CET313 Artificial Intelligence - Assignment

Jerome Thayananthajothy (229472295)
BSc (Hons) Computer Systems Engineering Top Up

Hate Speech Detection System API with Forum Website Integration

Word count 5106

Introduction

The sudden and unexpected growth of social media platforms as the likes of Twitter and other such community forums has certainly made a huge impact on the pursuit of mankind towards the future, specifically in the communication and content publishing space. Inversely this has resulted in a contribution towards the propagation of so-called hate speech, a form of hurtful and insulting language adopted by humans to exacerbate their pretence of superiority among others not to mention the operation of hateful activities (Badjatiya, P. et al, 2017). The hidden nature and convenience provided by such platforms have contributed to the breeding of hate speech which then eventually results in hate crimes. All of this is done in a virtual environment past the reach of conventional law enforcement.

The definition of "Hate Speech" according to Nockleby, J. 2000 is "any communication that disparages an individual or a group on the grounds of some characteristics such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other categorization". Recently there has been a noticeable increase in hate speech owing to current events, especially in the UK (Guardian, 2017). Various surveys and studies focusing mostly on the youth have reported a growth in hate speech and related hate crimes. The basis of these attacks is ethnicity, religious beliefs, gender, and or sexual orientation (EANews, 2017). The seriousness of this matter has gained increasing recognition and according to reports, a range of measures to amend the issue has been initiated internationally.

In recent years efforts on a large scale have also been made for the automatic detection of hate speech in the domain of social networking. These commonly consist of semantics analysis procedures of the content using methods such as Natural Language Processing (NLP) and Machine Learning (ML). The aforementioned task is accomplished by scanning textual content and classifying them into non-hateful or hateful, where it may also be able to recognize the types of hate speech depending on the data provided for the model to learn from. It has been observed that even though methods employed currently show optimistic results, the evaluations performed by these systems are mostly biased and classify content that is usually not considered to be hateful content to be determined as hateful content in opposition to detecting actual hate content. A few more studies have shown that trailblazing methods that detect content that is considered to be "sexist messages" can only acquire an F1 score of between 15% and 60% points under the detection of non-hate messages (Park, J. 2017). These findings show that it is more difficult to identify hate content and relative types as opposed to non-hate content. Although, it can be argued from a practical perspective that the capability to precisely and exhaustively recognize and determine particular types of hateful language is more profitable.

This project uses an extension of the model provided by the PyTorch framework and a large collection of English Twitter datasets to train and test them so as to perform to an almost desirable level of competence in detecting hateful language by means of sentiment analysis and a neural network.

Literature Review on Prototype Identification

Owing to the multitude of information available on the world wide web, there is indeed strong incentive to study machine learning-based hate speech detection capabilities (Mondal, M. et al. 2017). The study contributes to an important factor in reducing crime that is resulted from hate speech and protecting people's opinions. This section of this report focuses on the various studies previously performed on the subject of hate speech detection. It is also worth mentioning that such studies have also resulted in lowering the rate of hate speech which results in a reduction in crime (Sanoussi, M. et al. 2022).

Even though there are numerous studies available, hate speech is still proving to be a problem and a challenge. This is because of the reason that both humans and hate speech detection systems even the best have issues identifying hate speech because of the convolution and variation of hate speech classifications. Often a more intense form of prejudice, Hate speech is considered to be somewhere in between what is considered to be aggressive and abusive textual content. Even still the afore mentioned share a derogatory aspect. Alternatively homophobic, sexist, and religious hate is comparatively dissimilar as it focuses on and target a class of individuals or a gender. Studies show that the separation and classification of the concepts are painstakingly difficult (Ayo, F. et al. 2020). Certain definitions of hate speech are debatable; tweets considered racist and homophobic for instance, are often more probable to be determined as hate speech rather than classed as other harmful or unacceptable content types. Since this is the case there is no way to be definite about whether a provocative textual content can be categorized as hate speech (Strossen, N. 2016). The wise use of the type of short textual documents much like a tweet on Twitter is cause for serious issues; they prove to be problematic to the traditional approaches such as bag-of-words model, resulting in the inadequacy of data because of a deficit in contextual information (Comito, C. et al. 2019). Added to this the dataset collected may vary because of these categorizations and meanings, making it difficult to collate machine learning models (MacAvaney, S. et al. 2019).

There are several other studies on the subject of hate speech however, most only tend to target one particular aspect of the subject and are comparatively old. The study by Ayo, F. et al. (2020) focused on creating a generic metadata architecture for the qualification aspect of hate speech based on a set of an already agreed-upon set of score groups using semantic analysis. The study by Chetty, N. et al, (2018) was about the types of hate speech that was focused on gender, religious beliefs, and ethnicity-related to cyberterrorism and legal frameworks of an international calibre; although it has to be noted that this study did not take into consideration Twitter or machine learning specialized datasets. Most of the work cited by Paz, M. et al. (2020). is regarding legal literature that defined what exactly hate speech is for the purpose of imposing penalties on those who commit such crimes. The study by Matamoros-Fernández, A. et al. (2021) concentrated on the geographical aspects of hate speech and social media diversity. There are few papers aimed at the dedicated study of English hate speech.

The process of properly identifying hate speech is difficult, mostly because of the accessibility of quality datasets along with the resources for the development of robust ML models with quality performance of an considerable standard. The detection of what is considered to be offensive content is troublesome for a number of reasons, which also includes word discombobulation, complications in pointing out racist slurs, and language of abusive nature that is disguised as professionally written content so that it breaches sentence divisions. Such works like SemEval and HASOC tasks are in need of more rugged, and reliable datasets with large quantities of data. This is because of the wide use of hate speech detection systems. Even still datasets of a smaller size range are not enough for arriving at conclusions. For instance, a few research were done using only 200 tweets. While this produced good results it raised concerns regarding performance generality. Hence it can be said that the capacity of the dataset alone does not assure diversification (Bender, E, et al. 2021).

Reflection on the Prototype identification

It was determined that a large number of systems that functioned to identify hate speech and offensive content already existed in the industry. Though the datasets were lacking and definitions for hate speech varied and were constantly changing, systems of powerful qualities that performed the same functions with enhanced and efficient capabilities already existed, this project is not to propose a new and improved method or design for such systems but an implementation of such a system where it has or at the least has not been integrated into a real-world application like a forum website where the main subject is a person's conversation and subsequent replies by other people. The identification and inspiration for this project were easy enough to achieve for it was a follow-up of the practical activities provided by the module (CT313 Artificial Intelligence).

