A picture containing text

Description automatically generated

Using machine learning to detect hate speech and offensive language through instant messaging

Final Report

DT282

BSc in Computer Science International

**Daniel Krasovski**

**C18357323**

**Svetlana Hensman**

School of Computer Science

Technological University, Dublin

**08/04/2022**

Abstract

Since the invention of the internet, one of the most used technologies is instant messaging. While this in itself is not bad in anyway, it brings along a new way to spread offensive and hateful messages and statements. Especially in a medium such as Discord where there are many young users and no built in moderation or detection of such messages. Users can be exposed to very hateful and offensive messages at an age where they should not. This is the problem I am attempting to solve in my this final year project. I am going to be using Machine Learning and NLP to detect Hate speech and Offensive messages real time and use the built in tools in discord to deal with the messages through various methods.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Daniel Krasovski

04/04/2022

Acknowledgements

I would like to Acknowledge my project supervisor Svetlana Hensman for guiding me and making sure I am always on the right track. I would also like to thank my great friends that I made in this course Ire Adabari, Raphael Ofeimu, Kyle Butler and Nikolay Malyshev for all the help, advice, and encouragement I received during the previous 4 years.

Table of Contents

[1. Introduction 7](#_Toc100074824)

[1.1. Project Background 7](#_Toc100074825)

[1.2. Project Description 7](#_Toc100074826)

[1.3. Project Aims and Objectives 8](#_Toc100074827)

[1.4. Project Scope 9](#_Toc100074828)

[1.5. Thesis Roadmap 9](#_Toc100074829)

[*Design* 9](#_Toc100074830)

[*Development* 10](#_Toc100074831)

[*Testing and Evaluation* 10](#_Toc100074832)

[*Redevelopment* 10](#_Toc100074833)

[*Conclusions and Future Work* 10](#_Toc100074834)

[2. Literature Review 11](#_Toc100074835)

[2.1. Introduction 11](#_Toc100074836)

[2.2. Alternative Existing Solutions to Your Problem 11](#_Toc100074837)

[Hate speech by PyAntony 11](#_Toc100074838)

[Hate Speech Detection and the Problem of Offensive Language 11](#_Toc100074839)

[2.3. Technologies you’ve researched 11](#_Toc100074840)

[Programming languages 12](#_Toc100074841)

[Machine learning 12](#_Toc100074842)

[Machine Learning Frameworks 13](#_Toc100074843)

[Discord API wrapper 14](#_Toc100074844)

[OpenCV and pytesseract 14](#_Toc100074845)

[BERT Language model 14](#_Toc100074846)

[2.4. Other Research you’ve done 14](#_Toc100074847)

[2.5. Existing Final Year Projects 15](#_Toc100074848)

[2.6. Conclusions 15](#_Toc100074849)

[3. Project Design 16](#_Toc100074850)

[3.1 Introduction 16](#_Toc100074851)

[3.2. Software Methodology 16](#_Toc100074852)

[Agile 16](#_Toc100074853)

[Rapid Application Development 17](#_Toc100074854)

[3.3. Overview of System 18](#_Toc100074855)

[3. Prototype Development 19](#_Toc100074856)

[Discord 19](#_Toc100074857)

[Backend 20](#_Toc100074858)

[Introduction 21](#_Toc100074859)

[Prototype One 21](#_Toc100074860)

[Prototype Two 25](#_Toc100074861)

[Prototype Three 26](#_Toc100074862)

[4. Project Development 28](#_Toc100074863)

[4.1. Introduction 28](#_Toc100074864)

[4.2. Development of model 28](#_Toc100074865)

[5. Testing and Evaluation 28](#_Toc100074866)

[5.1. Introduction 28](#_Toc100074867)

[5.2. Plan for Testing 28](#_Toc100074868)

[User feedback 29](#_Toc100074869)

[5.3. Plan for Evaluation 29](#_Toc100074870)

[Test plan 29](#_Toc100074871)

[6. Issues and Future Work 31](#_Toc100074872)

[6.1 Introduction 31](#_Toc100074873)

[6.2 Issues and Risks 31](#_Toc100074874)

[6.3 Plans and Future Work 31](#_Toc100074875)

[6.3.1 GANTT Chart 32](#_Toc100074876)

[Bibliography 33](#_Toc100074877)

# 1. Introduction

## Project Background

Multiple research has shown that social media posts and other online messages can cause an increase in acts of violence. For example in Germany it was shown that hate speech on social media and messaging apps can lead to an uptick in hate crimes.(1)

With the amount of content being posted online being far too greater for any human to preview beforehand. This is also not taking into account the bias’s they could have. So to solve this problem I am going to attempt to solve this problem by using machine learning to train a model that can differentiate between hate speech, offensive language and just normal messages.

