Online Bilingual Abuse Detection Using Natural Language Processing and Machine Learning

Santa Sian – 001108298

April 2024

University of Greenwich

BSc Hons Computer Science

Supervisor: Najma Taimoor

Word Count: 13,455

Table of Contents

Abstract

The growing prevalence of cyberbullying justifies worries about the negative consequences of social media's expansion. This growth is accompanied by a greater dependence on social media sites, where the right to free speech is frequently abused. Recent data reveals that around 45.5% of individuals report having encountered cyberbullying at some stage in their lives. This figure marks a significant increase from the rates observed in 2007 and even shows a rising trend from 2018-2019, underscoring a disturbing trend in digital communication. Although specific measures have been implemented to mitigate this issue, they are hampered by their application limitations or the inadequacy of their underlying algorithms. The primary objective of this study is to examine innovative methods for identifying and automatically detecting cyberbullying instances across various social media platforms, including tweets, comments, and messages. In pursuit of this goal, this project uses a dataset comprising authentic Twitter messages in English and Hindi (commonly known as Hinglish). The focus is on pinpointing the foundations of cyberbullying in different languages, specifically during its formation in real-time. This research project seeks to improve the effectiveness of cyberbullying detection mechanisms by developing a novel live chat system that mimics these conversations on OSNs and helps reduce online abuse altogether. This is achieved by employing sophisticated Deep Learning models, such as BERT, and harnessing the capabilities of Natural Language Processing (NLP).

Preface

As I approached the completion of my Computer Science degree, I had a dilemma in selecting a subject for my research. My profound fascination with forensics first led me to pursue a project in that field. During my exploration of many topics, I had a strong interest in a project suggested by our faculty called "Online Abuse Detection Using Natural Language Processing." This project resonated with my increasing concern with the ethical dimensions of technology. It provided an opportunity to explore a field that, while distinct from forensics, shared the common objective of safeguarding individuals and communities. Shifting my focus from forensics to combating internet abuse via technology was both a tough and interesting experience. The experience broadened my academic and professional horizons, allowing me to utilise my computer science knowledge in a sector that has societal significance. The project combined my early passion for forensics with my recently acquired fascination with utilising NLP and Machine Learning (ML) to address real-world challenges.

The reason I decided to concentrate on identifying online harassment, particularly in a bilingual setting, was due to my realisation that the internet serves as a global medium that transcends linguistic and national boundaries. In our ever-evolving modern era where digital communication plays a vital role in social interaction, it is imperative to prioritise the security of online platforms. This project offered a chance to contribute to this crucial field, aligning with our Computer Science program's dedication to implementing realistic resolutions for complex problems. Throughout the course of this project, I have combined my technical proficiency with an increasing understanding of the ethical dimensions of computers. My exploration of addressing the issue of detecting and handling online harassment in a couple of languages has brought me into the complex realm of NLP and contextual machine learning, with a primary emphasis on BERT (Bidirectional Encoder Representations from Transformers). The utilisation of this advanced technology has improved my technical proficiency and deepened my understanding of the complexities associated with ethical AI development.

In retrospect, I see this project as a pivotal moment in my academic career. The experience propelled me outside of my comfort zone, bridging my initial intrigue with forensics to the overarching objective of protecting individuals in the digital realm. As I conclude this crucial phase of my studies, I am grateful for the opportunity to have participated in a demanding and highly informative project, shaping my aspirations and abilities as a developing computer scientist.

Acknowledgements

I would like to thank my supervisors, for their continued support and assistance through my progression, giving useful advice even when I had doubts about my project idea. Thank you for allowing me to be vocal about my thoughts and giving critical feedback that allowed my thought process to steer in the correct direction. I would also like to thank all the YouTube video creators who assisted in related topics regarding tuning a custom dataset with models like BERT. Stack Overflow assisted greatly in troubleshooting my product code. I would finally like to thank me for believing in myself and persevering through a challenging time in my academic career.

List of Tables

List of Figures

List of Abuse Detection Figures

List of Buffer Figures

1. Introduction

The internet is currently the most important worldwide communication tool available, providing a space for the exchange of thoughts, information, digital media such as images and videos, views and day-to-day updates. Due of its diversity, social media has become large and dynamic. It includes prominent online social networks (OSNs) such as Facebook, WhatsApp, LinkedIn, Instagram, Twitter, YouTube, and Instagram. As of now, social media usage is widespread, with Dean (2023) reporting that in 2020, about 3.9 billion people were engaging with these sites, an estimated figure expected to reach 4.41 billion by 2025. Backlink's data from 2024 indicates that 80.08% of the global population aged 18 and above are social media users, with a number of 4.95 billion, which is a 25.6% increase from 2020. Despite its many advantages, social media also has its downsides. Misuse of these platforms has led to harassment based on caste, colour, gender, culture, orientation, and background. Many ideas and contents face harsh criticism and neglect. This increase in critical and harassing behaviours has escalated bullying, particularly cyberbullying. This form of bullying inflicts reputational damage and mental distress on the victims, leading to severe outcomes like depression, self-harm, and, in some cases, even suicide. Given the profound impact on mental health, addressing cyberbullying is essential and urgent.

As shown in Figure 1 below, recent research indicates that up until 2021, 45.5% of people have at some point in their life, experienced cyberbullying. Among teenagers, 60% have faced some form of cyberbullying. Furthermore, a staggering 87% of young online users acknowledge witnessing cyberbullying (Patchin 2022). Studies reveal that girls are more frequently targeted by cyberbullies compared to boys, with 36% of girls reporting such experiences against 26% of boys (Noviantho and Ashianti 2017). As the increasing prevalence of cyberbullying has led to significant health concerns for those affected, the harmful impact of abusive content on social media highlights the pressing requirement for creative solutions to quickly detect and prevent such behaviour (Sharon et al. 2022).

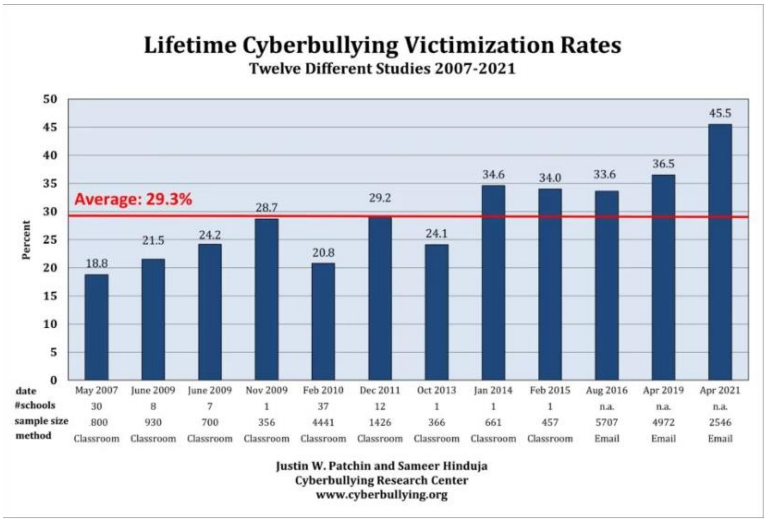


Figure 1 - Cyberbullying victim rates 2007-2021 (Patchin 2022)

As Shah, Phadtare and Rajpara (2022) mention, several third-party applications have been developed to address online abuse, but their reliance on basic keyword-matching techniques often results in inaccurate predictions. The manual entry of data into these systems can reflect the subjective views of the developers, leading to potential biases. A significant limitation of these approaches is the need for a comprehensive, labelled database, often relying on biased datasets. Some methods utilise lexicons, a common strategy for detecting abusive words. However, this approach narrows the application's effectiveness and overlooks hurtful statements that do not contain explicitly abusive language.

This project's goal is to contribute to resolving cyberbullying by detecting and categorising intimidating or threatening texts or messages. The aim is to fine-tune, train and employ a model capable of classifying cyberbullying in both English and Hinglish, utilising NLP and ML techniques with Deep Learning models like BERT and seamless integration alongside the creation of a chat application capable of determining the toxicity of text in group chats. This system aims to identify and prevent the use of hateful or offensive language, thereby mitigating the severe consequences of cyberbullying. It does so by alerting users to problematic content, raising awareness, and implementing temporary restrictions or expulsion from chats for repeat offenders. Furthermore, the project introduces the integration of a buffer system in the front end for contextual analysis, addressing an expected shortfall in current abuse detection systems, which often misinterpret the context, wrongly labelling non-toxic words as toxic. The effectiveness of this proposed solution will be assessed through a comparative analysis with similar research endeavours. This study has a great deal of practical significance and is well-positioned to act as a basis for further research on the identification of cyberbullying.

2. Literature Review

2.1 Overview of OSN (Online Social Networking) and the rise of Abuse

Online Social Networks have dramatically revolutionised the way individuals interact and communicate with each other. Social platforms like Twitter, Snapchat, Facebook, TikTok and Instagram have expanded social interaction beyond physical limits, enabling connectivity on a global scale (Kaplan and Haenlein 2010). These networks are now essential to modern social life, serving as venues for personal expression, sharing information and building communities (Boyd and Ellison 2007).

However, the widespread use and anonymity offered by these platforms have also led to a crucial issue, which is the ongoing increase in online abuse. This encompasses cyberbullying, hate speech and various types of harassment. Smith et al. (2008: p. 376) described the term *cyberbullying* as "an aggressive, intentional act carried out by a group or individual using electronic forms of contact, repeatedly and over time against a victim who cannot easily defend himself or herself." Their definition articulates the ongoing and extensive nature of online abuse.

The prevalence of Online abuse is escalating, with numerous studies indicating a worrying rise in such incidents. A Pew Research Center survey revealed that around 41 percent of Americans had faced some version of harassment online (Duggan 2017). The nature of this abuse varies, ranging from insulting name-calling and intentional embarrassment to stalking, physical threats and sexual harassment. One of the survey results illustrated in Figure 2 shows an increase in various forms of online harassment that Americans have faced over the years.

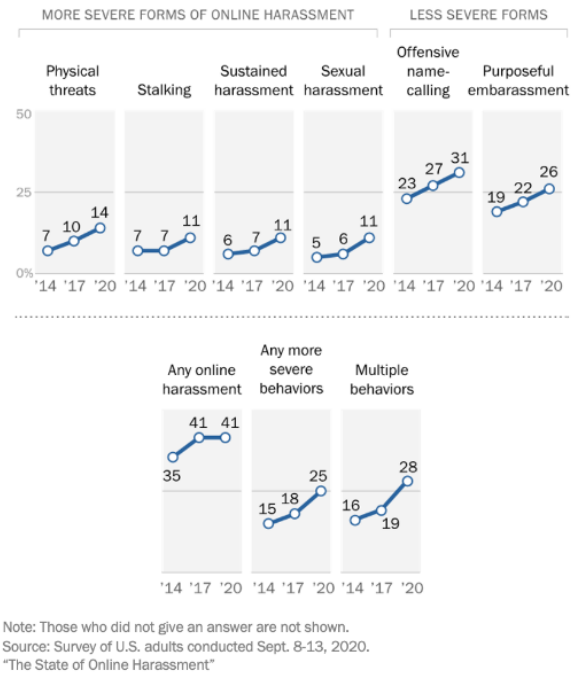


Figure 2 - Results from survey, taken from (Pew Research Center 2021)

The anonymity provided by most Online Social Networks is what often encourages these users to partake in such actions without fearing immediate consequences. The effects of online abuse are profound and varied, with victims frequently suffering considerable emotional and psychological harm. Hinduja and Patchin (2010) observed that individuals subjected to cyberbullying often show heightened symptoms of depression, anxiety and loneliness, which can have may have long-lasting effects on their general wellbeing, self-esteem, and mental health. Nixon (2014) further emphasised this, noting the severe and lasting psychological effects of such abuse on people, particularly teenagers, leading to grave depression and, in many cases, suicidal thoughts and a multitude of other profound mental health conditions resulting from their online interactions.

Viewed from a societal angle, online abuse can create an environment of intolerance and aggression, eroding the sense of unity and belonging that these OSNs aims to foster. This issue is a significant barrier to the inclusive and respectful dialogue that is crucial for the health of democratic societies (Citron 2014). Solidifying a precise definition of abuse is challenging, as varying standards across online communities influence what is deemed abusive, a complexity highlighted by Chandrasekharan et al. (2018). Definitions of different forms of abuse often overlap and lack clarity. However, in its broadest sense, abuse is identified as any act intended to belittle or harm a specific individual or group (Mishra et al. 2020).

Although these Online Social Networks have positively reshaped how society interacts, the increasing occurrence of online abuse poses a significant challenge. Tackling this intricate problem necessitates a comprehensive approach that incorporates legal, educational and technological strategies. The subsequent sections will explore the complexities and awareness surrounding online abuse, examine the problem domain, discuss methods to counteract abuse and explore how traditional and contextual machine-learning techniques can be employed for abuse detection.

2.2 Challenges and Awareness of Online Abuse

2.2.1 Challenges

Navigating the ever-changing world of Online Social Networks presents significant difficulties in managing and moderating online interactions. The sheer amount of content users produce, and its dynamic nature makes manual supervision unfeasible. This reality has led to the need for automated tools for content moderation and identifying abusive behaviour (Talukder 2019). However, these systems encounter complex challenges due to the nuanced and context-dependent nature of human communication, an extension of the issues previously noted by Chandrasekharan et al. (2018).

The intricacy and diversity of abusive language poses a major challenge to automated abuse identification. Nobata et al. (2016) underscored the fact that abuse can be highly context-dependent, often requiring insights into cultural and situational nuances that automated systems very commonly miss. This complexity makes it hard for these systems to accurately distinguish harmful content from harmless, often leading to incorrect flagging and oversight. Furthermore, creating annotated datasets to train machine learning models is a formidable task. These datasets need extensive resources to compile and are susceptible to biases, which can impact the effectiveness and impartiality of the models developed, as discussed by Waseem and Hovy (2016).

The continuous evolution of online language and tactics used in abuse adds another layer of complexity to content moderation. Abusers frequently change their language to bypass detection, employing tactics like code words or modified spellings, a phenomenon called adversarial stylometry (Gitari et al. 2015). Moreover, Davidson et al. (2017) highlighted the fine line between hate speech and derogatory language in digital environments, further complicating the task of content moderation. Automated systems must delicately navigate this grey area to efficiently identify and mitigate abuse while carefully not violating peoples free speech rights.

Additionally, given that language is a central factor in modelling abuse, the primary focus has historically been on English, with subsequent attention to German, Hindi, and Dutch. However, recent initiatives have been made to develop datasets in other languages, including Arabic (Mubarak et al. 2017) and Chinese (Su et al. 2017). Hinglish (Mathur et al. 2018a) has also been seen to be used by a few researchers, and this project will also utilise this type of dataset (Mishra et al. 2020). This expansion into multilingual datasets mirrors the global presence of online abuse and underscores the necessity for systems capable of understanding and moderating content within various linguistic contexts. This was also crucial in determining the project's primary focus. Consequently, the project was aimed at identifying abuse not only in English but in Hindi as well. Various significant studies that will be discussed in the later sections of this report, support the reasoning for this decision. Jhaveri, Ramaiya and Chadha (2022) discussed the challenges in detecting abusive comments in Indian languages, especially Hindi. They used sophisticated models like ‘XLM-RoBERTa’ and ‘MuRIL’ to

highlight the importance of considering language specifics and the value of using transliterated data in recognising abuse. In addition, Gupta et al. (2022) developed 'MACD', a detailed human-verified dataset for Indic languages. This dataset and the innovative 'AbuseXLMR' model illustrate the feasibility of constructing effective multilingual abuse detection models, particularly for commonly spoken languages such as Hindi. Moreover, a Machine Learning approach was presented by Chopra et al. (2023) to identify instances of hate speech via written content posted on social media platforms. that combine Hindi and English coding (code-mixed texts), employing Facebook's fastText for efficient feature extraction. Their research reinforces the strategy for bilingual abuse identification, highlighting the necessity for specialised techniques in dealing with code-mixed languages.

2.2.2 Awareness

Recent studies have significantly increased awareness and thus resulted in the development of methods to detect online abuse. As mentioned earlier, Duggan (2017) conducted an extensive study that shed light on the widespread nature of online harassment, emphasising the importance of efficient and effective abuse detection and prevention strategies. This heightened awareness has also spread towards educational efforts, such as those discussed by Livingstone et al. (2011), which strive to educate and empower users, particularly children and young teenagers, about responsible internet usage and the lasting effects of their digital footprint.

In addressing these issues, Kumar, Cohen and Golab (2019) introduced an innovative attention-based Neural Network technique for identifying abusive language on OSNs. Their work demonstrated the effectiveness of using contextual and semantic analysis in identifying toxic content. Similarly, the comprehensive survey by Mishra et al. (2020) in NLP presented various computational strategies and datasets employed in abuse detection. Their research analysed the strengths and limitations of these methods and suggested guidelines for ethical considerations and explainability in this context. Furthermore, Talukder (2019) explored the issue of friend abuse on networks like Facebook and introduced two innovative solutions, 'AbuSniff', which aims to detect and prevent abuse from existing Facebook friends, and 'Flock', a method designed to avert abuse during the process of friend invitations.

Amnesty International's investigation into online abuse against women, conducted through an IPSOS MORI poll, revealed that nearly one in four women across eight countries had experienced online abuse or harassment, with 41% feeling that their physical safety was at risk (Amnesty International 2017). These findings highlight the tangible consequences of digital abuse and the critical need to address this issue. The ethical aspects of creating and implementing automated abuse detection systems are also gaining vast attention. As emphasised by Binns (2017), these systems must be equitable, transparent and allow users to challenge decisions. This approach ensures that efforts to combat online abuse do not infringe upon individuals' privacy and right to free speech. European Commission (2016) unveiled a Code of Conduct aimed

at combating unlawful online hate speech, which is a noteworthy instance of legislative action directed at addressing online harassment.

Furthermore, authors such as Jane (2020) have stressed the importance of collective actions and the establishment of community norms in the digital space to counteract abuse. These strategies involve not just technological solutions but also shifts in societal and cultural attitudes to tackle the underlying reasons for online abuse. Although progress has been achieved in increasing awareness and creating solutions, effectively handling online abuse demands continuous dedication, innovation and collaboration across various sectors.