The problem with suggesting an improvement for such a system as hate speech detection would require more extensive research and advanced study into the subject as a lot of focus has already been applied to the matter. So, instead of an enhancement or improvement that is unlikely to make an impact, an actual integration of the system into a niche partition of social media such as online forums or even any other websites where discussions by people are involved would benefit greatly from this project especially as it provides an API to analyse and indicate if a given content is hateful and or offensive.

An abundance of literature was available on the subject of hate speech detection so sufficient advice and assistance were available throughout the research phase of the project. Hence the research and review were easily done without difficulty in acquiring study material. The inspiration behind this project was, of course, a leaf out of the book of our dear lecturer and mentor Mrs. Kate MacFarlane's project to achieve almost the same goal. It must be said that the guidance and inspiration provided by her are greatly appreciated.

Development

The project consists of two parts; one a functioning forum discussion website where users can log in and create new discussions (threads) and to these discussion other users can make replies to; thereby propagate a conversation with each other. Text inputs are provided to type in user replies and the content is saved using a structured database that is attached to the project. The second part is the actual AI module wrapped with an API that allows access to receive and send data. The AI module is a neural network that is built using basic concepts with a manually trained model and datasets acquired through third-party means.

The idea is for when a user creates a new thread or posts a new reply the textual content before being saved into the database is sent to the AI module API and is analysed for offensive and or hateful content. If the AI model evaluates the content to be acceptable then the user's data is saved into the database and is displayed on the forum website, otherwise an alert message is sent to the user creating the text post to notify the user of the unacceptable content present in their posts. The below diagram illustrates the flow of the idea.

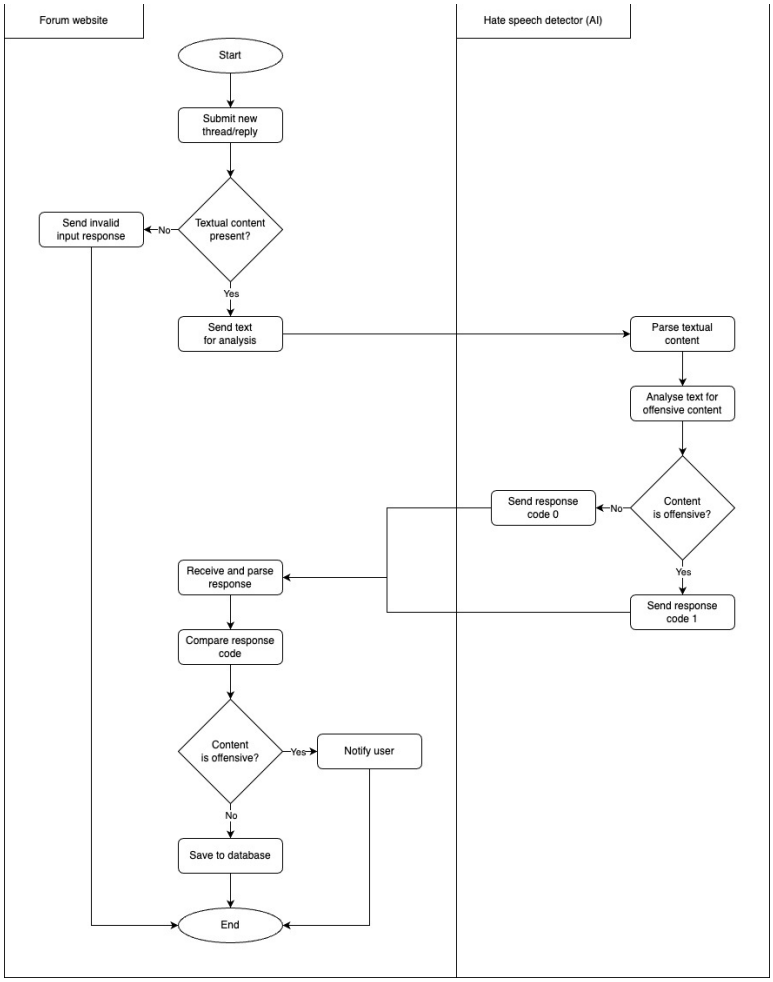
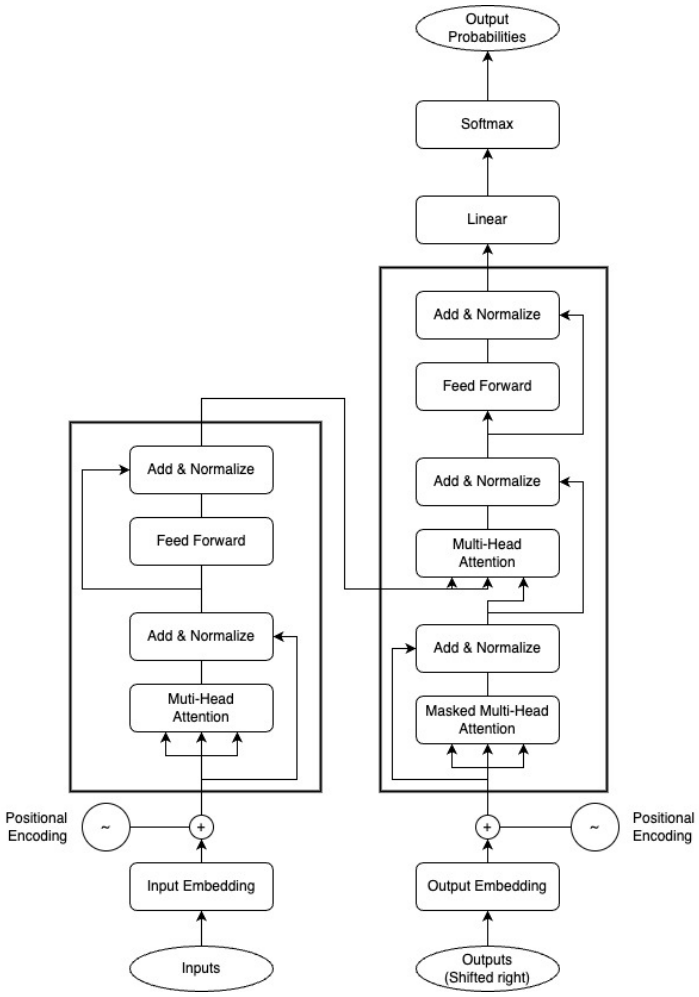


Figure 1: Full project API flowchart

Hate speech detector (Artificial Intelligence Python module)

The AI module is a neural network which performs hate speech detection that when textual content is input, analyses the sentiment of the sentences and determines if the content is considered hateful, offensive or is acceptable. The out is binary meaning the text is determined to be either unacceptable (contains hate/offensive content) or acceptable. The diagram of the design and flow of the model used for this module is given below.



