

Defending Fun

Using Deep Learning and Natural Language Processing in Moderation Systems for
Esports Titles

Thomas Vernon

1907240



Swansea University
Prifysgol Abertawe

Department of Computer Science
Adran Gyfrifidureg

2023 – 2024

1 DECLARATION

Statement 1

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed (candidate)

Date

Statement 2

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by citations giving explicit references. A bibliography is appended.

Signed (candidate)

Date

Statement 3

The University's ethical procedures have been followed and, where appropriate, ethical approval has been granted.

Signed (candidate)

Date

2 ABSTRACT

Toxicity is the word used to describe the negative behaviour that some people exhibit when communicating with others while online, particularly in a manner that they would not necessarily show if communicating face-to-face [1]. In the realm of online multiplayer games, these toxic interactions are often prevalent, and it is usually the focus of research when trying to combat it. It can take many forms, but it mostly condenses down to any action performed in-game by a player, with the intention of upsetting or abusing another player.

All online multiplayer games face the issue of toxicity between players in one form or another. Competitive and esports titles allow players to compete against one another, often grouped in teams. For these teams to be able to collaborate with each other, a greater level of communication is needed compared to non-competitive games. This often takes the form of a text chat, or a voice chat system to allow the teams to communicate with one another, and even sometimes to the opposing team(s). The unfortunate truth is, when a form of communication is granted to players, there will inevitably be bad actors that aim to abuse it. Players may get angry or upset at themselves or others, and this can materialise into toxicity or abuse towards other players.

This project aims to look at the usage of deep learning and natural language processing to detect toxicity through the analysis of real game data from Valve Software's "Counter Strike", and to evaluate its effectiveness and viability with feedback from real people. In the process, we look at the methods that are currently employed in existing systems across other eSports titles, to further assess viability. The overall objective of this paper is to contribute to the effort of tackling toxicity in gaming, or perhaps even on the wider internet, which will hopefully create a safer online environment in the future for all.

It was found that while the application of a deep learning model to detect toxicity in the context of Counter Strike is feasible, careful tweaking and fine tuning is needed for high accuracy, which is vital if harsh punishments are being administered based off these predictions.

3 ACKNOWLEDGEMENTS

Thanks to my lecturers and supervisors for supporting me for all these years.

Thanks to my mother, father, brother and sister for caring for my passions.

Thanks to all the friends I made along the way.

And a very special thanks to Sam, Riley, Matt and Charlie for being the greatest teammates ever, and my best friends.



4 Table of Contents

1	Declaration.....	2
2	Abstract	3
3	Acknowledgements	4
5	Introduction.....	6
5.1	Background.....	6
5.2	Motivations	6
5.3	Objectives.....	6
6	Existing Systems and Analysis	8
6.1	What Is Toxicity?.....	8
6.2	Counter Strike (Valve Software).....	8
6.3	Faceit.....	9
6.4	Valorant (Riot Games).....	10
6.5	Rainbow Six: Siege (Ubisoft).....	11
7	Development.....	12
7.1	Counter Strike as a Platform	12
7.2	Counter Strike 2 and Its Problems	12
7.3	Demo Sourcing Difficulties	12
7.4	Python and Anaconda.....	13
7.5	Demo Parsing Tools	13
7.6	Detoxify.....	14
7.7	Corpus Cleaning.....	14
7.7.1	Spelling Correction	15
7.7.2	Acronym and Word Replacement	15
7.8	The Final Score	16
8	Survey	19
8.1	Survey Results.....	20
8.2	Response Analysis.....	22
9	Findings and Discussion	24
10	Conclusion	26
11	References.....	27

5 INTRODUCTION

5.1 BACKGROUND

The eSports video game genre has been rapidly growing in popularity over the past decade, attracting people from many different locations and backgrounds to come together and collaborate to beat their opponents. This collaboration is often supplemented with the use of an in-game communication system, such as text chat or voice chat – essential to convey creative ideas and tactics. However, this comes with its downsides, and some users will inevitably abuse the communication afforded to them and hurt the experience of others that they are playing with. Because of this, game developers have introduced various systems to attempt to combat this. Looking at the current top 5 largest team-based eSports titles based on tournament prize pool [2], these are Dota 2, Counter Strike, Rainbow Six Siege, Rocket League and Valorant. While only some of these titles allow for voice-chat communication between players, all of them allow for text-chat communication, which can be abused by bad actors to be toxic to other players. The methods for detecting this toxicity can differ from game to game, each with varying degrees of effectiveness.

It is worth noting that the moderation of text chat compared to voice chat seems to be more common and effective. This is perhaps because text chat is much easier to parse and does not require speech-to-text conversion, which is additional processing power and a potential user privacy invasion that game developers may not want to implement.

5.2 MOTIVATIONS

As a long time player of Counter Strike, and with some game time in Valorant and Rainbow Six: Siege, I have spent thousands of hours competing in Esports. As a result, I am no stranger to the different methods that bad actors will carry out when attempting to abuse others online. Counter Strike's system is known for being particularly ineffective, and any time spent playing Valorant (a game created to directly compete with Counter Strike) will show what a good moderation system can do to better the player experience. Toxicity has no place anywhere, especially in online gaming. It is always a shame to see, and it ruins the experience for all the people involved.

My main motivations for this project stemmed from looking for new and novel ways to combat this toxicity that I experienced in Counter Strike, in the hopes that it will add to the sphere of knowledge that can improve the online gaming and Esports experience for everyone. Furthermore, Counter Strike being a particularly open platform (which will be explored later in this paper) makes it the perfect candidate for researching and testing new technologies related to Esports.

5.3 OBJECTIVES

The main objective of this project is to create a piece of software that can take a demo file from the game Counter Strike, and parse that file for all its the text chat data. It will then take this text chat data, and sort it into a format that can be easily analysed and manipulated. After sorting, a deep learning model will be used to predict the toxicity of each of the messages from each player in that match. These predictions will be used to create a “toxicity score” assigned to each player, which could then be used to penalise that player in some way to prevent further offenses if the software were used in a real-world context. A survey with complete examples of these

predictions will be created, and responses to this survey will be used to determine the strengths and weaknesses of the software implementation, and what could be changed to improve it for future implementations.

6 EXISTING SYSTEMS AND ANALYSIS

There are many systems that exist in online multiplayer games, some with different approaches and varying degrees of effectiveness. To help with the development of the detection system, and to better assess its effectiveness, I familiarised myself with some of these systems that I may not have encountered, and further researched the systems that I have seen before.

6.1 WHAT IS TOXICITY?

Before looking at the existing systems that are in place to detect toxicity, it is important to understand what toxicity is, and what it means to be toxic. The relevant form of the word “toxicity” or more precisely in this case “online toxicity”, is defined by The Decision Lab as encompassing “rude, aggressive and degrading attitudes and behaviour that are exhibited on online platforms” [1]. Esports titles are very much included in the class of “online platforms”, but it is important to note that the toxicity exhibited in these games both have similarities and differences to that seen on more general online platforms such as social media and online forums. Encompasses all the traits as mentioned by The Decision Lab, but toxicity in online games (when limited to text chat) can also include:

- Ghosting: giving information about your own team to the opponents to punish players on your own side
- Skill judgement: making demeaning comments about another player’s skill
- Spam: repeating (sometimes meaningless) words or phrases to annoy other players

This will be important to pay attention to when creating a toxicity detection system, as while they may not be seen as very negative to outsiders unfamiliar with a competitive Esports environment, to players that either perceive the stakes to be higher, or take the game more competitively, they are likely held at a similar importance compared to more general forms of toxicity seen elsewhere.