2.3 Problem Domain

The challenge of detecting bilingual abuse is particularly complex and unique, mainly because of the inherent linguistic and cultural diversity in bilingual or multilingual interactions. Although most current research on automated abuse detection concentrates on English, the international scope of OSNs requires efficient solutions for various languages.

The varied linguistic landscape in OSNs presents a considerable obstacle for abuse detection systems. The study by Toliyat et al. (2022) into the identification of hate speech targeting Asians on Twitter during the COVID-19 pandemic illustrated that these systems often perform less effectively in languages other than English due to the absence of extensive datasets and unique language characteristics. Unlike English, numerous languages do not have detailed annotated datasets essential for training machine learning models for precise abuse detection. Furthermore, the complexity of interpreting context increases in multilingual settings. As Fortuna and Nunes (2018) highlighted, the complexities of cultural context, slang, and idiomatic phrases significantly differ across languages, creating challenges for models trained on data from a single linguistic background. This issue becomes even more pronounced in bilingual environments where code-switching, or alternating between two or more languages, is frequent.

The complexity in detecting bilingual abuse involves accurately recognising abusive content in various languages while considering their contextual and cultural subtleties. Sanguinetti et al. (2018) pointed out the challenges in identifying nuanced forms of abuse like irony and sarcasm, which can differ significantly across languages. The continuous evolution of online language, marked by new slang and code words, further complicates this task. Machine translation could be a viable approach, but as Vidgen and Derczynski (2020) suggested in their comprehensive review of abusive language training data, directly translating text from one language to another for abuse detection risks losing context and causing misinterpretations, particularly in languages with a smaller online footprint. Their research underscores the need for large, diverse, and theoretically grounded training datasets to minimise biases in effective abuse detection systems. They also delve into the challenges of compiling such datasets and discuss the social and ethical implications of sharing

and using these datasets. This includes recognising the complexities and risks associated with managing diverse linguistic data in the realm of abuse detection. This approach faces difficulties with colloquial language and new terms frequently used in online conversations. Technically, developing robust bilingual abuse detection systems requires overcoming challenges like scarce annotated data and crafting advanced algorithms capable of grasping context in multiple languages. It can also be argued that collecting and analysing text data for abusive content raises privacy concerns, especially if user data is used without consent.

The works of Badjatiya et al. (2017) using DL on hate speech detection in twitter comments (tweets) shows potential, but these models need a wide array of training data. Their thorough experiments with 16,000 annotated tweets revealed that deep learning techniques, including FastText, CNNs, and LSTMs, were significantly more effective than traditional character/word n-gram approaches. However, this challenge of resource-intensive training must be considered when furthering the development of this project, as it will be costly due to the requirement of computational power from advanced GPUs, which necessitates the purchase of compute units. The study highlights the capability of deep learning to accurately classify tweets as racist, sexist, or neither, using complex natural language constructs and sophisticated machine learning frameworks. From an ethical point of view, the creation of bilingual abuse detection systems must take into account fairness and bias. Davidson et al. (2019) emphasised that algorithms might unintentionally reinforce biases if they are not meticulously crafted and evaluated, especially in multilingual settings where cultural sensitivities differ. Balancing the detection of abusive language with the protection of people's right to freedom of speech also requires careful consideration to avoid censoring legitimate expressions or opinions.

2.4 Anti-Abuse Approaches

Numerous research studies have been carried out to develop effective strategies for dealing with bilingual abuse. Early stages of automated abuse detection utilised traditional ML methods such as SVMs (also known as Support Vector Machines), Logistic Regression, Naïve Bayes, Tree based algorithms like Random Forests and Decision Trees. These techniques relied on manually created features like keyword filters and heuristic rules, establishing a fundamental understanding of online content. However, as Burnap and Williams (2015) noted, despite achieving a fair accuracy, these models had a significant shortcoming: their inability to completely grasp the subtleties and context of natural language. They primarily treated the text as an unstructured collection of words, overlooking the importance of word order and contextual subtleties, which are essential for recognising abusive content. Li (2017) also pointed out these limitations, emphasising their failure to comprehend the semantic interconnections within the text.

Introducing the concepts of Deep Learning has resulted in a major transformation in the realm of automated abuse detection, shifting focus to more sophisticated models that can discern deeper semantic meanings. This change is characterised by using CNN's (also known as Convolutional Neural Network) and RNN's (also known as Recurrent Neural Network), including the LSTM (also known as Long Short-Term Memory) networks. These advanced models, skilled in deciphering complex language patterns, provide improved performance compared to the traditional approaches. Zhang and Wallace (2017) underscored the efficiency of CNNs in detecting local, position-invariant features in text, surpassing previous methods in identifying abusive content. Their analysis of CNNs in sentence classification tasks demonstrated the models' proficiency in managing a variety of linguistic subtleties. This is attributed to their ability to learn from extensive data sets without depending on manually created rules. Similarly, Badjatiya et al. (2017) demonstrated the efficacy of deep neural networks in recognising patterns of abusive language within social media texts. CNNs, known for their extraordinary pattern recognition abilities, were tailored for analysing textual data, successfully capturing local dependencies in the text. Furthermore, RNNs, especially LSTM networks as investigated by Zhou et al. (2016), significantly improved the processing of longer sequences. This advancement was pivotal in addressing the sequential aspect of language, facilitating a more nuanced comprehension of context in sentences and paragraphs, which is vital for detecting abusive language.

The introduction of contextual DL models like BERT (Bidirectional Encoder Representations from Transformers) and similar transformer-based architectures also marked a significant advancement in the field. As explained by Devlin et al. (2019), BERT represents a major breakthrough with its profound bidirectional nature, enabling it to understand the complete context of a word by examining the surrounding words. This feature is particularly vital in abuse detection, where the whole meaning of a word can be entirely altered by its context. BERT employs two principal pre-training tasks: the masked language model (also known as MLM) and next sentence prediction (also known as NSP). The MLM task aids the model in learning contextual

relationships between words by predicting tokens that are randomly masked in a sentence. Meanwhile, NSP trains the model to comprehend the relationship between two sequential sentences, a critical skill for applications like question answering and natural language inference. This project will utilise the BERT model and its accompanying Tokenizer, leveraging its ability to capture complex linguistic patterns and contextual nuances. The integration of BERT into abuse detection has notably enhanced the precision and efficiency in identifying harmful content. BERT's capability to deal with language ambiguity and complexity distinguishes it as an effective instrument for moderating online platforms. The study conducted by Vidgen and Derczynski (2020) illustrated BERT's effectiveness, showcasing its ability to differentiate various types of offensive language, including hate speech, with greater accuracy compared to earlier models.

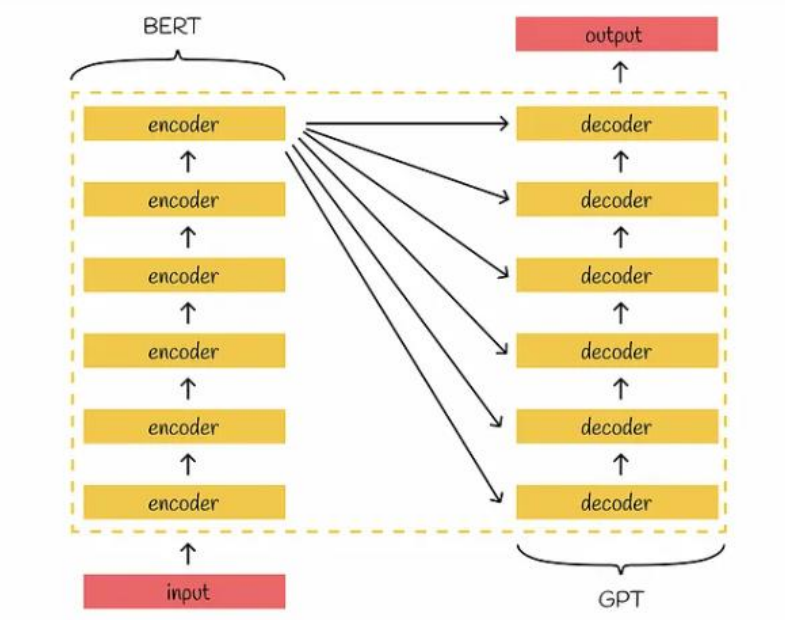


Figure 3 - Transformer Architecture (Efimov 2023)

Despite the overall technological advancements, the field of abuse detection, particularly in bilingual settings, still faces significant challenges. A primary concern, as discussed by Vidgen and Derczynski (2020) and Sap et al. (2019), is the need to balance the effectiveness of these models in spotting abusive content while not violating the notion of free speech. Another critical issue is the inherent biases present in these models, which mirror the prejudices found in their training data. As Binns (2017) also stated, it is crucial to ensure fairness, transparency, and accountability in these automated systems, given the ethical implications of automated moderation.

To address these challenges in abuse detection, there is a growing support for a collaborative and multi-modal approach. This strategy encompasses not only technological solutions but also societal and educational initiatives. For example, Salminen et al. (2020) proposed the integration of text analysis with additional data types, like user behaviour patterns, in order to improve the efficiency of detection systems. This cooperative

endeavour, which involves technology companies, policymakers, educators, and the broader community, is crucial for fostering safer and more inclusive online spaces.

In further advancing the field, some researchers have investigated the adaptability of contextual models like BERT in varied linguistic and cultural contexts, which is vital for bilingual abuse detection. Sun et al. (2019) showcased BERT's effectiveness in managing multilingual datasets, an important aspect for its application in diverse linguistic environments. The model's capability to perform well in various languages stems from its pre-training on large and varied text corpora. This adaptability makes BERT an excellent option for projects concentrating on bilingual or multilingual abuse detection.

Expanding on that, creating and applying manually curated datasets have become essential in differentiating abusive from non-abusive content and training computational models. These datasets, generally filled with extensive text from online exchanges on OSNs, are the critical resources for crafting and enhancing tools designed to detect and prevent online abuse. For example, Nobata et al. (2016) developed four datasets using comments from Yahoo! News and Finance, labelling each as “abusive” or “clean”. There has also been significant research on abusive language on Twitter. Waseem and Hovy (2016) compiled a substantial collection of tweets, categorizing each as “racism”, “sexism”, or “neither”. Davidson et al. (2017) created a dataset of around 25,000 tweets, each manually classified as “racist”, “offensive but not racist”, or “clean”. However, Mishra et al. (2020) pointed out that their data collection method, which relied on the presence of specific abusive words, did not accurately represent the proper distribution of classes in real life. Figure 4 below provides a detailed list of some of these datasets, and their corresponding research papers and languages.

Dataset link	Associated paper	Language	Size	Source	Composition
DATA-TWITTER-WH	(Waseem and Hovy, 2016)	English	17k	*Twitter*	*Racism, Sexism, Neither*
DATA-TWITTER-W	(Waseem, 2016)	English	7k	*Twitter*	*Racism, Sexism, Both, Neither*
DATA-TWITTER-DAVID	(Davidson et al., 2017)	English	25k	*Twitter*	*Racist, Offensive, Clean*
DATA-TWITTER-F	(Founta et al., 2018)	English	80k	*Twitter*	*Abusive, Spam, Hateful, Normal*
DATA-WIKI-ATT	(Wulczyn et al., 2017b)	English	116k	*Wikipedia talk page*	*Personal attack, Clean*
DATA-WIKI-AGG	(Wulczyn et al., 2017b)	English	116k	*Wikipedia talk page*	*Aggressive, Clean*
DATA-WIKI-TOX	(Wulczyn et al., 2017b)	English	160k	*Wikipedia talk page*	*Toxic, Clean*
DATA-FOX-NEWS	(Gao and Huang, 2017)	English	1.5k	*Fox news*	*Hateful, Non-hateful*
DATA-GAZZETTA	(Pavlopoulos et al., 2017b)	Greek	1.6M	*Gazzetta*	*Accept, Reject*
DATA-FACEBOOK	(Kumar et al., 2018)	Hindi & English	15k	*Facebook*	*{Covertly, Overtly, Non}-aggressive*
Arabic News	(Mubarak et al., 2017)	Arabic	32k	*Aljazeera News*	*Obscene, Offensive, Clean*
GermEval 2018	(Wiegand et al., 2018c)	German	8.5k	*Twitter*	*Abuse, Insult, Profanity, Other*
Ask.fm	(Samghabadi et al., 2017)	English	5.6k	*Ask.fm*	*Invective, Neutral*

Figure 4 - Summary of datasets (Mishra et al. 2020)

Besides BERT, other transformer-based models such as ‘RoBERTa’ (also known as Robustly Optimized BERT Pretraining Approach) and ‘GPT’ (Generative Pretrained Transformer) have also been examined for their effectiveness in abuse detection. Liu et al. (2019) examined the improvements made in RoBERTa, which refines the training process of BERT and enhances its performance in Natural Language Processing tasks. In a similar vein, Radford et al. (2019) introduced ‘GPT-2’, demonstrating its advanced capability in

understanding context and generating human-like text. This contributes significant insights into the subtleties of language usage in abusive content, highlighting the model's potential in this domain.

Researchers have also begun exploring the application of multi-task learning in Neural Networks for the specific purpose of identifying abusive behaviour. The research conducted by Rajamanickam et al. (2020) revealed that training Neural Networks on both emotion classification and abuse detection tasks concurrently increases their efficacy in detecting abusive content. Recognising the emotional tone of comments, whether it be disgust, anger, joy, fear, or optimism, has been shown to improve the accuracy of abuse detection in Twitter posts. This approach signifies the importance of understanding the emotional context in effectively identifying abusive behaviour online.

Furthering from there, scholars have started to include user qualities and identification variables into their models to forecast the likelihood of abusive conduct among users who possess specific traits. This technique is called ‘user-profiling’. In order to identify cyberbullying, Dadvar et al. (2013) included user age in addition to conventional lexicon-based criteria. In a similar vein, Galán-García et al. (2016) enhanced their study by considering variables including the location, time of publication, and language used in Twitter user profiles. This method of including users' personal characteristics indicates a more thorough approach to spotting possible internet abuse.

The technique of Zero-shot learning (also known as ZSL) with prompting is emerging as a notable method for detecting hate speech, offering a solution to the limitations of conventional supervised learning methods. Plaza-Del-Arco, Nozza and Hovy (2023) pointed out the effectiveness of this approach, especially for languages with scarce labelled data. This method utilises large language models and infuses task-specific knowledge through prompting, enabling the detection of hate speech without the need for extensive labelled data. The study underlines the significance of prompt selection on outcomes and indicates that recent large language models have the capability to match or even surpass the performance of fine-tuned models. This makes it a viable option for languages with fewer resources. Furthermore, the authors demonstrate how language models that have been fine-tuned can be trained to achieve state-of-the-art performance on a variety of benchmarks without the need for specialised dataset fine-tuning. Consequently, Zero-shot learning with prompting emerges as a valuable strategy in combating abuse, particularly in detecting hate speech across diverse linguistic environments.

2.5 Notable Related Works

In the space of cyberbullying detection, various studies have already been conducted to identify abuse in English. However, the detection of abuse in Hindi faces significant challenges due to language complexities and a scarcity of available datasets. Hindi, with over 600 million speakers worldwide, stands as the third most spoken language. It is also recognised as one of the 22 official languages of India, in a nation that ranks as the world's most populous to date. According to Shewale (2024), India is also ranked second in the world in countries that have the most social media users, with a number of 755.47 million users, predicted to reach 1.2 billion by 2027. Beyond India, Hindi is also used in several other countries, including Nepal, the USA, Mauritius, and more. This makes Hindi a secondary focus for this project, alongside English. The project primarily deals with detecting bilingual online abuse in English and Hinglish (also called Roman Hindi). Subsequently, additional research was undertaken to explore significant methods in this area. Sameer (2022) addressed the pressing matter of hate speech on social media, particularly analysing a blend of English and Hindi-English tweets from Twitter. Their approach involved classifying hate speech using character-level embeddings of tweets and DNN (also known as Deep Neural Networks). They developed two frameworks: a CNN and a hybrid of CNN and LSTM algorithms, enhancing previous methods with character-level embedding. They trained their models on an imbalanced (original) and an oversampled (balanced) version of the training data, testing them to achieve an 88.97% accuracy rate. The outcomes of their study underscore the potential effectiveness of deep learning models in identifying hate speech in mixed English and Hindi-English tweets, offering important contributions to the field of NLP and ML, including approaches like BERT, for bilingual online abuse detection.

In addition, Nafis et al. (2023) presented a novel dataset consisting of 10,000 tweets in English and Hindi-English code-mixing. The dataset includes manual annotations for identifying aggressive and derogatory language. They meticulously optimised multiple pre-trained language models using this dataset to address the issue of cyberbullying. The researchers improved several models, namely BERT, RoBERTa, XLM-R, HingRoBERTa, and Bernice. They provided the macro F1-scores for the total dataset, as well as for the code-mixed and non-code-mixed parts individually. The HingRoBERTa model, specifically developed for Hindi-English code-mixed language, outperformed other models in the code-mixed area. The authors also performed both quantitative and qualitative evaluations of inaccurate forecasts, offering significant insights for future investigations. The BERT model demonstrated an accuracy of 60.99% for monolingual text, 61.94% for code-mixed text, and 62.05% overall in recognising offensive language.

Mathur et al. (2018b) achieved a significant breakthrough in the field of bilingual online abuse detection by employing NLP and ML techniques. The researchers created the Multi-Input Multi-Channel Transfer Learning-based model (MIMCT) to detect offensive Hinglish tweets. This model outperformed existing classification models and established a new standard in this domain. The work utilised multiple primary word

embeddings, such as Glove, Twitter word2vec, and FastText, along with secondary features, to train a multi-channel CNN-LSTM architecture. This architecture was enhanced using transfer learning, as it was previously trained on English tweets. The MIMCT model outperformed typical supervised classification models and demonstrated superiority over previous transfer learning-based CNN and LSTM models. This highlights its efficacy in addressing the intricacies of objectionable text categorisation in Hinglish. The research highlights the potential of transfer learning in classifying offensive Hinglish content, aligning with the usage of Deep Learning models such as BERT in detecting bilingual online abuse. This highlights the significance and efficiency of deep learning methods in handling the complexities of multilingual social media information. The most successful model they used was a Support Vector Machine that was improved with the TF-IDF technique, resulting in an F1-score of 72.3%.