This model is custom built with elements extended from PyTorch and Twitter datasets to train it. The following diagram show the architecture of the whole system. The main focus of the kernel is to train a simple transformer from only PyTorch to classify hate speech in the provided textual content. The Python package TorchText was also used as it helped prepare the data and in addition had excellent documentation as well as various blogs and tutorials were referenced on how to use them properly (Pai, A. 2020).

As this module will be classifying content only the encoder part of the transformer is required. To achieve this the transformer encoder layer available through PyTorch was used.

In order to accommodate various devices this module would be potentially run on a small conditional block was added to decide to whether to use the devices CPU or if available the GPU.

Dataset

The dataset used for this module was obtained from Kaggle and had the Creative Commons type of public license meaning the provided data is freely available for use in the public domain within the set parameters of the law of course. This dataset is from a paper on Automated Hate Speech Detection by Davidson et al, (2017).

This dataset uses data obtained from Twitter and was used to study hate speech identification. The texts are categorized as, hateful speech, language of offensive nature or acceptable language. It should be noted that due to the nature of the study this dataset contains explicit content such as vulgar language and content generally considered offensive.

Installation

The pre-requisites to run this module on a machine is to have Python and Anaconda installed. Any version between 3.6 and 3.9 is acceptable. The module was developed and run on an Anaconda environment. The environment and all associated Python packages required for this module has been saved to an *environment.yaml* file and the module should be ready to run once the environment is set up correctly.

After install Python and Anaconda, to set up the environment run the following command.

```
➜  forumbot git:(main) ✗ cd ai
➜  ai git:(main) ✗ conda env create --file environment.yaml
```

This should set up the environment and install all the required packages.

Once the environment has been set up to run the API use the following command.

```
➜  forumbot git:(main) ✗ cd ai
➜  ai git:(main) ✗ conda activate torchenv && flask run
```

Please make sure to run all the above given command within the *ai* directory.

Once the command is executed successfully the following output should be received. This indicates that the AI and its wrapper API is up and running, ready to be used.

```
ai git:(main) ✗ conda activate torchenv && flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
nltk_data] Downloading package punkt to
nltk_data]     /Users/thavarshan/nltk_data...
nltk_data]   Package punkt is already up-to-date!
nltk_data] Downloading package stopwords to
nltk_data]     /Users/thavarshan/nltk_data...
nltk_data]   Package stopwords is already up-to-date!
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

This shows that the hate speech detection system API is accessible through the local address http://127.0.0.1:500.

Forum website

The forum website is built using PHP, JavaScript and MySQL. The website is functional for the most part and works just like a forum website is expected to work except for the editing and deleting parts of the posts. These were excluded on the basis of having less time to implement them. It must be noted that a web framework was used to quickly build a prototype web application without spending too much time on it.

Basic functions of the forum website include browsing list of user discussions, viewing individual discussions including replies, creating new discussions and posting replies to individual discussions.

Installation

Before being able to run the forum website, it should be ensured that PHP and Composer is installed on the machine. If the website is going to be run on macOS, PHP and Composer can be installed via Homebrew. In addition, it is recommended to install Node and NPM.

After installing PHP and composer, run the following command inside the *web* directory of the project.

```
➜  forumbot git:(main) ✗ cd web
➜  web git:(main) ✗ composer install
```

*Figure 2: Assume **forumbot** to be the project directory.*

Running that command should output the following:

```
  - Installing sebastian/comparator (4.0.8): Extracting archive
  - Installing sebastian/code-unit (1.0.8): Extracting archive
  - Installing sebastian/cli-parser (1.0.1): Extracting archive
  - Installing phpunit/php-timer (5.0.3): Extracting archive
  - Installing phpunit/php-text-template (2.0.4): Extracting archive
  - Installing phpunit/php-invoker (3.1.1): Extracting archive
  - Installing phpunit/php-file-iterator (3.0.6): Extracting archive
  - Installing theseer/tokenizer (1.2.1): Extracting archive
  - Installing sebastian/lines-of-code (1.0.3): Extracting archive
  - Installing sebastian/complexity (2.0.2): Extracting archive
  - Installing sebastian/code-unit-reverse-lookup (2.0.3): Extracting archive
  - Installing phpunit/php-code-coverage (9.2.19): Extracting archive
  - Installing phar-io/version (3.2.1): Extracting archive
  - Installing phar-io/manifest (2.0.3): Extracting archive
  - Installing myclabs/deep-copy (1.11.0): Extracting archive
  - Installing doctrine/instantiator (1.4.1): Extracting archive
  - Installing phpunit/phpunit (9.5.26): Extracting archive
  - Installing spatie/backtrace (1.2.1): Extracting archive
  - Installing spatie/flare-client-php (1.3.1): Extracting archive
  - Installing spatie/ignition (1.4.1): Extracting archive
  - Installing spatie/laravel-ignition (1.6.1): Extracting archive
  - Installing tightenco/ziggy (v1.5.0): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

  INFO  Discovering packages.

  inertiajs/inertia-laravel ..... DONE
  jenssegers/agent ..... DONE
  laravel/fortify ..... DONE
  laravel/jetstream ..... DONE
  laravel/sail ..... DONE
  laravel/sanctum ..... DONE
  laravel/tinker ..... DONE
  nesbot/carbon ..... DONE
  nunomaduro/collision ..... DONE
  nunomaduro/termwind ..... DONE
  spatie/laravel-ignition ..... DONE
  tightenco/ziggy ..... DONE

83 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
➜  web git:(main) ✗
```

Figure 3: Successful installation of dependencies.

Once the above steps are complete, the project environment and application unique key must be generated.

```
➜  web git:(main) ✗ cp .env.example .env && php artisan key:generate

  INFO  Application key set successfully.
```

The database does not require much attention as a simple SQLite database is used and is built into the application itself. Simply run the following command to create a SQLite database file and the database should be ready to go.

