MULTI-CLASS CLASSIFICATION OF TEXTUAL DATA: DETECTION AND MITIGATION OF CHEATING IN MASSIVELY MULTIPLAYER ONLINE ROLE PLAYING GAMES

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

By

NAGA SAI NIKHIL MAGULURI
B.Tech., Jawaharlal Nehru Technological University, India, 2015

2017
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Naga Sai Nikhil Maguluri ENTITLED Multi-class Classification of Textual Data: Detection and Mitigation of Cheating in Massively Multiplayer Online Role Playing Games BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Michelle A. Cheatham, Ph.D.
Thesis Director

Mateen M. Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on Final Examination

Michelle A. Cheatham, Ph.D.

Tanvi Banerjee, Ph.D.

Mateen M. Rizki, Ph.D.

Robert E.W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School
ABSTRACT

Maguluri, Naga Sai Nikhil. M.S., Department of Computer Science and Engineering, Wright State University, 2017. Multi-class Classification of Textual Data: Detection and Mitigation of Cheating in Massively Multiplayer Online Role Playing Games.

The success of any multiplayer game depends on the player’s experience. Cheating/Hacking undermines the player’s experience and thus the success of that game. Cheaters, who use hacks, bots or trainers are ruining the gaming experience of a player and are making him leave the game. As the video game industry is a constantly increasing multibillion dollar economy, it is crucial to assure and maintain a state of security.

Players reflect their gaming experience in one of the following places: multiplayer chat, game reviews, and social media. This thesis is an exploratory study where our goal is to experiment and propose a new way to detect, mitigate cheating in Massively Multiplayer Online Role Playing Games by performing a multiclass classification on these unstructured textual data to categorize cheaters and victims with good classification accuracy that is acceptable for practical applications.

In this thesis, First, we have studied the current situation regarding cheating and anti-cheating in online games. Second, we have studied various Natural Language Processing and Machine learning methods and tools for text classification. Third, a general
method for automatic player categorization is proposed and finally, its performance is evaluated by experimenting on various datasets.
# Table of Contents

1. Introduction .............................................................................................................. 1  
   1.1. Overview ........................................................................................................ 1  
   1.2. Motivation for Cheaters .................................................................................. 4  
   1.3. Problem Statement ......................................................................................... 4  
   1.4. Current Trends ............................................................................................... 5  
   1.5. Purpose, Scope, and Contribution ................................................................. 6  
   1.6. Methodology ................................................................................................... 7  

2. Literature Survey ..................................................................................................... 9  
   2.1. Cheating in Online Gaming ........................................................................... 9  
      2.1.1. Overview ................................................................................................. 9  
      2.1.2. Game Types ......................................................................................... 10  
      2.1.3. Qualitative risk analysis of Game types ............................................... 10  
      2.1.4. General Architecture of MMO and MO games .................................. 11  
      2.1.5. Cheats and Exploits ............................................................................. 13  
      2.1.1. Anti-cheat Software .............................................................................. 21  
      2.1.2. Related Work ....................................................................................... 23  
   2.2. Machine Learning ............................................................................................ 26  
      2.2.1. Overview ............................................................................................... 26  
      2.2.2. Types of Machine Learning ................................................................. 27  
   2.3. Natural Language Processing ......................................................................... 28  
      2.3.1. Overview ............................................................................................... 28  
      2.3.2. N-grams ............................................................................................... 29  
      2.3.3. Tokenization and Sentence Segmentation .......................................... 29  
      2.3.4. Term Frequency – Inverse Document Frequency ................................ 30  
      2.3.5. Stemming .............................................................................................. 30  
   2.4. Automatic Text Categorization ....................................................................... 31  
      2.4.1. Overview ............................................................................................... 31  
      2.4.2. Logistic Regression ............................................................................... 31  
      2.4.3. Naïve Bayes ......................................................................................... 33  
      2.4.4. Random Forest ..................................................................................... 34  
      2.4.5. Support Vector Machine ...................................................................... 35  
      2.4.6. Related Work ....................................................................................... 35
3. Building Classifier ..............................................................................................................40
  3.1. Introduction ..................................................................................................................40
  3.2. Identification of Keywords ...........................................................................................42
  3.3. Datasets ........................................................................................................................43
      3.3.1. Data Collection SM-GEN, SM-CSGO .................................................................43
      3.3.2. Data Collection MC-TF2C, MC-TF2S .................................................................47
      3.3.3. Data Collection RV-CSGO ....................................................................................48
  3.4. Labeling the data ..........................................................................................................49
  3.5. Preprocessing the data ...............................................................................................52
      3.5.1. SM-GEN, SM-CSGO ..........................................................................................52
      3.5.2. MC-TF2C, MC-TF2S .........................................................................................55
      3.5.3. RV-CSGO ............................................................................................................56
  3.6. Feature Extraction .......................................................................................................56

4. Experiment and Results ....................................................................................................65
  4.1. Introduction ..................................................................................................................65
  4.2. Classifiers ....................................................................................................................65
  4.3. Evaluation Metric .........................................................................................................66
  4.4. Feature Selection .........................................................................................................67
  4.5. Performance Analysis .................................................................................................70

5. Conclusion and Future Work ............................................................................................84

Bibliography ..........................................................................................................................88
List of Figures

Figure 1. A vendor selling hacks for Counter-Strike: Global Offensive on eBay 3
Figure 2. Methodology 7
Figure 3. Qualitative risk analysis on game type 11
Figure 4. General architecture of MMO and MO games 12
Figure 5. Screenshot of a player using aimbot 14
Figure 6. Screenshot of a player using wallhack 15
Figure 7. Screenshot of a player using Anti-aim 16
Figure 8. Screenshot of a player using multi-hack 17
Figure 9. Screenshot of a player using no-smoke 18
Figure 10. Screenshot of a player using ESP hack 19
Figure 12. Proposed algorithm workflow 41
Figure 13. Identification of keywords 42
Figure 14. Word cloud of keywords 43
Figure 15. Data collection from Twitter 45
Figure 16. Word cloud of victim after data cleaning 59
Figure 17. Word cloud of cheater after data cleaning 59
Figure 18. Word cloud of victim before data cleaning 60
Figure 19. Word cloud of cheater before data cleaning 61
Figure 20. Performance evaluation of single features 68
Figure 21. Performance evaluation of different combination of features ...................... 69
Figure 22. Confusion matrix and data distribution plot for feature D on SM-GEN train dataset .......................................................... 71
Figure 23. Confusion matrix and data distribution plot for feature S on SM-GEN train dataset .......................................................... 71
Figure 24. Confusion matrix and data distribution plot for feature P on SM-GEN train dataset .......................................................... 72
Figure 25. Confusion matrix and plot for feature L on SM-GEN train dataset .......... 72
Figure 26. Confusion matrices of different classifiers starting from top left LR, NB, RF, SVM for feature combination D-S-P-L .......................................................... 73
Figure 27. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on SM-GEN test data set .............................................. 74
Figure 28. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on SM-CSGO data set .............................................. 76
Figure 29. Filled map showing geographic locations where cheating is predominant. 78
Figure 30. Map showing locations in the USA where cheating is dominant .......... 78
Figure 31. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on MC-TF2C data set .............................................. 79
Figure 32. Boxplot of damage ........................................................................ 80
Figure 33. Box plot of kills ............................................................................ 81
Figure 34. Boxplot of scores .......................................................................... 82
Figure 35. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on RV-CSCO data set ................................................................. 83
List of Tables

Table 1. Video Game types classification ................................................................. 10
Table 2. Cheat availability for various FPS game ......................................................... 20
Table 3. Hack developing and selling Websites ............................................................ 21
Table 4. Different Anti-cheat mechanisms ....................................................................... 22
Table 5. Anti-Cheat software providers ........................................................................... 23
Table 6. Systemic classification of cheating in video games by Yan and Randell (2005) ................................................................. 23
Table 7. Statistics of ALL_TWEETS dataset .................................................................. 45
Table 8. Data extracted from JSON ................................................................................ 47
Table 9. Relevant information available in MC-TF2S ..................................................... 48
Table 10. Relevant information available in MC-TF2C ..................................................... 48
Table 11. Information available in RV-CSGO .................................................................. 49
Table 12. Stats of SM-GEN dataset ................................................................................ 51
Table 13. Stats of SM-CSGO dataset ............................................................................... 52
Table 14. Stats of MC-TF2C dataset ................................................................................ 56
Table 15. Stats of RV-CSGO dataset ............................................................................... 56
Table 16. Personal pronouns ............................................................................................ 62
Table 17. Some example documents and their extracted features ...................................... 63
Table 18. Interrogative pronouns ...................................................................................... 64
Table 19. Extracted Features and their representation .......................................................... 64
Table 20. Single Feature Evaluation on SM-GEN ................................................................. 68
Table 21. Combination of features evaluation on SM-GEN .................................................. 69
Table 22. F-scores of classifiers on SM-GEN test dataset ..................................................... 75
Table 23. F-scores of classifiers on SM-CSGO dataset .......................................................... 76
Table 24. F-scores of classifiers on MC-TF2C dataset ......................................................... 79
Table 25. F-scores of classifiers on RV-CSGO dataset .......................................................... 83
ACKNOWLEDGEMENTS

This work would not have been possible without the financial support from the Wright State University Department of College of Engineering and Computer Science under Assistantship Award Scholarship. I am especially indebted to Dr. Ramana Grandhi, Executive Director of International Collaborations and Graduate Programs for the financial support, and Dr. Michelle Cheatham, co-lead of the Data Semantics (DaSe) Lab and my thesis director, who have been supportive of my career goals and who worked actively to provide me with the protected academic time to pursue those goals.

I am grateful to all of those with whom I have had the pleasure to work during this and other related projects. Each of the members of my Thesis Committee, Dr. Tanvi Banerjee and Dr. Mateen Rizki, has provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general. I would especially like to thank Dr. Mateen Rizki, the chairman of Department of Computer Science and Engineering for extending the funding for my Master’s degree.

Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to thank my parents and my sister, whose love and guidance are with me in whatever I pursue. They are the ultimate role models who provide unending inspiration.
1. Introduction

1.1. Overview

The revenue of the gaming industry is booming year over year. In 2015 there were more than 155 million Americans who played games, and this averages to at least two players in each game-playing household. The international gaming industry was worth $91.5 billion in 2015 and is expected to reach $107 billion in 2017, with a whopping increase of 9.4% year over year (Essential Facts About the Computer and Video Game Industry, 2016). Competitive gaming is now considered a professional sport in the United States (Makuch, 2013).

The success of any online game depends on a player’s experience. Cheating undermines the player’s experience and thus the success of that online game. Yan and Randell (2005) define cheating as follows:

“Any behavior that a player uses to gain an advantage over his peer players or achieves a target in an online game is cheating if, according to the game rules or at the discretion of the game operator (i.e. the game service provider, who is not necessarily the developer of the game), the advantage or the target is one that he is not supposed to have achieved.”
When a player buys a game, and installs it on their device, the developers have no control over the device of the player. But the player has access to all of the device hardware. Hence the files, memory, services, drivers, executables and finally the game is not secure. A client can hack the game in many ways: Patches can be applied on executables to change the game behavior, Game data files can be changed to manage the game properties, Network packets can be captured, decoded and altered to modify the game commands.

Hacking/cheating is not new to the gaming industry. Cheaters, who use hacks, bots or trainers are ruining the gaming experience of other players and can make them leave the game. Many online multiplayer games, such as Blizzard’s Diablo and Ensemble Studio’s Age of Empires, which were best sellers in the early days of their release, have lost their significant reputation later because of cheating. Many tournaments were canceled, as the gamers quit due to lack of trust in the game. (Pritchard, 2000).

The gravity of problems created by cheating in a game depends on its type. If a game is a single player, then there is nothing to worry about as the cheater is only affecting himself and is happy in doing so. The success of a company is at stake if the game is multiplayer only. As the number of people who play games has risen (Essential Facts About the Computer and Video Game Industry, 2016) and as it is more fun to play with/against other players instead of the computer, the online gameplay is becoming an integral part of the gaming industry as it draws a huge audience. Hence, it is more important for the developers to ensure that the experience of every online game player is authentic and candid. The anonymity of the internet is encouraging players to cheat, and cheating pursuits increase with the success of the game. One can get a bigger picture of how serious the
problems created by cheating to an online game are, by searching for that massively multiplayer online game in shopping websites like eBay where several sellers are getting real money by selling some cheat or hack. Figure 1 shows a vendor selling hacks for Counter-Strike: Global Offensive.

![Figure 1. A vendor selling hacks for Counter-Strike: Global Offensive on eBay](image)

To tackle cheaters, video game developers have to spend millions of dollars and resources in releasing multiple patches to invalidate the cheat. The release of patches is not enough to stop the cheaters as they will find another way around (Pritchard, 2000). These days, cheaters are using advanced tools that are easy to use and can bypass the cheat detection models. One way to prevent cheating is to run the game on hardware, whose internals are not in control of the player by using cloud services. But, most games require huge amounts of disk space and huge processing power to run. Running these graphic
intensive games in the cloud is not a good idea as it costs millions of dollars for game developers and will effectively increase the cost of a game. Also, the cloud requires skilled programmers and frequent maintenance. In our thesis, we have explored a much more cost-effective way of mitigating the impact of cheaters.

1.2. Motivation for Cheaters

Researchers (Pritchard, 2000; Spohn, 2002; and Consalvo, 2005) have found out the real motives that drive players to cheat in an online multiplayer game. These motives include: to dominate in gameplay, to get unstuck, to annoy other players, to crush opponents, to gain in-game items at ease, to skip boring parts of gameplay and to make other players think that he is a God (that is to make others believe that he has good gaming skills).

1.3. Problem Statement

This master’s thesis is an exploratory study where our goal is to experiment and answer the question “can natural language processing and machine learning tools for text classification be effectively used for automatic identification of cheaters/victims in Massive Multiplayer Online Role Playing Games (MMORPGs)?”
1.4. Current Trends

Extensive research is carried out in the domains of automatic text classification and detection of cheating in online gaming. There are many software and hardware-based models, which help in the detection and prevention of cheating (Kim et. al., 2005; Laurens et. al., 2007; Feng et. al., 2008; Chapel et. al., 2010; Pao et. al., 2010; Galli et. al. 2011). However, most of these methods can detect a particular cheating technique and prevent that technique. Game developers often tend to not take cheating seriously even though it has grave consequences on a game. Most multiplayer game developers rely on some third party anti-cheat solution providers (Valve Anti-cheat, PunkBuster, Game Guard) which are unsophisticated in detecting cheating (Quintin, 2010; Meer, 2010).

Using Natural Language Processing (NLP) computer systems, the input human language can be processed, and desired output is achieved. Even though NLP has been successful in developing several applications like sentiment analysis of a product or movie, spam filtering, information extraction, question answering and text summarization (Speriosui et al., 2011; Brody and Diakopoulos, 2011; Jiang et al., 2011, Punuski et al., 2006; Youn et al., 2007, Xiao-li et al., 2009, Ramage et al., 2010; Jin et al., 2011), little research has been done in applying NLP towards game studies. In gaming, NLP has been used to create a novel gameplay where the player’s speech is converted into actions and are carried out in the game simulation. NLP has been used to analyze a player’s reviews of a game to obtain the player’s experience of that game (Mulholland et. al., 2015). But there
is no significant research in detection/mitigation of cheating in MMORPGs using textual analytics, and this thesis is the first step in achieving this goal.

1.5. Purpose, Scope, and Contribution

Players exhibit their gaming experience in one of the following places: multiplayer chat, game reviews, and social media. The purpose of this thesis is to experiment and propose a new way that helps in detection and mitigation of cheating in MMORPGs by performing a multiclass classification on these unstructured textual data with good classification accuracy that is acceptable for practical applications. We hope that this thesis inspires other researchers to identify other machine learning based approaches based on this report and experiment to expand it.

The scope of this thesis is limited to the English language, and all the datasets are assumed to contain valid data, that is the statements made by users/players are believed to be non-fiction. The datasets used in this thesis are associated with First Person Shooting (FPS) games and can be applied to any Massively Multiplayer Online (MMO) game with minor alterations.

We have studied various natural language processing and machine learning methods and tools for text classification. Also, we have reviewed current situation regarding cheating and anti-cheating in online games and a general method for automatic classification of a new document as a cheater, victim or neutral case is proposed and evaluated by experimenting on various datasets.
1.6. **Methodology**

Extracting textual and natural language content, identification of keywords to separate data with cheating information from normal gaming data, identification of features for automatic player categorization, building logistic regression, naïve Bayes, random forest and support vector machine classifiers, analyzing the performance of each classifier on different data sets. Retrieval of useful information from datasets after classification are the principal motives of this thesis. The methodology adopted to achieve these motives is summarized in Figure 2.

![Figure 2. Methodology](image)

To get an understanding of the field and to discover suitable ML algorithms and NLP methods for player categorization; first, we have reviewed relevant textual classification literature. After the relevant literature review, we came up with a generalized method that can be applied to any online multiplayer game and can do automatic player
categorization using text classification methods. The proposed general method is implemented and developed as a software framework so that support for various algorithms and tools can be achieved. The implemented framework is evaluated by performing several experiments of player categorization on different datasets of real gaming data. The framework’s performance and statistically analyzed results are documented. From these results, conclusions were drawn, and future work is proposed.

As the players exhibit their gaming experience in social media, multiplayer chat or reviews, the gaming data has been taken from three sources: Twitter, logs.tf and Steam. We have obtained data of user posts in social media regarding cheating from Twitter. We have obtained multiplayer chat logs and stats from logs.tf. Steam is an entertainment platform that offers games on PC, Mac, and Linux and provides paid access to games. Cross-platform multiplayer is one feature provided by steam where players from different platforms can all join in an online multiplayer game and play. We have obtained game reviews from Steam.

This thesis is organized as follows. Chapter 2 gives background and related work in the fields of NLP, ML, automatic text categorization and cheating in online gaming. Chapter 3 describes the automatic text classification models implemented in this work. Chapter 4 presents the experimental setup and discussion on the results obtained. Chapter 5 summarizes our work and outlines possible future extensions to the current work.
2. Literature Survey

This chapter introduces the specific concepts within cheating in online gaming, machine learning, natural language processing, and automatic text categorization that are important and essential to understand the main body of this thesis. This chapter is meant to give an overview of concepts for the reader with minimal to no knowledge of these fields and includes the current theoretical and methodological contributions of other researchers in those areas relevant to our thesis.

2.1. Cheating in Online Gaming

2.1.1. Overview

Cheating is predominant in most of the current multiplayer games. In this section, we summarize the game types, provide a general architecture of multiplayer gaming, various known methods of cheating, detection techniques and related work in detection and mitigation of cheating in online multiplayer gaming. From background study, it is interesting to note that, the game genre has a huge impact on the type of cheats available for a game. For example, most or all first person shooting (FPS) games are exposed to aimbots and wallhacks.
2.1.2. Game Types

The following table gives a possible video game type classification and the rules related to cheating.