There are also other issues apart from real-world violence with being exposed to hateful messages. Mental Health is also very affected by hateful messages. A study found that exposure to hateful messages leads to greater stress among college students.(2)

The goal of my project is to try to detect and deal with these kind of messages.

Hate speech and offensive language can be very subjective. And there is no one label that can be put upon hate speech or offensive language. So what one person could find offensive, another could not. In this project I will be attempting to deal with the most extreme of both of these.

## Project Description

My project is going to be a machine learning model that when trained using appropriate data will be able to detect hate speech and offensive messages. To test and showcase the possibilities of what to do once the messages have been detected, I will be using the Discord API to make a discord bot. I will be using the discord.py wrapper which is written in python.(3)

The complexity of this project comes from the machine learning and real time detection. Every time a message is sent in a chat it will be scanned and then dealt with appropriately. Making sure the model is trained adequately and the differentiation between hate speech and offensive language will also be very import to get done correctly. The main data set which I will be using comes from the paper Automated Hate Speech Detection and the Problem of Offensive Language.(4) The data is a collection of tweets containing lexicon from hatebase.org. I will also attempt to implement a feature where the dataset could continuously get updated and retrained to improve accuracy and include more up to date lexicons.

Through my previous experience with machine learning algorithms and NLP, The difficult part of this project is going to be performance. Making sure the model can be trained in adequate time and that the algorithm does not need to run for each time a prediction is required. Furthermore connecting the backend machine learning and the front end Discord API will also be a challenge and will have the possibility of running into issues that I could not foresee.

The approach for the project that I will take will be one of Feature-Driven Development(FDD). FDD is related to Scrum but instead of being delivery focused, it is feature focused.(5) I will work on the main feature of message detection and classification and once it is finished move onto new features and only move one once the feature is finished or is completely abandon features that are not feasible to implement.

The screenshots below show a possibility of a response once a valid detection is found

Text

Description automatically generated

Figure(1)

## Project Aims and Objectives

The overall aim of this project is to provide a Discord moderation bot that can be put onto servers that have hundreds of messages sent per minute, where a group of moderators would not be able to handle spotting, analysing and dealing with such messages. The features that I hope to have implement by the end of the project are:

Hate speech detection – scan every message sent and use a prediction model to get a probability if the message sent contains hate speech

Offensive language detection – same as hate speech except it will be for offensive language

And if I get these features finished I hope to work on the following afterwards:

Data collection – let users report messages that do not get flagged and have moderators look over it and decide if it fits into any category

Adaptability – work on the adaptability and attempt to make it work on other messaging apps

Increase languages supported, currently it will only work in English. Find a feasible way to add European languages such as German and French.

## Project Scope

This project will not end up solving the whole problem of offensive and inappropriate content being posted online. I will just be focusing on the instant messaging side of the problem. I will also be looking more at the technical aspect and how difficult each aspect would be to implement on a large scale. I will also not be making a standalone texting app to showcase this, I will instead be making a framework that could be easily adapted and implemented into other messaging apps.

## Thesis Roadmap

Research

For research, this section will be about me researching similar projects to mine and what they managed to achieve. I will also be doing some research into the advantages and disadvantages of having a system like this. I will also be looking into the ethical and moral issues that could come up with such a project and find the best way to overcome these issues. It will also contain any other relevant research I will have completed for this project.

### Design

For the design of the project, It will be about the choices I made for how it will be development and the reasoning for it. It will also contain the different datasets and methods I could have used to help with my development. Finding the correct machine learning platform and discussing the reasoning for which one I went will also be in this section.

### Development

This chapter will break down the entire development process, it will go into depth about any decisions made, any code I used and how they were changed and adapted. The challenges I encountered and how I overcame them will also be discussed in this chapter

### Testing and Evaluation

This chapter will go into detail about how I tested and evaluated the project. From testing the model, to seeing how it all works together. I will also go into about how easy it would be to adapt, change and implement new features

### Redevelopment

This chapter will discuss anything that was changed in the later stage of the project. It will also go into the reasons behind it and what effect it had on the project over all

### Conclusions and Future Work

The final chapter will be an overview of the whole project, what was able to be completed and what other features could be added in the future

# 2. Literature Review

## 2.1. Introduction

In this chapter I will be discussing the key areas of research I have done. I will go into detail about existing solutions, other similar projects that have been done and different technologies I have looked at and comparing the differences between each option I could have went with. I also done some research into Discord and the different API wrappers I could have used.