```
➜  web git:(main) ✗ touch database/database.sqlite
```

Next is to run migrations to create the required tables and columns for the database and also to seed data into the database so that the forum website contains sample content. Do so by running the following command.

```
➜  web git:(main) ✗ php artisan migrate:fresh --seed

  Dropping all tables ..................................... 9ms DONE

  INFO  Preparing database.

  Creating migration table ................................ 2ms DONE

  INFO  Running migrations.

  2014_10_12_000000_create_users_table ................... 1ms DONE
  2014_10_12_100000_create_password_resets_table ........ 1ms DONE
  2014_10_12_200000_add_two_factor_columns_to_users_table  1ms DONE
  2019_08_19_000000_create_failed_jobs_table ............ 1ms DONE
  2019_12_14_000001_create_personal_access_tokens_table  1ms DONE
  2022_11_20_160958_create_categories_table ............. 0ms DONE
  2022_11_20_161007_create_threads_table ................ 1ms DONE
  2022_11_20_161016_create_replies_table ................ 0ms DONE
  2022_11_20_161600_create_sessions_table ............... 1ms DONE

  INFO  Seeding database.
```

The backend of the website is now set up and the API should be available to run, but in order for the actual website with the user interface to be accessible to the user the front-end JavaScript needs set up.

First the Node.js dependencies need to be installed. This can be accomplished by running the following command (in the same directory - *web*).

NOTE: just like PHP and Composer the front-end part requires Node.js installed on the machine.

```
➜  web git:(main) ✗ npm run install
```

The following output should indicate the successful completion of the installation.

```
added 128 packages, and audited 129 packages in 5s

26 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

NOTE: Servers to run both API (forum, AI) have been built in and only require simple commands to run them.

Now that the JavaScript modules have been installed, the frontend code need to be compiled to a usable bundling. To do that, run the following command.

```
➜  web git:(main) ✗ npm run build
```

All that is left now is to run the forum website server. Do so by running the following command.

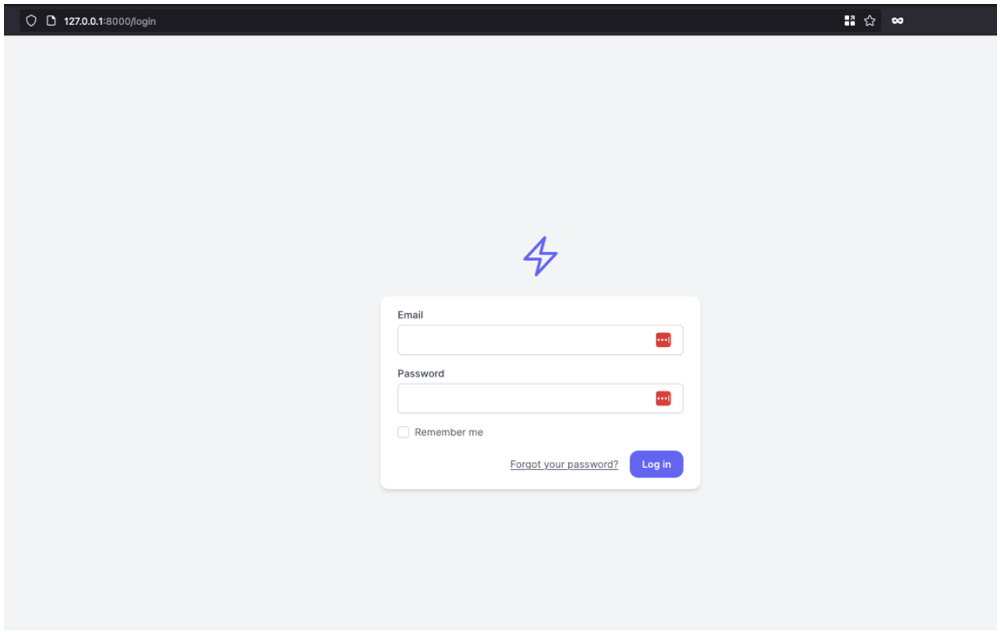
```
➜  web git:(main) ✗ php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server
```

You may now be able to access the forum website on your browser through the address displayed on your terminal window e.g., http://127.0.0.1:8000.

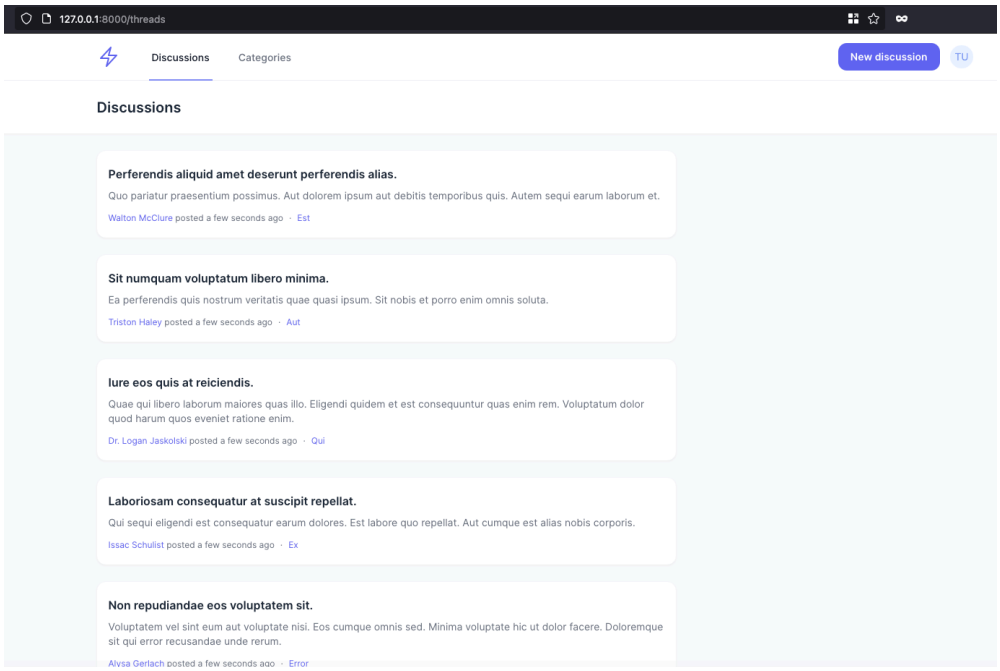
When the given address is accessed through the browser the following screen should be visible.