Used as inspiration for this project, Shah, Phadtare and Rajpara (2022) developed a model for detecting cyberbullying in English and Hinglish also using NLP and ML techniques. They gathered real-time data from Twitter, WhatsApp chats and YouTube comments in both languages. During preprocessing, they eliminated unnecessary characters, hashtags, numeric data, and stopwords. They employed NLP techniques like tokenisation, lemmatisation, and vectorisation to ready the data for analysis. Various machine learning algorithms were trained, showing promising accuracy and speed, mainly when used with a count vectorizer for feature selection. The study found that Linear SVC and SGD (Stochastic Gradient Classifier) were more effective in classifying and predicting bullying messages in Hinglish. They required less time for training and prediction compared to other algorithms. The most successful model in their research was Linear SVC, which achieved a high accuracy rate of 0.95% in identifying cyberbullying messages in Hinglish. However, a fundamental limitation of their approach was its reliance on traditional machine learning methods, which might not fully grasp the intricate linguistic nuances of Hinglish. This limitation presents an opportunity for enhancement through advanced DL models like BERT, which are more adept at capturing text’s contextual and semantic meanings in bilingual datasets. Therefore, while traditional methods showed promising outcomes, integrating BERT into this project could overcome these limitations and potentially enhance the accuracy and efficiency of bilingual online abuse detection.

To further justify the use of BERT, more research was conducted in order to see other scholars' outcomes. The study by Saini et al. (2023) employed BERT to identify cyberbullying on online platforms, achieving remarkable accuracy rates of 96.88% and 97.34% on two distinct datasets. These findings highlight BERT's ability to effectively differentiate between toxic and non-toxic content. Paul and Saha (2020) also made a significant contribution by employing BERT for cyberbullying detection, achieving state-of-the-art results across various real-world data sources such as Formspring, Twitter, and Wikipedia. This demonstrates the model's adaptability across different online communication contexts. Additionally, Caselli et al. (2021) introduced an iteration of BERT, termed HateBERT, designed explicitly for detecting abusive language. This

model, trained on a comprehensive dataset of Reddit comments from banned communities, showcases BERT's flexibility in identifying targeted abusive language, further reinforcing the potential of BERT in this domain.

Complementing these studies, previous research has also employed BERT Base Uncased, a model variant that processes text in lowercase, to detect cyberbullying instances. Most recently, Parikh and Dalvi (2024) demonstrated the effectiveness of this approach, revealing that BERT Base Uncased achieved a test accuracy of 85.81% and an F1-score of 0.8566. This performance surpassed other transformer models such as DistilBERT-Base-Uncased, ELECTRA, and MobileBERT-Uncased. The study highlights the superiority of this version of BERT over traditional machine learning algorithms, particularly in its capability for real-time detection of cyberbullying.

2.6 Conclusion of Literature Review

The performed study provides a thorough framework for tackling the increasing problem of online abuse. It provides a comprehensive analysis that examines the growing importance and worldwide acknowledgement of online abuse. It also addresses the various problems associated with this issue, including psychological and technical factors contributing to its widespread occurrence. It provides a comprehensive analysis of the several strategies devised to combat abuse in the modern era of technological progress. One major issue is the limited availability of precisely labelled information and the intricacies of linguistic subtleties. The research emphasises that although many studies concentrate on identifying online abuse, mainly in English, because of its extensive usage, there is a significant disregard for languages other than English. In order to address this gap, the suggested approach entails identifying and flagging offensive language in both English and Hindi-English. This project plans to extend the work of Shah, Phadtare and Rajpara (2022), by using their dataset as a foundation for creating an anti-abuse model, utilising BERT as the deep learning model of choice. This approach leverages the strengths of BERT in language processing to enhance the effectiveness of online abuse detection. Upon completing the training process, this model will be incorporated into a real-time chat platform specifically created to simulate everyday interactions in online social networks. Its purpose will be to detect and prevent instances of online abuse. A novel contribution in the form of a dynamic buffer in the front end will also be introduced. The subsequent segments of this research will delve into the crucial components of formulating this solution.

3. Analysis and Requirement Specification

3.1 Analysis

The thorough research undertaken has validated the project's legitimacy and yielded valuable insights into the two primary methodologies: machine learning and deep learning. Before settling on the project's defining approach, a critical analysis will assess each approach, delineating their strengths, weaknesses and suitability in addressing the complexities of language and cultural distinctions.

Historically, again having been previously reviewed, text analysis and abuse detection have depended on traditional ML algorithms. These models are based on statistical learning and require manual extraction of features from text data. The feature extraction process in ML is both creative and methodical, requiring deep domain expertise to pinpoint relevant features such as word frequencies, n-grams, and syntactic patterns (Gao and Huang 2017). One significant benefit of ML models is their interpretability. For instance, decision trees provide clear insight into how decisions are made, which is particularly vital in sensitive areas like abuse detection, where understanding the rationale behind a classification is crucial (Ribeiro, Singh and Guestrin 2016). Moreover, ML models are often acclaimed for their training and prediction efficiency, using significantly less computational power than their Deep Learning counterparts (James et al. 2013). This efficiency makes them well-suited for situations with limited computational resources.

Nevertheless, the dependence on manual feature extraction is a major limitation of ML models. This approach is often time-consuming and may fail to encompass all linguistic subtleties, especially in bilingual settings where cultural and contextual elements are crucial. Additionally, conventional ML models frequently have difficulty comprehending context and long-range dependencies in language. This shortcoming is especially apparent in abuse detection, where the meaning and implication of words can be completely altered by the context of their use (Lai et al. 2015).

In contrast, Deep Learning marks a significant shift in NLP compared to traditional methods. BERT has been groundbreaking in its capacity to contextualise each word in a sentence by looking at the words that come before and after it (Devlin et al. 2019). This ability to understand context is vital for accurately identifying nuanced and context-specific abuse, particularly in bilingual environments. An additional key advantage of BERT and other Deep Learning models is their capability for transfer learning. With training on extensive text datasets, BERT can be fine-tuned using smaller datasets while maintaining high performance, a benefit especially relevant in specialised fields like abuse detection (Sun et al. 2019). Furthermore, Deep Learning models excel in processing complex patterns in data, which is crucial for detecting subtle and indirect forms of abuse that simpler models might miss.

However, DL models present their own challenges. They demand considerable computational resources for training and inference, which may be a constraint in some situations (Strubell, Ganesh and McCallum 2019). This certainly poses as a project constraint, as personal funding is required to purchase the additional computational units needed to run the faster GPUs imperative for effective model training. Additionally, DL models often face criticism for their lack of interpretability. The intricate interactions of layers and neurons in models like BERT make it hard to trace and comprehend the decision-making process, a crucial issue in fields where accountability is vital (Samek, Wiegand and Müller 2017). Moreover, despite the benefits of transfer learning, DL models typically need large datasets to achieve their optimum performance.

Although ML models have strengths in terms of interpretability and computational efficiency, the intricate and subtle nature of language in abuse detection necessitates a more advanced approach, one that models like BERT uniquely offer. Its adeptness in managing complex language patterns and the efficiency gained through fine-tuning renders it an appropriate option for this project despite the hurdles of high computational resource demands and reduced interpretability.

ML and DL exhibit notable differences in processing large amounts of data. Machine learning algorithms often perform well with smaller datasets, frequently requiring human manipulation of features. In contrast, deep learning models, especially those using neural networks like BERT for natural language processing, have more proficiency in handling extensive datasets. These deep learning models benefit from their natural ability to autonomously identify complex patterns in data, a characteristic that increases with more extensive data sets. Considering the large dataset used in this project, it is suitable to use a DL approach due to its inherent ability to analyse and derive insights from data autonomously. BERT excels at managing large amounts of data and surpasses traditional machine learning approaches in terms of both language comprehension complexity and efficiency. It does this by using pre-existing knowledge during training (Alaskar and Saba 2021).

Additionally, addressing the challenge of real-time context analysis in a server-client chat system is also essential. While BERT is proficient in understanding context, its static nature after training hinders its ability to assess evolving chat contexts. To address this, implementing a dynamic buffer system in the server side of the GUI is proposed. This system aims to maintain a dynamic history of recent messages for each user, serving as a short-term memory of 5 minutes, thus providing the trained BERT model with a continuously updating context for analysis. This strategy helps to counterbalance the binary nature of the dataset (texts labelled as toxic or not), which does not inherently include conversational context. By incorporating a dynamic buffer system, BERT can be more effective in analysing sequences of messages, enhancing abuse detection capabilities. This integration ensures real-time responsiveness and effectively navigates the dataset's constraints and the model's limitations.

3.2 Requirement Specification

- The Programming language used throughout the development of the product is Python due to its Programming power and adaptability.
- Google Colaboratory (known as Google Colab) is Online Cloud Service hosted by Google, that provides an environment for writing and executing Python code in separate blocks. This was chosen for the development of the classification model as it provides access to powerful GPU's that can be used to train Deep Learning models like BERT as they require additional computational power.
- An internet connection is also required to connect to the Google Colab cloud service.
- The selected IDE (Integrated Development Environment) for the project is Visual Studio Code, known for being versatile, efficient and its wide range of extensions and powerful debugging tools.
- Necessary libraries for the backend development such as pandas, numpy, matplotlib, seaborn, string, regular expressions, warnings, nlkt, sklearn and transformers, as well as necessary libraries for the frontend development such as os, socket, threading, tkinter, datetime, torch and collections for importing defaultdict and deque.

3.2.1 Functional Requirements

- The system must accurately detect abusive content in text messages. This involves analysing the text inputs and classifying them as toxic or non-toxic.
- It should also have support for bilingual support, so not just text in English but also understand the text in Roman Hindi and whether it is abusive or not.
- The final trained model should be integrated with the GUI, and housed in the server, so that when a message is sent between clients, it is passed through the model for identification before it is received by the other client. So, this also requires it to be processed in real-time.
- It requires a user-friendly GUI interface, which will be developed in VS Code, using Tkinter and sockets and threads, so as to allow users (clients) to send and receive messages.
- The GUI should be able to display whether a message has profanity and let the other client know that the message has been hidden.
- Through the use of the dynamic buffer, the system should detect the context of the conversation, thus it should not misclassify words which would normally be classed as abusive.
- The system should give out 2 warnings when abusive content is detected, and upon the third attempt, the client should be kicked out of the server.

3.2.2 Non-Functional Requirements

- Well commented sections of code, to help understand and differentiate each part of implementation. It is also standard good practice to have clean code. The programming language to be used is Python.
- The product should aim to maintain high accuracy in detecting abusive text, to minimise the false positives and false negatives. This means that training to be undertaken properly to achieve this.
- The GUI should be easy to navigate, with clear visuals such as buttons and text boxes, which would signpost the user where they can type and what to press to send a message.
- The response time should be minimal to ensure a smooth user experience in the real-time chat.
- The project should be compatible with different machines. Ensuring everything is kept under one place, and file names and locations are kept consistent, will ensure the product working seamlessly regardless of device.

4. Product Design

4.1 Description of Dataset

This project requires a dataset that includes English tweets and a combination of English and Hindi tweets. Furthermore, it is crucial to categorise these tweets as either hate speech or non-hate speech. Locating a publicly accessible dataset with only these characteristics has proven diligent. Therefore, to ensure the reproducibility of the experimental analysis, we used regularly utilised datasets in academic research. Upon conducting extensive study, a dataset was discovered in the publication authored by Shah, Phadtare and Rajpara (2022). The English dataset, created by the authors, was sourced from Kaggle and consists of real-time tweets from Twitter and communications retrieved from several OSNs. It contains a total of 15,307 rows. Regarding their Hinglish dataset, they manually collected tweets from Twitter and conversations from WhatsApp and YouTube comments. The dataset has approximately 3,000 rows. They combined them in order to obtain a more extensive dataset. The dataset is comprised of two columns, specifically labelled as 'text' and 'label'. The label is composed of -1 and 0 values, representing whether the content is harmful or non-toxic, respectively. The authors emphasise that this dataset contains a wide range of negative words that users frequently employ in their daily interactions. Nevertheless, it is evident that the dataset exhibits a significant bias towards English and lacks an equitable representation of Hindi remarks, hence constituting a constraint. Nevertheless, despite this drawback, the dataset was utilised in the project due to the scarcity of datasets available in this particular format and creating one manually would be a time-consuming endeavour.

Upon full analysis of the dataset, a few discrepancies were noticed. As shown in Table 1 below, the dataset was highly biased towards English text as the authors only put together 2,819 toxic comments for Hindi, and a staggering low number of 222 for non-toxic, compared to the 8,877 of toxic in English and 6,431 of non-toxic text.

Table 1 - Default Dataset Figures

Language	Hate Class	Number of Messages
English	Toxic (-1)	8,877
	Non-Toxic (0)	6,431
Hindi-English (Roman Hindi)	Toxic (-1)	2,819
	Non-Toxic (0)	222
Total		**18,349**

On attempting to rectify this, a few attempts at a solution were undertaken. The first was to try and use a Google translation API to try and translate the English text into Hindi and append it at the bottom of the dataset, however after further research, Google translation API's do not support Hindi in the Roman Hinglish version, it only supports the version of Hindi where the use of actual Hindi letters are used. The other solution was to use OpenAI's ChatGPT to generate records of text in order to fill in that gap of toxic and non-toxic. It was very time consuming however it proved to be successful. When prompted by the author, ChatGPT responded with a 'list of about 1200 generated conversational texts in Roman Hindi' (OpenAI ChatGPT 2024). This method seemed effective, however it is imperative to note that this text is generated by Artificial Intelligence, so it does not have the same quality of text that would be manually extracted from actual real conversations. This could hinder performance; however, it was the only solution left. Table 2 below shows the new figures after altering the dataset.

Table 2 - Dataset after manually adding more non-toxic Hindi records.

Language	Hate Class	Number of Messages
English	Toxic (-1)	8,877
	Non-Toxic (0)	6,431
Hindi-English (Roman Hindi)	Toxic (-1)	2,815
	Non-Toxic (0)	1,431
Total		**19,554**

[Conversation with OpenAI ChatGPT was deleted, however the specific prompt used was as follows: "*generate random roman Hindi sentences that I can use for a conversation in Hindi, remember it has to be written in english, like "deepak chahal itna bada hijda niklega kisko pata tha"*"]

4.2 GUI Design

Figure 5 below illustrates the logical interaction between the user and the GUI while Figure 6 shows the visual representation of the GUI.

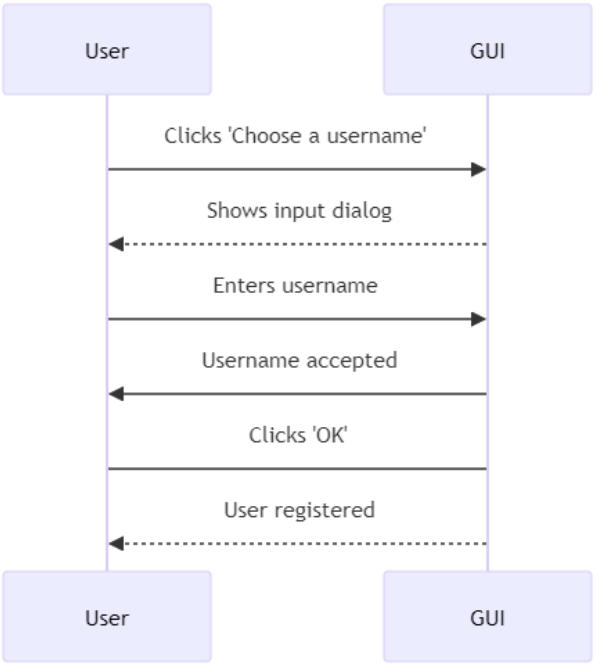


Figure 5 - Interaction between user and GUI

Figure 6 - Visual representation of GUI design

4.3 Sequence Diagram

Figure 7 below shows the Sequence Diagram which maps out the structure of how the model works in practice.

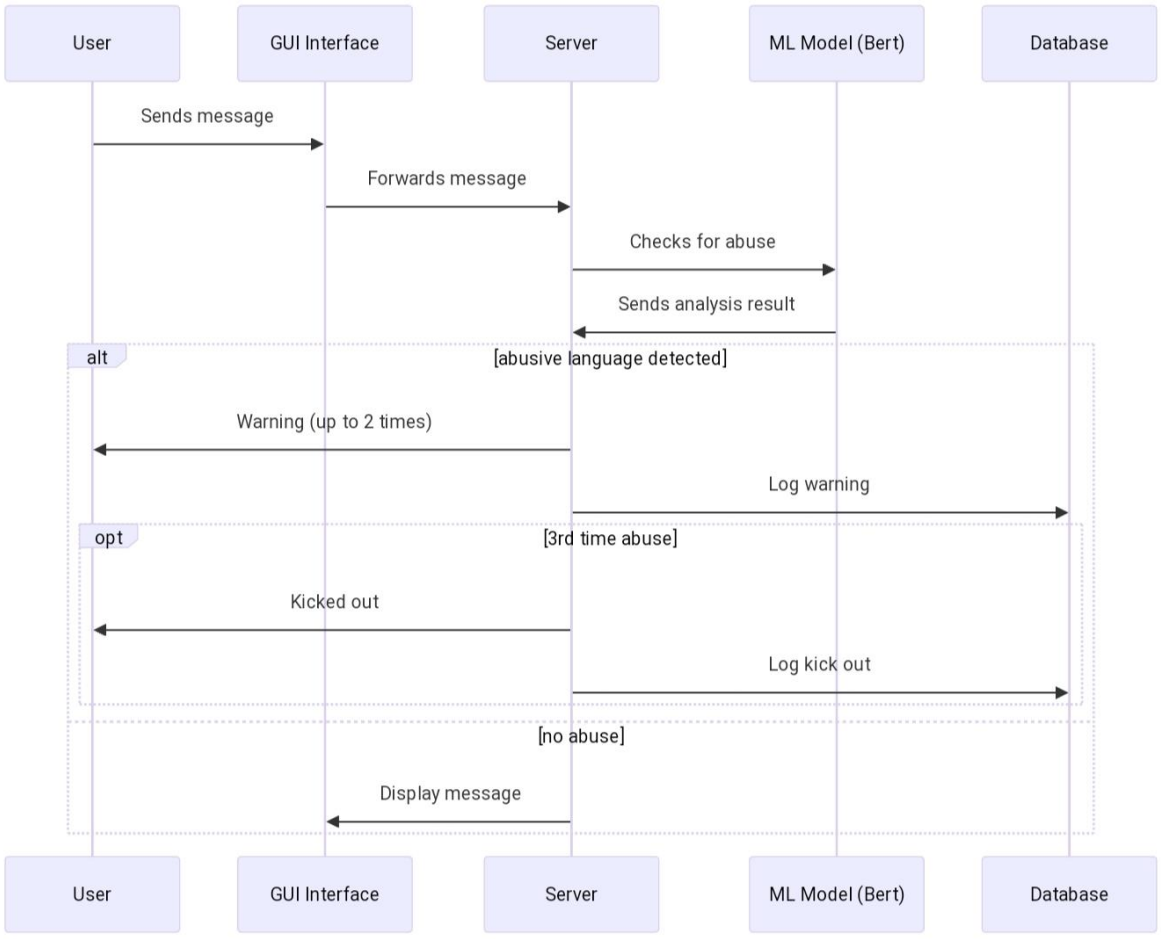


Figure 7 - Sequence Diagram

4.4 State Diagram

Figure 8 below illustrates a more practical understanding of how the product works.