<table>
<thead>
<tr>
<th>Game Type</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Player</td>
<td>Cheating can be allowed. Even game developers themselves facilitate cheats for players. There are no potential effects on another players’ gameplay.</td>
</tr>
<tr>
<td>Local Area Network/Peer to Peer</td>
<td>It is the wish of players on that network, whether they want to play with cheats or not. Verbal rules are made by players or by a local admin.</td>
</tr>
<tr>
<td>Split Screen Multiplayer</td>
<td>It is the wish of players/friends, whether they want to play with cheats or not. Verbal rules are made by players.</td>
</tr>
<tr>
<td>Multiplayer Online</td>
<td>Players must agree to the End User License Agreement (EULA). Cheating or hacking is forbidden in these games.</td>
</tr>
<tr>
<td>Massive Multiplayer Online</td>
<td>Players must agree to the End User License Agreement (EULA). Cheating or hacking is prohibited in these games.</td>
</tr>
</tbody>
</table>

2.1.3. Qualitative risk analysis of Game types

We can now present a qualitative risk analysis by considering the chance of occurrence of cheaters and the number of players affected. The highest risk of cheating can be observed in MO and MMO games in Figure 3.
2.1.4. General Architecture of MMO and MO games

As the MO and MMO have a high risk of cheating, our focus is on these game types. The general architecture of an online multiplayer game consists of an admin or game master (who has created a multiplayer game instance and has some special privileges), normal players (one or more) and a game server. Figure 4 shows the architecture of an online multiplayer game, the hardware and software components that are usually used by the game software are shown.
Figure 4. General architecture of MMO and MO games

Client

Software: OS, Drivers, Video Rendering API (DX, Open GL), Game client with special abilities

Hardware: CPU, Memory, Video card, Network card

Server

Database: Client Information, Game Configuration, Game Statistics

Game Server Application

Internet

Admin

Player

Client

Software: OS, Drivers, Video Rendering API (DX, Open GL), Game client

Hardware: CPU, Memory, Video card, Network card
2.1.5. Cheats and Exploits

Anything that is used to gain an unfair advantage in a game play by a player is known as cheating. Game developers may include cheats in single player games, but they are strictly banned in multiplayer games. A glitch or bug is an exploit in a game code that is abused by cheaters to attain an unfair advantage. There are several ways through which a player can achieve unfair advantage in gameplay, which are described as below:

- displaying critical gameplay information
- modifying game behavior
- automating or simulating actions
- code injection into the remote process by DLL injection or thread hijacking.
- manipulating remote process data
- installing new drivers into the operating system
- changing the operating system configuration

Following are the cheats that are frequently used in an FPS game, and screenshots of some cheats are given wherever it is applicable.

- Aimbot: One of the most used hacks, which is also used in combination with other hacks. A program or script through which a player can get aid by locking the target automatically and giving them a fast headshot. Figure 5 is a screenshot from the game Counter-Strike, where the player is using an aimbot. As aimbot, helps in the automatic locking of a target, you can see a red colored
square on the opposite team's player which turns green once it is locked, and then the hacker can shoot him.

![Screenshot of a player using aimbot](image)

**Figure 5.** Screenshot of a player using aimbot

- **Wallhack:** A hack through which a player can see through walls, move through walls, and shoot through walls. Figure 6 shows the screenshot of a hacker using a wallhack: the wall in front of the hacker becomes transparent and shows him the players behind the wall.

- **Lagswitch:** A hack through which a player can cut off the outbound signal for a duration of 1 to 10 seconds, and during this time a player can move around and attack while being invulnerable. For others, players using lag switch looks like they are teleporting.
- Spinbot: A hack using this it becomes impossible for other players to lock and shoot, as the target always spins around. Players use spin bot along with aimbot so that they can aim and shoot with ease while spinning around makes it difficult for other players to hit them.

![Figure 6. Screenshot of a player using wallhack](image)

- Trigger-bot: A program or script that can shoot automatically when the crosshair of a player is on the enemy. Perfect headshots can be achieved when trigger bot is used along with aimbot as one aids to aim and the other to shoot.

- No-clip: A hack through which a player can control the character camera to move it in any direction and can also pass through walls.
- No-recoil: A program or script that eliminates the recoil of the gun, making the crosshair anchored to a point so that the player can do a continuous shot comfortably.

- Maphack: A program or script that can reveal the entire game map area, which helps a player to find the location of other players and shoot them from far away.

- No-spread: A hack that directs all of the bullets from a gun to a single point so that, a player can shoot other players far away with high accuracy.

- Anti-aim: A hack that flips the hitboxes, that is, turns the body of the player's character upside down so that the player does not lose significant hit points since bullets hitting the head now cause only foot damage. (Generally hit points are more for the head).

Figure 7. Screenshot of a player using Anti-aim
• **Silent-aim**: A hack that allows a player to shoot other players who are outside of the crosshair but inside the player's Point of View (POV). This means the player can shoot the other player when he enters the player's POV without even aiming at him. Figure 7 shows the screenshot of a hacker using silent-aim; we can observe that even though the crosshair is not exactly on the opposite player, he is shot as he is in the POV of the hacker.

• **Multihack**: A combination of two or more hacks listed in the table. Usually, vendors who sell hacks use this word. Figure 8 shows the screenshot of a hacker using a multi-hack, the hacker can select any combination of hacks from the hack menu.

![Figure 8. Screenshot of player using multi-hack](image-url)
• No-smoke: A hack which gives the player the ability to see through smoke from a smoke bomb. Figure 9 shows the screenshot of a hacker using a no-smoke hack; the hacker can see clearly since the smoke from the smoke bomb is slim.

• No-flash: A hack which gives the player the ability to see through the flash from a flash bomb.

• ESP: Extrasensory perception gives the player the ability to know contextual information like another players' health, ammo, and location. The figure shows the screenshot of a hacker using ESP, which is showing some additional information about the other players.

Figure 9. Screenshot of a player using no-smoke
Figure 10. Screenshot of a player using ESP hack

Table 2 gives the cheat availability in some of the most popular FPS games, and from this table, it is evident that most of the FPS games have the same type of bots. Hence genre of the game plays an active role in identifying the types of cheats available for that game.

Table 3 gives a list of websites that sell game cheats.
<table>
<thead>
<tr>
<th>Game</th>
<th>Aimbot</th>
<th>Wallhack</th>
<th>Lagswitch</th>
<th>Spinbot</th>
<th>Triggerbot</th>
<th>Noclip</th>
<th>Norecoil</th>
<th>Nospread</th>
<th>Anti-aim</th>
<th>silent-aim</th>
<th>MapHack</th>
<th>Multihack</th>
<th>Ragehack</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Cheat availability for various FPS games.
<table>
<thead>
<tr>
<th>Website</th>
<th>Number of games</th>
</tr>
</thead>
<tbody>
<tr>
<td>X22 Cheats</td>
<td>33</td>
</tr>
<tr>
<td>ArtificialAiming</td>
<td>40</td>
</tr>
<tr>
<td>DamnCheaters</td>
<td>29</td>
</tr>
<tr>
<td>AimJunkies</td>
<td>33</td>
</tr>
<tr>
<td>Virtual Advantage</td>
<td>18</td>
</tr>
<tr>
<td>Catalyst Hax</td>
<td>14</td>
</tr>
<tr>
<td>Optimal Aim</td>
<td>12</td>
</tr>
</tbody>
</table>

2.1.1. **Anti-cheat Software**

Anti-cheating measures are the actions taken by game developers to curb cheating in online gaming. Developers didn't think much of game security in the early 1990s, but as the online gaming became popular year after year and emerged as massively multiplayer online gaming, seriousness towards game security has increased. This section gives an overview of anti-cheat software mechanisms and current software used by various games.

Anti-cheat software providers do not reveal the inner mechanisms employed by them. Some common mechanisms employed by anti-cheat software that are known publicly are listed in Table 4.
Table 4. Different Anti-cheat mechanisms

<table>
<thead>
<tr>
<th>Anti-cheat Mechanism</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File checksums</td>
<td>Checksums are calculated for critical game files, typically using the MD5-algorithm. Before joining the game, checksums for the player's game files are computed and verified with the list of checksums in the server. The player is not allowed to join the game if there is any mismatch of checksum.</td>
</tr>
<tr>
<td>Process monitoring</td>
<td>Game hacks often run as separate processes. Anti-cheat software checks for these processes and terminates the game if it finds any suspicious process.</td>
</tr>
<tr>
<td>Memory Scanning</td>
<td>Anti-cheat software also scans the memory of the computer to detect any suspicious behavior.</td>
</tr>
<tr>
<td>Dynamic Memory Addressing</td>
<td>Game hacks try to modify the game variables in memory. To achieve this, the hacker has to first find the value of the variable that he must change and then alter the value. This can be stopped by dynamic memory addressing, that is to move the critical game data around memory randomly.</td>
</tr>
<tr>
<td>Ban list</td>
<td>Every anti-cheat software maintains a list that contains names of players who are caught cheating in the online multiplayer game. A user who is caught cheating in a game secured by an anti-cheat software is not allowed to play in any other games secured by that anti-cheating software.</td>
</tr>
</tbody>
</table>

There are several anti-cheat software providers in the market. The mechanisms implemented by this software are not disclosed. Some of the top anti-cheat software and the number of online multi-player games covered by them are tabulated below.
Table 5. Anti-Cheat software providers

<table>
<thead>
<tr>
<th>Anti-cheat software</th>
<th>Number of MMO games protected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valve Anti-Cheat</td>
<td>50</td>
</tr>
<tr>
<td>PunkBuster</td>
<td>23</td>
</tr>
<tr>
<td>GameGuard</td>
<td>13</td>
</tr>
<tr>
<td>DMW World</td>
<td>8</td>
</tr>
<tr>
<td>HackShield</td>
<td>8</td>
</tr>
<tr>
<td>UCP</td>
<td>8</td>
</tr>
<tr>
<td>The Warden</td>
<td>5</td>
</tr>
</tbody>
</table>

2.1.2. Related Work

In this section, we summarize the related previous work in detecting cheating behavior in MMORPGs using artificial intelligence. Matt Pritchard is the first to propose a taxonomy, the different ways by which a cheater can exploit the game, with regards to cheating, and later there are several taxonomies proposed by researchers (GautheierDickey et. al., 2004; Kuecklich 2004; Consalvo 2005) regarding cheating. The taxonomy proposed by Yan & Randell (2005) is the most used one and is described in the following table.

