='upper left')
plt.show()
```



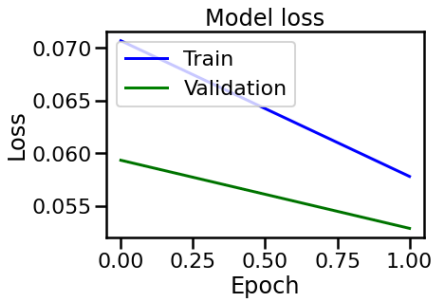


Figure 26 & 27. Loss and Accuracy values for the CNN model.

Various methods like tunning the layers, filters and introduction of max pooling layers, batch normalisation and dropout layers were added to the model and helped to achieve better results.

7.1.3 LSTM-CNN

The following results were achieved for the hybrid model, the training loss was decreasing from 0.0714 to 0.0558 while validation also showed the same trend, it decreased from 0.0543 to 0.0519, on the other hand, the training accuracy increased from 0.9296 to 0.9934 and the validation accuracy stayed consistent at 0.9941, the figures 28 & 29 helps visualise the model accuracy and loss for the hybrid model.

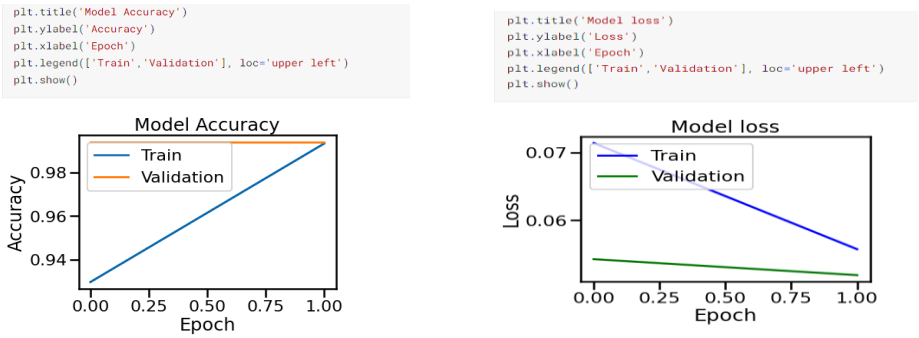


Figure 28 & 29. Loss and Accuracy values for the LSTM-CNN model.

The hybrid model had more layers compared to the rest of the models, but tuning methods and the introduction of global max-pooling layers, max-pooling layers and dropout had a good impact on the results.

7.2 Evaluation

After getting promising results from the training and validation sets it is now time to test and evaluate the model with a testing data set which is an unseen dataset reserved for testing.

The testing data set went through the same pre-processing and feature engineering stages as the training data and is made ready to be used for testing.

The “predict” function was used function to generate outputs for the inputs present in the test data.

All three models were submitted on Kaggle to find out the private and public scores of each of the models.

Table 2: The Public and private scores of 2 submissions on Kaggle.

Deep Learning Model Accuracy Scores		
Model	Kaggle Score	
	Private	Public
LSTM	0.97581	0.97324
CNN	0.95931	0.95847
LSTM - CNN	0.96861	0.96608

The Accuracy scores were high and significant for all the 3 models, as seen from the figure above, a private score of 0.97581 and a public score of 0.97324 was achieved by the LSTM model which was the highest among all the 3 models.

The hybrid model of LSTM-CNN also performed remarkably and achieved a private score of 0.96861 and the public score of 0.96608 and achieved the second spot. Out of the 3 the lowest performance was seen in the CNN model; it was able to achieve a private score of 0.95931 and a public score of 0.95847. Even though in comparison to the other 2 models the model was ranked third on its own it performed well, and a few improvements might help achieve better results.

7.3 Areas Of Improvement

The performance for all the 3 deep learning models was significant there are some areas which can be improved and thus might result in achieving better accuracy scores.

Hyperparameter tuning, all the models were tuned differently from each other, and the hypermeters selected were based on a trial-and-error method. Instead of using the trial-and-error method the use of advanced techniques such as grid search for example from the Talos auto ml library can help remarkably, this technique helps in tuning the hyperparameter meaning it runs all the different combinations of the parameters and shows you the results, this can help improve the performance of the models.

Another area which can be improved is the list of stop words, certain stop words that just belong to this dataset can be added to the list and this might help improve the performance.

8 Conclusion

In conclusion, the problem of toxic comments on online platforms is a significant challenge that affects individuals, communities, and society. The emergence of Deep Learning and Artificial Neural Networks has opened new possibilities for addressing this problem, and this thesis has explored the potential of several promising deep learning models.

The review of existing research has revealed that Convolutional Neural Networks, Recurrent Neural Networks, and Transformers are among the most effective models for detecting and classifying toxic comments. These models have demonstrated high accuracy, fast processing speeds, and robust performance in various contexts, including single-language and multi-language data.