## 2.2. Alternative Existing Solutions to Your Problem

There were multiple research papers with code along it. They all used different datasets however the 2 that I found were most applicable to me were: hate-speech by PyAntony.(6) and Automated Hate Speech Detection and the Problem of Offensive Language by Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber.(4)

### Hate speech by PyAntony

The dataset is from a white supremacist forum. There are 2392 observations with 1914 for training and 478 for testing. It uses the BERT modelling approach and has an API for prediction using post request. The project compares two approaches of the BERT model, Fine-tuning and embedding. With fine-tunning appearing to be more effective with 82% accuracy and evaluation loss of 0.43

This project uses the BERT model for its natural language processing(NLP). It was created in google in 2018 and has been used in its search engine for nearly almost every engine language query. It is highly advanced and looks like to be the best option for classifying text. I will attempt to implement the BERT model for my finished model.

### Hate Speech Detection and the Problem of Offensive Language

This paper uses a more simplistic approach to the model. It uses the sklearn library and logistic regression for its model. The interesting apart which would apply to my project would be the dataset. It is a sample of 25,000 tweets containing lexicon from hatebase.org. It was then manually coded by CrowdFlower workers whether the tweet contained hate speech, offensive language or neither. This dataset will be the one I will work with from the start and develop a protype with.

## 2.3. Technologies you’ve researched

### Programming languages

From an early stage I knew that for this project I will be coding most of it, if not all, in python. Python is a high level open source language. It has a high amount of libraries that allow for machine learning, data analysis and data visualisation. The language is also syntactically simple and there are extensive online resources for python and with over 60% of machine learning developers using python. It seemed like the perfect option to go with. (7)

The other option was R. R is a functional language that is primarily used for data analytics and for the visual of data. There are various resources and tutorials for it, However it does lack features compared to python and my lack of familiarity to it is the reason I did not go with it.

### Machine learning

Machine Learning (ML) is a sub discipline of Artificial Intelligence and aims to understand and develop self-improving algorithms. There are three basic machine learning paradigms. These are supervised learning, unsupervised learning and reinforcement learning.

#### Supervised Learning

Supervised learning is a way of developing and training a model using a set of pre-existing training instances. A issue is presented to the machine learning agents, and they must make a choice. This problem's right solution is already known. In the event of a loss, the model is updated after the decision has been determined. This is due to an error, which is the difference between the algorithm's option and the actual right solution. The goal of supervised learning is to generalize the training data such that the algorithm can finally make the correct decision for new issues with no prior instances. (8)

#### Unsupervised Learning

In a similar fashion to supervised learning, Unsupervised learning utilises data sets to train machine learning algorithms. The difference lies in the goal, which is to enable the learning agent to recognise patterns in a data set, by determining how features are distributed, which it can further utilise to create clusters of similar data, extract the most important and crucial features of the data and generate new data based on the original. This training process can be applied to a neural network called auto encoder. It consists of two parts and attempts to map the output of a pair to the input by learning the identity function.(9)

#### Deep Learning

Deep learning, a branch of machine learning based on artificial neural networks, is another sub-discipline of machine learning. Deep learning makes use of neural networks to try to reproduce the way our brains work. Data is fed into artificial neural networks, which deep learning uses to train itself. The data is constantly analysed at each of the levels of the neural network, with each level focusing on different parts of the data and what might be used to differentiate it. The deeper the neural network goes, the more objects it will search for, such as eyes and noses, before attempting to determine if it comprises a face. (10)

#### Natural Language Processing

To use machine learning with text it was essential to learn about Natural Language Processing(NLP). NLP is defined as the automatic manipulation of natural language using text. It has been around for over 50 years and grew with the rise of computers.(11) There are many uses of NLP anything from speech recognition, to natural language generation. The parts which I am going to attempt to implement in my project are:

* Text Classification – Text classification is a machine learning technique that categorizing text into organized groups. It can be used to understand if any text is negative or positive, what topic it is about and detect what language is being used.(12)
* Sentiment analysis – this is the process of computationally identifying and categorizing opinions from a text. It is used to determine whether the attitude used in the sentence Is positive, negative or neutral.
* Text summarisation – This is the process in which large texts are summarised for quicker consumption(13)

### Machine Learning Frameworks

There are lots of machine learning frameworks available for python from TensorFlow to Scikit-Learn.