6.2 COUNTER STRIKE (VALVE SOFTWARE)

The current system in counter strike [3] punishes players by automatically hiding their communications in game from other players, triggered by a certain number of toxicity reports (shown in figure 1). It acts as if a player has already “blocked” their communications, and thus a player can simply “unblock” them if they wish to see and hear the offending player’s text and voice chat (demonstrated by “noobtray” in figure 2).

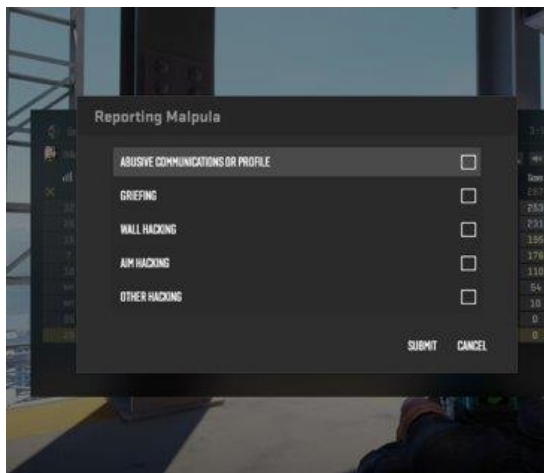


Figure 1: Report categories in Counter Strike



Figure 2: A player with an “automatic mute”

It’s a commonly known fact that toxicity is a problem in Counter Strike [4], which demonstrates that its moderation system is largely ineffective. It’s relatively common to encounter players that have been automatically blocked due to their behaviour, but also common to receive text and verbal abuse from people that haven’t been automatically blocked.

6.3 FACEIT

Faceit is a third-party matchmaking system service for Counter Strike. This means that players can play and compete on a service that is completely separated from Counter Strike’s services. An improved anti-cheat software (used to detect and prevent cheating), better matchmaking system, and a paid priority matchmaking queue are all incentives to use the platform. These incentives could also contribute to a less toxic atmosphere in general, as players that use Faceit could generally take the game more seriously.

Faceit has two systems in place to combat toxicity. The first is a rather rudimentary report-based score called “FBI” short for Faceit Behaviour Index [5] (figure 3). It has a base score of 1000, and positive or negative reports will cause the score to increase or decrease respectively. No reports will push the score towards 1000. Although it still has its flaws due to relying purely on user reports and has almost no repercussions or punishment, its increased presence in the game could act as a deterrent.

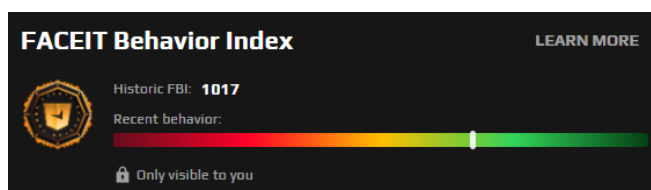


Figure 3: The Faceit Behaviour Index

The second, much newer system Faceit has in place is dubbed “Minerva” [6]. Its aim is to mimic the presence of an administrator (admin for short) in the server for each game played, and to take action if toxic behaviour is detected. It does this by analysing the text chat from each game using an AI model trained on existing Faceit match data. This AI will deliver warnings for toxic abuse detected in the context of the conversation (a very important point that will be encountered later in this paper), and flag spam messages when detected. Repeat offenders will receive

progressively harsher punishments in the form of queue cooldowns (the inability to enter the matchmaking queue to play a game) and are immediately active after the offense takes place.

The existence of a repercussion is far more effective than none, and a very valuable deterrent to prevent offenders. Also, the fact that the punishment is relatively harsh, demonstrates Faceit's confidence in the system's accuracy. In their article announcing Minerva, Scuri explains how total toxic messages sent on the platform were reduced by around 20% after the system was first put into place [6], proving its success (figure 4).

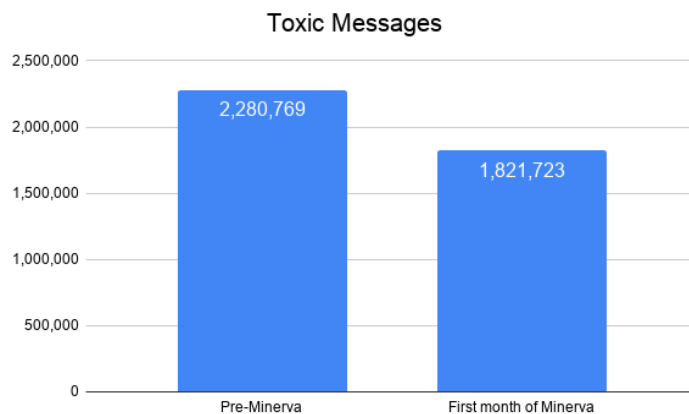


Figure 4: The effect of the Minerva system

6.4 VALORANT (RIOT GAMES)

At first glance, Valorant's toxicity detection and punishment system looks very similar to Faceit's. Much like Minerva, it seems like they use an AI model that they are actively training to detect toxic words and phrases [7]. If these words and phrases are considered "zero tolerance", that player will face immediate repercussions, in the form of communication restrictions and escalating to matchmaking bans (again, much like Faceit's Minerva). The severity of these bans can be extremely severe for chronic offenders and can be as long as a year, which shows the confidence they have in the detection system.

In their article explaining their system, they show that they experienced an increase in game restrictions and game bans while rolling it out (figure 5) [7]. However, they are careful to highlight that the numbers are "not necessarily an indication that toxicity in Valorant has gone down as a result", and that after surveying players, they noticed that "the frequency with which players encounter harassment in our game hasn't meaningfully gone down". They conclude that this information shows that their work up to that point is at best "foundational" [7]. However, the fact that they have seen no increase in perceived toxicity despite the game's booming popularity, demonstrates that the system is not failing. Comparing personal experience playing on the platform to official Counter Strike matchmaking, it's vastly superior.



Figure 5: Game bans on Valorant in 2021

6.5 RAINBOW SIX: SIEGE (UBISOFT)

While I do not have as much experience with this game as I do with Valorant and Counter Strike, Rainbow Six: Siege (often shortened to “R6” or just “Siege”), is also among the top five most popular esports titles, and is thus also relevant and interesting to look at in the context of this project. Contrasting the previous two games mentioned in this section, it seemingly lacks good text chat moderation of any kind, fully relying on Ubisoft’s built-in system. While it does have a zero tolerance on certain words that may be insults, slurs and profanity, penalties are given at the mere mention of these words, even if they are not directed at other users in the form of toxicity. As a result, players can simply omit these known words from their toxic comments, or replace some of the characters with similar lookalikes, still achieving their desired outcome [8].