However, the use of Deep Learning and Artificial Neural Networks for toxic comment detection also presents some challenges. For instance, the models need to be trained on diverse and representative datasets to ensure that they can generalize to new data effectively. Additionally, the models must be able to account for the complex nuances of language, including sarcasm, irony, and cultural references.

Future research should focus on addressing these challenges and improving the performance of Deep Learning and Artificial Neural Networks for toxic comment detection. This may involve exploring new architectures, optimizing hyperparameters, and incorporating other types of data, such as images and videos.

Despite these challenges, the findings of this thesis suggest that Deep Learning and Artificial Neural Networks hold great potential for tackling the problem of toxic comments on online platforms. By providing accurate and efficient solutions for detecting and classifying toxic comments, these models can help to promote healthy and respectful online interactions and contribute to building safer and more inclusive digital communities.

References

1 OpenWeb. (n.d.). Toxic Online Comments: How They Happen, and How To Stop Them. [online] Available at: https://www.openweb.com/blog/toxic-online-comments-how-they-happen-and-how-to-stop-them. [Accessed 12 Jan 2023].

2 SAS (2019). Machine Learning: What it is and why it matters. [online] Sas.com. Available at: https://www.sas.com/en_us/insights/analytics/machine-learning.html. [Accessed 2 Feb 2023].

3 Medium. (n.d.). Medium. [online] Available at: https://towardsdatascience.com/accuracy-vs-loss-8592571536b. [Accessed 5 May 2023].

4 Burns, E. (2022). What Is Machine Learning and Why Is It Important? [online] SearchEnterpriseAI. Available at: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a. [Accessed 20 Jan 2023].

5 Mahapatra, S. (2018). *Building a Deployable ML Classifier in Python*. [online] Medium. Available at: https://towardsdatascience.com/building-a-deployable-ml-classifier-in-python-46ba55e1d720. [Accessed 12 Feb 2023].

6 Sydorenko, I. (2021). What Is a Dataset in Machine Learning. [online] labelyourdata.com. Available at: https://labelyourdata.com/articles/what-is-dataset-in-machine-learning#:~:text=A%20dataset%20in%20machine%20learning%20is%2C%20quite%20simply%2C%20a%20collection. [Accessed 1 March 2023].

7 Thaddeussegura (2021). *'Train, Validation, Test Split' explained in 200 words. - Data Science*. [online] Available at: https://thaddeus-segura.com/train-test-split/. [Accessed 5 May 2023].

8 Carty, D. (2021). Training, Validation and Testing Data Explained. [online] Applause. Available at: https://www.applause.com/blog/training-data-validation-data-vs-test-data. [Accessed 25 Jan 2023].

9 Seldon (2021). Supervised vs Unsupervised Learning Explained. [online] Seldon. Available at: https://www.seldon.io/supervised-vs-unsupervised-learning-explained#:~:text=The%20main%20difference%20between%20supervised.
[Accessed 20 March 2023].

10 Javatpoint (n.d.). Regression vs Classification in Machine Learning - Javatpoint. [online] Available at: https://www.javatpoint.com/regression-vs-classification-in-machine-learning.
[Accessed 11 March 2023].

11 IBM (2020). *What is Supervised Learning?* [online] www.ibm.com. Available at: https://www.ibm.com/cloud/learn/supervised-learning#:~:text=Supervised%20learning%2C%20also%20known%20as.
[Accessed 17 Jan 2023].

12 Javatpoint (n.d.). Unsupervised Machine learning - Javatpoint. [online] Available at: https://www.javatpoint.com/unsupervised-machine-learning.
[Accessed 17 Jan 2023].

13 Mueller, J. and Luca Massaron (2016). Machine learning for dummies. Hoboken, New Jersey: John Wiley & Sons, Inc.

14 www.sciencedirect.com. (n.d.). Artificial Neural Network - an overview | ScienceDirect Topics. [online] Available at: https://www.sciencedirect.com/topics/computer-science/artificial-neural-network.
[Accessed 25 Feb 2023].

15 v7labs (n.d.). The Essential Guide to Neural Network Architectures. [online] Available at: https://www.v7labs.com/blog/neural-network-architectures-guide#h2
[Accessed 5 May 2023].

16 GeeksforGeeks. (2017). Underfitting and Overfitting in Machine Learning. [online] Available at: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/.
[Accessed 19 Feb 2023].