#### TensorFlow

TensorFlow is an opensource end to end machine learning platform. It has a wide range of tools, libraries and community resources that enable users to develop machine learning algorithms easily and efficiently.(14) TensorFlow can create models for anything from image recognition to NLP. It works by allowing the developers to create structures that describe how data moves through a graph. Each node in the graph represents a mathematical operation with each edge between the nodes being a multidimensional array(Tensor). The main supported language is Python, however there are community built API’s for other languages. The mathematical operations however are all computed through high-performance C++ binaries.

#### Scikit-Learn

Scikit learn is an opensource machine learning and data analysis library in python. It is considered by many to be the gold standard for machine learning in python. Scikit learn provides many algorithms for supervised and unsupervised learning and is built upon technology such as NumPY, pandas and matplotlib. Scikit learn is very easy to setup and learn to use. It is high level which allows you to define a model for prediction in a short amount of time and code.

### Discord API wrapper

The Discord API wrapper that I will be using for this project is Discord.py. I went with discord.py because of my previous experience using it and it’s easy to use, feature rich and async ready. One concerning issue at the start of the project was that the mainline development was ceased. However by the time of writing this, development has restarted and is very active.

### OpenCV and pytesseract

OpenCV is an open source library that is used for real time computer vision. In this project I use it to process images for text extraction. Once the image is processed, I then use pytesseract to find all the text in the image and convert it to a string. This will then later be run against the model to check for hate speech and offensive language.

### BERT Language model

Bidirectional Encoder Representations from Transformers (BERT) is a transformer based machine learning method for NLP developed by google in 2018. By late 2021 google has been using BERT in most of its English language queries. BERT will be used in this project to create the multi label text classification to detect both hate speech and offensive language

## 2.4. Other Research you’ve done

Other research that I have done is looking into the side effects and consequences of hate speech and offensive content on the internet. Especially on young people, there can be long lasting negative effects.(15) Further more I have also looked into the issues of dealing with hate speech and offensive language and how to attempt at creating an unbiased model

## 2.5. Existing Final Year Projects

From my brief look at the previous final year projects, I was able to find one project that was similar to mine. It is titled Anti-Bullying with Machine Learning and it is by Shane O’Neill. The project uses various combinations of machine learning and text mining to build predictive models to detect cyber bullying. Mr O’Neill tested his models on 4 different datasets labelled for bullying or abusive content. He found that the data quality is very important and no one model can be used for all the datasets. He also created a demonstration interface where a user can enter any sentence and it would predict if the sentence entered contains bullying. This final year project contains very useful information and is very insightful.(16)

## 2.6. Conclusions

From all this research, I feel very confident in the development of this project. The extensive technologies available and the vast amount of similar research papers into this topic gives me a very good platform to improve on all the previous work. Connecting to the discord API will also be very approachable due to the simple nature of the wrapper and with my previous experience using it.

Requirements Table

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Priority** |
| Hate speech | Accurately detect hate speech | HIGH |
| Offensive Language | Accurately detect offensive language | HIGH |
| Discord API | Connecting the model to the discord API | HIGH |
| Reporting system | Allow users to report messages that are not detected | MEDIUM |
| Spam detection | Detect when a user is spamming | MEDIUM |
| Dataset improvement | Improve the dataset for more accurate results | MEDIUM |
| Level of moderation | Make it possible to choose how “strict” the detections will be. | MEDIUM |
| Other languages | Look into supporting other languages | LOW |
| Text summarisation | Summarise messages that have been sent | LOW |
| Toxicity metric | Have a metric for each user, to see how offensive/ hateful they are | LOW |
| Fake news | Detect when fake news is being sent. | LOW |

# 3. Project Design

## 3.1 Introduction

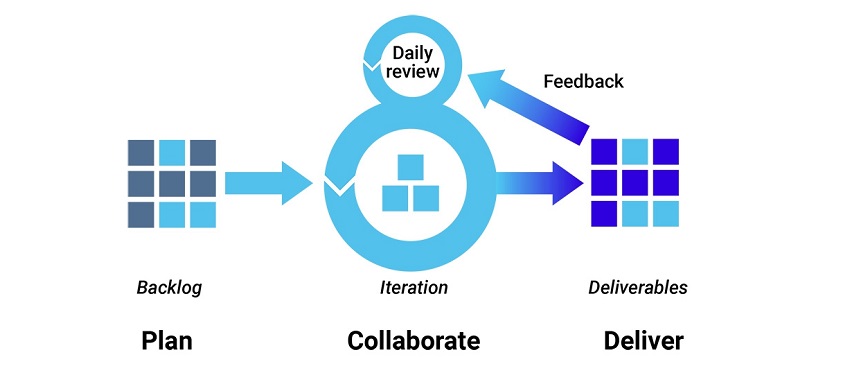
In this chapter, I will be discussing how I will design and develop the prototype for this project. I will start with the methodology where I will discuss which method I chose and why. I will also compare other methods I could have went with. Afterwards I will discuss the overview of the system. This will be about how the front end and the backend will interact and how they will work. Lastly I will end with how I want to the prototype to look and function. I will also talk about the Development of the prototype in this section also as I done both of the steps at the same time. This is will also reduce the amount of information that will be repeated by having the development in its own section.