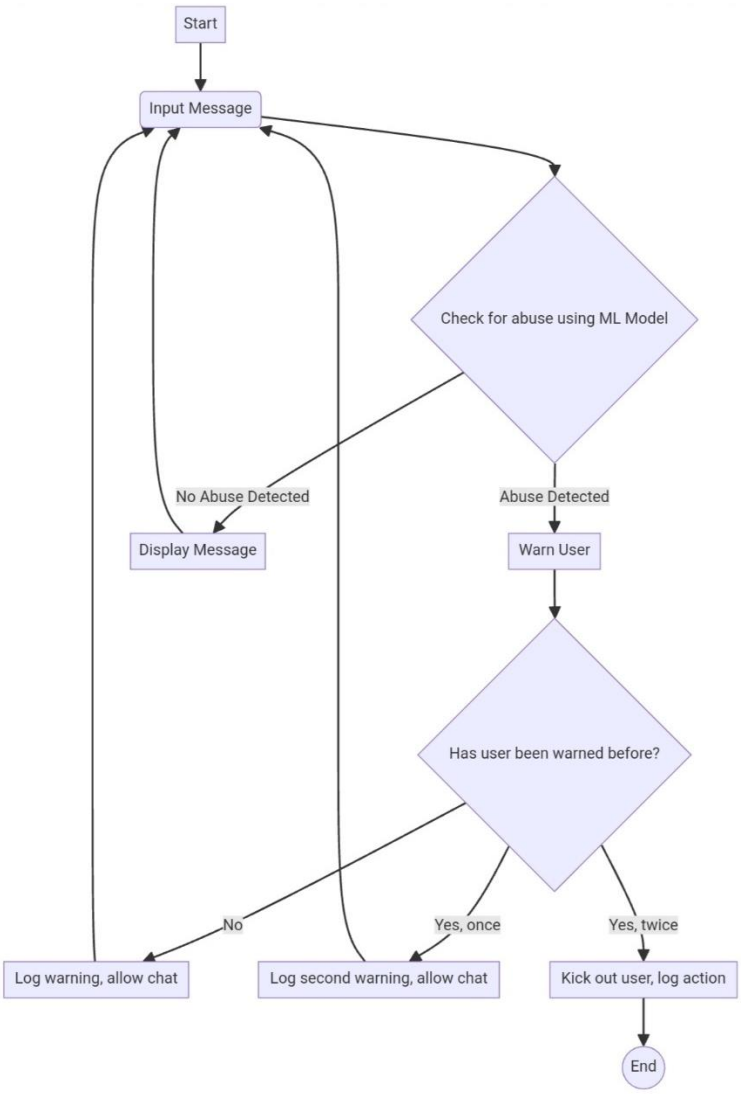


Figure 8 - State Diagram

5. Product Implementation

5.1 Backend

The backend of the project was coded in Google Colab. The next sections will discuss each process undertaken in the production of the final model. Source code used for the basis of the backend was taken from Nichite (2022).

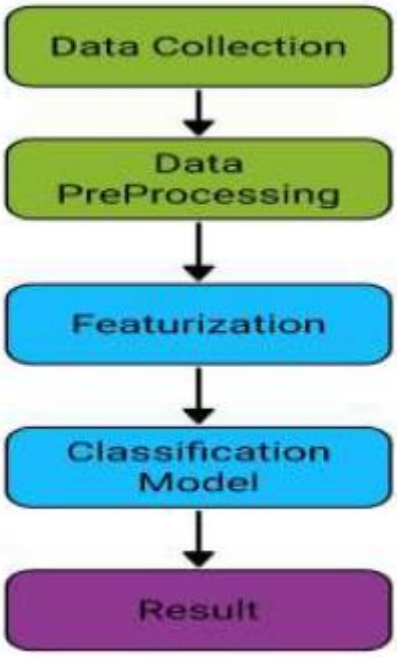


Figure 9 - General steps for cyberbullying detection (Khan and Bhat 2022)

5.1.1 Installing Libraries

The first step is to force install the required libraries such as “accelerate” for efficient hardware utilisation, “transformers” for accessing the BERT model and related functionalities and “scikit-learn” for general ML tasks. The Cuda cache is then cleared to ensure starting with as much available GPU memory as possible. All the necessary libraries required for the project are then imported.

5.1.2 Loading Dataset

Using the Pandas library, the CSV file is read into a Pandas DataFrame called “data”. The dataset labelled “dataset2.csv” is stored in a personal Google Drive for ease of access.

5.1.3 Dataset Manipulation

The default dataset contains “-1” and “0” as the classification of text in the “label” column, thus in order to convert this into simple binary format for ease of training, the default heading for the text, which is called

“headline” is manually changed to “text”. And the data that contains “-1” in the label column, is replaced with “1”. This step ensures that the data is in the right format and structure for the following modelling steps.

5.1.4 Data Analysis and Visualization.

The data is then analysed through steps of visualization, where the number of total texts are given in total, and then split into the number of toxic texts and non-toxic texts, as shows below:

No of texts: 19387

No of toxic: 11692 (60%)

No of non-toxic: 7696 (39%)

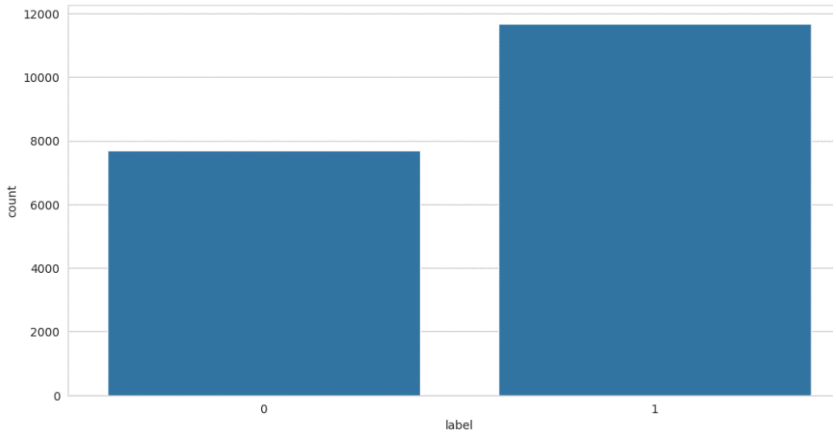


Figure 10 - Distribution of toxic and non-toxic messages

After further inspection, it can be noted that there is a null value in the first column “text”, so the next step is to drop it. Then any duplicates are dropped, and the new figures of data are shown as follows:

No of texts: 18198

No of toxic: 10619 (58%)

No of non-toxic: 7579 (41%)

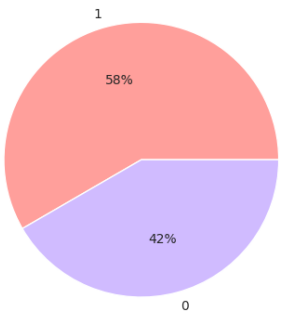


Figure 11 - Pie Chart for visualization

5.1.5 Data Preprocessing

Cleaning the dataset before testing it with any ML or DL model can be highly beneficial as data cleaning is an essential step that can impact the model performance in ways such as reducing noise (that comes in the form of irrelevant information and typos), handling missing values, standardisation (conversion of all text to the same case, removing unnecessary punctuation and possibly correcting spelling mistakes), improving context understanding (by removing irrelevant words from the text) and tokenisation compatibility (separating concatenated words or handling special characters). However, it is vital to avoid overcleaning as some noise, such as slang or specific jargon, might be necessary for the model to understand the context.

This code section creates a list of punctuation characters and defines a cleaning function. It checks if the input is a string; if not, the function returns an empty string. The text is then converted into lowercase, removing extra whitespaces, links, punctuations, non-ASCII characters and URLs, single characters, numbers and extra spaces. Finally, the function returns the cleaned text. Further, the cleaning function is applied to the text data, and the data frame is then reorganised to replace the old text with the newly cleaned text. *Stopwords* are words that do not add much meaning, such as articles, prepositions, conjunctions and pronouns. The next step is refining the cleaned text data by removing these common words. The final step is lemmatisation. This process is where words are recused to their base or root form. A *WordNetLemmatizer* object is initialised. This popular NLP library in Python uses Wordnets' lexical database to find a word's base or root form. A function is created that performs lemmatisation on a given string, then tokenises the string into individual words, where each word is then lemmatised using the *WordNetLemmatizer*, and finally, the lemmatised words are then joined back into a single string. Finally, the new lemmatised text is applied to the previously cleaned text.

5.1.6 Loading BERT and Tokenizing

Here, the BERT model and its Tokenizer are initialised from the transformers library by Hugging Face. The model and Tokenizer being used is the *bert-base-uncased*. Tokenisation is very important as every word and piece of punctuation turns into a "token" for the model to understand and process. Bert uses wordpiece tokenisation, which searches through all the frequencies of the common characters that happen next to each other. It will take a certain number of them and represent them as meaningful tokens. This involves converting the word into a tokenised output, which is in the form of a dictionary, where three keys are obtained: *Input IDs* (token IDs for each sentence), *Token type IDs* (used if there are multiple sentences per input) and *Attention Masks* (a mask that indicates which tokens the model should pay attention to, and which are padding). For each key, the value is a list of lists. The next step involves preparing the dataset for training and validation using a train-test split and tokenising the text data for input into the BERT model. The columns are then converted into a list as the tokeniser expects a list of strings and then sets them to "X" (list of entries from the text column in the dataset) and "y" (label column) objects. It is then split into 80% for training (14,558) and

20% for validation (3,640), using the *train_test_split* function from the Scikit-learn library. The training and validation data is then tokenised. This process is crucial as the model can be trained on one set of data and evaluated on another. This helps assess the performance and ability to generalise to new, unseen data.

5.1.7 Creating Custom PyTorch Dataset

A *Dataset* class, specifically designed for handling tokenized data for use with the BERT model, is created by extending PyTorch's *torch.utils.data.Dataset* base class. This custom Dataset class needs to be fine-tuned for BERT, which involves transforming the tokenised output into a format compatible with this Dataset class. To establish this class, it is necessary to implement three essential functions: *__init__, __getitem__,* and *__len__*. The *__init__* function serves as the constructor, initialising the Dataset object with 'encodings' (a dictionary containing tokenized X_train data from the BERT tokenizer) and 'labels' (a list matching the input data). The *__getitem__* function retrieves a specific example from the dataset based on an index, returning it as a dictionary. In this structure, 'self.encodings' holds the tokenized information, and the retrieval process involves extracting key-value pairs from this dictionary, like input IDs and their corresponding lists, along with token type IDs and attention masks.

Upon obtaining the key value "val", there is an extraction of "key" for utilisation as a dictionary key in the form {key: torch.tensor(val[idx])}. This approach deviates from employing the value "val" as a mere list of lists. Instead, from such a nested list structure, the value corresponding to a specific example index "idx" is extracted, thereby facilitating the acquisition of both the dictionary key and the particular example in tensor representation. Subsequently, a label at a given index is retrieved, culminating in the return of an item manifested as a dictionary. The final method, *__len__* simply returns the length of the input IDs.

The development of a training dataset was accomplished through the integration of tokenized training data, and similarly, a validation dataset was constituted by incorporating tokenized validation data. The presence of a *__getitem__* method enables the processing of the training dataset; by inputting an index "idx", the method yields a dictionary representation of the training data pertinent to that specific example. Examining records within the training dataset reveals that the input IDs do not constitute a list of lists but rather a singular list represented in a tensor format. Thus, selection is focused solely on the list associated with the particular index. This approach maintains the consistency of the key, as the same keys are employed. However, in terms of value, instead of selecting from a list of lists, a singular example is extracted, elucidating the operational mechanics of the *__getitem__* function.

Finally, a function called *compute_metrics* is established to process evaluation outputs. Typically, evaluations of Transformers involve two key elements: labels and predictions. The function receives a parameter *p*, a tuple encompassing predictions and actual labels. This function derives metrics like accuracy, recall, precision, and F1-scores.

5.1.8 Setting Class Weights

Class imbalance is a common problem, especially in text classification tasks where certain classes are less represented. This issue can bias the learning process, favouring models that lean towards the more common class. This project uses a dataset comprising two classes, where the class label is represented in a binary format: 1 indicating toxicity, and 0 as non-toxicity. The challenge arises from the unequal distribution of these classes, potentially resulting in a model that does not accurately reflect the less common class. Addressing this imbalance with standard preprocessing techniques like oversampling (such as using Synthetic Minority Oversampling Technique (SMOTE)), undersampling and random oversampling is complex. These methods alter the original data distribution by creating synthetic data (as SMOTE does), removing samples (as in undersampling) or repeating existing samples (as in random oversampling). Such approaches may lead to overfitting to the oversampled class, particularly if the synthetic or duplicated samples do not represent the actual data distribution accurately.

On the contrary, class weights enable training on the unaltered original dataset, maintaining its distribution. This method is computationally efficient and minimises the risk of overfitting while preserving the model's integrity and interpretability. Singh (2020) supports this approach, describing it as a more natural and less invasive way to address imbalance in the learning process. This also aligns with the strategy Nafis et al. (2023) adopted, which employed class weights to manage data imbalance in their research. Using a weighted loss function during training places greater emphasis on the minority class. This can be achieved in PyTorch by applying class weights to the *CrossEntropyLoss* function.

The *compute_class_weight* function from the Scikit-learn library is used to determine weights that inversely reflect class frequencies. It is set as 'balanced' to automatically modify weights to be inversely proportional to their frequencies of classes in the input data. Then, the unique classes in the training label are calculated, and the training label data is used to determine the frequency of each class. These weights are then incorporated into the BERT model's training regime through a custom training class, *CustomTrainer*, which extends Hugging Face's *HfTrainer* class. At the heart of this method is the *compute_loss* method, modified within the *CustomTrainer.* This method applies class weights to the loss function, particularly to Cross-Entropy Loss. By assigning greater weights to less represented classes, the model incurs a higher penalty for incorrectly classifying samples from these groups than more common ones. This incentivises the model to focus more on learning from the minority classes, aiming for a more equitable and balanced learning process. The customised loss calculation is crucial in preventing the model from favouring the majority class, ensuring it values each class according to its prevalence in the dataset. This technique of integrating class weights is especially effective for deep learning models like BERT, which might otherwise tend to overfit the predominant class in uneven datasets.

5.1.9 Defining Custom Trainer

In ML and DL, essential training parameters like the number of epochs and batch size are crucial for achieving the model's optimal performance. A test trainer is utilised to identify the most effective settings, employing random search across a hyperparameter grid. This grid includes epochs ranging from 2, 3, 4, 5 and 10 and batch sizes of 8, 16, and 32. Insufficient epochs can lead to underfitting, where the model fails to learn from the training data adequately. Conversely, too many epochs might cause overfitting, where the model overly specialises in the training data and performs poorly on new data. While smaller batch sizes can enhance model generalisation and regularisation, they may prolong training time. Larger batch sizes improve computational efficiency and speed up training but can negatively impact generalisation if excessively large. The aim is to balance these factors, and random search helps pinpoint the optimal parameters. Additionally, early stopping is implemented to halt training when there is no further improvement, marking the best performance. This procedure is extensive and computationally intensive. Ultimately, the ideal parameters were identified as 4 epochs and a batch size of 16.

The primary Trainer and its necessary arguments are configured, establishing a distinct output directory for the training results. The process involves using the BERT model, the training and evaluation datasets and predefined compute metrics. The Trainer will process the evaluation object *p* through the compute function, which handles predictions and labels. It selects the maximum prediction and then calculates metrics by comparing these predictions with the labels. Additionally, the Trainer accepts configurable arguments related to the training. This includes setting parameters like 4 epochs, a batch size of 16, applying an evaluation strategy at the end of each epoch, and logging progress every 10 steps.

5.1.10 Plot Learning Curve

A custom function is created to display the learning curve using the results obtained from the trainer. Figure 12 below shows the progression stats with each epoch being trained on.

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.295200	0.159631	0.950000	0.962381	0.951507	0.956913
2	0.148900	0.182379	0.951648	0.943534	0.975518	0.959259
3	0.125200	0.199237	0.951099	0.944698	0.973164	0.958720
4	0.005100	0.230754	0.953297	0.950230	0.970810	0.960410

Figure 12 - Performance over 4 epochs

Figure 13 Below illustrates the models performance by showing the overall training and validation loss over 4 epochs, and the training accuracy overtime.