The repercussions of the toxicity are extremely severe, with only three strikes (administered after manual human review) before receiving a permanent game ban [9]. This is disproportionately more severe than any of the previous systems that have been looked at considering how rudimentary and simple it is, and the community’s overall negative sentiment on the system despite complaints of toxicity in the game show that it is likely not the way forward.

The fact that Ubisoft still relies on manual reviews to administer punishments shows that their existing system is inaccurate, and that they do not have an automated system accurate enough to administer these punishments for them. This may be acceptable at the scale of Siege, as it is considerably smaller than both Counter Strike and Valorant, but should the game boom in popularity for one reason or another, it would likely be infeasible for Ubisoft to continue using this system. The above research could also be used as evidence that this manual review system is ineffective and show that a more robust automated system may need to be put into place.

7 DEVELOPMENT

This section will detail the development and implementation of the project, highlighting the successes and challenges that were met in the process.

7.1 COUNTER STRIKE AS A PLATFORM

Valve has a history of releasing titles that are inspired or derived from community-made mods (short for modification) of their own games. The first iteration of Counter Strike (2000) was initially a mod of Valve's Half Life (1998) [10]. As a result of this, many if not all of Valve's games released since have direct mod support from valve, in the form of various creation tools and high accessibility to the game's internal working structure. The current modern iteration of Counter Strike is no different, and this makes it one of the most open platforms currently in the sphere of Esports titles. As a result, it is commonly used for researching and testing new technologies by third parties that do not have an affiliation with Valve. This could be why third-party matchmaking services such as Faceit and ESEA can co-exist with the Counter Strike game and are very rarely seen on other platforms. This was also the main reason why it was chosen as the platform for the implementation of the project.

7.2 COUNTER STRIKE 2 AND ITS PROBLEMS

A reader familiar with Counter Strike may have noticed that up until this point, I have referred to the game in question only as "Counter Strike". There are two iterations of the game that are currently relevant. Those are Counter Strike: Global Offensive (2012) (or CSGO for short) and the newly released Counter Strike 2 (2023) (or CS2 for short) [11]. CS2 brought many changes and improvements, but its release controversially replaced CSGO, rendering it unplayable. Its release also lined up almost perfectly with the start of this project in its planning stages, which meant that various useful tools and libraries for working with CSGO that had been developed over its lifetime, were not compatible with the newer iteration of the game. As a result, CSGO has still been used as a base for this project even though it is outdated. Despite this, it is almost certain that the tools and libraries used for this project will either be updated or have CS2-counterparts that will be created within the coming years which can be used instead.

7.3 DEMO SOURCING DIFFICULTIES

A "demo" file (suffixed with .dem) is a file that can represent a match played in CSGO or CS2 in its entirety. Every player name, movement, shot and grenade throw can be almost perfectly replicated and played back using these files. They can be parsed offline (IE: outside of the Counter Strike live game view) using various parsing tools, and text chat data from each user can also be extracted.

The release of CS2 saw changes in how these demo files are recorded, across both official "Valve" matchmaking and Faceit matchmaking services. One of these being that chat logs of messages being sent by players in-game were not saved. In the context of my project, this makes these demo files useless. Also, demo files recorded in CS2 are not compatible with parsing libraries that have been created for CSGO, as the internal structure of the respective demo files for each iteration of the game are different.

Another issue is that demo files are only available for a certain amount of time after the game is played, on both Valve and Faceit matchmaking. So, the chances of being able to acquire CSGO

demos with chat history to analyse were nil, as by the time the project reached its development phase, CSGO had been replaced by CS2 for many months. Fortunately, this was overcome with using a collection of personally backed up CSGO demo files, all of which have their chat history intact.

7.4 PYTHON AND ANACONDA

The chosen development setup was the use of Python Jupyter Notebooks coupled with VSCode as an IDE. Python is a modern and popular choice for data science and artificial intelligence applications, and there is plenty of documentation and support available online in those areas if needed. The notebook format is a popular choice for data scientists as it has improved quality of life features suited to those tasks, and it allows for easier visualisation and a better workflow for data processing. Anaconda was used due to its good virtual environment handling and its comprehensive list of included libraries, as well as its ease of use for installing new ones.


7.5 DEMO PARSING TOOLS

As previously mentioned, because CSGO has been around for a long time, there are many different demo file parsing libraries to choose from across many different languages. As Python is being used, a popular library for that language “Awpy” was chosen. It was built on the “demoinfocs” project for Golang and is very user friendly. Documentation is provided by the developer, which was used heavily in the development of the software for the project. Below (figure 6) is a few lines of code that use Awpy to extract all the data from a demo file and put it into JSON format, ready for parsing.

```
1 filename = 'exampledemos/681371d2-9b49-4292-b867-97d688215dde.dem'
2
3 demo_parser = DemoParser(
4     demofile = filename,
5     demo_id = "testgame1",
6     parse_chat = True
7 )
8
9 data = demo_parser.parse()
```

Figure 6: Using Awpy to extract data from a demo file

Afterwards, some code was written (figure 7) to sift through the JSON and make the data much more manipulatable. It creates a dictionary where each element is an individual player, each with sub arrays containing the respective unique IDs, steam IDs (used by steam to identify every player on the platform) and their chat history for that match.



```

1  #create a list of all the players in the game
2  playerList = []
3  for chat in data['chatMessages']:
4      if chat['steamID'] not in playerList:
5          playerList.append(chat['steamID'])
6
7  #create a dictionary of players and their chat messages
8  playerChat = {}
9  for player in playerList:
10     playerChat[player] = []
11     for chat in data['chatMessages']:
12         if chat['steamID'] == player:
13             playerChat[player].append(chat['text'])
14
15  organisedChat = {}
16
17  #give each player a unique ID for qol
18  ID = 0
19  for each in playerChat:
20     organisedChat[ID] = {'steamID': each, 'chatLog': playerChat[each]}
21     ID += 1
22

```

Figure 7: Extracting relevant data from the JSON

7.6 DETOXIFY

For my software to be able to calculate the toxicity scores of each player, a model would be needed to take the text chat as an input and then output a prediction. Initially in my unfamiliarity with the subject, it was planned to either train or fine-tune a model using existing counter strike data that I had scraped or sourced myself. However, it was made clear after speaking to some people with experience in the field that this was rather outside the scope of this project. It would require a lot of processing power and good knowledge for the tuning, both of which are not readily available to a third-year computer scientist. One of these people had written some software to analyse the toxicity and sentiment of tweets and was happy to share their code. It was through this that I was made aware of the detoxify library, and it looked like a good fit for this project.

Detoxify is a free and open-source library built using Pytorch, and provides various deep learning models trained on the 1st and 2nd Jigsaw challenges (toxic comment classification challenges co-created with google). One of these models is multilingual, which is ideal in the context of Counter Strike (and other esports titles) where multiple nationalities and backgrounds find themselves in the same environment competing with and against one another.

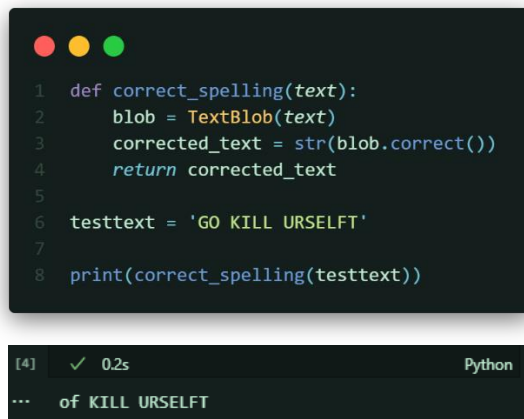
The output of the Detoxify multilingual model is a list of predicted features with the following names: toxicity, severe_toxicity, obscene, identity_attack, insult, threat and sexual_explicit.