The GitHub link for the project is:<https://github.com/Meapy/FYP---Realtime-hatespeech-detection-on-discord-messages>

## 3.2. Software Methodology

### Agile

The Agile development method Is used to reduce risk such as bugs, cost and changing requirements when adding new functionality. There are multiple forms of agile development including scrum, extreme programming(XP) and feature driven development. The main benefit of Agile development is that it allows for releases in iterations. This allows the developers find bugs and fix them earlier on, and let the users to use the software earlier. One big downside however, is that when working in big teams, it requires good communication and high time commitment to make it work efficiently.

Figure 2(17)

Like I mentioned earlier in the paper, I will be using Feature driven development(FDD). FDD is iterative and incremental with the goal being delivering results as frequently as possible. It also encourages the use of proper documentation at all levels to improve progress and track the results. FDD works using five basic steps,

1. Build overall model
2. Create feature list
3. Plan each feature
4. Design the feature
5. Build the feature

The first two steps are done at the start while the final 3 steps are repeated for all the features in the project. Most of the time will be spent on designing and building the feature in FDD.(5)

### Rapid Application Development

Rapid application development(RAD) is a shortened development approach that has the goal of delivering a high quality system with a low cost of investment. It facilitates the ever changing demands and requirements from the users and allows the developers to quickly adjust.

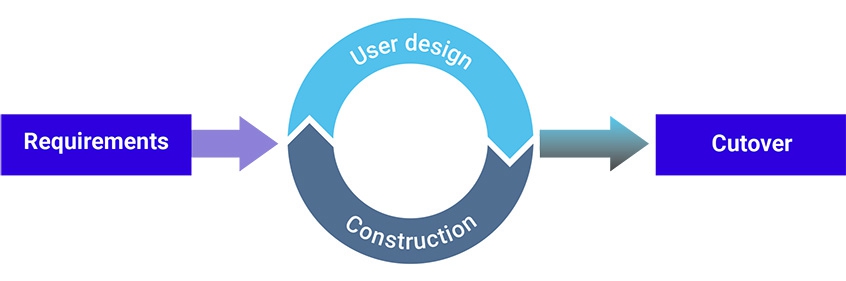


Figure 3(16)

The 4 stages of RAD are requirements, user design, construction and cutover. The middle two phases are repeated until the product meets all the requirements. RAD is most effective for small projects that are time sensitive. (17)

## 3.3. Overview of System

With the FDD approach being used for this project. Each feature will be planned, implemented and tested one at a time. Once one feature is finished, I will move onto the next one. I decided to go with this approach because I would like to implement as many features as possible without having some features not fully completed or implement.

The design and code will be developed in stages. Where each feature will be researched and designed before the implementation will being. This will make sure that the development of each feature will be done with most efficient method and will reduce the risk for starting a feature that will not be feasible to implement. The approach will consist of :

1. Design and implementation of the twitter dataset prediction with the discord API and have the prediction working. accuracy will not matter too much as this will be the basis for being able to connect the front end with the backend.
2. Design other feature (such as proper prediction)
3. Implement the feature
4. Test the feature
5. Repeat steps 2-4 until no other features could be implemented. (due to time or all features have been developed)

The next diagram will show the two layers of the program and how they will communication with each other.

Diagram

Description automatically generated

Figure 4

# 3. Prototype Development

## Discord

The front end messaging channels will be the discord group chats. To add the bot to a server, first you have to have the correct permission and afterwards you just have to click on a link and then select the server you want to add it to. This is how this step looks:

Graphical user interface, application, Teams

Description automatically generated

Figure 5

Once the bot is in the server, it will be able to read the messages sent in every channel in the server. You are able to change the permissions of which channels the bot is able to see by changing it’s permissions. As seen in this image:

Graphical user interface, text, application

Description automatically generated

Figure 6

Once the permissions are set up and you gave the bot to view a channel. You will be able to see the bot come up on the members list on the right hand side. The bot will now be able to view the messages and interact with them. This is how the users will be able to interact with the bot.