You may now be able to use the following credentials to access the forum website.

Email: example@email.com
Password: **password**

Once successfully logged in the following screen should be visible.



Evaluation

The finished prototype was evaluated using manual test methods such as manually sending HTTP requests to the API and evaluating the response received. Since the prototype is of two parts tests were mostly focused on the AI part of the prototype and the forum website was less focussed on since it had unit and feature tests written in code.

The test plan included in this document explains the approach taken for testing and overall framework for the testing of the prototype. This document explains the following:

- Test strategy: the rules the tests will be standardized on, including the designated content of the project like objectives and assumptions.
- Execution strategy: explanation of how the tests were performed and the process undertaken to identify and fix defects.

As mentioned previously the prototype is of two parts the forum website and the actual ai module. The forum website will simulate the real-world circumstances of potential hate speech infliction and the ai will perform the main validation and classification of the content submitted to the forum. The forum website does use and is included with a built-in database and possess the related implementation of integrating the database and performing actions on it but those functionalities were omitted in this evaluation since it is irrelevant to the project scope.

The objective of this evaluation is to ensure the prototype functions the way it is intended to according to the specifications. Since the main function of the module was to analyse text and score it on the basis of how offensive it is, tests comprised of simply sending HTTP requests with a body containing textual content with various types of sentiment to the API and evaluating the response received.

Assumptions:

- Sample data be available in the system prior to start testing
- Further action will be taken in the case a defect is identified
- Exploratory testing will be carried out once the build for each part of the prototype is ready for testing
- Performance tests will not be considered
- Test environment will be available
- The system will be akin to a black box and treated as such

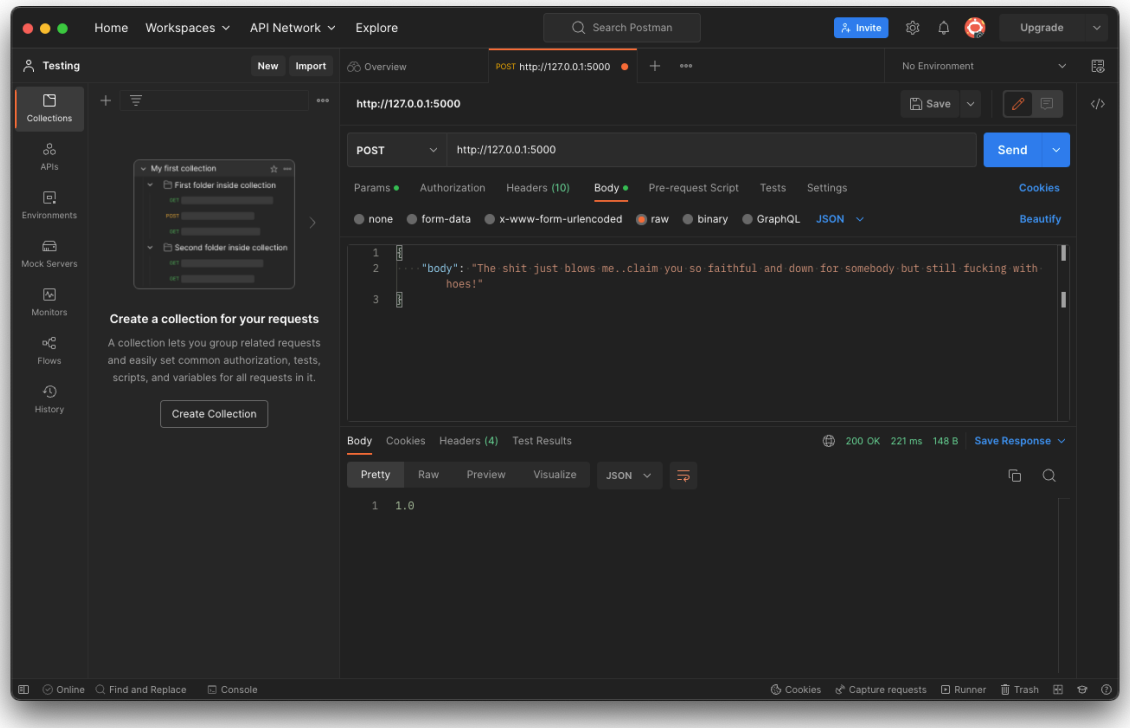
Test Cases

Test Case ID	001	Test Case Description	Test Text Analysis and Classification		
Created By	Jerome	Reviewed By	Jerome	Version	1.0
Tester's Name	Jerome	Date Tested	12-Dec-2022	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Terminal/Postman application	1	body = “Lemmie eat a Oreo & do these dishes. One oreo? Lol.”
2		2	body = “Why the eggplant emoji doe? he say she looked like scream lmao.”
3		3	
4		4	

Test Scenario	Verify on submitting offensive and acceptable content that proper classification is performed.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Fill in payload data	Request body should contain parameters.	As Expected	Pass
2	Send HTTP request	An HTTP request is made and accepted by the server.	As Expected	Pass
3	Receive response	A string response of either “1.0” or “0.0” is received.	As Expected	Pass