Table 6. Systemic classification of cheating in video games by Yan and Randell (2005)

<table>
<thead>
<tr>
<th>Type of Cheat</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheating by social engineering</td>
<td>Tricking honest players to reenter user id and password.</td>
</tr>
<tr>
<td>Cheating by collusion</td>
<td>In MMO games it is not allowed to know about certain details of a player, but by conspiring, cheaters get the information from other players.</td>
</tr>
<tr>
<td>Cheating by exploiting misplaced trust</td>
<td>The game developer places too much trust in the client side, but the cheater modifies the game behavior by changing game data.</td>
</tr>
<tr>
<td>Cheating by compromising game servers</td>
<td>If the game server or host system is not secured, then the cheater can modify the game programs on the server.</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cheating by compromising passwords</td>
<td>The password of a player is stolen by a cheater who then gains access to the player’s virtual assets and information.</td>
</tr>
<tr>
<td>Cheating by abusing game procedures or policies</td>
<td>The Cheater achieves an advantage by abusing the operating procedure of a game, for example, by turning off or disconnecting the game when he is about to lose.</td>
</tr>
<tr>
<td>Cheating related to Internal misuse</td>
<td>A Cheater with system administrator privileges can abuse these privileges.</td>
</tr>
<tr>
<td>Cheating by exploiting lack of secrecy</td>
<td>If the communication messages are not encrypted then the cheater can modify the information in packets by simple insert, delete and modify commands.</td>
</tr>
<tr>
<td>Cheating by denying service to peer players</td>
<td>To delay the actions of opponent players, a common technique used by hackers is to overflow their network connection.</td>
</tr>
<tr>
<td>Cheating by modifying client infrastructure</td>
<td>The cheater changes the client infrastructure instead of changing the game files. For example, Wallhack is achieved by modifying drivers.</td>
</tr>
<tr>
<td>Cheating by exploiting a bug</td>
<td>Developers must be careful otherwise cheaters exploit the defects in their code for their benefit.</td>
</tr>
<tr>
<td>Timing cheating</td>
<td>The cheater delays his actions until he knows the actions of everyone else.</td>
</tr>
<tr>
<td>Cheating related to virtual assets</td>
<td>The cheater takes real money from players by promising to give them virtual game assets in return and failing to keep that promise.</td>
</tr>
<tr>
<td>Cheating by lack of authentication</td>
<td>In certain countries like Korea, there is no proper mechanism for authentication, so cheaters can exploit and access the opponent players' computers.</td>
</tr>
<tr>
<td>Cheating by exploiting machine intelligence</td>
<td>The cheater utilizes Artificial Intelligence (AI) to complete the tasks in the game.</td>
</tr>
</tbody>
</table>
Cheating by exploiting misplaced trust, cheating by denying service to peer players, cheating by modifying client infrastructure, cheating by exploiting a bug, timing cheating, cheating by exploiting machine intelligence are relevant to this thesis.

Kim et. al. (2005), proposed a technique to detect cheaters that are using bots/auto programs by analyzing the sequence of actions performed by a player. A bot is a program that can produce automatic mouse and keyboard events. They have converted the sequence of actions carried out by a player into attributes to train a decision tree and achieved higher accuracy.

Laurens et. al. (2007), proposed a cheating detection technique that uses unsupervised machine learning to detect an anomaly in the behavior of the player. The proposed technique can detect wallhack in an FPS game using the concept of the trace. A trace is a mechanism which provides the information of what the player is looking at? The abnormal behavior of a player can be measured by monitoring and by calculating a final score from the four parameters: the frequency of illegal traces, the sequence of illegal traces, the distance to world traces and the distance to entity traces.

Chapel et. al. (2010), proposed a cheating detection method that is based on the behavior of the player. They have developed a probabilistic model that assigns ranks for every player based on their game results and can detect potential cheaters based on statistical tests. The rank of cheaters is assumed to be inflated.

Pao et. al. (2010), proposed a technique to detect cheaters in FPS games, that are using bots, by measuring the dissimilarities between the trajectories of an honest player
and a cheater using a bot. The movements of an honest player are tracked. The movement trajectory that diverges from that track is labeled as a cheater and is used to train a supervised classifier.

Galli et. al. (2011), proposed a real-time cheating detection technique that is based on supervised machine learning. They have used naïve Bayes, random forest, decision trees, neural networks and support vector machine classifiers for automatic detection of cheaters in the FPS game Unreal Tournament III. The training data has a sequence of actions performed by a player and is labeled as a cheater if they find any suspicious behavior in the player's actions. They have achieved a classification accuracy of 90%.

This thesis differs from previous work in such a way that, our work is the first to detect cheaters and victims using textual data (social media, multiplayer chat, and reviews) about online multiplayer games.

### 2.2. Machine Learning

#### 2.2.1. Overview

Machine learning, a type of artificial intelligence, gives the “ability to learn” to computers without being explicitly programmed. The goal of machine learning is to look for patterns in the data and to develop computer programs that can change program actions accordingly when exposed to new data. Tom M. Mitchell (1997) defines machine learning as
“A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$."

In simple terms, building a machine learning classifier is an inductive learning process in which, at the time of classification of a new document, relevant features of the new document are recognized and compared with a set of training documents. Machine learning applications include medical diagnoses, text classifications, and computer visions.

The statistical representation of a machine learning classifier can be summarized as follows: for a given data set $\{(x_1, y), (x_2, y), \ldots, (x_n, y)\}$ find a training classifier $h: X \to Y$ which can assign instances to their object class more accurately. For example, in spam filtering, the input $x_i$ is a message, and the output $y_i$ takes the value “spam” or “not spam.”

### 2.2.2. Types of Machine Learning

Supervised learning is a machine learning task that is used fairly commonly in classification systems and often the goal of supervised learning is to make the computer learn a classification system with the training instances that are labeled with the correct result. Digit recognition is an example of supervised learning.

Unsupervised learning is a machine learning task in which the goal is to make the computer learn to group instances without pre-determined categorizations but by identifying similarities between the inputs. Anomaly detection is an example of unsupervised learning.
Semi-supervised learning is a machine learning task in which the goal is to train a classifier using a small portion of the labeled data and a large portion of the unlabeled documents.

Reinforcement learning is a machine learning task that allows the classifier to gain domain knowledge based on the feedback from the environment and by reinforcement learning the software agents can determine the ideal behavior in a particular context automatically.

Transfer learning is a machine learning task, where the goal is to store the knowledge gained during training in one type of problem and transfer that knowledge to a related task in another similar type of problems.

Learning to learn is a machine learning task, where the classifier trains by itself using its previous experience.

2.3. Natural Language Processing

2.3.1. Overview

Natural Language Processing (NLP) is a field of computer science and linguistics with a focus on interactions between computers and human languages. NLP applications include machine translation, information retrieval, and conversation agents. As NLP is a vast field, in this section we focus on the concepts needed for later chapters in this thesis.
2.3.2. N-grams

N-grams of text are extensively used in text mining applications. Given a textual input, an N-gram is a contiguous sequence of n items in the text. The items can either be characters or words. The number of N-grams in a sentence $K$ with $X$ number of words is given as follows

$$Ngrams_k = X - (N - 1)$$  \hspace{1cm} (1)

N-grams can be used for various tasks including developing features for a supervised machine learning model. When developing a language model, n-grams can be used to develop unigram (N=1), bigram (N=2) and trigram (N=3) models.

2.3.3. Tokenization and Sentence Segmentation

In lexical analysis, the process of splitting a text into symbols, words or phrases by locating word boundaries, ending and starting points of a word, is known as tokenization, and these words are called tokens. Splitting words by spaces and punctuation marks is the simplest form of tokenization. It is often called word segmentation. The process of dividing a text into its component sentences by identifying sentence boundaries between words in different sentences is known as sentence segmentation. Splitting sentences can typically be done by looking for punctuation marks.
2.3.4. Term Frequency – Inverse Document Frequency

Term frequency inverse document frequency (tf-idf) is a numerical statistic that reflects the importance of a term \( t \) in document \( d \) in a corpus. Tf-idf is the product of two statistics. Term frequency \( tf(t, d) \) is the raw frequency \( f_{t,d} \) of a term \( t \) in document \( d \).

\[
tf(t, d) = f_{t,d}
\]  
(2)

Inverse document frequency \( idf(t, D) \) gives information about whether the term \( t \) is common or rare across all documents \( D \). Inverse document frequency is often scaled logarithmically as follows

\[
idf(t, D) = \log \frac{N}{1 + n_t}
\]  
(3)

Where \( N \) is the total number of documents and \( n_t = |\{d \in D: t \in d\}| \) is the number of documents \( d \in D \) with term \( t \). The \( tfidf(t, d, D) \) is multiplication of term frequency \( tf(t, d) \) and \( idf(t, D) \)

\[
tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) = f_{t,d} \cdot \log \frac{N}{1 + n_t}
\]  
(4)

2.3.5. Stemming

Stemming is a process of term normalization and is used to reduce derivationally related words into their lowest forms. Stemming removes the differences between inflected forms of a word, by chopping the morphological and inflectional ends of derived words. Stemming suffers from two issues: under-stemming and over-stemming. The inability of a stemmer to reduce words with the same meaning to the same root is known as under-
stemming. For example, “jumped” reduces to “jump” but “jumping” reduces to “jumpi”. The tendency of a stemmer to reduce words with distinct meaning to the same root is known as over-stemming. For example, “universe” and “university” reduces to “univers.”

2.4. Automatic Text Categorization

2.4.1. Overview

Automatic text categorization is a supervised learning task where a training set of labeled documents is provided and, based on the likelihood suggested by the training set, a pre-defined category label is assigned to new documents. Figure 11 illustrates the steps that are to be carried out for a supervised classification. In the first phase, the labeled input data is used to train the classifier. In the next phase, a new document is presented to the trained classifier, and it must assign a category to the new document. In this section, we discuss various text classification techniques, and at the end of this section, we discuss the related work.