## Backend

### Introduction

Throughout the development of this project, there were 3 prototype machine learning models for the hate speech and offensive language detection built. With the first two prototypes having major issues that prevented me from using the model. The reasons why will be explored in this section along with the benefits and drawbacks of each model.

### Prototype One

With the first prototype, I created a simple script that connect the bot to the servers it was added to. Afterwards I created a command that would allow a user to pass a message to the bot for prediction and the bot would respond with its prediction. Here is a code snippet of the discord bot using the discord.py API wrapper.

For the Classification I used the Automated hate speech detection and problem of offensive language project as the basis for which I made the classifier. However it would not work without making changes. The changes I had to make to make the classifier were:

1. Update the print statements
2. Update Unicode to str
3. Save the tweet tags to a file (this was done to speed up the training process as obtaining the tweet tags each time took a considerable amount of time)
4. Update the pipeline
5. Create the prediction function

Here is a code snippet of the prediction function:



With the Both of them connected, here is how it works in a discord channel:

Graphical user interface, text, application

Description automatically generated

Figure 7

#### Prototype issues

I came across many issues while developing my prototype. I will discuss the issues in 3 parts:

1. Existing solution
2. Adapted solution
3. Prototype solution

##### Existing Solution

The first issue I came across was when I cloned the hate-speech-and-offensive-language repository was to get it to work. Both the Jupiter notebook file for python 3.6 and the final classifier would not run because the code was out of date. For the classifier the model files were also made with an older version of Scikit learn. The code was not maintained from 2019 so there was multiple things I had to change to get it to run. So I began to edit the code to make the adapted solution

##### Adapted Solution

For the adapted solution, there were multiple changes I made, the simple fixes were the print statements and Unicode issues and the more complex issues where the performance issue, model training and then prediction. For the performance, the code would get the tweet tags every time the model was retrained so I solved this issue by saving the tweet tags to a txt file and then checking if the file existed every time the code ran. Code snippet on next page



Afterwards I had to fix an issue with the pipeline by updating the solver to libliner because the penalty was l1, another solution was changing the penalty to l2. Once these issues were solved, I was able to run the jupyter notebook file. The Accuracy however was significantly worse as seen by comparing both of the confusion matrixes:

Chart, waterfall chart

Description automatically generatedChart, waterfall chart

Description automatically generated

Figure 8 (original confusion matrix) (18) Figure 9 (my confusion matrix)

##### Prototype Solution

For the standalone classifier to the protype, I worked off the final classifier in the repository and off the adapted solution. I could not get any of the pickle files(for the vectorizer) or the model file to work. So they are created every time the code is ran. From my testing this does not have a noticeable effect on the speed of the model. The prediction class I also had to completely remake myself. In the end the classifier does work however the predictions are not consistent and there are a lot of false positives and false negatives, so for the final solution I will look into creating my own model and classifier.

#### Prototype Conclusion

In the end this prototype was completely random with the predictions and the only positives that came out of the project were the connection to the discord API and a general understanding of the machine learning method used, the dataset and how to work with it and knowing how in the end, It will all work and look. In the end, a lot was learned and no time was wasted.

### Prototype Two

The 2nd prototype was created to work exactly the same with the discord API code so that was not changed. This time I created a model myself using TensorFlow. It was an unsupervised neural network that was trained for pattern matching. The way It worked was having 3 tags, for hate speech, offensive language and neither. Each of the sentences were taken and tokenized and assigned to the classes. Code below



Once the sentences were tokenized and assigned to the classes. we created the model. the model was a neural network with 3 layers. The code below:



The model was trained with a batch size of 128 and took around 10 minutes to train per epoch. with testing it, the accuracy and training loss levelled off after 7- epochs. The results were not good with an accuracy of 73% and a training loss of 0.67.

#### Prototype issues

With further testing the model performed poorly, with most of the predictions, prediction messages to be offensive. This is because of a few reason.

1. The dataset was large and lots of words were shared between all 3 of the classes
2. The split of the data with 77% being offensive,16% being neither and 5% being hate speech, just by the large number of offensive data, it always thought it was offensive content.

This model could be implemented to detect offensive content alone by filtering for offensive words. However a the dataset would have to be completely changed. These unsatisfactory results made me abandon this model as well and move onto a more complex solution. Using the BERT Model.