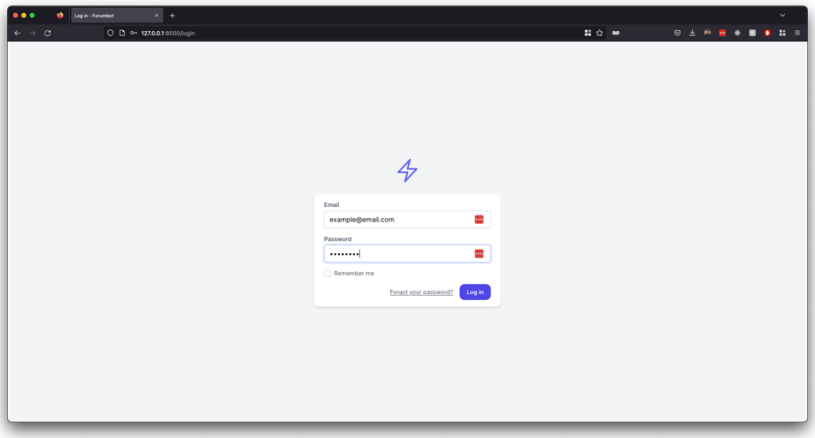


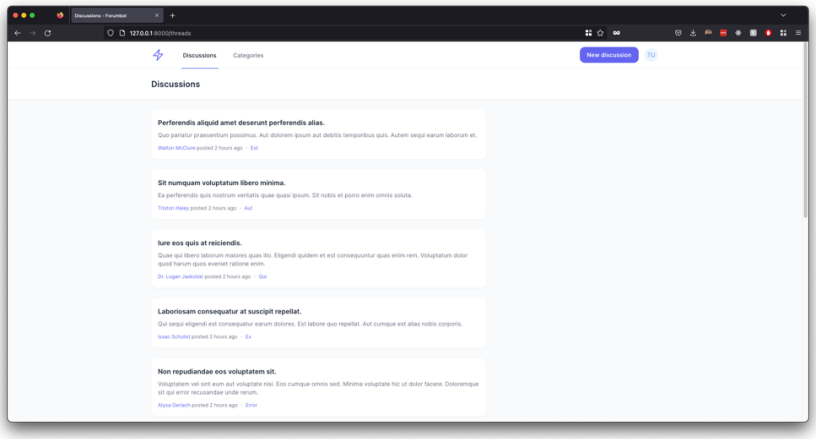
Test Case ID	002	Test Case Description	Test Forum Login Functionality		
Created By	Jerome	Reviewed By	Jerome	Version	1.0
Tester's Name	Jerome	Date Tested	12-Dec-2022	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Chrome/Firefox browser	1	Email = example@email.com
2		2	Password = password
3		3	
4		4	

Test Scenario	Verify on entering valid credentials the user is able to login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://127.0.0.1:5000/login	Login page should be displayed	As Expected	Pass
2	Enter email and password	Credentials can be input	As Expected	Pass
3	Click Login	User is logged in and redirected to discussions list page	As Expected	Pass



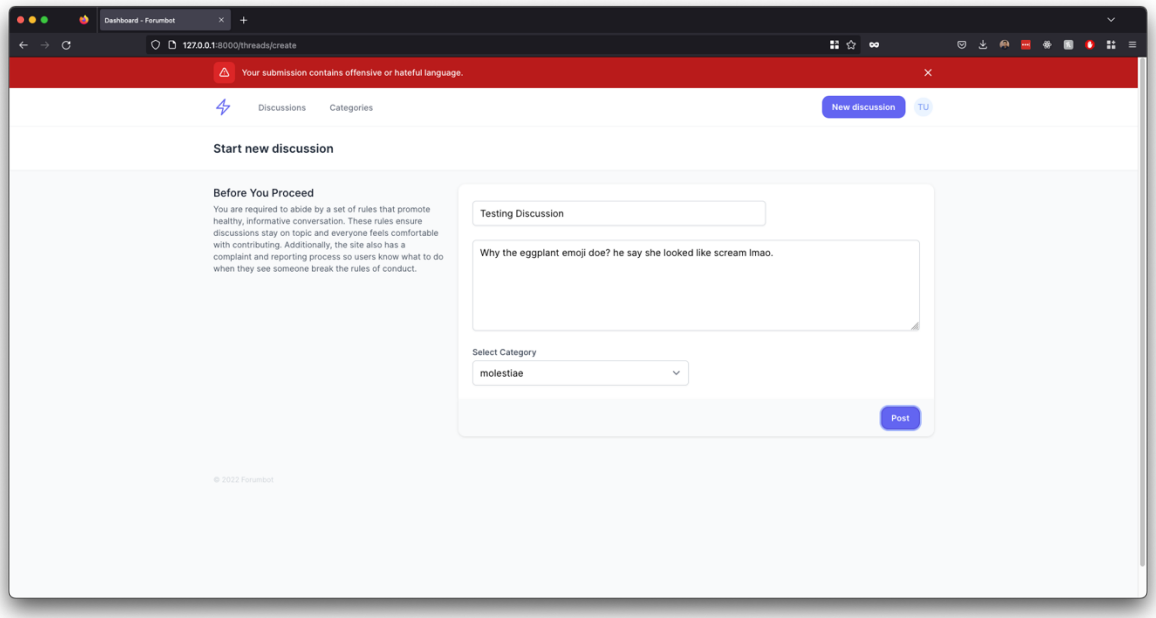


Test Case ID	003	Test Case Description	Test Forum Discussion Creation & Evaluation for Offensive Content		
Created By	Jerome	Reviewed By	Jerome	Version	1.0
Tester's Name	Jerome	Date Tested	12-Dec-2022	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Chrome/Firefox browser	1	title = “Testing Discussion”
2		2	body = “Why the eggplant emoji doe? he say she looked like scream lmao.”
3		3	category = random
4		4	

Test Scenario	Verify on submitting offensive and acceptable content that proper classification is performed and proper response is displayed.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://127.0.0.1:5000/threads/create	The form to create threads should be displayed	As Expected	Pass
2	Enter title, body and choose a random category	The content is accepted in to the input area	As Expected	Pass
3	Receive response	Since offensive content is posted, a banner with red background should appear at the top of the page.	As Expected	Pass