17 365 Data Science. (2021). *Overfitting vs. Underfitting: What Is the Difference?* [online] Available at: https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/.
[Accessed 1 April 2023].

18 Kevin khieu and Neha Narwal (n.d.). Detecting and Classifying Toxic Comments. [online] Available at: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf
[Accessed 1 April 2023].

19 kaggle.com. (n.d.). Toxic Comment Classification Challenge. [online] Available at: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52644
[Accessed 5 May 2023].

20 Darko Androcec (20) Machine learning methods for toxic comment classification: a systematic review. [online] Available https://www.researchgate.net/publication/349929587_Machine_learning_methods_for_toxic_comment_classification_a_systematic_review
[Accessed 10 April 2023].

21 Wikipedia. (2021). Project Jupyter. [online] Available at: https://en.wikipedia.org/wiki/Project_Jupyter.
[Accessed 11 April 2023].

22 Wikipedia Contributors (2019). Python (programming language). [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Python_(programming_language) .
[Accessed 29 April 2023].

23 Goyal, K. (2020). Data Preprocessing in Machine Learning: 7 Easy Steps To Follow. [online] upGrad blog. Available at: https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/.
[Accessed 16 April 2023].

24 NTKL. (n.d.). nltk.tokenize package — NLTK 3.5 documentation. [online] Available at: https://www.nltk.org/api/nltk.tokenize.html.
[Accessed 1 May 2023].

25 Brownlee, J. (2017). Why One-Hot Encode Data in Machine Learning? [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.
[Accessed 10 Feb 2023].

26 Princeton University (2019). WordNet | A Lexical Database for English. [online] Princeton.edu. Available at: https://wordnet.princeton.edu/.
[Accessed 4 May 2023].

27 Wikipedia. (2022). Stop word. [online] Available at: https://en.wikipedia.org/wiki/Stop_word.
[Accessed 10 May 2023].

28 Jason Brownlee (2017) Data Preparation for Variable Length Input Sequences. [online] Available at: https://machinelearningmastery.com/data-preparation-variable-length-input-sequences-sequence-prediction/ [Accessed 4 Jan 2023].

29 Rao, P. (2019). Transfer Learning in NLP for Tweet Stance Classification. [online] Medium. Available at: https://towardsdatascience.com/transfer-learning-in-nlp-for-tweet-stance-classification-8ab014da8dde. [Accessed 12 May 2023].

30 Fasttext.cc. (2019). fastText. [online] Available at: https://fasttext.cc/. [Accessed 12 May 2023].

31 Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. [online] arXiv.org. Available at: https://arxiv.org/abs/1408.5882. [Accessed 4 March 2023].

32 Randolph (2023). *Deep Learning for Multi-Label Text Classification*. [online] GitHub. Available at: https://github.com/RandolphVI/Multi-Label-Text-Classification [Accessed 5 May 2023].

33 Tang, D., Qin, B. and Liu, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. [online] Available at: https://doi.org/10.18653/v1/D15-116. [Accessed 14 May 2023].

34 Team, T.A. and Team, T.A. (n.d.). *Text Classification with RNN – Towards AI — The Best of Tech, Science, and Engineering*. [online] Available at: https://towardsai.net/p/deep-learning/text-classification-with-rnn. [Accessed 4 March 2023].

35 Nyuytiymbiy, K. (2021). Parameters and Hyperparameters in Machine Learning and Deep Learning. [online] Medium. Available at: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac. [Accessed 13 Feb 2023].

36 TensorFlow. (n.d.). tf.keras.layers.Dropout | TensorFlow Core v2.1.0. [online] Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout. [Accessed 4 May 2023].

37 TensorFlow. (n.d.). tf.keras.layers.Dense | TensorFlow Core v2.3.0. [online] Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense [Accessed 2 May 2023].

38 Team, K. (n.d.). Keras documentation: GlobalMaxPooling2D layer. [online] keras.io. Available at:

https://keras.io/api/layers/pooling_layers/global_max_pooling2d/
[Accessed 5 May 2023].

Additional Code and Information

Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import keras
import re
import string
import nltk

import seaborn as sns
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.stem.snowball import SnowballStemmer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.metrics import roc_auc_score , accuracy_score ,
confusion_matrix , f1_score
```