7.7 CORPUS CLEANING

Corpus cleaning is a pre-processing step that is done to text before inputting it into a natural language processing model. It involves reducing or removing unwanted data from the input to increase the accuracy of the output. In this case, a couple of processes were tested, one of which was used in the final software.

7.7.1 Spelling Correction

The spelling correction feature of the natural language processing library “Textblob” was tested to attempt to correct spelling where the model was struggling to recognise mistakes. However, this was dropped as there were numerous examples where this made the output *worse*, instead of improving it. Below is one example (figure 8) of the text input 'GO KILL URSELF' (taken from a player from one of the demo files).



```
1 def correct_spelling(text):
2     blob = TextBlob(text)
3     corrected_text = str(blob.correct())
4     return corrected_text
5
6 testtext = 'GO KILL URSELF'
7
8 print(correct_spelling(testtext))
```

[4] ✓ 0.2s Python

... of KILL URSELF

Figure 8: Textblob spelling correction mishaps

As we can see, it was corrected to “of KILL URSELF”, which not only does it fail to correct the incorrect spelling of “yourself”, but it also changes a word that is correct in the first place. When this sentence was fed into the detoxify model, the “corrected” sentence scored 0.6 under the “toxicity” feature, whereas the original sentence scored a higher 0.8. Perhaps this function was being used incorrectly, but this along with some other similar examples led me to bypass this step of the cleaning, as it was felt that it could not be trusted to perform correctly in its state at the time.

7.7.2 Acronym and Word Replacement

One issue that was encountered, was that the Detoxify would fail to recognise certain toxic words and phrases that would only be used in the context of a counter strike game, as counter strike-specific data was not included in its training data. These could be false negatives or positives, such as the term “bot” almost exclusively being used to refer to a bad or unskilled player, or such as the shortened word “ez” often being used in the demeaning context of “this game is easy (you are bad at the game)”.

To try and circumvent this issue, a dictionary was created of acronyms and words that the software would use to find and replace these problematic words as they came up. It was made sure to replace the word with a phrase that better conveyed that word’s meaning, rather than its literal translation. Below is a small excerpt of that dictionary (figure 9), and below that is the code written that performs the processing using that dictionary (figure 10).

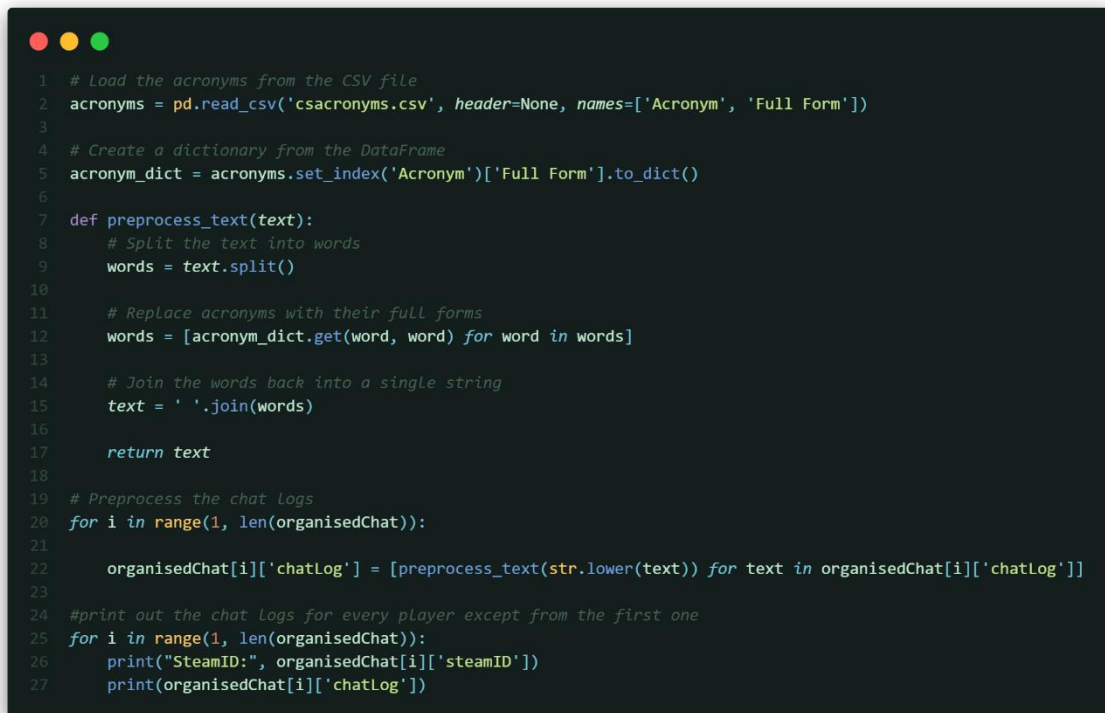


```

1 ns,nice shot
2 nt,nice try
3 noob,bad player
4 smurf,experienced player
5 smurfing,an experienced player
6 ez,you're bad at the game
7 izi,you're bad at the game

```

Figure 9: An excerpt of the acronym and word replacement dictionary



```

1 # Load the acronyms from the CSV file
2 acronyms = pd.read_csv('csacronyms.csv', header=None, names=['Acronym', 'Full Form'])
3
4 # Create a dictionary from the DataFrame
5 acronym_dict = acronyms.set_index('Acronym')['Full Form'].to_dict()
6
7 def preprocess_text(text):
8     # Split the text into words
9     words = text.split()
10
11     # Replace acronyms with their full forms
12     words = [acronym_dict.get(word, word) for word in words]
13
14     # Join the words back into a single string
15     text = ' '.join(words)
16
17     return text
18
19 # Preprocess the chat logs
20 for i in range(1, len(organisedChat)):
21
22     organisedChat[i]['chatLog'] = [preprocess_text(str.lower(text)) for text in organisedChat[i]['chatLog']]
23
24 #print out the chat logs for every player except from the first one
25 for i in range(1, len(organisedChat)):
26     print("SteamID:", organisedChat[i]['steamID'])
27     print(organisedChat[i]['chatLog'])

```

Figure 10: Using the dictionary to clean the corpus

It is worth noting that this dictionary is by no means exhaustive and limited to English, as they were written as the problematic words and phrases were encountered. Perhaps if a much larger curated dictionary for Counter Strike specific phrases was created (one could not be found), this would work better. This step would also likely not be needed if the model that is being fed this text was trained or fine tuned on Counter Strike specific data.

7.8 THE FINAL SCORE

After the text has been scraped from the demo files, sorted and processed, it can now be fed into the Detoxify model. Fortunately, it supports the input of arrays of strings (and will output a corresponding output array of the predicted features of those strings), which was the format of the chat history under each player in the sorted chat history dictionary.