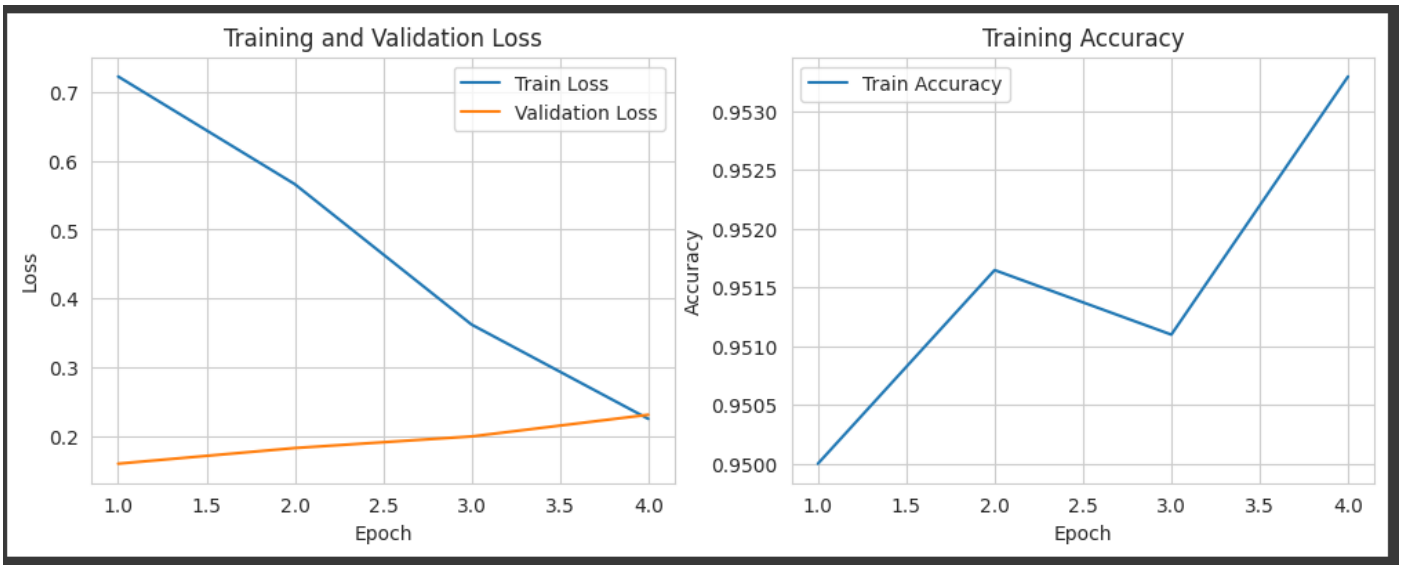


Figure 13 - Learning Curve

5.1.11 Get Confusion Matrix and Classification Report

Figure 14 below shows the confusion matrix and classification report of the results obtained from training the model.

```
Confusion Matrix:
 [[1408  108]
 [  62 2062]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      1516
           1       0.95      0.97      0.96      2124

    accuracy                           0.95      3640
   macro avg       0.95      0.95      0.95      3640
weighted avg       0.95      0.95      0.95      3640
```

Figure 14 - Confusion Matrix and Classification Report

5.1.12 Saving Model

The challenge with using 'pickle' to save the entire model is ensuring that the 'torch.load' process, embedded within the pickle loading, appropriately respects the 'map_location' parameter. This can be a complex endeavour. An alternative approach involves utilising 'joblib,' but it needs to be specifically tailored for PyTorch models and may not efficiently handle the intricacies of deep learning models, such as device-specific tensor data. Therefore, for PyTorch models, particularly complex ones like BERT, it is advisable to opt for a 'State Dictionary Only' approach, employing 'torch.save' and 'torch.load.' This method preserves the model's state dictionary, a Python dictionary that maps each layer to its parameters (weights and biases). This approach offers greater flexibility in saving PyTorch models, facilitating easy mapping of the saved model to different devices, whether equipped with a GPU or CPU.

5.2 Frontend

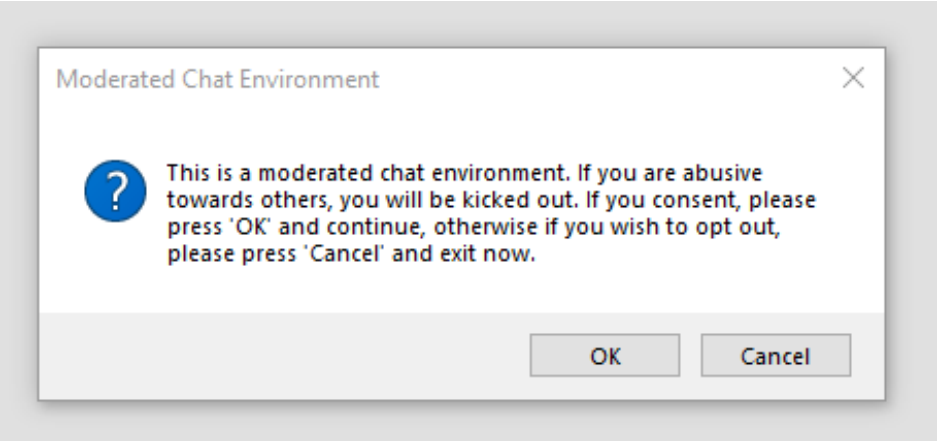
5.2.1 chat_client.py

This script is responsible for the client-side operations of the chat application. It employs Python's *tkinter* library to craft a Graphical User Interface (GUI) that facilitates interactions like sending messages to the server and showing responses from other clients. A TCP/IP connection is also established by specifying the server host and port. The script utilises Python's *socket* library for the creation of a socket object and connection to the server. The decision to use a TCP/IP socket is intentional due to its reliability in transmitting data, which is crucial for real-time chat applications. The *ChatApp*, built from the Python library's "tk.Tk," leverages Tkinter's features to develop the GUI. This includes a chat display area with a scrollbar, a list box for displaying active users, a space for entering messages, and a send button. The design focuses on user-friendliness and interactivity, aiming to provide users with a chat application that is easy to navigate and use, as the application's objective is to imitate a typical OSN chat system.

A vital feature of the script is its ability to manage the sending and receiving of messages. The function for sending messages, namely *send_msg*, is linked to the GUI's send button and the enter key. It captures the user's input and sends it to the server. At the same time, the script is equipped to handle incoming messages. This task is handled by a separate thread named *receive_msg*, which is dedicated to message reception. This ensures the application stays responsive and interactive, even in active network communication. Using the *threading* library from Python in this approach is a crucial design element. It prevents the GUI from becoming unresponsive, a common issue in network-connected applications. Additional functionalities like error handling and user notifications, such as updates about user activities, disconnection alerts and warnings against abuse, are also included. Upon starting the client application, users are given an alert window, stating

the presence and rules of the moderated environment, and whether they consent or not. This is to ensure the application complies with the ethical issue concerning freedom of speech and respecting their autonomy and rights.

Figure 15 - Alert window



If chosen to continue, they are able to set their username. Furthermore, the script provides capabilities for recording chat messages and displaying them in the chat box, each accompanied by a timestamp. This acts as a method for keeping records. All the logged messages are saved in a text file.

The script allows for creating up to three clients, but only two clients are necessary to demonstrate the application's functionality. Essentially, the script aims to provide a smooth and user-friendly chat experience supported by a strong and dependable communication link with the server.

5.2.2 chat_server.py

This script is responsible for the server-side operations and detecting abuse within the chat application. It begins by setting up BERT and its tokenizer, a necessary step because loading BERT's trained model as a state dictionary requires the model's architecture already in place. The saved state dictionary has the trained weights and biases but not the model's structure. Initially trained on a GPU, the model was modified for use on a CPU due to the server's hardware limitations. This involved ensuring the tensor computations worked with CPU processing, thus justifying the next step of moving the model to CPU and setting it to evaluation mode. The script defines the same host and port addresses as the client script and initialises two lists to monitor connected clients and their names. A unique addition to the script is the creation of a dynamic buffer, acting as a temporary memory for the system, helping to understand the context of conversations.

```python
# `MessageBuffer` class to handle the storage and retrieval of messages with a dynamic retention period.
# it stores messages along with their timestamps in a deque, which is efficient for appending and popping elements.

# __init__ - constructor that creates a new messagebuffer object
# retentionperiod is an argument that sets the duration for which the messages are retained, so 5 minutes
# initalises two attributes, self.messages and self.retention_period

class MessageBuffer:
    def __init__(self, retention_period=timedelta(minutes=5)):
        self.messages = deque()
        self.retention_period = retention_period

    def add_message(self, message):
        timestamp = datetime.now()
        self.messages.append((timestamp, message))
        self.clean_old_messages()

    def get_context(self):
        return " ".join(message for _, message in self.messages)

    def clean_old_messages(self):
        current_time = datetime.now()
        while self.messages and current_time - self.messages[0][0] > self.retention_period:
            self.messages.popleft()

#creates a defaultdict where each key is a client name and the value is a messagebuffer object
recent_messages = defaultdict(MessageBuffer)
#an empty dictionary that keeps track of the number of warnings for each client
warnings = {}
```

Figure 16 - MessageBuffer class

The *MessageBuffer* class oversees message storage and retrieval with a flexible retention period, enabling the system to grasp the conversation's flow, not just isolated messages. A double-ended queue (deque) from Python's collections module is employed for efficient message management. A deque efficiently adds and removes elements from both ends with O(1) complexity, making it suitable for maintaining a recent message window in memory.

An empty deque is then initialised, and messages are appended to the right end of it. Each message is stored as a tuple consisting of its timestamp and message. The *get_context* method iterates through the deque, concatenating the message components of each tuple into a string that reflects the latest conversation context. This aids in comprehending the conversation's current tone and subject matter, enhancing the effectiveness of abuse detection. The *clean_old_messages* method uses a while loop to check the oldest message in the deque. If it is older than the retention period of 5 minutes, it is removed from the front of the deque. This uses the deques capability of quick removals from the front and ensures a consistent flow of messages within the retention period. The *recent_messages* function establishes a dictionary-like structure, where each key represents a client's name, and each value corresponds to a *MessageBuffer* object. An empty dictionary is also set up to monitor the count of warnings issued to each client.

In parallel, the script sets up a TCP server socket, which acts as the listening port for incoming client connections. Then, the *classify_with_bert* function uses the model to categorise incoming text as either regular or toxic by comparing the probability against a 0.5 threshold to determine a true or false value.

```
def classify_with_bert(text):

    model.eval()
    model.to('cpu')

    # Prepare the text for BERT using the tokenizer
    inputs = tokenizer.encode_plus(
        text,
        add_special_tokens=True,
        max_length=512,
        return_tensors='pt',
        padding='max_length',
        truncation=True
    )

    # Get predictions from the model
    with torch.no_grad():
        outputs = model(**inputs)

    # Convert logits to probabilities
    probs = torch.nn.functional.softmax(outputs.logits, dim=-1)

    # we compare the probability to the threshold 0.5 to get a True/False value
    is_abusive = probs[:, 1] > 0.5

    return is_abusive.item()
```

Figure 17 - Classify with bert function

Additionally, the script defines another function responsible for distributing the list of active users to all clients. It also features a function for logging chat messages in a text file. Furthermore, the script introduces a *broadcast* function that manages the message distribution from one client to another. This function also oversees the handling of client connections and disconnections.

```python
#in this function, messages are added to the buffer as they are received, and the context is retrieved by concatenating the recent messages within the retention period.
# this function continuously receives messages from a client, processes them, checks for toxicity, and handles warnings and disconnections accordingly.
def handle(client):
    name = names[clients.index(client)]
    while True:
        try:
            raw_message = client.recv(1024)
            if not raw_message:
                print(f"Empty message, removing client.")
                remove_client(client)
                break
            message = raw_message.decode('utf-8')
            print(f"Received message: {message}")
            name = names[clients.index(client)]

            recent_messages[name].add_message(message)
            context = recent_messages[name].get_context()

            print(f"Classifying message for abuse detection with context")
            is_toxic = classify_with_bert(context)
            if is_toxic:
                warnings[name] = warnings.get(name, 0) + 1
                if warnings[name] == 1:
                    warning_message = "WARNING: Stop using profanity and behave decently"
                elif warnings[name] == 2:
                    warning_message = "WARNING: This is your final warning, you will be kicked out if you bully again"
                elif warnings[name] >= 3:
                    client.send("DISCONNECT You have been removed from the chat due to repeated bullying.".encode('utf-8'))
                    remove_client(client)
                    broadcast(f"{name} has been removed from the chat due to repeated bullying.", None, False)
                    break
                client.send(warning_message.encode('utf-8'))
                if warnings[name] < 3:
                    hidden_msg_notification = "Offensive language was detected and hidden."
                    broadcast(hidden_msg_notification, client, True)
            else:
                broadcast(f"{name}: {message}".encode('utf-8'), client, exclude_sender=False)
        except ConnectionError:
            print(f"Client {name} disconnected.")
            remove_client(client)
            break
        except Exception as e:
            print(f"An error occurred: {e}")
            break
```

Figure 18 - Handle function

Clients are managed in individual threads using the *handle* function, which processes client messages. When a new message is received, it is not only the message itself that is analysed for potential abuse but also its context. The system retrieves a concatenation of recent messages from the *MessageBuffer* up to the current one, all within the retention period. This approach enables the abuse detection model to make more nuanced decisions by considering the message in the broader context of the conversation. The *handle* function also incorporates an abuse detection and warning mechanism. This system employs the BERT model to scrutinise messages for abusive language. If abuse is detected, the server refrains from broadcasting the original content. Instead, it sends a cautionary message to the sender, discouraging such behaviour, and informs other clients that an offensive message has been identified and concealed. After three instances of abuse, the client is automatically disconnected from the server. Upon a client's disconnection, they are removed from the list of active clients, and a notification of their departure is broadcast to all clients. This automated moderation plays a crucial role in the script, ensuring a safe and respectful chat environment.

6. Testing and Integration

6.1 Testing all Models trained

Table 3 - Testing of Models

Training arguments	Model used	Dataset used	Parameters	Results	Text preprocessing	Notes
Manual inputting	BERT	Stock	Epochs – 10 Batch size - 20	Accuracy - 94%	With	-
	BERT	Stock	Epochs – 10 Batch size - 20	Accuracy - 96%	Without	Performs better than model with preprocessing.
	MBERT	Stock	Epochs – 10 Batch size - 20	Accuracy - 94%	With and Without (No change)	Text cleaning had no effect on performance
High epoch and batch size, but with early stoppage	BERT	Stock	Epochs – 30 Batch size - 32	Accuracy - 95%	With	Stopped at 5 epochs due to early stoppage
	BERT	Stock	Epochs – 30 Batch size - 32	Accuracy - 95%	Without	Stopped at 4 epochs due to early stoppage
Random Search	BERT	Stock	Epochs – 4 Batch size - 16	Accuracy - 95%	With	Parameters searched over – Epochs - [2, 3, 4, 5, 10] Batch size - [8, 16, 32] Best parameters chosen – Epochs - 4 Batch size - 16 Loss: 0.302
	BERT	Stock	Epochs – 4 Batch size - 16	Accuracy - 96%	Without	
	MBERT	Stock	Epochs – 4 Batch size - 16	Accuracy - 94%	With	
	MBERT	Stock	Epochs – 4 Batch size - 16	Accuracy - 95%	Without	

Custom modified dataset	BERT	Updated	Epochs – 4 Batch size - 16	Accuracy - 95%	With and without	Text cleaning had no effect on metric performance however affected the performance of the message buffer.
EXTRA						
Manually inputting	BERT	Stock	Epochs – 1 Batch size - 8	Accuracy - 94%	With	No performance change throughout different parameters
	BERT	Stock	Epochs – 5 Batch size - 8	Accuracy - 94%	With	
	BERT	Stock	Epochs – 10 Batch size - 8	Accuracy - 94%	With	
	BERT	Stock	Epochs – 10 Batch size - 32	Accuracy - 94%	With	
	BERT	Stock	Epochs – 1 Batch size - 8	Accuracy - 95%	Without	Slight increase in performance without text cleaning.
	BERT	Stock	Epochs – 5 Batch size - 8	Accuracy - 95%	Without	
	BERT	Stock	Epochs – 10 Batch size - 8	Accuracy - 95%	Without	
	BERT	Stock	Epochs – 10 Batch size - 32	Accuracy - 96%	Without	Higher accuracy using a higher epoch and batch size

6.2 Integration of Model with GUI

After training various models, a version with no text preprocessing is taken as the final model for the abusive detection chat application. This can be justified due to reasons such as:

- BERT is specifically designed to comprehend the contextual meaning of words inside phrases. Eliminating stopwords or modifying words by lemmatisation can affect the context and significance of sentences. The performance of the DL model may be compromised if the text is oversimplified or modified since it heavily depends on the whole context for correct predictions, thus leading to the loss of essential information. Unlike typical NLP models, BERT can effectively employ stopwords to enhance its comprehension of sentence structures and word connections. Stopwords play a vital role in providing important contextual clues that enhance the accuracy of the model's predictions.
- BERT is also pre-trained on an extensive collection of material, enabling it to acquire a distinct set of words. Applying lemmatisation might convert words into forms not present in BERT's acquired vocabulary.
- BERT is highly robust when confronted with noise and variances in text since it has already been trained using many datasets, including texts with informal language, errors, and numerous peculiarities. Hence, excessively overcleaning the text may not be vital and might hinder its effectiveness.

6.3 Testing the GUI with user inputs

The Abuse Detection and Prevention Figure and Message Buffer Figures in the Appendix section of the report denote the application working, by testing different conversational texts to identify whether the model predicts the text correctly or not and understands the context within conversation.

7. Product Evaluation

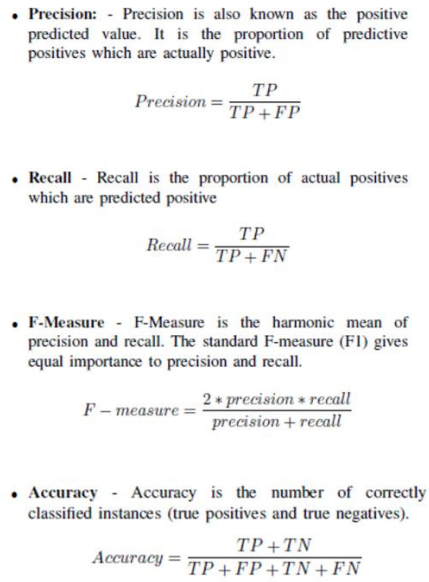
7.1 Metric Analysis

When assessing the performance of the BERT model trained on parameters of 4 epochs with a batch size of 16, the learning curve offers a thorough understanding of the model's learning patterns. The convergence of training and validation loss suggests that the model is well-fitted, as the training loss consistently decreases, and the validation loss remains consistently low. This indicates that the model is balanced and can generalise effectively to new data. Significantly, the training and validation loss curves stay consistent, strengthening the idea that the model has grasped the data's fundamental patterns without memorising the training set. The accuracy curve for the training data shows a consistent increase, indicating that the model's capability to categorise the training samples accurately improves with each epoch. An observed decline between the second and third epochs necessitates a more thorough investigation; it may indicate a brief setback in learning that the model successfully overcomes by the fourth epoch, as demonstrated by the subsequent increase in accuracy. This variation can be ascribed to several reasons, such as the learning rate or the unpredictable nature of the training batches. It indicates an opportunity to enhance the stability of the training process through optimisation.