The code imports several Python libraries at the beginning of the script using the import statement.

Loading Data

```
training_data  =  pd.read_csv(r'C:\Users\Dhruval\Downloads\jigsaw-
toxic-comment-classification-challenge\train.csv\train.csv')

testing_data  =  pd.read_csv(r'C:\Users\Dhruval\Downloads\jigsaw-
toxic-comment-classification-challenge\test.csv\test.csv')

test_target =  pd.read_csv(r'C:\Users\Dhruval\Downloads\jigsaw-toxic-
comment-classification-challenge\test_labels.csv\test_labels.csv')
```

The code reads the training data and testing data from CSV files using the pd.read_csv() function.

Data Exploration

```
comments = training_data.drop(['id','comment_text'],axis = 1)
for i in comments.columns :
```

```
    print("Percent of {0}s: ".format(i),
round(100*comments[i].mean(),2), "%")

classes = {}
for i in list(comments.columns):
    classes[i] =  comments[i].sum()
n_classes = [classes[i] for i in list(classes.keys())]
classes = list(classes.keys())
d = {'Toxicity': classes, 'Counts': n_classes}
df = pd.DataFrame(data=d)

ax = sns.barplot(y= "Counts", x = "Toxicity", data = df,
palette=("Blues_d"))
sns.set_context("poster")
```

The code uses the describe () function to get descriptive statistics of the training data.

The code calculates the percentage of each type of toxic comment using the mean of each column and prints it to the console.

The code creates a bar plot showing the distribution of toxic comments using the sns.barplot() function.