A method was written (figure 11) that loops through the dictionary containing each individual player in the organised dictionary, feeds the chat history from that player into the model, and adds two scores to each player. Those two scores being “average” and “sum”. “Average” takes the predicted feature scores from every chat message and returns an average from each individual message. “Sum” is instead a cumulation of all the individual feature scores from each individual message added together. Initially, “average” was used to calculate the outputted toxicity score, but it meant that players that sent similarly toxic messages, but one sending many more than the other, would achieve a similar score. “sum” was therefore created and used to more heavily penalise players that sent more toxic messages than others.

```

1  #calculate the average of each characteristic for each player
2  averages = {}
3  sums = {}
4  for player in organisedChat:
5      input = organisedChat[player]['chatLog']
6      output = analyze_toxicity(input)
7      sum_output = {}
8      for key in output:
9          sum_output[key] = sum(output[key])
10         output[key] = sum_output[key] / len(output[key])
11         averages[player, organisedChat[player]['steamID']] = output
12         sums[player, organisedChat[player]['steamID']] = sum_output
13 print(averages)
14 print(sums)

```

Figure 11: Calculating the final scores given to each player

The final score for each user was then calculated by summing up all the separate features into a single number, so only one value was presented for each player in the game (figure 12). This was chosen because every one of the features the model was designed to detect and output all should not be present in a competitive online Esports environment, as the person viewing this information should not have to learn how the system works to use it. Because of this, the irrelevant information is abstracted away. A weighting calculation was also implemented (figure 13) to allow heavier or lighter weighting for different features if needed, but for now they were all set to equal weighting, as not enough feedback and experience with the model had been gained to meaningfully adjust them.

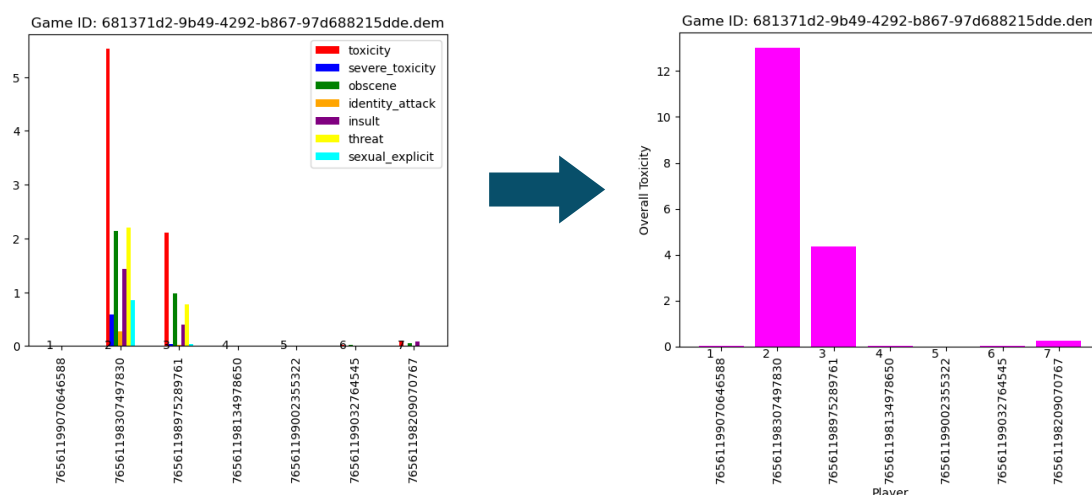


Figure 12: Consolidating the predicted features into a single score



```
1  weights = {
2      'toxicity': 1,
3      'severe_toxicity': 1,
4      'obscene': 1,
5      'threat': 1,
6      'insult': 1,
7      'identity_attack': 1,
8      'sexual_explicit': 1
9  }
10
11  # calculate the overall toxicity for each player and add it to the sums dictionary
12  for player in sums:
13      overall_toxicity = 0
14      for key in weights:
15          overall_toxicity += sums[player][key] * weights[key]
16      sums[player]['overall_toxicity'] = overall_toxicity
17
```

Figure 13: Weighting implementation

Presenting the output as a single score for each player was deemed to be important, as this needs to be easy to understand both in the context of the following survey, and if this score were to be presented to players in a real-world implementation of the software.

8 SURVEY

The purpose of the survey is to gather as much opinion on the output of the software as possible, as opposed to just one. The results of the survey will be used to suggest improvements and adjustments both within and outside the scope of this project.

The survey was sent to friends and acquaintances, most of which are familiar with Counter Strike and the language that is often used in the game. They were left unguided, fully relying on the instructions in the introduction and their own intuition to complete the survey to avoid potential biases that could be introduced by the survey creator. All data recorded from the survey was fully anonymised to protect user privacy, and all the people that took the survey did so voluntarily.

Four games were chosen with plenty of chat activity from multiple players. The games' demo files were fed into the program, and the outputs recorded. The matplotlib library was used to represent the output from each game, in the form of a single chart with each individual player along the x axis, and their respective scores on the y axis.

Each question of the survey contains one of those charts, and the chat history from the corresponding game. It asks the person taking it to check a box corresponding to each individual player, if they agree with the predicted score. If they have any comments to say on the outcome, there is an "other" box they may tick and fill to record their thoughts.

Before the questions, a short introduction was written to introduce the project and survey to the people taking it, and to explain what is required from them (figure 14).

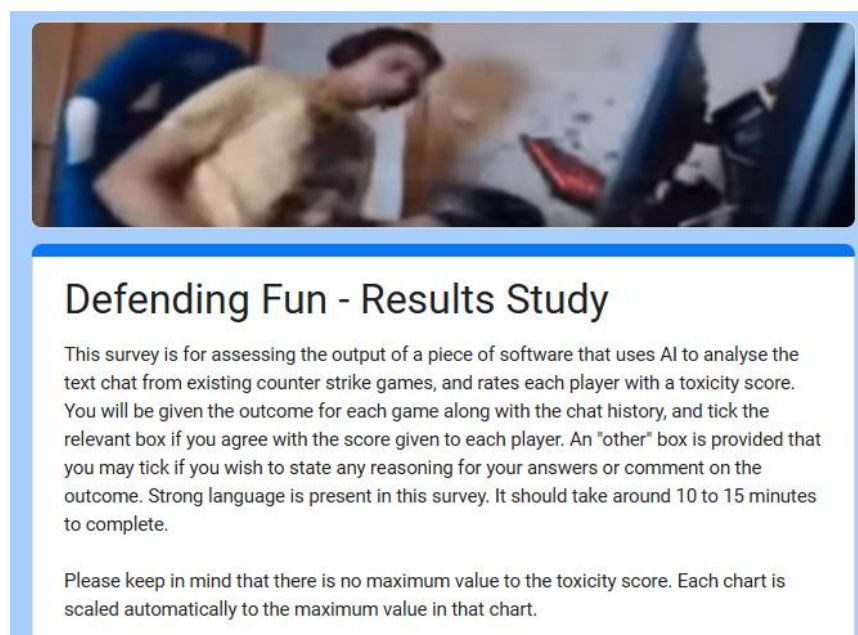


Figure 14: The survey introduction

Below shows an example of the first question of the survey (figure 15). The questions will be attached in full along with the completed code for the project.