2.4.2. Logistic Regression

Logistic Regression (LR) is a simple classification algorithm to predict a discrete variable. For example, consider the case of predicting binary outcomes such as this patient will get heart disease in the next two years, as a classification problem with discrete values ‘0’ or ‘1’ as output. In LR, for each class in y, we try to predict the probability that a given
\(i^{th}\) input \(x^{(i)}\) belongs to the class \(y\) and assign the class with the maximum probability to the input \(x^{(i)}\).

Figure 11: Different Phases in Automatic Text Categorization

For multi-class problems (where the dependent variable is nominal and has more than two values), multinomial logistic regression (MLR) or softmax regression is a classification model that generalizes LR for multiclass problems. MLR, given a set of independent variables, computes the probabilities of different possible classes of a dependent variable. Decision rules are then made to select the class with the highest probability when a new document appears for classification. LR can handle nonlinear effects and is more robust as the normal distribution of independent variables is not needed, but it requires more data to achieve stable results.
2.4.3. Naïve Bayes

A probabilistic classifier based on Bayes' theorem with a naïve assumption that classes are independent of each other is a Naïve Bayes (NB) classifier. The algorithm classifies texts by analyzing the presence of each word of a test document with training documents, i.e., by calculating the probability of that test document belonging to different classes. NB classifier works by using a MAP (maximum a posteriori) decision rule, which constructs a decision rule \( d \) such that a document will be labeled with the class that yields the highest posterior probability.

The posterior probability can be calculated by Bayes theorem by assuming all of the features \( X_1, X_2, \ldots, X_n \) are conditionally independent.

\[
Posterior \ Probability = \frac{Prior \times Likelihood}{Evidence} \tag{5}
\]

There are several variants of the NB classifier which differ by the assumptions they make regarding the distribution of posterior probabilities. Multinomial Naïve Bayes (MNB), a classic approach for text classification, compares each word in the document that must be characterized with the words of training data of each class. In an MNB classifier, the distribution of features in a document are modeled as multinomial, i.e., the probability of a document given its class is multinomial distribution.

Even though it is easy to implement an NB classifier, the performance of an NB classifier varies in the literature. In some cases, an NB classifier performs better than any other classification methods (Chai et. al., 2002) since only a small training set is enough.
for each class, but in some cases, it performs poorly (Yang et. al., 1999; Joachims 1996; Joachims 1998).

2.4.4. Random Forest

Random Forest (RF) is an ensemble based learning model. RF can be used for classification, regression, and other such tasks. The main idea behind ensemble methods is that a group of "weak learners" can form a "strong learner".

The RF algorithm: Let ‘M’ denotes the number of features and ‘N’ denotes the number of training samples. Split the training set for each decision tree such that all ‘N’ training samples are considered by that tree in ‘n’ times (Generally, n=100 to 200). While choosing training set for the current tree, one-third of the cases are left out of samples, and as the trees are added to the forest this left out data can be used to get the estimate of classification error. While making a decision, classifier considers ‘m’ (m<M) features out of ‘M’ features at each node of the tree (Generally, $m = \sqrt{M}$). The data is run down on each tree, and for each case pairs, proximities are calculated. Proximities, normalized at the end of the run can be used to fill missing data.

The forest error rate of RF algorithm can be reduced by increasing the strength of each tree by choosing Strong classifiers, classifiers with a low error rate, and decreasing the correlation between any two trees in the forest. RF model can be tuned using the parameters n, the number of trees considered for growing RF classifier, and m, the number of features selected at every node.
RF is one of the most accurate learning models, and for many data sets, it achieved the highest accuracy. RF can run efficiently on large datasets. But RF tends to overfit the data for some classification tasks (Liaw and Wiener, 2002; Ham et. al., 2005).

### 2.4.5. Support Vector Machine

Literature Review suggests that, for automatic text categorization, SVM is one of the best techniques (Furey et. al., 2000; Tong and Koller, 2001; Tong and Chang, 2001). SVM works by constructing hyperplanes in the search space that can best separate objects. Determination of optimal boundaries separating different objects that is to find the maximum-margin hyperplane is the key for SVM classifiers. Select the two hyperplanes that are parallel that can separate two classes, and have distance between them as large as possible and the region between these hyperplanes is known as margin. The training samples which are present on these hyperplanes are known as support vectors and hence the name of the model. Maximum-margin hyperplane lies exactly in between these hyperplanes. A classic approach to a multi-class classification problem is to combine several binary SVM classifiers. In SVM model the over-fitting of data can be avoided by regularization, kernel parameters and choice of kernel. But the determination of these parameters itself is a difficult task (Cawley and Talbot, 2010).

### 2.4.6. Related Work

In this section, we summarize the literature in the area of automatic text categorization. Most work in text classification has used a Bag-of-Words (BOW) model.
Cavnar and Trenkle (1994) proposed an N-gram-based text classification model to classify news articles. The N-gram based model is based on the fundamental idea that some words in human language occur more often. According to Zipf’s law and as restated by Cavnar and Trenkle (1994):

“\textit{The nth most common word in human language text occurs with a frequency inversely proportional to }n\text{.”}

The above statement implies that for a particular domain there always exists a set of words that are used more often. The N-gram model for text classification means that a set of most frequent words used in some articles of a particular domain will remain the same for other articles of the same domain and from experiments on N-gram language model we can conclude that this model can be reliable for text categorization. Tan et. al., (2002) showed improvement in F-measure and break even points by adding bigrams to the standard unigram / BOW model.

To enhance the feature representation Cai and Hofman (2003) have used context models: concept-based document representation. To extract semantics to achieve robustness and reliability towards linguistic variations (vocabulary and word choice), they have used probabilistic latent semantic analysis.

Ho (1995) has proposed a method to construct tree based classifiers and established that forests of trees splitting with hyperplanes can gain accuracy when randomly restricted to select subspaces (the subset of feature dimensions) of the feature space. Leo Breiman (2001), influenced by Ho’s work, has proposed random forests. The key to random forests
is to build a classifier with a set of decision trees (each decision tree is grown randomly in selected subspaces of data). In his following series of papers, Brieman (2004) established that a substantial gain in classification and regression accuracy could be achieved, by using these ensemble trees.

For solving the two-class pattern recognition problem, support vector machine (SVM) is a learning approach introduced by Vapnik (1995). However, for practical applications with multiple objects, there are several studied methods. Duan & Keerthi (2005) has carried out an empirical study on multi-class SVM models: one-versus-all winner-takes-all, one-versus-one max-wins and pairwise coupling. Consider a multiclass classification problem with M number of classes and $w_i, i = 1, 2, ..., M$ are the classes. In the one-versus-all winner-takes-all model, M binary classifiers are built. For a test document $\vec{t}$, the class with the highest value of $p_i$ is assigned where $p_i$ is ith classifier output function trained on examples $w_i$ as one class and all others as another class. In the one-versus-one max-wins voting method, a total of $M(M - 1)/2$ classifiers are formed by constructing a binary classifier for each pair of distinct classes. For a new test document $\tilde{t}$, votes are taken from each classifier $C_{ij}$ (trained on examples from $w_i$ as positive and $w_j$ as negative), and the class with most votes is assigned. In pairwise coupling method, the key idea is to combine outputs of all one-versus-one binary classifiers (under the assumption that the output of each binary classifier is the posterior probability of the positive class) to obtain the estimated priori probability. For a new test document $\tilde{t}$, the class with the highest value of $p_i = P(w_i | \tilde{t})$ is assigned.
Pang et.al., (2000) have performed document classification by considering the overall sentiment of the document that is whether a document expresses a positive or negative opinion. They have examined the performance of sentiment classification using three different machine learning calculations: support vector machine, maximum entropy, and naïve Bayes. Turney (2002) has used unsupervised learning model for the sentiment analysis task, which utilizes a Parts of Speech (POS) tagger to identify modifiers and intensifiers in a document. Classification is done based on the calculated Semantic Orientation (SO) score of phrases. Dave et.al., (2003) have developed a web based opinion mining tool that crawls data from the web, creates attributes, and aggregates opinions for a given product. Features are extracted by using Information Retrieval (IR) techniques, and results of various metrics are tested. Pang and Lee (2004) have developed a text categorization method that connects a classifier to the subjective gathering to avoid misleading information for polarity classification. The subjective portion of a text is obtained using a minimum cut in graph technique. Whitelaw et.al., (2005) have introduced a sentiment classification method based on appraisal groups. The appraisal adjectives list is used for classification, and the list is obtained from semi-automated methods. Li et.al., (2011) have introduced several semi-supervised learning models with dynamic subspace generation for imbalanced sentiment classification and to solve the problem of manually labeled data by using an under-sampling technique.

Research on understanding short texts language has gained more attention in recent times. Twitter, a social media platform is a central point for short text data for many researchers. Jiang et al. (2011), Speriosu et al. (2011), Brody et al. (2011) have used Twitter
feeds for sentiment analysis. Tumasjan et al. (2010), has used Twitter feeds for opinion mining related to political issues. Existing NLP approaches that have achieved high accuracy on normal data sets fail miserably on sentence level data sets (Guo et. al., 2013). Researchers have achieved high accuracies on short text classification problems by developing context-based models that can analyze the syntactic structure and extract the semantic meaning of sentences (Cai and Hofman, 2003).
3. Building Classifier

3.1. Introduction

We have observed that a player describes his gaming experience or opinions in the following places: multiplayer chat, game reviews, and social media. In this thesis, we have created five data sets from these areas, SM-GEN, SM-CSGO, MC-TF2C, MC-TF2S, and RV-CSGO for training and testing of the machine learning classifiers. As shown in figure 12, implementation has two phases: training phase and testing phase. The decision boundaries 1, 2 shown in the figure are measures of satisfactory f-scores of various classifiers. If not satisfied, we proceed back to the data preprocessing steps, make some changes, extract new features and compute f-scores once again. If satisfied, in the training phase, the trained models are saved, and in the testing phase, we will proceed to evaluation measures.