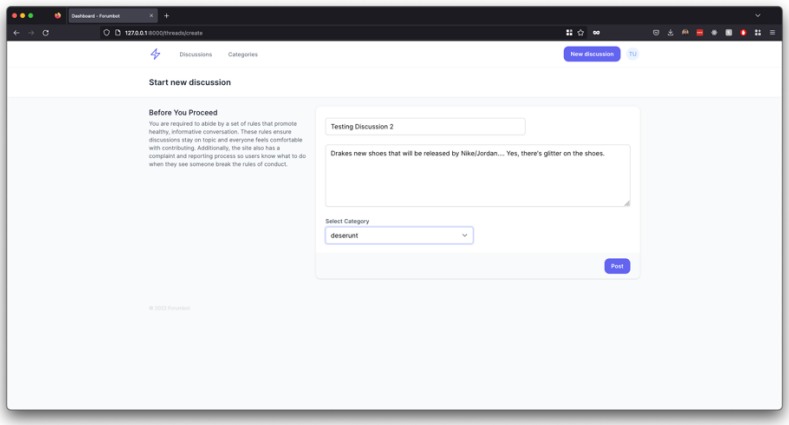


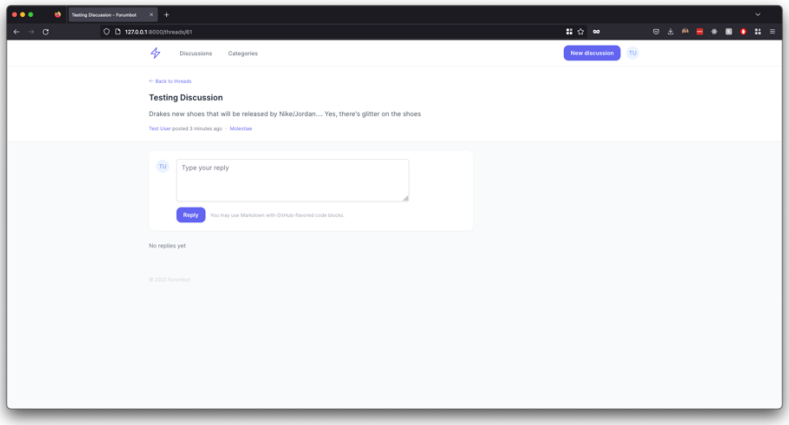
Test Case ID	004	Test Case Description	Test Forum Discussion Creation & Evaluation for Acceptable Content		
Created By	Jerome	Reviewed By	Jerome	Version	1.0
Tester's Name	Jerome	Date Tested	12-Dec-2022	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Chrome/Firefox browser	1	title = “Testing Discussion 2”
2		2	body = “Drakes new shoes that will be released by Nike/Jordan.... Yes, there's glitter on the shoes.”
3		3	category = random
4		4	

Test Scenario	Verify on submitting offensive and acceptable content that proper classification is performed and proper response is displayed.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://127.0.0.1:5000/threads/create	The form to create threads should be displayed	As Expected	Pass
2	Enter title, body and choose a random category	The content is accepted in to the input area	As Expected	Pass
3	Receive response	Since acceptable content is posted, the user should be redirected to the discussion page.	As Expected	Pass



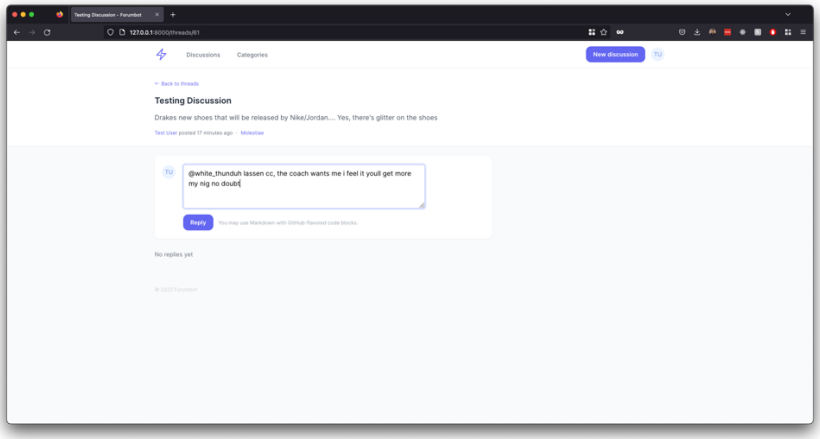


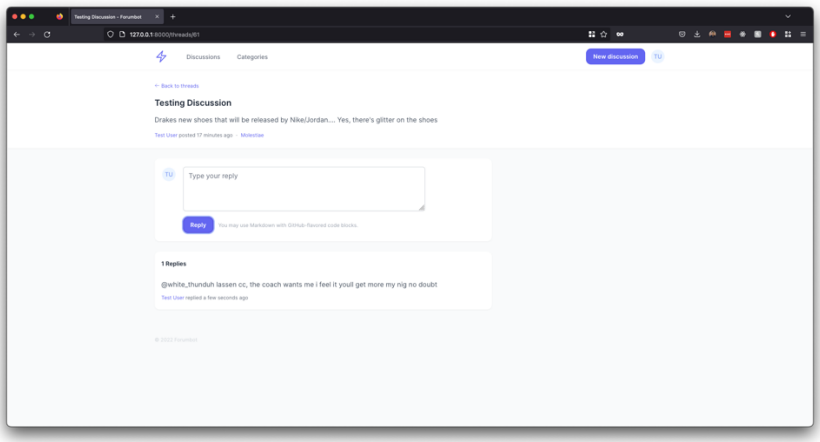
Test Case ID	005	Test Case Description	Test Forum Reply Posting & Evaluation for Acceptable Content		
Created By	Jerome	Reviewed By	Jerome	Version	1.0
Tester's Name	Jerome	Date Tested	12-Dec-2022	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Chrome/Firefox browser	1	body = "@white_thunduh lassen cc , the coach wants me i feel it youll get more my nig no doubt"
2		2	
3		3	
4		4	

Test Scenario	Verify on submitting offensive and acceptable content that proper classification is performed and proper response is displayed.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://127.0.0.1:5000/threads/61	The form to create threads should be displayed	As Expected	Pass
2	Enter text reply	The content is accepted in to the input area	As Expected	Pass
3	Receive response	Since acceptable content is posted, the reply should appear on the page.	As Expected	Pass