The classification report and confusion matrix provide additional evidence of the model's strong performance. A 95% accuracy demonstrates the model's effectiveness, as evidenced by precision scores of 0.96 for class 0 and 0.95 for class 1. These values indicate a high rate of accurate positive predictions out of all positive predictions produced, and a low rate of false positives. The recall scores of 0.93 for class 0 and 0.97 for class 1 indicate that the model has a high sensitivity and can accurately identify the most positive samples. The F1-scores, representing the harmonic mean of precision and recall, are 0.94 for class 0 and 0.96 for class 1. These numbers indicate a harmonious balance between the two metrics and demonstrate the model's high accuracy in handling both classes. Nevertheless, the slight discrepancy between precision and recall can indicate tiny favouritism towards a particular class, which could be rectified by fine-tuning the model or implementing data augmentation techniques. In general, the model demonstrates a praiseworthy ability to make accurate predictions. However, the subtle variations in the accuracy curve and the minor differences in performance measures for specific classes highlight areas that should be improved.

To further understand the readings from the matrix, the results can be analysed as follows: TP (True Positives) and FP (False Positives) are the values that have been predicted correctly. TN (True Negatives) and FN (False Negatives) are the values that have been mispredicted. As observed from Figure 14, TP = 1408, TN = 2062, whilst FP = 108 and FN = 62. This means that out of (1408+2062) = 3470 test instances, the algorithm misclassified (108+62) = 170. As shown in the figure below, the accuracy is calculated by dividing the total number of true positives and negatives by the sum of all the values in the matrix. This is how the overall

accuracy of 95% was obtained. Figure 19 below shows the actual definitions of the metrics and how they are all calculated.



- **Precision:** - Precision is also known as the positive predicted value. It is the proportion of predictive positives which are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** - Recall is the proportion of actual positives which are predicted positive

$$Recall = \frac{TP}{TP + FN}$$

- **F-Measure** - F-Measure is the harmonic mean of precision and recall. The standard F-measure (F1) gives equal importance to precision and recall.

$$F - measure = \frac{2 * precision * recall}{precision + recall}$$

- **Accuracy** - Accuracy is the number of correctly classified instances (true positives and true negatives).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Figure 19 - Definition of metric scores (Shah, Phadtare and Rajpara 2022)

7.2 Evaluation Against Related Notable Works

Sameer (2022) was able to use the benefits of character-level embedding representations of tweets and DNN to achieve their final results. The author set a high number of 50 epochs to train their CNN and CNN-LSTM models and applied oversampling techniques to their training dataset to handle the imbalance data, their best model obtained a detection accuracy of 88.97%. As their dataset has the same number of columns as the dataset used in this project, theirs is larger as it has a total of 32,551 tweets. Regardless, this projects model which uses BERT was able to obtain a 95% accuracy score, thus outperforming their model and showing the current performance reach of DL models.

Nafis et al. (2023) focused on monolingual, code-mixed and combined dataset evaluations utilising models such as BERT, RoBERTa, XLM-R, HingRoBERTa and Bernice. However, their BERT model was only able to demonstrate an accuracy of 60.99% on monolingual, 61.94% on code-mixed and 62.05% overall. They used an official Twitter API to obtain their tweets, which classifies their dataset as more integral than the one used in this project; however, in terms of accuracy scores, the BERT model in this project outperforms their model.

Mathur et al. (2018b) introduced a novel HOT dataset and created a MIMCT model that consisted of two main components: primary and second inputs and CNN-LSTM binary channel neural network, outperformed existing classification models in the domain by using multiple primary embeddings and secondary extracted semantic features as inputs to train a multi-channel CNN-LSTM architecture, that has been pre-trained on

English tweets through transfer learning. As their methodology is completely different from that of this project, a comparison of metric performance is conducted. Their SVM with TF-IDF baseline gave a peak performance of 72.3% F1-score, but their MIMCT model with maximum input features gave an F1-score of 89.5%. Compared to the BERT model trained in this project, which obtained an F1-score of 96%, the BERT model outperforms theirs in metrics.

In contrast to the traditional ML and NLP approach utilised by Shah, Phadtare and Rajpara (2022) in their study, this approach integrates advanced DL techniques. While the authors focused on applying ML algorithms and NLP for feature extraction and classification of text to identify cyberbullying, this project's approach builds upon their foundational work by incorporating BERT to enhance the accuracy and efficiency of cyberbullying detection. Despite using BERT, the results obtained from this approach align closely with the best-performing model in their study, which employed the Random Forest classifier with TF-IDF as the feature selection model. This outcome suggests that while BERT did not significantly outperform the traditional ML models used in their work, it nonetheless demonstrates the robustness and comparability of BERT in the context of cyberbullying detection. BERT models utilise token embeddings and positional encoding instead of fixed-length vector representations like TF-IDF. Therefore, unlike the approach used by Shah, Phadtare and Rajpara (2022), there was no necessity for an independent TF-IDF vectorization or maintaining a vocabulary file, as was done in their research. By leveraging their dataset and making a few improvements to enhance the quality and balance of languages, and integrating BERT into the analysis, this project approach contributes to the evolving landscape of natural language processing and deep learning, offering a more nuanced and context-aware method for identifying cyberbullying in social media content, while also validating the effectiveness of the traditional machine learning models employed in the original study. Figure 20 below illustrates that compared to the exact conversational text used in Buffer Figure 5 in the appendix, their system misclassifies many of the words used, meaning theirs does not understand context and has a severely unbalanced dataset favouring English more than Roman Hindi. This proves that this project's attempt at detecting conversational context and general content in Hinglish surpasses theirs, regardless of similar model accuracy scores.

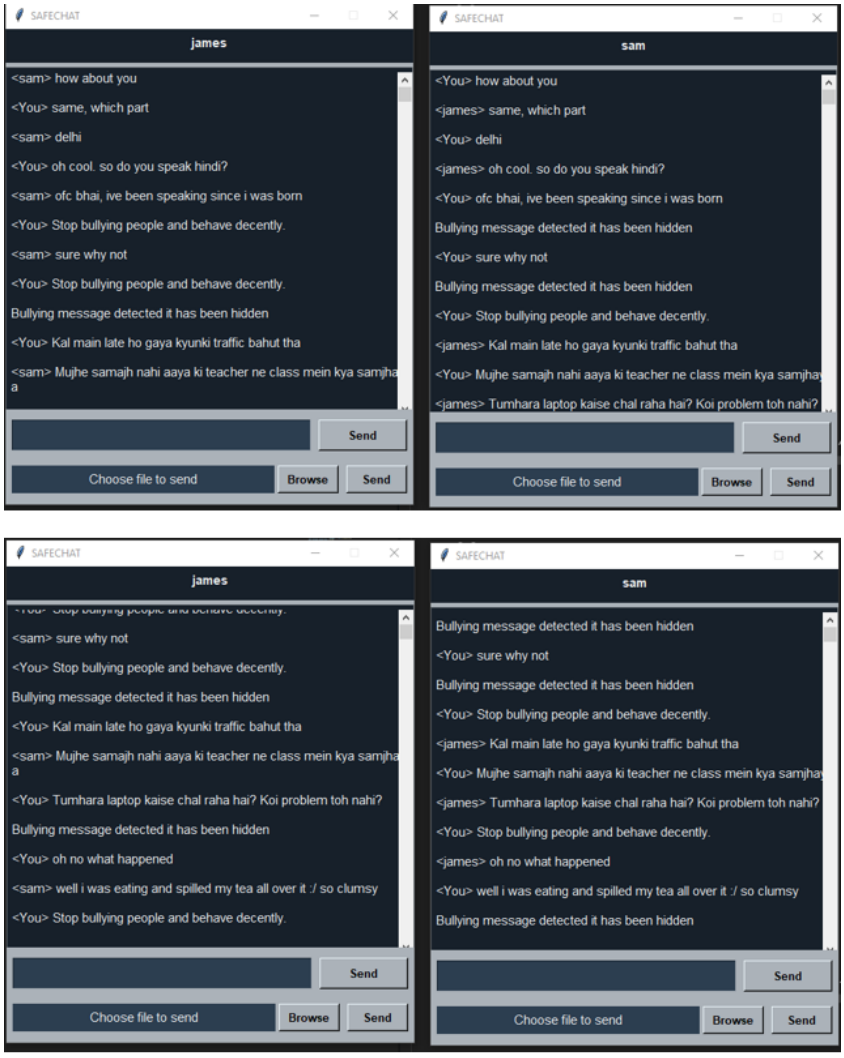


Figure 20 - Comparing model performance against (Shah, Phadtare and Rajpara 2022)

8. Conclusion of Report

To conclude, the project under review has extensively delved into a comprehensive exploration of literature within the domain of online abuse, commencing by elucidating the fundamental aspects of abuse, subsequently progressing towards strategies for addressing this issue, the different methodologies used, ranging from traditional to machine learning and deep learning methods, and then diving into some related works which served as a comparative analysis and inspiration in designing the product for this project. The report delineated two predominant approaches in tackling online abuse, primarily ML and DL, and then further articulated a set of functional and non-functional requirements integral to the project. The design of the product was thoroughly examined, followed by phases such as the backend and frontend implementation, product testing and evaluation against metrics and the results produced from the related works which were previously discussed. Besides the significant contribution of accurately detecting and mitigating abusive content in an OSN setting, this project's notable innovation is the development of a dynamic buffer, functioning as a short-term memory, aiding in contextual understanding amidst the constraints posed by dataset limitations. The product developed

represents a progressive step forward in this field. However, the concluding sections of this report are dedicated to discussing the limitations of the current system and proposing future avenues for enhancement.

8.1 Limitations

The necessity for computational power is paramount in training sophisticated models. Utilising a cloud-based environment such as Google Colab, which does not support a complete graphical user interface system, presents particular challenges. The model was initially saved using the pickle library. However, difficulties arose when loading the model onto a server in Visual Studio Code. Errors indicated issues in deserializing the object on a CUDA device. This problem was traced to the fact that the model, having been trained and saved in a GPU environment, required explicit mapping to the CPU when being loaded on a CPU-only system. Attempts were made to utilise CUDA with a personal computer equipped with a high-performance GPU. However, despite installing all necessary CUDA toolkits, persistent issues were encountered in integrating it with Visual Studio Code. Consequently, the viable solution was to subscribe to Google Colab Pro, taking advantage of the advanced GPUs available in its paid version, however this meant paying £60 for compute units needed to train the models, which was very costly. Hardware requirements are also a limitation, as running the chat application on a laptop with a mobile CPU was tremendously slower than running it on a desktop computer with a mid to high-end CPU.

The system will unfortunately still misclassify standard text as abusive at times due to the message buffer concatenating all recent messages for each client, so for example if an abusive word is said at first, but followed with a non-abusive sentence or phrase, the system might still classify it as abusive since it is joined with the previous message.

As the model has been saved as a separate state dictionary, it cannot learn from unseen data, so this is a major constriction. Another limitation is the dataset used, favouring English as it has more English content than Roman Hindi. This means the model needs more Hindi text to learn from. There was an attempt to rectify this using generative AI to generate additional Hindi text; however, this technically needs to be revised in order to maintain the integrity and meaning of the texts and limits the model from learning as this is text generated by AI and not taken from real word conversations. This could also be why the system may misclassify some Hindi text as abusive when it is not.

8.2 Future Work

Possible improvements to the model involve partitioning datasets according to the language used and enhancing the model's training and understanding abilities for each language. A practical method involves using a thorough English dataset and applying a translation model to transform it into another language, then storing it as a distinct dataset. This strategy guarantees the presence of comparable amounts of data in both languages to facilitate the learning process of the model. In the present version, a single dataset containing a combination of English and Roman Hindi is used, where the Hindi material is transcribed using the English alphabet. The model can be modified to process a dataset effectively containing Hindi text written in its conventional script. This would allow the model to acquire knowledge from special characters, a feature presently not addressed by BERT, MBERT, or even Google Translate, as these systems are not trained on Romanized Hindi text. More testing needs to be done with the model to check for false positives and negatives when detecting the conversation's context. As for the classification of abusive text, the model performs as expected. In addition, a dataset can be created that supports conversational content, thus making it more straightforward for the BERT model to learn the context by constantly learning and improving from the conversations in the dataset. Contextual awareness is especially relevant in datasets where each item is a constituent of a comprehensive sequence. This is applicable to situations such as conversations, where each message functions as a response to the previous one, time-series data, or narrative materials, such as novels or a series of related articles.

Another future improvement can lie in detecting the type of abuse. The dataset used in this project limits this as it only has two headings, with text and label. Further datasets composed of different types, such as racism, sexism, insult, and so forth, can be curated. A primary challenge in this will be in translating the texts if it is done in a bilingual/multilingual setting. However, if it is made possible and done successfully, this enhancement can let the user know what type of abuse has been detected and prevented.

Finally, further testing can be conducted with the set threshold in the server script, to see whether there is any performance increase or decrease using different values instead of 0.5.

9. List of References

Alaskar, H. and Saba, T. (2021). Machine Learning and Deep Learning: A Comparative Review. *Algorithms for Intelligent Systems*, pp.143–150. doi:https://doi.org/10.1007/978-981-33-6307-6_15.

Amnesty International UK (2017). *Online abuse of women widespread*. [online] www.amnesty.org.uk. Available at: https://www.amnesty.org.uk/online-abuse-women-widespread [Accessed 7 Jan. 2024].

Badjatiya, P., Gupta, S., Gupta, M. and Varma, V. (2017). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. [online] doi:https://doi.org/10.1145/3041021.3054223.

Binns, R. (2017). *Fairness in Machine Learning: Lessons from Political Philosophy*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3086546 [Accessed 11 Jan. 2024].

Boyd, D.M. and Ellison, N.B. (2007). Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, [online] 13(1), pp.210–230. doi:https://doi.org/10.1111/j.1083-6101.2007.00393.x.

Burnap, P. and Williams, M.L. (2015). Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. *Policy & Internet*, 7(2), pp.223–242. doi:https://doi.org/10.1002/poi3.85.

Caselli, T., Basile, V., Mitrović, J. and Granitzer, M. (2021). *HateBERT: Retraining BERT for Abusive Language Detection in English*. [online] ACLWeb. doi:https://doi.org/10.18653/v1/2021.woah-1.3.

Chandrasekharan, E., Samory, M., Jhaver, S., Charvat, H., Bruckman, A., Lampe, C., Eisenstein, J. and Gilbert, E. (2018). The Internet's Hidden Rules. *Proceedings of the ACM on Human-Computer Interaction*, [online] 2(CSCW), pp.1–25. doi:https://doi.org/10.1145/3274301.

Chopra, A., Sharma, D.K., Jha, A. and Ghosh, U. (2023). A Framework for Online Hate Speech Detection on Code Mixed Hindi-English Text and Hindi Text in Devanagari. *ACM Transactions on Asian and Low-Resource Language Information Processing*. doi:https://doi.org/10.1145/3568673.

Citron, D.K. (2014). *Hate Crimes in Cyberspace - Introduction*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2616790 [Accessed 6 Jan. 2024].

Dadvar, M., Trieschnigg, D., Ordelman, R. and de Jong, F. (2013). Improving Cyberbullying Detection with User Context. *Lecture Notes in Computer Science*, pp.693–696. doi:https://doi.org/10.1007/978-3-642-36973-5_62.

Davidson, T., Bhattacharya, D. and Weber, I. (2019). *Racial Bias in Hate Speech and Abusive Language Detection Datasets*. [online] ACLWeb. doi:https://doi.org/10.18653/v1/W19-3504.

Davidson, T., Warmsley, D., Macy, M. and Weber, I. (2017) "*Automated Hate Speech Detection and the Problem of Offensive Language*", Proceedings of the International AAAI Conference on Web and Social Media, 11(1), pp. 512-515. doi: 10.1609/icwsm.v11i1.14955.

Dean, B. (2023). *Social network usage & growth statistics: How many people use social media in 2021?* [online] Backlinko. Available at: https://backlinko.com/social-media-users [Accessed 10 Jan. 2024].

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North*, 1. doi:https://doi.org/10.18653/v1/n19-1423.

Duggan, M. (2017). *Online Harassment 2017*. [online] Pew Research Center: Internet, Science & Tech. Available at: https://www.pewresearch.org/internet/2017/07/11/online-harassment-2017/ [Accessed 6 Jan. 2024].

Efimov, V. (2023). *Large Language Models: BERT — Bidirectional Encoder Representations from Transformer*. [online] Medium. Available at: https://towardsdatascience.com/bert-3d1bf880386a [Accessed 20 Jan. 2024].

European Commission. (2016). *The EU Code of conduct on countering illegal hate speech online*. [online] Available at: https://commission.europa.eu/strategy-and-policy/policies/justice-and-fundamental-rights/combatting-discrimination/racism-and-xenophobia/eu-code-conduct-countering-illegal-hate-speech-online_en [Accessed 7 Jan. 2024].

Fortuna, P. and Nunes, S. (2018). A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys*, [online] 51(4), pp.1–30. doi:https://doi.org/10.1145/3232676.

Galán-GarcÍa, P., Puerta, J.G.D.L., Gómez, C.L., Santos, I. and Bringas, P.G. (2016). *Supervised machine learning for the detection of troll profiles in twitter social network: application to a real case of cyberbullying*. [online] philpapers.org. Available at: https://philpapers.org/rec/GALSML [Accessed 8 Jan. 2024].

Gao, L. and Huang, R. (2017). Detecting Online Hate Speech Using Context Aware Models. *RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning*. [online] doi:https://doi.org/10.26615/978-954-452-049-6_036.

Gitari, N.D., Zhang, Z., Damien, H. and Long, J. (2015). A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4), pp.215–230. doi:https://doi.org/10.14257/ijmue.2015.10.4.21.

Gupta, V., Roychowdhury, S., Das, M., Banerjee, S., Saha, P., Mathew, B., Vanchinathan, H.P. and Mukherjee, A. (2022). *Multilingual Abusive Comment Detection at Scale for Indic Languages*. [online] openreview.net. Available at: https://openreview.net/forum?id=HCnb1TByvx7 [Accessed 10 Jan. 2024].