In this chapter, we explain data collections methods used in collecting data from sources (Twitter, logs.tf and Steam), rules laid to label the data, different preprocessing steps or cleaning steps applied on data and extraction of numerical features from textual data.
Figure 12. Proposed algorithm workflow
3.2. Identification of Keywords

As the application of NLP in the field of gaming is relatively new, there is no proper research in keywords that can distinguish data with cheating information from that of normal gaming data and hence identification of those keywords is a paramount task. Figure 13 illustrates the methodology used to identify the keywords. We have studied different means by which players cheat in FPS games and came up with a final list of keywords which are shown in the word cloud, Figure 14. In this thesis, we are confined to FPS games, but the same can be applied to any genre game as well, by identifying the different cheating methods employed by players to cheat in those games.

![Figure 13. Identification of keywords](image)

Several of the keywords we identified bought in irrelevant data from Twitter, but these keywords can be used to separate data with cheating information in the multiplayer...
chat and game review datasets. These keywords include: bots, God mode, Fast reload, No smoke, No flash, and ESP.

Figure 14. Word cloud of keywords

3.3. Datasets

3.3.1. Data Collection SM-GEN, SM-CSGO

Twitter is one of the most popular social media platforms where users post their opinions. Recently, Twitter is the favorite dataset for many of the NLP researchers as it possesses unique qualities like 140-character uniform length, widespread diversity, real-time data stream and real life conversations. Many product based companies rely on Twitter to get the opinions of users on their product.
Twitter offers two APIs for researchers to obtain the data from their corpus: Search API and Streaming API. Twitter provides various Representational State Transfer (REST) APIs to provide programmatic access to read and write Twitter data. The Twitter Search API is one of the RESTful APIs provided by Twitter. It makes keyword searches that were written in the preceding seven days. Twitter has established some rate limits which are segmented based on the type of authentication, user or application. The rate limit for applications is 450 requests per 15-minute window, and for users, it is 180 requests per 15-minute window. An application has been created with the name "Detection and Mitigation of Cheating in MMORPGs" in the Twitter's Developer page, and all the necessary access keys and tokens of the application were obtained. Using these application access keys and tokens a Twitter client was developed which can search Twitter by keywords using Twitter search API and return the tweets in JSON format. The Twitter client keeps an eye on the rate limits and goes into an idle state when the rate limit is reached. The big drawback of the Search API is that it can only give the tweets that are written in the previous week. Initial tweets are gathered using the Twitter search API.

Unlike the Search API, the Streaming API returns real time tweets for the input keywords. Several streaming endpoints are provided by Twitter for developers: public streams, user streams, site streams each have a particular use case. Public streams, most often used by researchers, are suitable for data mining the streams through the entire public data of Twitter. All of the tweets obtained from all of the APIs are filtered to pull tweets that are in the English language only.
Together with Search API and Streaming API, we have created a dataset ALL_TWEETS that has the statistics shown in Table 7. SM-GEN and SM-CSGO are two data sets from Twitter. Both datasets consist of tweets that are related to cheating in online FPS games. SM-GEN does not focus on a particular game; it has tweets related to cheating from different FPS games like Counter Strike, Modern Combat, Battle Field, Team Fortress, etc. SM-CSGO only has tweets related to the online multiplayer game Counter-Strike Global Offensive. SM-GEN and SM-CSGO are the datasets derived from ALL_TWEETS with some filters.

![Data collection from Twitter](image)

**Figure 15. Data collection from Twitter**

<table>
<thead>
<tr>
<th></th>
<th>Tweets</th>
<th>Duplicate Tweets</th>
<th>Retweets</th>
<th>Original Tweets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALL_TWEETS</strong></td>
<td>13560</td>
<td>310</td>
<td>3587</td>
<td>9973</td>
</tr>
<tr>
<td><strong>SM-GEN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SM-CSGO</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 7. Statistics of ALL_TWEETS dataset*
SM-GEN is the data set obtained from ALL_TWEETS, with the following filters.

- Removal of irrelevant tweets: Because of the spaces in the keywords some tweets with only one-half of the keyword are returned by Search and Streaming APIs. For example, consider the keyword “wall hack,” the APIs have returned tweets that have only “wall” in their post. All of these irrelevant tweets, that are not related to cheating in online gaming, were programmatically removed by examining whether each tweet has at least one keyword identified in the word cloud, Figure 14.

- Removal of Retweets: Retweets are not a point of interest in this thesis.

- Removal of tweets with length less than three words. Tweets with words less than three words, have no information hence, these tweets are filtered out.

- Removal of tweets from Phantom Forces: Phantom forces is an FPS game which has legalized the usage of aim bots in its game by selling aim bots for a monthly subscription charge.

The SM-CSGO dataset is used for classifier testing purposes and is also obtained by applying all of the filters applied to SM-GEN, and also another filter is applied. SM-CSGO has 500 tweets.

- Removal of tweets with no mention of counter strike/global offensive/cs.

Both SM-GEN and SM-CSGO have the following data extracted from JSON data.
3.3.2. Data Collection MC-TF2C, MC-TF2S

MC-TF2C and MC-TF2S are two datasets which have multiplayer chat logs and stats of the Team Fortress 2 game respectively. The chat logs and stats are obtained from logs.tf which is an automatic stats generator and log parser for the game Team Fortress 2 with over 175,000 players and 1,580,000 matches being logged. Logs.tf provide an Application Programming Interface (API) to upload and search through entire log files of Team Fortress 2. The stats can be obtained in raw JSON format using http://logs.tf/json/<log_id> and stored as MC-TF2S. The logs can be obtained using http://logs.tf/logs/<log_id>.log.zip and stored as MC-TF2C. At the time of this thesis, there are a total of 1,573,129 chat logs and stats. Log ids ranging from 1,570,000 to 1,573,000 are obtained for this thesis. The raw JSON file of each stat file consists of several statistics of the game. Some of the useful stats of users are given in Table 9. All of these stats are automatically generated from corresponding log files by logs.tf.

<table>
<thead>
<tr>
<th>id</th>
<th>The id of the tweet</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_at</td>
<td>The time at which the tweet is published</td>
</tr>
<tr>
<td>text</td>
<td>The text of tweet</td>
</tr>
<tr>
<td>user_id</td>
<td>The user id of the user</td>
</tr>
<tr>
<td>description</td>
<td>The description of the user</td>
</tr>
<tr>
<td>time_zone</td>
<td>The time zone of the user</td>
</tr>
<tr>
<td>geo_location</td>
<td>The location of the user</td>
</tr>
</tbody>
</table>

Table 8. Data extracted from JSON
Table 9. Relevant information available in MC-TF2S

<table>
<thead>
<tr>
<th>Team</th>
<th>The team of the player.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The role of the player.</td>
</tr>
<tr>
<td>User-id</td>
<td>The id of the player.</td>
</tr>
<tr>
<td>Kills</td>
<td>A total number of kills in a game by the player.</td>
</tr>
<tr>
<td>dmg</td>
<td>Total damage done to opposite team players using different weapons by the player.</td>
</tr>
<tr>
<td>avg_dmg</td>
<td>Average damage done to opposite team players using different weapons by the player.</td>
</tr>
<tr>
<td>Headshots</td>
<td>A total number of headshots executed by the player.</td>
</tr>
<tr>
<td>deaths</td>
<td>A total number of times the player died in the game.</td>
</tr>
</tbody>
</table>

Along with the stats data, the chat data from each log file is also obtained. Each line in a log file from MC-TF2C has a date, player name, player age, player id, an annotation and a message. The annotations and the message followed by it are described in Table 10.

Table 10. Relevant information available in MC-TF2C

<table>
<thead>
<tr>
<th>entered the game</th>
<th>When a player enters the game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changed role to</td>
<td>When a player changes his role</td>
</tr>
<tr>
<td>triggered</td>
<td>actions of that player</td>
</tr>
<tr>
<td>Spawned as</td>
<td>The role of the player after getting killed</td>
</tr>
<tr>
<td>killed</td>
<td>If a player kills another player</td>
</tr>
<tr>
<td>say</td>
<td>Message of a player</td>
</tr>
</tbody>
</table>

3.3.3. Data Collection RV-CSGO

RV-CSGO is a reviews dataset obtained from the game counter strike global offensive which is scraped from the Steam website. Steam is a video game distribution
platform developed by the Valve Corporation that provides social networking services, digital rights management, and multiplayer gaming services. The steam platform is the largest PC gaming distribution platform. Steam has over 125 million registered accounts. Steam provides stronger anti-cheat measures under the name Valve Anti-Cheat (VAC) which was introduced in 2002, and it is estimated that over 5.2 million Steam accounts are banned by VAC as of March 2017 (GameMe 2017) because of cheating. We have collected 7000 reviews of Counter Strike Global Offensive, and each review has some additional information including, but not limited to what is listed in Table 11.

Table 11. Information available in RV-CSGO

<table>
<thead>
<tr>
<th>Num_found_helpful</th>
<th>Number of people marked the review as helpful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Username of the reviewer</td>
</tr>
<tr>
<td>Steam_id_number</td>
<td>Unique steam id of the reviewer</td>
</tr>
<tr>
<td>Total_game_hours_last_two_weeks</td>
<td>Total time spent by the reviewer in the past two weeks.</td>
</tr>
<tr>
<td>Total_game_hours</td>
<td>Total time spent by the reviewer on the game</td>
</tr>
<tr>
<td>Num_achievement_percentage</td>
<td>Percentage of a number of achievements by the reviewer during the total game hours.</td>
</tr>
<tr>
<td>Review</td>
<td>The review on game by the player</td>
</tr>
</tbody>
</table>

3.4. Labeling the data

For a machine to classify with confidence similar to humans, first we must provide a variety of examples and their labels, later we can obtain significant syntactic features like
single words, POS tags, n-grams, etc. To achieve that, algorithms need labeled data to make an educated guess on unseen instances. As humans are a reliable source of determining opinion, we have manually labeled the dataset SM-GEN using the following assumptions.