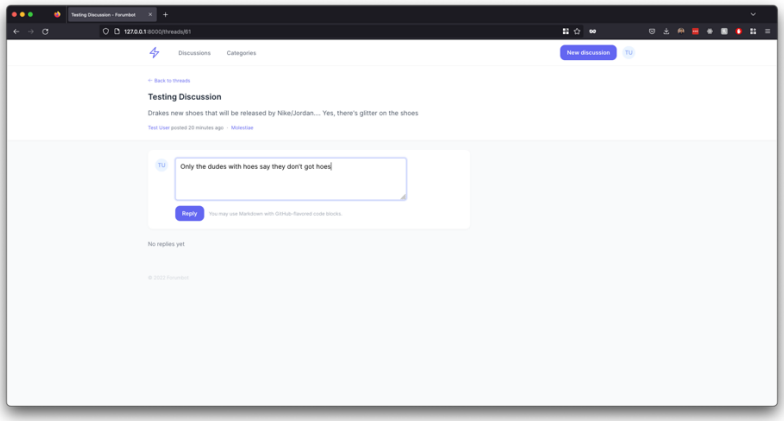


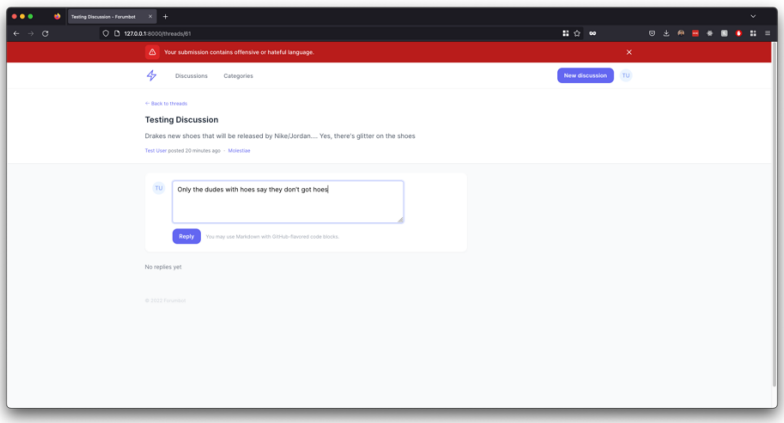
Test Case ID	006	Test Case Description	Test Forum Reply Posting & Evaluation for Offensive Content		
Created By	Jerome	Reviewed By	Jerome	Version	1.0
Tester's Name	Jerome	Date Tested	12-Dec-2022	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Chrome/Firefox browser	1	body = “Only the dudes with hoes say they don't got hoes”
2		2	
3		3	
4		4	

Test Scenario	Verify on submitting offensive and acceptable content that proper classification is performed and proper response is displayed.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://127.0.0.1:5000/threads/61	The form to create threads should be displayed	As Expected	Pass
2	Enter text reply	The content is accepted in to the input area	As Expected	Pass
3	Receive response	Since offensive content is posted, a banner with red background should appear at the top of the page.	As Expected	Pass





Conclusion

Updated and latest datasets of hate speech and offensive content from a variety of sources from different places are continually being developed. Even still many of these datasets prove to be small in comparison to an ideal dataset that would really make a difference, and also, they lack robustness because of how the data was gathered. Many other datasets do not contain enough data and lack semantic variety. On top of all of these shortcomings the language and region from where the data was collected make collation between these hate speech detection models problematic. Since the architecture of the hate speech identification models and also the definition of hate speech is varying and constantly changing it makes it a challenge to come to a consensus. The creation of unbiased hate or abusive datasets that are also large and diverse is described to be arduous and time-consuming and also requires experts on the subject. Further study revealed that the Scientific community at large especially those who are involved in the study of the subject of hate speech detection should seek to enhance and update currently available datasets. Even though many such learning models exist and are constantly being developed, the particular features available on such models should be carefully chosen and finely tuned especially where the classification of hate/non-hate detection is handled. By doing so we ensure the collection of finest quality features for hate speech identification, which also applies to conventional deep learning models. It was also observed that the existing literature does not indicate proper guidelines that define proper comparison between different datasets.

Regarding the academic theory and associated practical work that preceded this research and prototype development; they helped immensely towards the inspiration and the research of this project for it was through suggestions and resources provided by them was the research undertaken. The practical work especially was interesting but also highly educational and helped properly understand the concepts and mechanics behind such systems as the neural network, chatbots, and data analysis. The theory, on the other hand, was, to be frank, a bit dry and for the most part monotonous but helped a great deal nonetheless. Even though the theory and practical work contributed towards the conception and progression further help towards improving it and what other frameworks to use was received through separate research work and a few conversations with experienced academic personnel mainly the lecturer herself.

The development of the prototype itself was fairly straightforward since Python was an easy enough language to learn and has an enormous library of packages available for it. The prototype started by combining both the sentiment analyser for the chatbot and the concept and basic implementation of the neural network provided by the practical activities and was improved upon by adding further features and enhancements. The difficulty of properly understanding how a neural network worked to gain access to the information and then implement a proper system was challenging since the guidance was a bit lacking but was manageable in the end since plenty of information was available online.

References

- Ayo, F.E.; Folorunso, O.; Ibharalu, F.T.; Osinuga, I.A. (2020) Machine learning techniques for hate speech classification of twitter data: State-of-The-Art, future challenges and research directions. *Comput. Sci. Rev*.
- Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. (2021) On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? Association for Computing Machinery: New York, NY, USA.
- Chetty, N.; Alathur, S. (2018) Hate speech review in the context of online social networks. *Aggress. Violent Behav*.
- Comito, C.; Forestiero, A.; Pizzuti, C. (2019**)** Word embedding based clustering to detect topics in social media. *IEEE/WIC/ACM Int. Conf. Web Intell*. WI.
- Davidson, T. et al. (2017) Automated hate speech detection and the problem of offensive language, *Proceedings of the International AAAI Conference on Web and Social Media*.
- EANews. (2012) Countering hate speech online, Last accessed: July 2017, http://eeagrants.org/News/2012/.
- Guardian. Anti-muslim hate crime surges after manchester and london bridge attacks, Last accessed: July 2017, https://www.theguardian.com.
- Jo Ho Park and Pascale Fung. (2017) One-step and two-step classification for abusive language detection on Twitter. In *ALW1: 1st Workshop on Abusive Language Online*, Association for Computational Linguistics.
- John T. Nockleby. (2000) *Hate Speech*, pages 1277–1279. Macmillan, New York.
- MacAvaney, S.; Yao, H.R.; Yang, E.; Russell, K.; Goharian, N.; Frieder, O. (2019) Hate speech detection: Challenges and solutions. PLoS ONE.
- Matamoros-Fernández, A.; Farkas, J. (2021) Racism, Hate Speech, and Social Media: A Systematic Review and Critique. *Telev. New Media*.
- Mondal, M.; Silva, L.A.; Benevenuto, F. (2017) A measurement study of hate speech in social media. *HT 2017—28th ACM Conference on Hypertext and Social Media*.
- Pai, A. (2022) Text classification pytorch: *Build text classification model, Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2020/01/first-text-classification-in-pytorch/ (Accessed: December 12, 2022).
- Paz, M. A.; Montero-Díaz, J.; Moreno-Delgado, A. (2020) Hate Speech: A Systematized Review. *SAGE Op*en.
- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. (2017) Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion. International World Wide Web Conferences Steering Committee.
- Sanoussi, M.S.A.; Xiaohua, C.; Agordzo, G.K.; Guindo, M.L.; al Omari, A.M.M.A.; Issa, B.M. (2022) Detection of Hate Speech Texts Using Machine Learning Algorithm. *IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*.
- Strossen, N. (2016) Freedom of speech and equality: Do we have to choose. *JL Pol'y*.