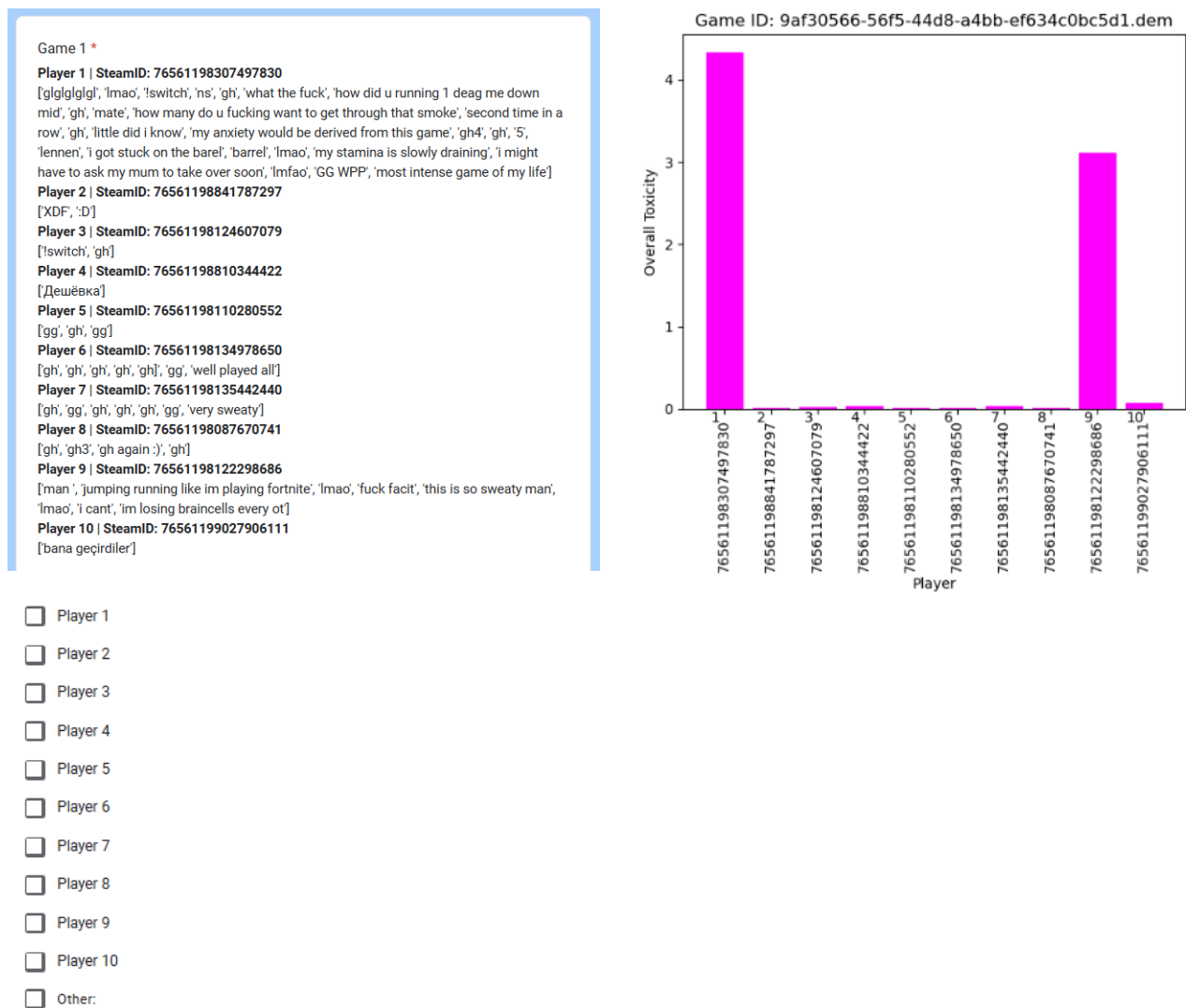


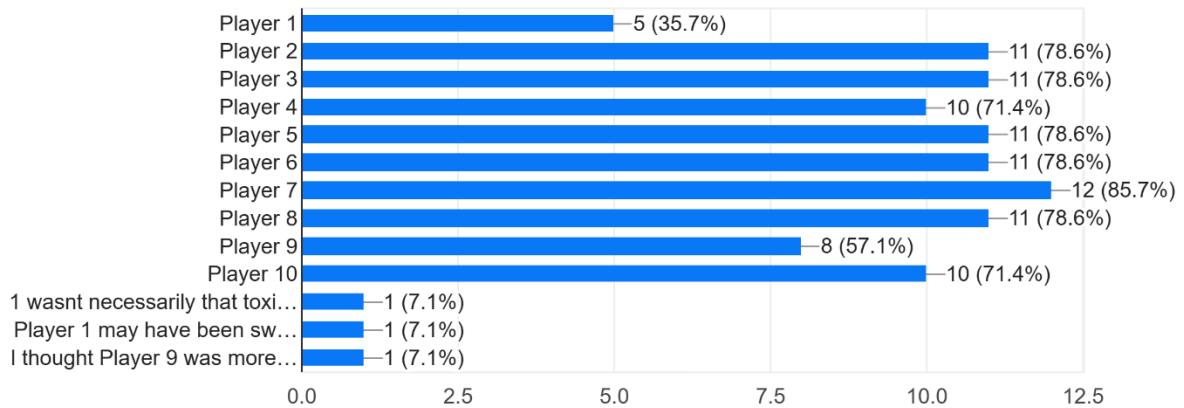
Figure 15: The first question in the survey as presented to respondents

8.1 SURVEY RESULTS

Below are the results of the survey as displayed by Google forms. The percentage next to each player is related to how many respondents agreed with the output of the program for that player in that game. Underneath those values are the comments that were written in the “other” box. The results will be attached in full in CSV format along with the completed code for the project.