A given tweet is labeled as a cheater if the tweet implies that

- The user himself has gained an unfair advantage over other players by using hacks.
- The user is a vendor that advertises various hacks for other players to buy.

Example tweets of cheaters:

- “@PzElyte i was that kid that would run around with aimbot/uav on mw2 lmao”
- “Selling The following Hacks for CS:GO -Aimbot - Walls -Trigger
  *Undetectable and working with proof! DM for prices! @ShoutGamers @ShoutRTs”
- “@ScufGaming @DavidVonderhaar I killed 24 people and I have aimbot”

A given tweet is labeled as a victim if the tweet implies that

- User has seen players using hacks
- User complains about hackers and exhibits his temperament using curse words.

Example tweets of victims:

- “@Covton he's using Aimbot and UAV lol”
- “@ATVIAssist i found one his name is Ludacris, please stop him his k/d is over 35 and his got aimbot”
- “@L7Panthers: every MW2 lobby i get is people using fucking aimbot”

A given tweet is labeled as neutral if the tweet is merely a statement about cheating where the user has no intention to achieve an unfair advantage by using hacks or does not accuse someone of cheating.

Example tweets for neutral

- “@StyLisStudios what is aimbot for?”
- “@KickFlipPenguin @Eighty7n @Soaz01 @ZeroPorridge Aimbot, gives very good accuracy”
- “@SirScoots @RustHacks hacking must be stopped.”

The statistics of datasets SM-GEN and SM-CSGO after annotation are shown in the following tables.

<table>
<thead>
<tr>
<th>Class</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheater</td>
<td>200</td>
<td>40</td>
</tr>
<tr>
<td>Victim</td>
<td>200</td>
<td>40</td>
</tr>
<tr>
<td>Neutral</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>500</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
### 3.5. Preprocessing the data

#### 3.5.1. SM-GEN, SM-CSGO

- Replace HTML character codes: If symbols like (<) for less than and (>) for greater than are used in the text, the browser cannot differentiate these with HTML tags. So, reserved characters in HTML are replaced by character entities. Hence the data from the web usually consists of HTML character entities which are to be replaced with their ASCII equivalents. A character entity may look like this: &entity_name or &#entity_number.

- CamelCase: The practice of writing two or more words with no intermediate spacing or punctuation but the starting of each word is capitalized is known as Camel case. The following regular expression pattern can help to identify these compound words. Example: CamelCase (Upper case), camelCase (Lower case).

\[
String\ camelCase = "[a - zA - Z][a - z] * ([A - Z0 - 9] + [a - z] *) + ";
\]
• Stop-words Removal: Stop words are the extremely common words that have nothing to do with the content and information retrieval results. By filtering out these stop words, the ambiguity in the data can be reduced and we can focus on relevant information. Even though stop words are most frequent words, there is no single universal stop word list used by all NLP tools. Stop words can mean different things for different applications. For example, in some applications where sentiment analysis has to be carried out adjective terms like 'good' are important. Hence, for different applications, the list will be different. In this thesis, we have considered determiners (a, an, the, etc.), conjunctions (for, and, so, etc.) and prepositions (in, under, before, etc.) as stop words.

• Tokenization: Text is a sequence of characters, words or phrases. Before application of any text processing methods, text must be tokenized, that is to segment the text into linguistic units such as punctuation, numbers, and words. These linguistic units after tokenization are the smallest units (also called tokens) which do not require any further decomposition. Even though the task of tokenization may seem simple if we split a sentence using the ‘space’ character, there are significant challenges like handling abbreviations, hyphenated words, mathematical and special expressions which are not taken care of. We have used the Stanford sentence tokenizer for this thesis.

• Spell Correction and Slang Conversion: As Twitter is limited to 140 characters, users often use shortened lingo to convey their thoughts. Also, users tend to misspell words often. Spell correction and slang conversion (for example ‘luv’
is converted into ‘love’) are critical as these words if not corrected or converted are counterproductive but by converting they may contribute to the feature vector. Spell correction and slang conversion can be done in many ways including but not limited to dictionary based, similarity/edit distance, hidden Markov model, and weighted edit distance. We have scraped an online tool http://www.lingo2word.com/ by Hazelwood (2001) which implements static dictionary based approach proposed by Raghunathan and Krawezyk (2009) for spell correction and slang conversion. On top of spell correction and slang conversion lingo2word also does word standardization, for example, ‘loooove’ is converted to ‘love,’ and Emoticons conversion, for instance ‘:)’ is converted into ‘Happy.’

- Removal of URLs and usernames: URLs, hyperlinks, and usernames add redundancy to the data. The following regular expression patterns can help to identify URLs, hyperlinks, and usernames.

  
  String urlPattern = (^www\/[\s\S]+) | (https?:\/[\s\S]+);  
  String userPattern = (RT)?@[\s\S]+;  

- Stemming: The idea behind stemming is to convert words into their base forms when grammatical placement of words is insignificant to your classifier. For this purpose, we have used the Porter Stemming algorithm by Martin Porter (1980).
3.5.2. MC-TF2C, MC-TF2S

- Filtering logs that are not in English: The data from logs.tf contain logs of different languages. We have used the langdetect library from Cybozu Labs (Shuyo, 2010) to take out English language logs only.

- Filtering chat logs and stats that have no information related to cheating: The 3000 logs and stats of the game Team Fortress 2 that we pulled from logs.tf contain both regular logs and cheat logs (logs in which there is information related to cheating). The logs that do not contain at least one of the keywords shown in Figure 14 are filtered out. We are left with 175 chat logs in which someone might have used hacks to attain an unfair advantage over other players.

- Getting chat from logs: The log files consist of all the information related to the game including but not limited to player’s roles, spawned time, weapon changes, kills, deaths, console messages and player’s chat. We have used a series of regular expression patterns to get players’ chat and used keywords to filter out the messages that have no information related to cheating. Now each log contains the player’s id, player’s team, player’s name and their message.

- All of the data preprocessing steps employed on tweets are now applied to the player’s messages: replace HTML character codes, camel case separation, removal of stop words, tokenization, spell correction, slang conversion and stemming.

The MC-TF2C now contains 175 chat logs of the game Team Fortress 2 which are suspected to involve some form of cheating. Each log file is labeled as YES if it contains
messages implying someone has cheated in the game else labeled as NO. MC-TF2S includes statistics of 3000 games including these 175 games.

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>96</td>
</tr>
<tr>
<td>NO</td>
<td>79</td>
</tr>
<tr>
<td>Total</td>
<td>175</td>
</tr>
</tbody>
</table>

3.5.3. RV-CSGO

The preprocessing steps applied on RV-CSGO are the same steps that are applied on MC-TF2C. Finally, we are left with 685 reviews in RV-CSGO. All these 685 reviews are labeled according to the rules provided in section 3.3.

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheater</td>
<td>64</td>
</tr>
<tr>
<td>Victim</td>
<td>401</td>
</tr>
<tr>
<td>Neutral</td>
<td>220</td>
</tr>
<tr>
<td>Total</td>
<td>685</td>
</tr>
</tbody>
</table>

3.6. Feature Extraction

A feature or attribute is a variable in which an observable phenomenon can be quantified and recorded. The success of a machine learning classifier depends critically on features being selected. The process of transforming arbitrary textual data into numerical features understandable for machine learning is known as feature extraction. We have
selected text-based features only, as our goal is to perform the classification task using the
textual data.

To extract meaningful information at first, we have extracted lexical features from the
textual content as these features have been successfully used for several classification tasks.
These lexical features involve unigrams or bag-of-words (BoW), bigrams and term
frequency inverse document frequency (tf-idf), which is simply a weighted version of the
former two. The unigrams model is a simplistic and intuitive method that can be combined
with scalable linear models to train classifiers, in which we assign an integer id for each
word occurring in the training data set and for each document, we compute and store the
count of the number of occurrences of each word as the value of the feature. But a collection
of a bag of words cannot secure the meaning of phrases and multi-word expressions. Hence
a collection of bigrams, consecutive word pairs is counted for features. If we feed word
counts directly to a classifier, the words with little meaningful information and with high
frequency shadow the interesting and meaningful words with less frequency. Term-
frequency and inverse-document-frequency re-weights these count features into floating
point values to reduce this shadowing of high-frequency words by multiplying the term-
frequency with its inverse-document-frequency. We have created a list containing the 200
most frequent unigrams and 125 most frequent bigrams.

The word clouds of cheaters and victims are shown in the figures 16 and 17
respectively. On closer scrutiny, we discovered that the cheater is joyous and happy in
cheating, while the victim is sad and unhappy. The cheater uses accolade words more often
to appreciate the hack that he uses, while the victim uses curse words to show his profanity
towards the cheater which is intuitive from the word clouds. Thus, the sentiment analysis of text from a cheater is often positive while that of a victim is negative. The neutral messages are often statements with no sentiment. Some examples are given in Table 17. We have used SentiStrength, an automatic sentiment analysis tool with up to human level accuracy, for this task. The task of SentiStrength is to estimate the positive and negative sentiment strengths, even for short texts with informal language. SentiStrength is a lexical based approach that makes use of sentiment related terms and can deal with standard linguistic methods such as emoticons, punctuation, and misspellings to express the sentiment. SentiStrength reports sentiment strength on a single scale of (-4 to +4). Feature scaling, a method using which the range of independent variables is standardized. Feature scaling is done to improve classifier’s accuracy. If the features are not scaled, then the classifier prioritizes feature with a broad range of values (Aksoy and Haralick, 2001). Hence, we have further reduced the scale to (-2 to +2), to match with the range of other features, with -2 representing extremely negative sentiment and +2 representing extremely positive sentiment while 0 represents a neutral sentiment.