### Prototype Three

The 3rd Machine learning prototype built was the one used in the final solution. This was created using TensorFlow and Tensorflow\_Text which facilitate the use of the BERT model. Implementing this model took a few more steps as it involved finding, downloading and then testing the dataset on the pretrained models. This was done through the TensorFlow hub website.(19) Here we can find pretrained encoders and pre-processors to use for the model. To create the prototype I used the bert-base-encased for the pretrained model and tokenizer and for the pre-processor and encoder used the smallest bert encoder model and its corresponding pre-processing model. Code snippets below:





Once BERT was setup, the rest of the model was created using a similar process as with the other 2 prototypes. However compared to the other models, this took longer to train and took approximately 35 minutes per epoch with around 10 -15 epochs before the accuracy and loss evened out at 87% and 0.35 respectively. This was very promising.

Chart, line chart

Description automatically generated

Figure 11

Testing the model came next and putting in sentences that the other prototypes classified incorrectly, this model got them correct. Testing with various types of hate speech, offensive sentences and just regular sentences. The model correctly classified the sentences at a satisfactory rate where It was comfortable to continue to use and work on the model to make it better.

# 4. Project Development

## 4.1. Introduction

The development of the project will be discussed in this section. This will include:

* Discord API
* Further development and fine tuning of the model
* Development of the moderation tools
* Image to text extraction and classification
* Data collection and reporting false flags
* Text processing and data cleaning

## 4.3. Development of Discord API

I started off development by creating the Discord bot. This was done by creating a developer account on the discord website.(20) Assigning the permissions for the bot. For this bot just giving it general administrator rights is sufficient. To add the bot to a server, you then go into the URL generator and give it the bot scope, administrator permissions and then click on the url and select the server to add it to. You are then brought to the discord application and this pop up appears to add to a server

Graphical user interface, application, Teams

Description automatically generated

Once this completed. the work on coding the bot began. The first thing that was coded to make the bot come online. This was done using the following code:



Each discord bot has a unique token that is used as a login key and it is very important to keep it private as it could give other people access to your bot if it becomes available. To secure the token, the dotenv library is used and the token is put inside a file named .dotenv and the token is then passed to the code when the bot is ran. Once the code is ran, you will see the bot come online and in the console it prints out which servers it is connect to as seen below

Text

Description automatically generated

## 4.3. Development of model

After the development and subsequent testing of the prototype model. I began work on fully developing the model.

# 5. Testing and Evaluation

## 5.1. Introduction

In this section, I am going to discuss the method for testing and evaluating this project. From using multiple models to test accuracy and getting feedback from users, I will be attempting to make the model more accurate as the development continues.

## 5.2. Plan for Testing

The project will be tested all the time thought the entire development cycle. From the speed of the model training, to the accuracy to make sure that it is working correctly.

To make sure that the project is never completely lost, I will be using GitHub for backup and version control. This will make sure that if there are any major issues that end up braking the project, the changes could be rolled back. It will also prevent any issue occurring with any accidental deletion of any files.

The project will also be continuously tested and evaluated by user feedback. I plan on asking multiple large discord servers to use the bot and report with any feedback and issues that they see with the bot. Doing this will help me find issues that I alone will not be able to find.

Furthermore I will also be comparing if the solution is more effective than having human perform the exact same task.

### User feedback

I will also be using user feedback to test the model. This will be for finetuning and making sure that the bot will not be under sensitive or over sensitive.

## 5.3. Plan for Evaluation

For evaluation of the project, I will be taking a combination of user feedback, model accuracy and error scores, and finally comparing on whether the bot does a better job on detecting and dealing with messages than a human can. Both on a large scale and on a small scale.

### Test plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Test Description** | **Expected Outcome** | **Pass?** |
| 1 | Does the classifier predict hate speech | Will predict hate speech at an adequate speed and deal with it adequately. By either removing the message, warning the user, muting the user, etc |  |
| 2 | Does the classifier predict offensive language | Outcome will be similar to hate speech and punishment will be dealt depending on the severity and previous occurrences |  |
| 3 | Are there a low number of false positives? | User testing and feedback will give a yes |  |
| 4 | Are there a low number of false negatives? | User testing and feedback will give a yes |  |
| 5 | Test cross validation scores | They are at an adequate level |  |
| 6 | Test Accruacy | Make sure the model’s accuracy score is correct and not irregular |  |
| 7 | Test for overfitting or underfitting | The dataset is correctly fitted to the model |  |

# 6. Issues and Future Work

## Introduction

There are many issues and future works that will be discussed in this section.

## Issues and Risks

The challenges that the author will come across while developing the project:

* Developing the model and classifier
* Ensuring that the model is fast
* Working with the discord API and connecting all the features
* ethical and moral dilemmas in working on such a project

How The author plan to approach these challenges:

* Complete adequate research and complete an online NLP course
* Testing between different training methods, train on different machines
* Continue working with the discord API and continue learning its features
* Read research papers and articles dealing with the issue

Risks that may come up in the project:

* Developing a model with long training speeds
* Not completing the project in time.