Model Code

CNN

```
from keras.layers import Input
from keras import Sequential, Model
from keras.layers import Embedding,LSTM,
GlobalMaxPool1D,Dropout,Activation, Dense
from keras.models import Sequential
from keras.layers import Dropout, Flatten, BatchNormalization, Conv1D
from keras import layers
from tensorflow.keras.losses import CategoricalCrossentropy
from keras.layers import MaxPooling1D
from keras.layers import Input, Embedding, Conv1D, MaxPooling1D,
Flatten, Dense, Dropout
from keras.models import Model

# Define the CNN Model.

inp = Input(shape=(maxpadlen,), dtype='int32')
embedding_layer = Embedding(len(word_index) + 1,
                            embedding_dim_fasttext,
                            weights=[embedding_matrix_fasttext],
                            input_length=maxpadlen,
                            trainable=False,
                            name='embeddings')(inp)
```

```
conv_layer = Conv1D(filters=128, kernel_size=5,
activation='relu')(embedding_layer)
max_pool_layer = MaxPooling1D(pool_size=5)(conv_layer)
flatten_layer = Flatten()(max_pool_layer)
dropout_layer = Dropout(0.2)(flatten_layer)
dense_layer = Dense(128, activation='relu',
kernel_initializer='he_uniform')(dropout_layer)
output_layer = Dense(6, activation='sigmoid',
kernel_initializer='glorot_uniform')(dense_layer)

model_2 = Model(inputs=inp, outputs=output_layer)
model_2.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

#Fit the Model on Training Data
model_info_2 = model_2.fit(x_train, y_train, epochs=2, batch_size=32,
validation_data=(x_val, y_val))
```

LSTM-CNN

```
from keras.layers import Input
from keras import Sequential, Model
from keras.layers import Embedding,LSTM,
GlobalMaxPool1D,Dropout,Activation, Dense
from keras.models import Sequential
from keras.layers import Dropout, Flatten, BatchNormalization, Conv1D
from keras import layers
from tensorflow.keras.losses import CategoricalCrossentropy
from keras.layers import MaxPooling1D

# Define the CNN Model.

inp=Input(shape=(maxpadlen, ),dtype='int32')
embedding_layer = Embedding(len(word_index) + 1,
                            embedding_dim_fasttext,
                            weights = [embedding_matrix_fasttext],
                            input_length = maxpadlen,
                            trainable=False,
                            name = 'embeddings')
embedded_sequences = embedding_layer(inp)
x = LSTM(50,
return_sequences=True,name='lstm_layer')(embedded_sequences)
x = Conv1D(filters=64, kernel_size=3, padding='same',
activation='relu', kernel_initializer='he_uniform')(x)
x = MaxPooling1D(3)(x)
x = GlobalMaxPool1D()(x)
x = BatchNormalization()(x)
x = Dense(40, activation="relu", kernel_initializer='he_uniform')(x)
x = Dropout(0.2)(x)
x = Dense(30, activation="relu", kernel_initializer='he_uniform')(x)
x = Dropout(0.2)(x)
preds = Dense(6, activation="sigmoid",
kernel_initializer='glorot_uniform')(x)

#Compile the Model.
model_2 = Model(inputs=inp, outputs=preds)
model_2.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])
```

```
#Fit the Model on Training Data.
model_info_2=model_2.fit(x_train,y_train, epochs=2, batch_size=32,
validation_data=(x_val, y_val))
```

LSTM

```
from keras.layers import Input
from keras import Sequential, Model
from keras.layers import Embedding,LSTM,
GlobalMaxPool1D,Dropout,Activation, Dense

# Define the LSTM Model.
inp=Input(shape=(maxpadlen, ),dtype='int32')
embedding_layer = Embedding(len(word_index) + 1,
                            embedding_dim_fasttext,
                            weights = [embedding_matrix_fasttext],
                            input_length = maxpadlen,
                            trainable=False,
                            name = 'embeddings')
embedded_sequences = embedding_layer(inp)
x = LSTM(50,
return_sequences=True,name='lstm_layer')(embedded_sequences)
x = GlobalMaxPool1D()(x)
x = Dropout(0.2)(x)
x = Dense(40, activation="relu", kernel_initializer='he_uniform')(x)
x = Dropout(0.2)(x)
preds = Dense(6, activation="sigmoid",
kernel_initializer='glorot_uniform')(x)

#Compile the Model.
model_1 = Model(inputs=inp, outputs=preds)
model_1.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

#Fit the Model on Training Data.
model_info_1=model_1.fit(x_train,y_train, epochs=2, batch_size=32,
validation_data=(x_val, y_val))
```