Hinduja, S. and Patchin, J.W. (2010). Bullying, Cyberbullying, and Suicide. *Archives of Suicide Research*, 14(3), pp.206–221. doi:https://doi.org/10.1080/13811118.2010.494133.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. *Springer Texts in Statistics*. New York, NY: Springer New York. doi:https://doi.org/10.1007/978-1-4614-7138-7.

Jane, E.A. (2020). Online Abuse and Harassment. *The International Encyclopedia of Gender, Media, and Communication*, [online] pp.1–16. doi:https://doi.org/10.1002/9781119429128.iegmc080.

Jhaveri, M., Ramaiya, D. and Chadha, H.S. (2022). *Toxicity Detection for Indic Multilingual Social Media Content*. [online] arXiv.org. Available at: https://arxiv.org/abs/2201.00598 [Accessed 10 Jan. 2024].

Kaplan, A.M. and Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of Social Media. *Business Horizons*, 53(1), pp.59–68. doi:https://doi.org/10.1016/j.bushor.2009.09.003.

Khan, A.A. and Bhat, A. (2022). A Study on Automatic Detection of Cyberbullying using Machine Learning. *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*. doi:https://doi.org/10.1109/iciccs53718.2022.9788299.

Kumar, D., Cohen, R. and Golab, L. (2019). *Online abuse detection: the value of preprocessing and neural attention models*. [online] ACLWeb. doi:https://doi.org/10.18653/v1/W19-1303.

Lai, S., Xu, L., Liu, K. and Zhao, J. (2015). Recurrent Convolutional Neural Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1). doi:https://doi.org/10.1609/aaai.v29i1.9513.

Li, H. (2017). Deep learning for natural language processing: advantages and challenges. *National Science Review*, [online] 5(1), pp.24–26. doi:https://doi.org/10.1093/nsr/nwx110.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. [online] arXiv.org. Available at: https://arxiv.org/abs/1907.11692 [Accessed 11 Jan. 2024].

Livingstone, S., Haddon, L., Görzig, A., Ólafsson, K., Leslie, H., Ólafsson, A. and Kjartan (2010). *Risks and safety for children on the internet: the UK report: summary of key findings Report Original citation*. [online] Available at:

https://eprints.lse.ac.uk/43731/1/Risks%20and%20safety%20for%20children%20on%20the%20internet%28lsero%29.pdf [Accessed 7 Jan. 2024].

Mathur, P., Rajiv Ratn Shah, Sawhney, R. and Debanjan Mahata (2018a). Detecting Offensive Tweets in Hindi-English Code-Switched Language. *Meeting of the Association for Computational Linguistics*. doi:https://doi.org/10.18653/v1/w18-3504.

Mathur, P., Sawhney, R., Ayyar, M. and Shah, R. (2018b). Did you offend me? Classification of Offensive Tweets in Hinglish Language. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. doi:https://doi.org/10.18653/v1/w18-5118.

Mishra, P., Yannakoudakis, H. and Shutova, E. (2020). Tackling Online Abuse: A Survey of Automated Abuse Detection Methods. *arXiv:1908.06024 [cs]*. [online] Available at: https://arxiv.org/abs/1908.06024 [Accessed 6 Jan. 2024].

Mubarak, H., Darwish, K. and Magdy, W. (2017). Abusive Language Detection on Arabic Social Media. *Proceedings of the First Workshop on Abusive Language Online*. doi:https://doi.org/10.18653/v1/w17-3008.

Nafis, N., Kanojia, D., Saini, N. and Murthy, R. (2023). *Towards Safer Communities: Detecting Aggression and Offensive Language in Code-Mixed Tweets to Combat Cyberbullying*.

Nichite, P. (2022). *Fine Tune Transformers Model like BERT on Custom Dataset.* [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=9he4XKqqzvE&t=911s [Accessed 23 Jan. 2024].

Nixon, C. (2014). Current perspectives: The impact of cyberbullying on adolescent health. *Adolescent Health, Medicine and Therapeutics*, [online] 5(5), pp.143–158. doi:https://doi.org/10.2147/ahmt.s36456.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y. and Chang, Y. (2016). Abusive Language Detection in Online User Content. *Proceedings of the 25th International Conference on World Wide Web - WWW '16*. [online] doi:https://doi.org/10.1145/2872427.2883062.

Noviantho, S.I. and Ashianti, L., 2017, November. Cyberbullying classification using text mining. *In 2017 1st International Conference on Informatics and Computational Sciences (ICICoS)* (pp. 241-246).

OpenAI ChatGPT (2024) ChatGPT response to Santa Sian, 20 February.

Parikh, R. and Dalvi, A. (2024). Identifying Instances of Cyberbullying on Twitter Using Deep Learning. pp.87–95. doi:https://doi.org/10.1007/978-981-99-6984-5_6.

Patchin, J.W. (2022). *Summary of Our Cyberbullying Research (2007-2021)*. [online] Cyberbullying Research Center. Available at: https://cyberbullying.org/summary-of-our-cyberbullying-research [Accessed 10 Jan. 2024].

Paul, S. and Saha, S. (2020). CyberBERT: BERT for cyberbullying identification. *Multimedia Systems*. doi:https://doi.org/10.1007/s00530-020-00710-4.

Plaza-Del-Arco, F., Nozza, D. and Hovy, D. (2023). *Respectful or Toxic? Using Zero-Shot Learning with Language Models to Detect Hate Speech*.

Radford, A., Wu, J., Rewon Child, Luan, D., Amodei, D. and Ilya Sutskever (2019). *Language Models are Unsupervised Multitask Learners*. [online] Available at: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe [Accessed 11 Jan. 2024].

Rajamanickam, S., Mishra, P., Ai, F., Yannakoudakis, H. and Shutova, E. (2020). *Joint Modelling of Emotion and Abusive Language Detection*. [online] Association for Computational Linguistics, pp.4270–4279. Available at: https://aclanthology.org/2020.acl-main.394.pdf [Accessed 8 Jan. 2024].

Ribeiro, M.T., Singh, S. and Guestrin, C. (2016). *'Why Should I Trust You?': Explaining the Predictions of Any Classifier*. [online] arXiv.org. Available at: https://arxiv.org/abs/1602.04938 [Accessed 11 Jan. 2024].

Rohit Kumar Kaliyar, Goswami, A., Sharma, U.K. and Kanika Kanojia (2023). ACDNet: Abusive Content Detection on Social Media with an Effective Deep Neural Network Using Code-Mixed Hinglish Data. *Communications in computer and information science*, pp.282–293. doi:https://doi.org/10.1007/978-3-031-35644-5_22.

Saini, H., Mehra, H., Rani, R., Jaiswal, G., Sharma, A. and Dev, A. (2023). Enhancing cyberbullying detection: a comparative study of ensemble CNN–SVM and BERT models. *Social Network Analysis and Mining*, 14(1). doi:https://doi.org/10.1007/s13278-023-01158-w.

Salminen, J., Hopf, M., Chowdhury, S.A., Jung, S., Almerekhi, H. and Jansen, B.J. (2020). Developing an online hate classifier for multiple social media platforms. *Human-centric Computing and Information Sciences*, 10(1). doi:https://doi.org/10.1186/s13673-019-0205-6.

Sameer, M. (2022). Hate Speech Detection in a mix of English and Hindi-English (Code-Mixed) Tweets. *Theses*. [online] Available at: https://repository.rit.edu/theses/11160/ [Accessed 10 Jan. 2024].

Samek, W., Wiegand, T. and Müller, K.-R. (2017). *Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models*. [online] arXiv.org. Available at: https://arxiv.org/abs/1708.08296 [Accessed 11 Jan. 2024].

Sanguinetti, M., Poletto, F., Bosco, C., Patti, V. and Stranisci, M. (2018). *An Italian Twitter Corpus of Hate Speech against Immigrants*. [online] ACLWeb. Available at: https://aclanthology.org/L18-1443/ [Accessed 8 Jan. 2024].

Sap, M., Card, D., Gabriel, S., Choi, Y. and Smith, N.A. (2019). The Risk of Racial Bias in Hate Speech Detection. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. doi:https://doi.org/10.18653/v1/p19-1163.

Shah, K., Phadtare, C. and Rajpara, K. (2022). Cyber Bullying Detection for Hindi-English Language Using Machine Learning. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4116143.

Sharon, R.A., Shah, H., Mukherjee, D. and Gupta, V. (2022). Multilingual and Multimodal Abuse Detection. doi:https://doi.org/10.48550/arxiv.2204.02263.

Shewale, R. (2024). *Social Media Users — How Many People Use Social Media In 2022*. [online] Demand Sage. Available at: https://www.demandsage.com/social-media-users/ [Accessed 29 Feb. 2024].

Singh, K. (2020). *How to Improve Class Imbalance using Class Weights in Machine Learning*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/ [Accessed 11 Jan. 2024].

Smith, P.K., Mahdavi, J., Carvalho, M., Fisher, S., Russell, S. and Tippett, N. (2008). Cyberbullying: its nature and impact in secondary school pupils. *Journal of child psychology and psychiatry, and allied disciplines*, [online] 49(4), pp.376–85. doi:https://doi.org/10.1111/j.1469-7610.2007.01846.x.

Strubell, E., Ganesh, A. and McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *arXiv:1906.02243 [cs]*. [online] Available at: https://arxiv.org/abs/1906.02243 [Accessed 11 Jan. 2024].

Su, H.-P., Huang, Z.-J., Chang, H.-T., and Lin, C.-J. (2017). Rephrasing profanity in Chinese text. In *Proceedings of the First Workshop on Abusive Language Online (ALW1), the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017),* pp. 18-24.

Sun, C., Qiu, X., Xu, Y. and Huang, X. (2019). How to Fine-Tune BERT for Text Classification? *Lecture Notes in Computer Science*, pp.194–206. doi:https://doi.org/10.1007/978-3-030-32381-3_16.

Talukder, S. (2019). *Detection and Prevention of Abuse in Online Social Networks Detection and Prevention of Abuse in Online Social Networks*.

Toliyat, A., Levitan, S.I., Peng, Z. and Etemadpour, R. (2022). Asian hate speech detection on Twitter during COVID-19. *Frontiers in Artificial Intelligence*, 5. doi:https://doi.org/10.3389/frai.2022.932381.

Vidgen, B. and Derczynski, L. (2020). Directions in abusive language training data, a systematic review: Garbage in, garbage out. *PLOS ONE*, 15(12), p.e0243300. doi:https://doi.org/10.1371/journal.pone.0243300.

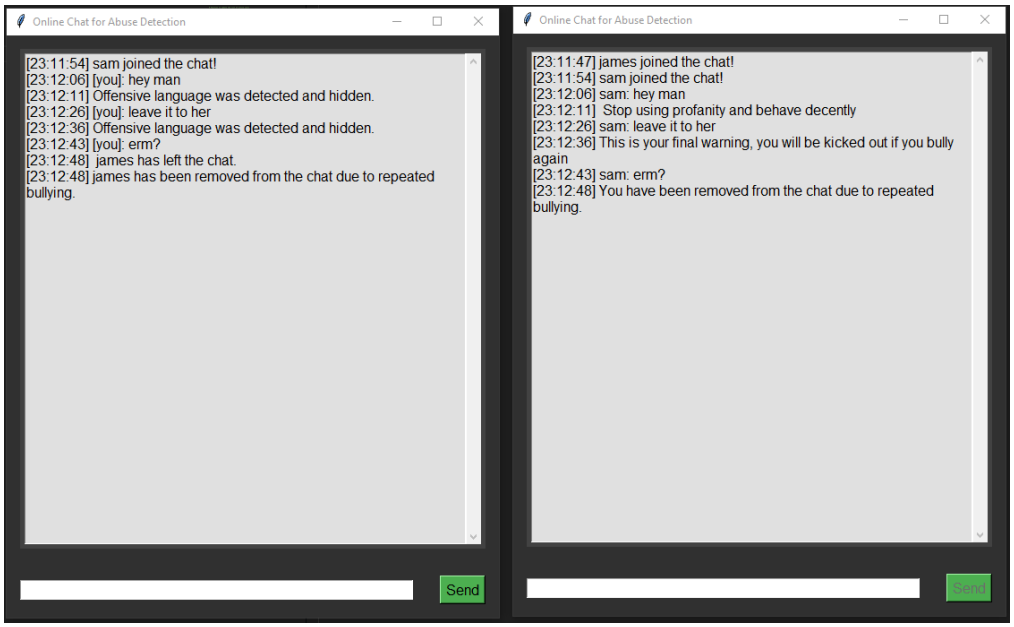
Waseem, Z. and Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *Proceedings of the NAACL Student Research Workshop*. [online] doi:https://doi.org/10.18653/v1/n16-2013.

Zhang, Y. and Wallace, B. (2017). *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification*. pp.253–263.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. and Xu, B. (2016). *Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling*.

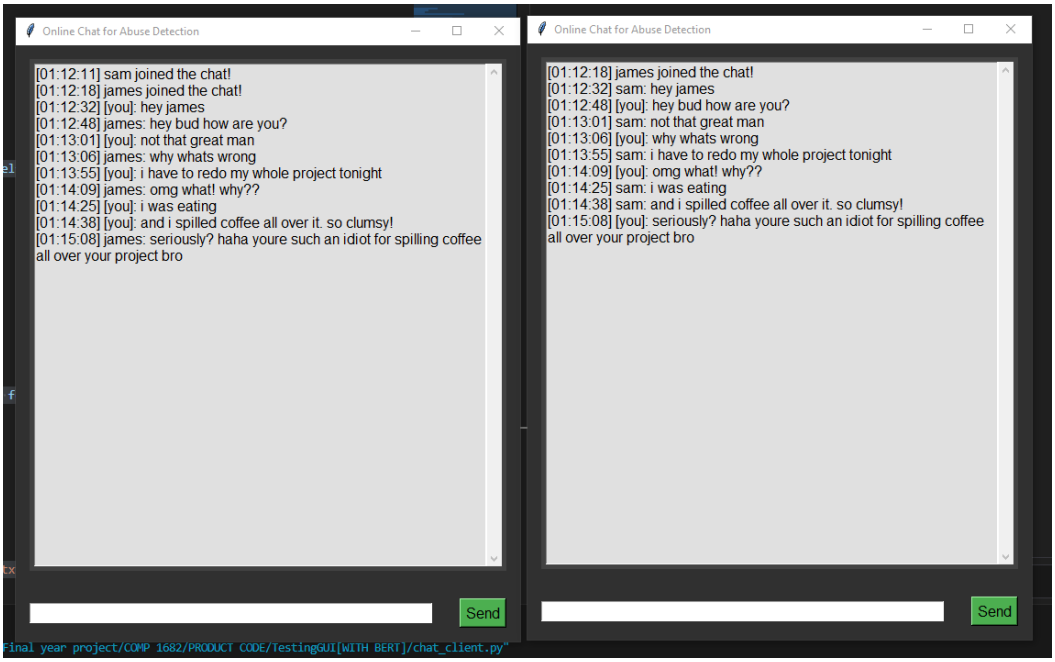
10. Appendices

10.1 Abuse Detection and Prevention

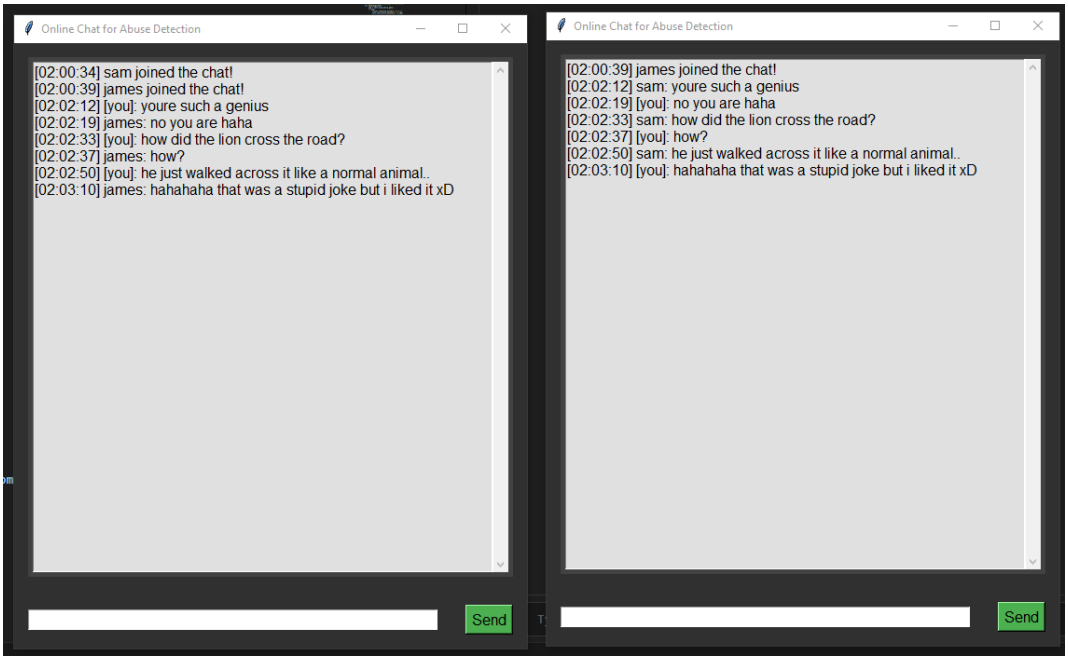


Abuse Detection Figure 1 - Showing the system removing the user after 2 warnings.