Dealing and overcoming the risks:

* Train the model less frequently, save the model to a file so you not need to retrain it
* Stick to the schedule and split the work into smaller chunks that are easier to do.

## Plans and Future Work

The plan for the project can be seen in the gantt chart below. The project will be completed with constant feedback and improvements on the overall mode, to ensure a highly accurate and functional detection model.

### GANTT Chart

Table

Description automatically generated

# Bibliography

1. Müller K, Schwarz C. Fanning the Flames of Hate: Social Media and Hate Crime [Internet]. Rochester, NY: Social Science Research Network; 2020 Jun [cited 2022 Jan 2]. Report No.: ID 3082972. Available from: https://papers.ssrn.com/abstract=3082972

2. Saha K, Chandrasekharan E, De Choudhury M. Prevalence and Psychological Effects of Hateful Speech in Online College Communities. Proc ACM Web Sci Conf ACM Web Sci Conf. 2019 Jun;2019:255–64.

3. Danny. discord.py [Internet]. 2022 [cited 2022 Jan 2]. Available from: https://github.com/Rapptz/discord.py

4. Davidson T, Warmsley D, Macy M, Weber I. Automated Hate Speech Detection and the Problem of Offensive Language. In: Eleventh International AAAI Conference on Web and Social Media [Internet]. 2017 [cited 2022 Jan 2]. Available from: https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665

5. What is FDD in Agile? [Internet]. Planview. [cited 2022 Jan 3]. Available from: https://www.planview.com/resources/articles/fdd-agile/

6. Antony. Hate speech prediction [Internet]. 2021 [cited 2022 Jan 3]. Available from: https://github.com/PyAntony/hate-speech

7. Top 5 Programming Languages and their Libraries for Machine Learning in 2020 [Internet]. GeeksforGeeks. 2020 [cited 2022 Jan 3]. Available from: https://www.geeksforgeeks.org/top-5-programming-languages-and-their-libraries-for-machine-learning-in-2020/

8. What is Supervised Learning? [Internet]. 2021 [cited 2022 Jan 14]. Available from: https://www.ibm.com/cloud/learn/supervised-learning

9. What is Unsupervised Learning? [Internet]. 2021 [cited 2022 Jan 14]. Available from: https://www.ibm.com/cloud/learn/unsupervised-learning

10. What is Deep Learning? [Internet]. 2021 [cited 2022 Jan 14]. Available from: https://www.ibm.com/cloud/learn/deep-learning

11. Brownlee J. What Is Natural Language Processing? [Internet]. Machine Learning Mastery. 2017 [cited 2022 Jan 3]. Available from: https://machinelearningmastery.com/natural-language-processing/

12. What is Text Classification? [Internet]. MonkeyLearn. [cited 2022 Jan 3]. Available from: https://monkeylearn.com/what-is-text-classification/

13. Text Summarization Approaches for NLP - Practical Guide with Generative Examples [Internet]. Machine Learning Plus. 2020 [cited 2022 Jan 3]. Available from: https://www.machinelearningplus.com/nlp/text-summarization-approaches-nlp-example/

14. TensorFlow [Internet]. TensorFlow. [cited 2022 Jan 3]. Available from: https://www.tensorflow.org/

15. Inappropriate or offensive content - Hwb [Internet]. [cited 2022 Jan 15]. Available from: https://hwb.gov.wales/zones/keeping-safe-online/inappropriate-or-offensive-content

16. O’Neill S. Anti-Bullying with Machine Learning Final Year Project Report. :67.

17. Top 4 software development methodologies | Synopsys [Internet]. Software Integrity Blog. 2017 [cited 2022 Jan 15]. Available from: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/

18. Davidson T. Automated Hate Speech Detection and the Problem of Offensive Language [Internet]. 2022 [cited 2022 Jan 15]. Available from: https://github.com/t-davidson/hate-speech-and-offensive-language/blob/507feccf0f3cf609409f765a08e0bbdac0a33d68/src/Automated%20Hate%20Speech%20Detection%20and%20the%20Problem%20of%20Offensive%20Language%20Python%203.6.ipynb

19. Home | TensorFlow Hub [Internet]. [cited 2022 Apr 5]. Available from: https://tfhub.dev/

20. Discord Developer Portal — API Docs for Bots and Developers [Internet]. Discord Developer Portal. [cited 2022 Apr 5]. Available from: https://discord.com/developers/applications