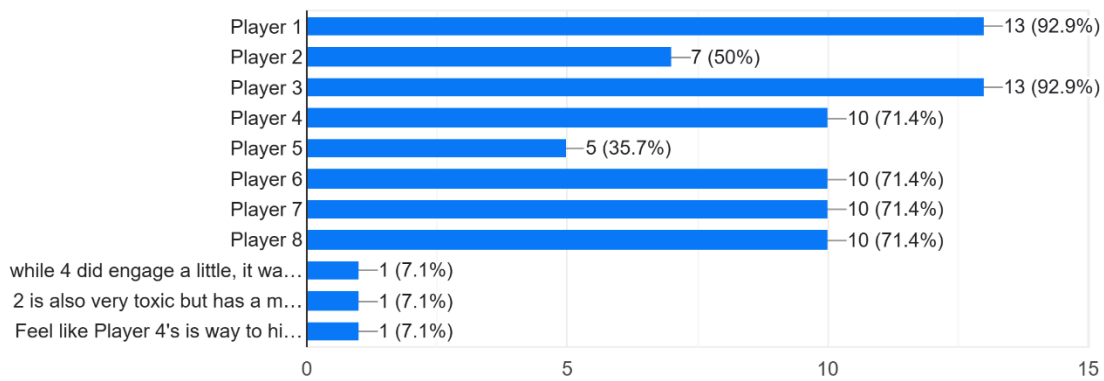
Game 1

14 responses



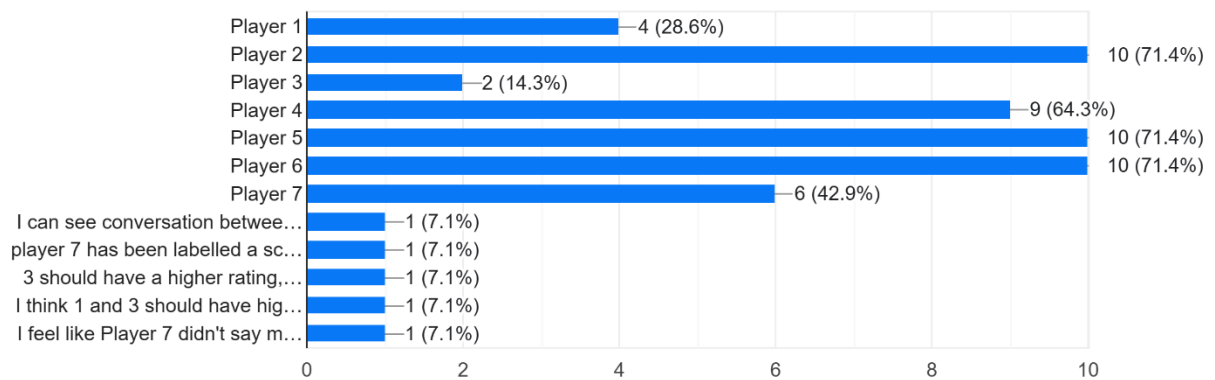
Game 2

14 responses



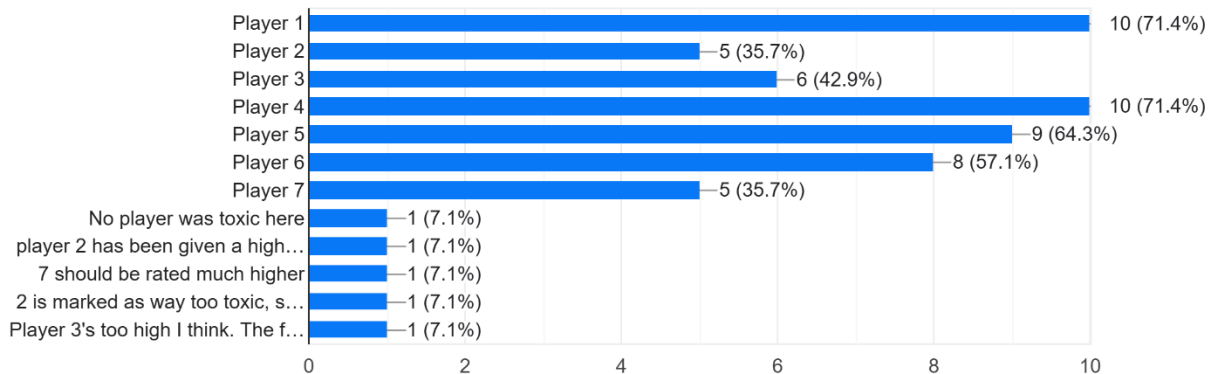
Game 3

14 responses



Game 4

14 responses



8.2 RESPONSE ANALYSIS

The responses were recorded and analysed through google forms, with a total of 14. While many of the responses agreed with the output of the program, there were a few instances where there was a majority disagreement.

The first question shows only a 36% agreement for player 1, and a 57% agreement for player 9. Both players had relatively high scores compared to the others, all of which were given very low toxic scores by the software and was agreed upon by most of the respondents. All the comments on the first game said they thought that player 1 “wasn’t that toxic” and that “player 9 was more toxic than 1). This disagreement could be due to numerous factors, and most of the other disagreements later in the results follow a similar pattern.

In retrospect, despite the warning at the end of the introduction (figure 14), the fact that the scaling across the y axis for each individual chart is different could still be misleading. For the first chart, the maximum score of around 4 was given and multiple respondents felt that this was too high, even though for a couple of the following charts, the maximum score exceeds 10 (which were mostly agreed upon). This was an oversight that could be corrected by scaling all the charts similarly, perhaps to the maximum score across all the games presented.

These disagreements could also be due to the evenly weighted nature of the score being calculated from the model’s outputted features. Currently, if any profanity or sexually explicit language is used, the model will give it an increased predicted score in its relative feature, and this will be evenly distributed relative to the toxic or severe toxic features. While this may be desirable in a game such as Valorant where such words and phrases are forbidden, these are allowed to be expressed in Counter Strike. As such, to improve the output of the program to prevent these disagreements in the context of Counter Strike, the weighting may be adjusted to more heavily weight outright toxicity as opposed to statements or profanity that are not directed at any given person.

Another weakness brought up by the survey is that the score may be improved if it did not fully rely on “sum”. Some respondents stated that some players that sent many minimally toxic messages should not have been rated with a higher score than players that sent fewer more toxic messages, despite the program rating them so. This could be overcome with combining both the

“average” score with the “sum” score, but likely weighted heavier towards the “sum” score. This would help to mitigate this perceived difference in the scores that the respondents thought that the players should have obtained, while still allowing the program to penalise players that send many toxic messages over those that do not.

9 FINDINGS AND DISCUSSION

Looking at the response analysis, and using the experience gained during the development process, it is clear that the accuracy of the model being used is not robust enough for the use case of the program. It is by no means a bad model, however, as it is proven to be very effective within the context it is designed for – general use. This is shown on Detoxify’s github where it is quoted to have achieved 98.86% accuracy on the Kaggle Leaderboard with its “original” model, 94.73% with its “unintended bias” model, and 95.36% with its multilingual model [12]. The shortcomings of this project can thus be attributed to the context that the Detoxify model was applied in, very much outside of its intended use case. This is backed up by multiple accounts during testing and from the survey respondents that show multiple players being given a lower score than others, despite being perceived as being more toxic, due to the use of Counter Strike-specific language that is seldom used elsewhere. Below we can see an example from game 4 in the survey, where player 2 was given a much higher score than player 3, despite player 3 having said more toxic chat messages. These are great examples of the aforementioned “Counter Strike-specific” language, such as “thanks for 33 elo” (elo – a term derived from a similar system used in competitive chess – are points that awarded to players when winning a match, and taken when losing a match, and is used to represent the overall skill of a player) serving the purpose of demeaning the skill of the player on the receiving end of the comment.