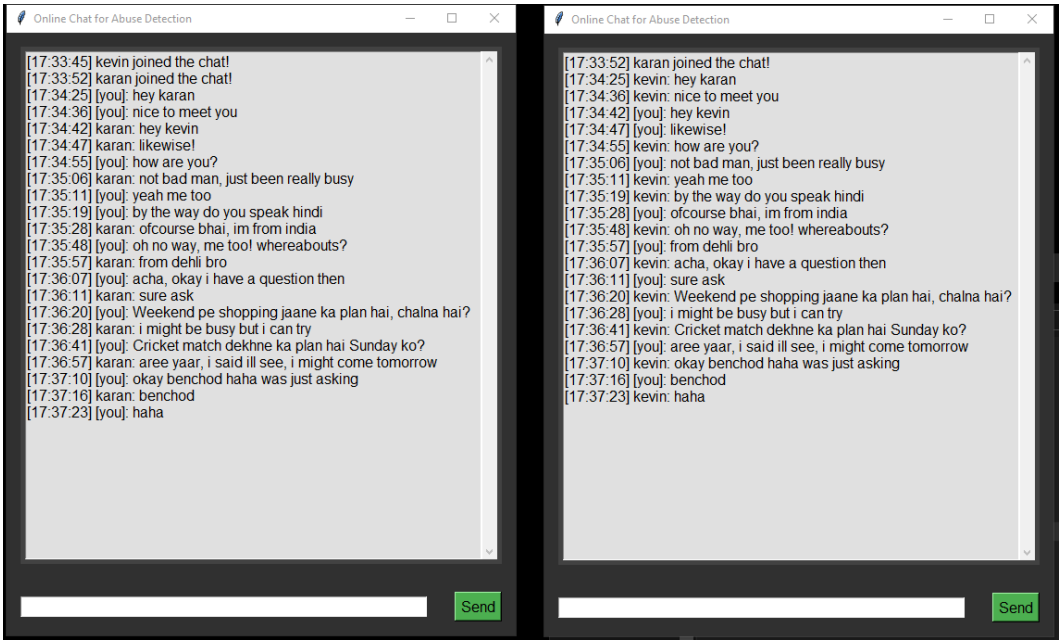
10.2 Buffer System



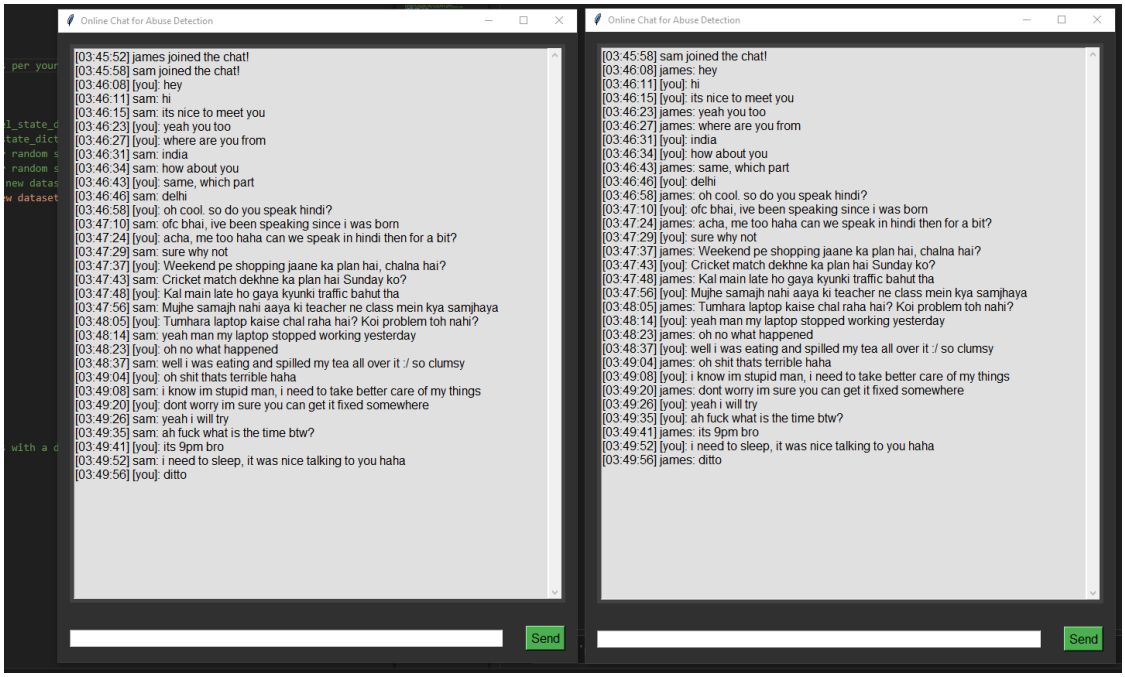
Buffer Figure 1 - "Idiot" not misclassified but understood within context.



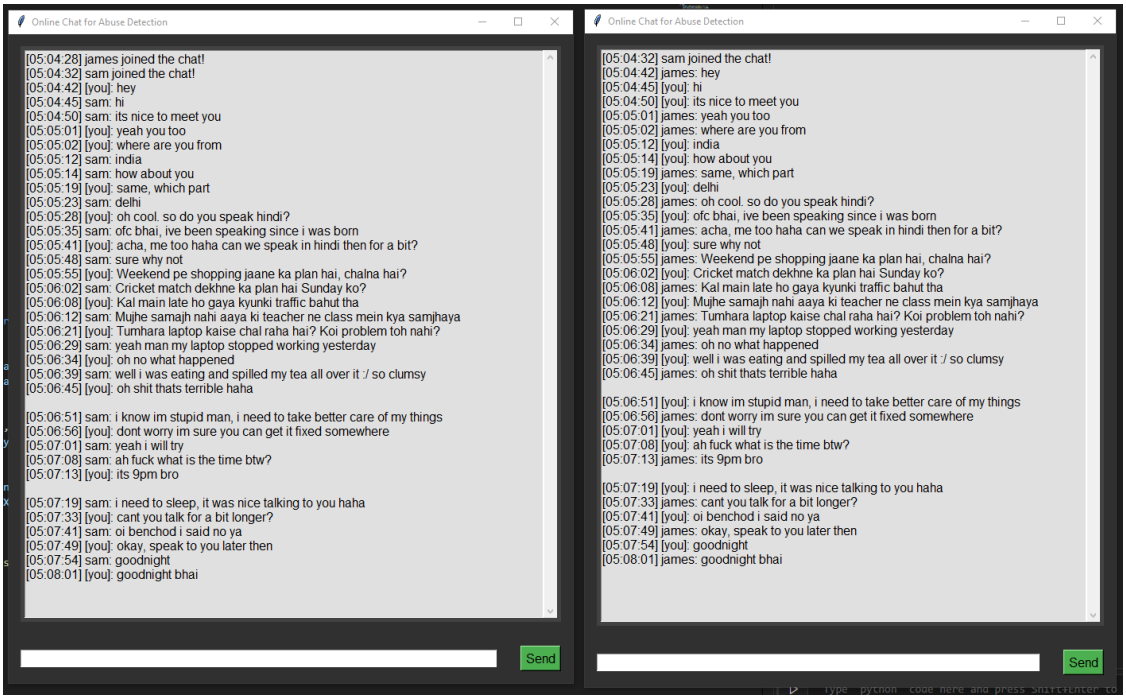
Buffer Figure 2 - "stupid" not misclassified but understood within context.



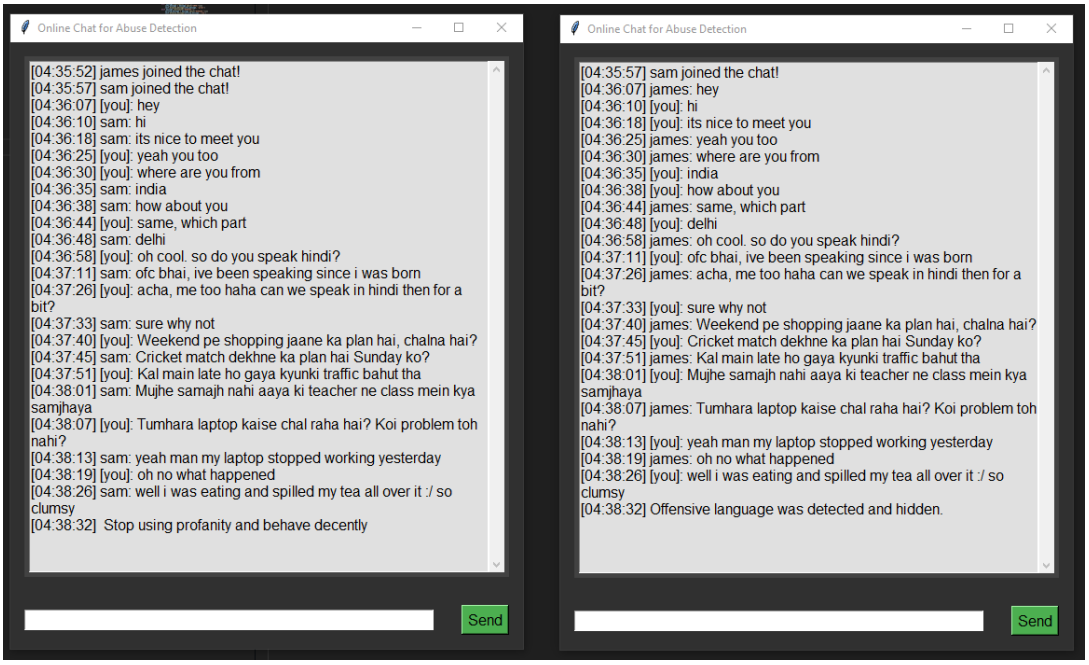
Buffer Figure 3 - "benchod" Indian word not misclassified but understood within context.



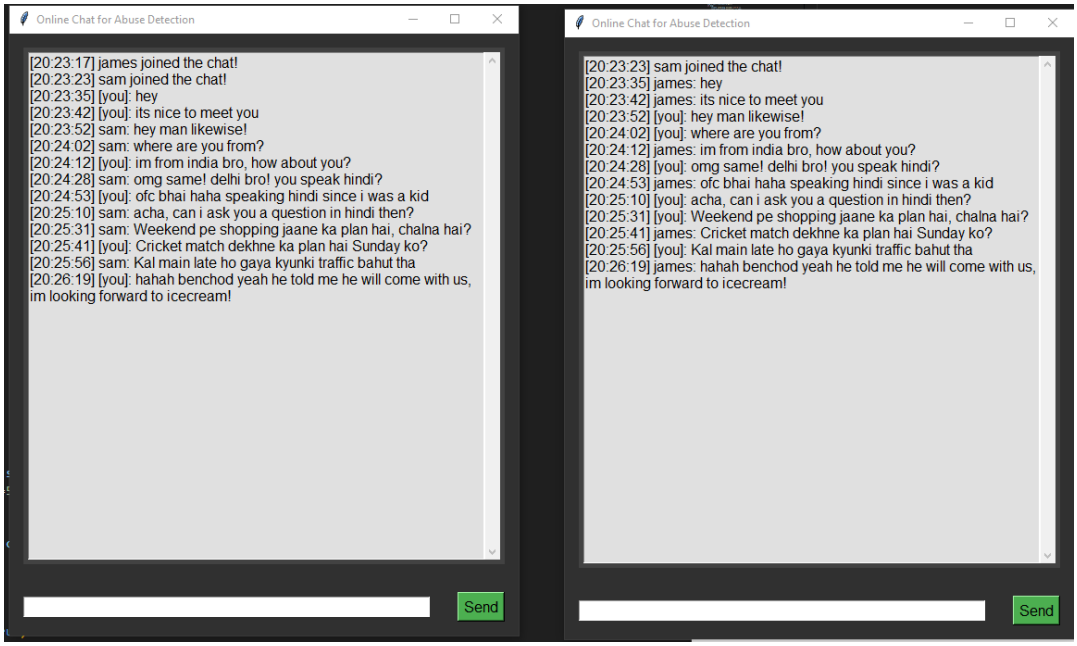
Buffer Figure 4 - Transition between English and Hindi, and words like "shit", "stupid", "fuck" are abusive but not misclassified as it understands within context.



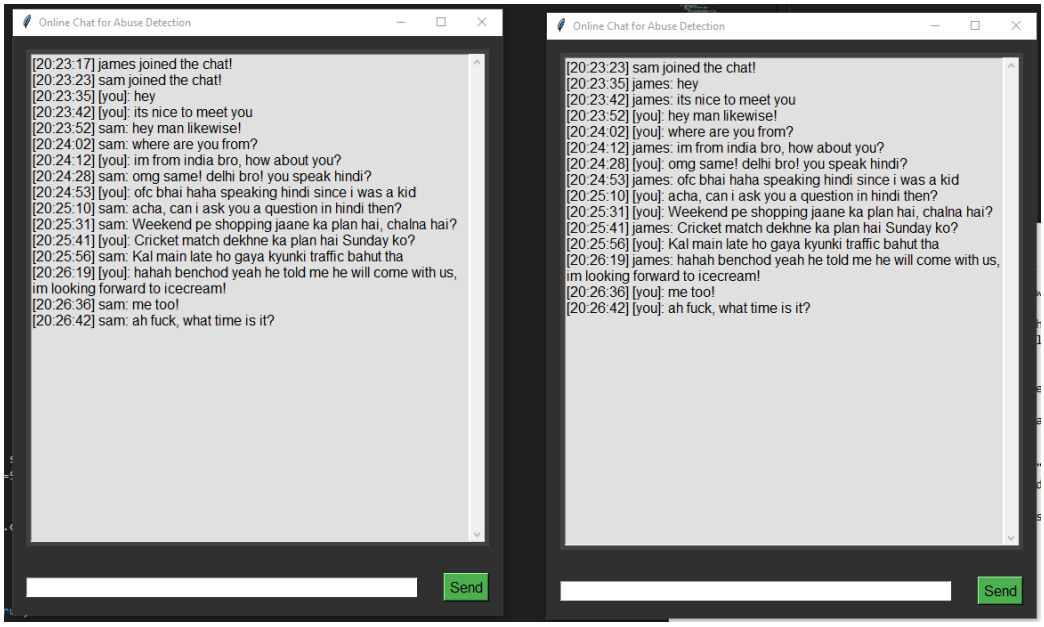
Buffer Figure 5 - Transition between English and Hindi, and English words like "shit", "stupid", "fuck" combined with Hindi word like "benchod" are abusive but not misclassified as it understands within context.



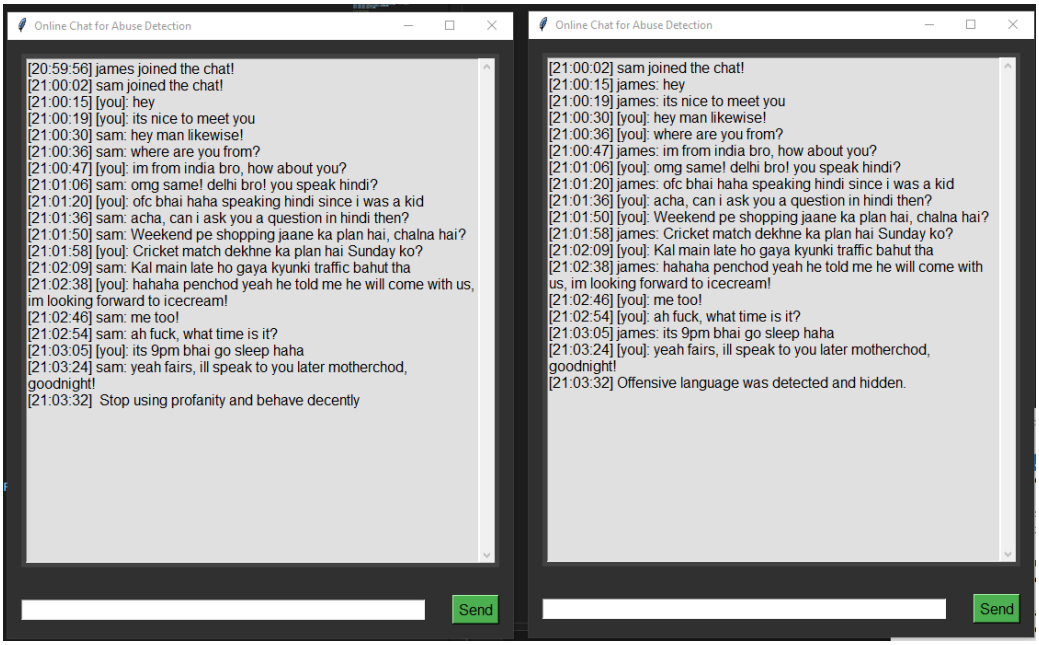
Buffer Figure 6 - NO BUFFER USED, and we can see it misclassifies word "shit" and does NOT detect context.



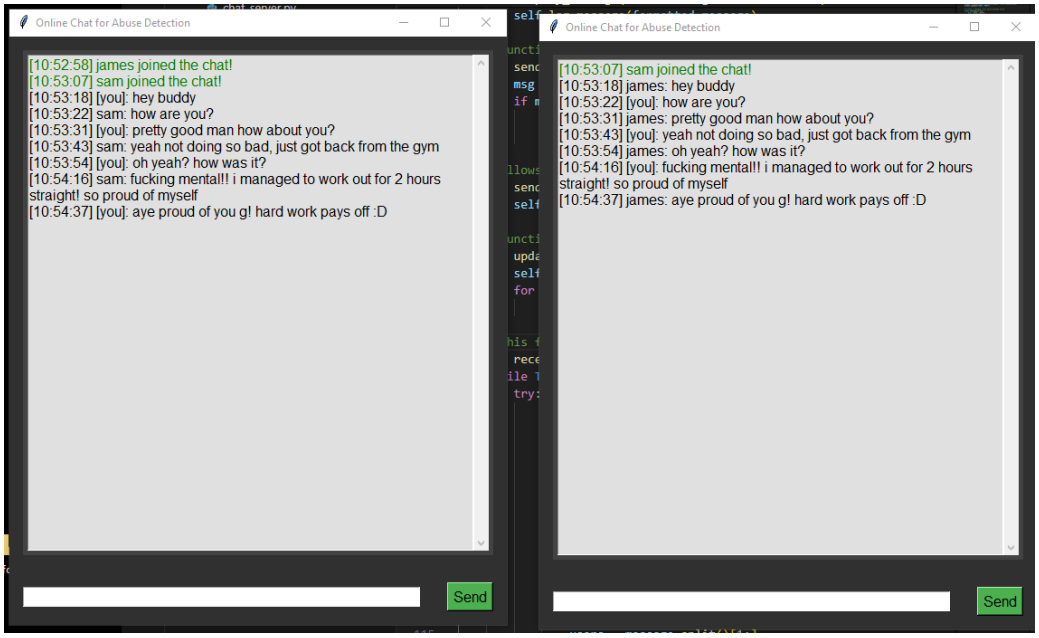
Buffer Figure 7 - Another example of using Hindi abusive text "benchod" but system understands context.



Buffer Figure 8 - "fuck" not misclassified but understood within context.



Buffer Figure 9 – "penchod" and "Motherchod" not misclassified but understood within context.



Buffer Figure 10 - "fucking" not misclassified but understood within context.