Player 2 | SteamID: 76561198866557264

['bot', 'bitches cant play', 'u suck']

Player 3 | SteamID: 76561198064323142

['ghosts cant talk', 'is that why i killed you', 'nice comeback', 'gh', 'you usually say gh at half time', 'btw', 'bot', 'L', 'reply', 'thanks for 33 elo', 'gg lol']

Game ID: 1d199287-4507-4b7f-bb6b-12679d632914 faceit vertigo with swansb.dem

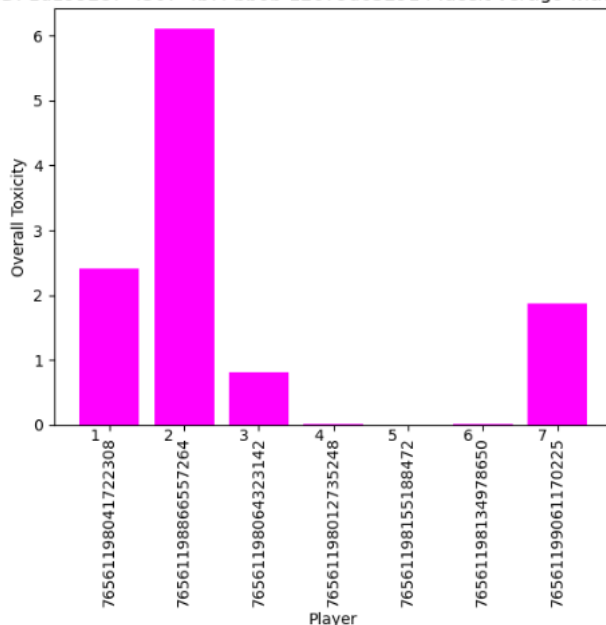


Figure 16: Excerpts from game 4 on the survey, showing the contextual shortcomings of the Detoxify model

To address the issue of context, the model could be fine tuned with Counter Strike specific data to allow the model to recognise and correctly evaluate phrases and acronyms that it previously could not. However, this would be very difficult for an individual, as access to large amounts of relevant data is hard to come by. Some large datasets of demo files of past matches do exist and

are available, but these often have all potentially identifying data removed from them (chat logs included) and are distilled purely down to gameplay only. Further work on this project would likely be required from a researcher affiliated with a large third party in the Counter Strike sphere with access to many match demos, such as Faceit or Leetify (or perhaps even Valve themselves), as the reliance on a personally saved collection of demo files simply isn't feasible when scaling the project up.

It is also interesting to note that the messages that the model has trouble understanding and correctly predicting are those that fall under the categories of "ghosting", "skill judgement" and "spam", all of which were identified in the initial research stage of this project as types of toxicity present in online games that do not appear in more general social online situations.

In the initial stages of this project, it was planned to compare the output of the program with an evaluation from an existing system, such as the report based Faceit Behavioural Index. This would have been ideal, as it is very close in functionality to the existing official toxicity detection and punishment system in the game and would provide valuable insight if it is worth replacing with a new deep learning approach. However, this number is only visible to the player that it belongs to, rendering this plan obsolete.

Another aspect that would have improved the development of this project, would have been if some of the existing systems that were analysed earlier could have been deconstructed and looked at internally. Understandably, their inner workings have been obscured from the public eye so they cannot be exploited by bad actors. This analysis would likely have been very valuable and may have provided some insight as to how I could tweak the weightings of the final score to achieve a more accurate prediction.

10 CONCLUSION

Taking everything learned so far, I believe this project was a success, despite its shortcomings. These shortcomings were by no means negative, as they highlighted important points that should be considered when creating a toxicity detection system with an AI-based approach. The first being that the system should be used and tested for a prolonged period before it is used to make any meaningful decisions related to punishments and game bans. With an extended deadline, this project could have dedicated a significant amount of time and space in this paper to this topic alone, through the tweaking of the implemented weighting system and the gathering of more real-world examples for testing. Another one of these points that this project demonstrated is very important, is the context that the model is trained in, coupled with the context that the model is applied to. The Detoxify model was trained on data that was largely different to some of the language used within Counter Strike, and when confronted with specific words and phrases, it understandably failed to recognise them. This could be remedied through the process of transfer learning and fine tuning the model with Counter-Strike specific data to allow the model to make more accurate predictions where it currently cannot. Should this project be continued, this should be the main point of focus.

Overall, this project provides a good proof of concept for the use of deep learning and natural language processing to identify toxic players in a Counter Strike game. It provides a framework for those that might wish to continue where this project left off, should a large dataset of demo files with included chat history become available in the future, or at the very least a guide for those wishing to familiarise themselves with the process of scraping and sorting text chat data from a Counter Strike game.

11 REFERENCES

- [1] T. D. Lab, "TDL Brief: Online Toxicity," The Decision Lab, 1 March 2001. [Online]. Available: <https://thedecisionlab.com/insights/health/tdl-brief-online-toxicity>. [Accessed 27 March 2024].
- [2] 2023. [Online]. Available: <https://escharts.com>. [Accessed 24 October 2023].
- [3] Valve, "Squelching the Noise," 6 February 2020. [Online]. Available: <https://blog.counter-strike.net/index.php/2020/02/28450/>. [Accessed 16 April 2024].
- [4] J. T. NPC, "zleague.gg," 13 January 2024. [Online]. Available: https://www.zleague.gg/theportal/navigating_turbulent_waters_addressing_toxicity_in_counter-strike_community/. [Accessed 21 April 2024].
- [5] "support.faceit.com," 28 November 2016. [Online]. Available: <https://support.faceit.com/hc/en-us/articles/207338350-The-FACEIT-Behavior-Index>. [Accessed 21 April 2024].
- [6] M. L. Scuri, "blog.faceit.com," 23 October 2019. [Online]. Available: <https://blog.faceit.com/revealing-minerva-and-addressing-toxicity-and-abusive-behavior-in-matches-9073914a51c>. [Accessed 21 April 2024].
- [7] "playvalorant.com," Riot Games, 2 October 2022. [Online]. Available: <https://playvalorant.com/en-us/news/dev/valorant-systems-health-series-voice-and-chat-toxicity/>. [Accessed 24 October 2023].
- [8] O. Richman, "siege.gg," 28 February 2022. [Online]. Available: <https://siege.gg/news/3264-why-is-rainbow-six-siege-toxic>. [Accessed 25 April 2024].
- [9] J. Donnelly, "'We have no regrets when it comes to banning toxic players,' says Ubisoft," pcgamer, 7 September 2018. [Online]. Available: <https://www.pcgamer.com/we-have-no-regrets-when-it-comes-to-banning-toxic-players-says-ubisoft/>. [Accessed 25 April 2024].
- [10] "SteamDB," 14 February 2024. [Online]. Available: <https://steamdb.info/app/10/>. [Accessed 25 April 2024].
- [11] "SteamDB - Counter Strike 2," 24 April 2024. [Online]. Available: <https://steamdb.info/app/730/>. [Accessed 25 April 2024].
- [12] L. Hanu, "Github.com - Detoxify," 1 February 2024. [Online]. Available: <https://github.com/unitaryai/detoxify>. [Accessed 25 April 2024].
- [13] V. Software, "blog.counterstrike.net," 6 February 2020. [Online]. Available: <https://blog.counterstrike.net/index.php/2020/02/28450/>. [Accessed 24 October 2023].

- [14 PajHOLA, "dignitas.gg," 2 May 2019. [Online]. Available:
] <https://dignitas.gg/articles/rocketleague-the-community-and-how-to-avoid-toxicity>.
[Accessed 24 October 2023].
- [15 M. Park, "PCGamer," 13 July 2018. [Online]. Available:
] <https://www.pcgamer.com/rainbow-sixtoxicity-ban/>. [Accessed 24 October 2023].
- [16 D. Soetjipto, "esports.gg," esports.gg, 31 August 2023. [Online]. Available:
] <https://esports.gg/news/dota-2/new-and-improved-dota-reporting-matchmaking/>.
[Accessed 24 October 2023].