Lexical based features may perform well in most cases, but in some cases, we may come across synonym words that are not present in the training set. So, we came up with a dictionary based feature in which we have created two dictionaries for the cheater and victim. The cheater dictionary (CD) and victim dictionary (VD) consists of the 200 most frequent words of cheaters and victims respectively after removing the outliers. We have used the dictionary.com developer API to get five synonym words for each of these most frequent words and stored them in their respective dictionaries. For each text input, a score
Figure 16. Word cloud of victim after data cleaning

Figure 17. Word cloud of cheater after data cleaning
with scale -2 to +2 is calculated by giving -1 for the words in VD and +1 for the words in CD.

Figures 18, 19 are the word clouds of victims and cheaters, respectively, before removing stop words and stemming. On closer scrutiny, we have discovered that a victim often mentions second or third person pronouns to refer to the person who is cheating, which can be seen in the word cloud of the victim. We have also discovered that the cheater often
mentions first person pronouns to refer to himself as being a user of hacks. Often in neutral cases, we have discovered that, there are no first, second or third person pronouns as the intention of the user is neither accusing someone nor referring to himself as a hacker. We have calculated a pronoun based score by giving -1 for each occurrence of second or third person pronouns and +1 for each occurrence of first person pronouns. The final scale for the pronoun based feature is (-2 to +2) with -2 representing a document referring to a second or third person and +2 representing a document referring to the first person while 0 represents no personal pronouns in the sentence.
Table 16. Personal pronouns

<table>
<thead>
<tr>
<th>First Person</th>
<th>I, me, we, us, myself, ourselves, my, our, mine, ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second Person</td>
<td>You, yourself, your, yours</td>
</tr>
<tr>
<td>Third Person</td>
<td>He, she, it, him, her, they, them, himself, herself, itself, themselves, their, theirs</td>
</tr>
</tbody>
</table>

In some cases, a user might mention both first, second and/or third person pronouns. For those cases, we have considered another feature which focuses on the pronoun that is closer to the cheat/hack word. The reason behind choosing the pronoun closer to cheat/hack word is that the cheat/hack word is most often the Object and pronoun is most often the Subject; in linguistic typology, Subject-Verb-Object (SVO) and Subject-Object-Verb (SOV) represents rigid word orders (Crystal, 1997). In the English language, SVO and SOV are the most frequently used word orders in a sentence structure where the Subject comes first, the Verb second and the Object third in the SVO sentence structure, while the Subject comes first, the Object second and the Verb third in the SOV sentence structure. The scale for the feature, location of the pronoun is (-1 to +1) with -1 representing the presence of a second or third person pronoun closer to the cheat/hack word and +1 representing the presence of a first-person pronoun while 0 represents that there is no pronoun in the given text.
<table>
<thead>
<tr>
<th>Message</th>
<th>Sentiment</th>
<th>Pronouns</th>
<th>Pronoun closer to hack word</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>@AimJunkies thank you so much I like this wallhack in csgo as it gives me so much fun to mess around.</td>
<td>Positive</td>
<td>I, me</td>
<td>I</td>
<td>Cheater</td>
</tr>
<tr>
<td>I spinbot in global offensive and it is the most satisfying thing lol.</td>
<td>Positive</td>
<td>I</td>
<td>I</td>
<td>Cheater</td>
</tr>
<tr>
<td>@rushhackreport <a href="https://t.co/QHcAs4Xqek">https://t.co/QHcAs4Xqek</a> this fucking hacker in seattle 2 server now. he use jumphack and aimbot plz <a href="https://t.co/er1yWJCBtb">https://t.co/er1yWJCBtb</a></td>
<td>Negative</td>
<td>he</td>
<td>he</td>
<td>Victim</td>
</tr>
<tr>
<td>@creativelesbian: &quot;hipilipity he has a menu with like aimbot triggerbot and stuff&quot; - deadly2016 smh u a bitch ass nigga gomd in skodnas…</td>
<td>Negative</td>
<td>He, you</td>
<td>he</td>
<td>Victim</td>
</tr>
<tr>
<td>@FaZeApex Im Going To Be Every other Nerd Ever And Say I Just Got Into Your Game And You Were Using Aimbot, #Exposed Hope CBass Kicks Your ass. smh</td>
<td>Negative</td>
<td>I, your, you</td>
<td>you</td>
<td>Victim</td>
</tr>
<tr>
<td>Who needs an aimbot when all that matters is #skill #BF4 #Battlefield4 #MLG #Twitch #XboxLive #XboxOne <a href="https://t.co/SPliZDHPR5">https://t.co/SPliZDHPR5</a> that's valve's aimbot detection</td>
<td>Neutral</td>
<td>None</td>
<td>None</td>
<td>Neutral</td>
</tr>
</tbody>
</table>

In another feature, we have considered whether or not the textual input contains a question. This is often indicative of a neutral case because the queries often refer to questions regarding a cheat/hack and do not implicit any information with regards to a cheater or victim. A given textual input is considered as a question if it contains any of the interrogative pronouns indicated in the following table or has a ‘?’ character. The feature will either be a 0 or 1, with 1 indicating a question and 0 indicating not a question.
Table 18. Interrogative pronouns

<table>
<thead>
<tr>
<th>Interrogative Pronouns</th>
</tr>
</thead>
<tbody>
<tr>
<td>What</td>
</tr>
<tr>
<td>Who</td>
</tr>
<tr>
<td>Which</td>
</tr>
<tr>
<td>Whose</td>
</tr>
<tr>
<td>Whom</td>
</tr>
</tbody>
</table>

Extracted features and their labels are shown in Table 19.

Table 19. Extracted Features and their representation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigrams</td>
<td>U</td>
</tr>
<tr>
<td>Bigrams</td>
<td>B</td>
</tr>
<tr>
<td>Term Frequency and Inverse Document Frequency (Unigrams)</td>
<td>TU</td>
</tr>
<tr>
<td>Term Frequency and Inverse Document Frequency (Bigrams)</td>
<td>TB</td>
</tr>
<tr>
<td>Dictionary based</td>
<td>D</td>
</tr>
<tr>
<td>Sentiment Analysis</td>
<td>S</td>
</tr>
<tr>
<td>Pronoun</td>
<td>P</td>
</tr>
<tr>
<td>Location of Pronoun</td>
<td>L</td>
</tr>
<tr>
<td>isQuestion</td>
<td>Q</td>
</tr>
</tbody>
</table>
4. Experiment and Results

4.1. Introduction

This chapter presents the experiments and results produced by accessing various evaluation metrics on the data sets. We have used the Waikato Environment for Knowledge Analysis (WEKA) tool for conducting machine learning experiments. WEKA is a workbench that contains a collection of tools for data pre-processing, regression, classification, clustering and visualization for data mining tasks (Hall et. al., 2009). In this chapter, first, we present the classifiers used. Then we continue to provide evaluation metrics used and the experiments performed to measure the success or failure of the approach.

4.2. Classifiers

For our experiments, we are comparing four different classifiers: Linear Regression (LR), Naïve Bayes (NB), Random Forest (RF) and Support Vector Machine (SVM). We evaluate the performance of individual features for our training dataset and finally select the features that can efficiently perform the classifying task. We test our datasets with these selected features on each trained classifier.
4.3. Evaluation Metric

In evaluating multi-class classification problems, computing the accuracy, that is the percentage of correctly predicted labels over all predictions, is not the best way to assess the performance of the classifier as a high accuracy classifier may classify a particular class accurately while making mistakes on other classes that are critical to the application.

Precision and recall are the two measures which can be computed for each class label, and a weighted average of class labels gives overall precision and recall. For a given class X and all the predicted labels, precision is a measure of “how many instances were correctly predicted?” and given all instances that should have the label X, recall is the measure of “how many of these were correctly captured?”

The $F_1$ score is a measure that combines both precision and recall and is commonly used to judge a classifier’s performance it is calculated by considering the average of all of the document/category pairs by giving equal weight to each document/category pair. For any classifier, the value of the $F_1$ score lies between 0 and 1, where 0 is the worst possible $F_1$ score, and 1 is the best possible $F_1$ score. The $F_1$ score is the harmonic mean of precision and recall measures which are defined as follows:

\[
p_i = \frac{TP_i}{TP_i + FP_i}, \quad r_i = \frac{TP_i}{TP_i + FN_i}
\]

Where $p_i$ and $r_i$ are the precision and recall measures calculated for each class label $i$. $TP_i$ is the number of true positives (the number of documents correctly labeled as belonging to class $i$), $FP_i$ is the number of false positives (the number of documents
incorrectly labeled as belonging to class \( i \), \( FN_i \) is the number of false negatives (the number of documents that should be labelled as belonging to class \( i \) but are not.) The \( F_1 \) measure for class \( i \) is calculated as follows:

\[
F_i = \frac{2pr_i}{p_i + r_i}
\]  

(7)

The global precision and recall values are obtained by calculating a weighted average of precision and recall for all class labels which is expressed as follows:

\[
p = \frac{\sum_{i=1}^{M} N_i p_i}{\sum_{i=1}^{M} N_i}, \quad r = \frac{\sum_{i=1}^{M} N_i r_i}{\sum_{i=1}^{M} N_i}
\]  

(8)

where \( M \) is the number of class labels and \( N_i \) is the number of instances of label \( i \).

The weighted average \( F_1 \) score is defined using global precision and recall values as follows:

\[
F_1(weighted - average) = \frac{2pr}{p + r}
\]  

(6)

### 4.4 Feature Selection

Given a set of features, feature selection or attribute selection, is the process of identifying a subset of features that is most effective for a particular classification task. The following table gives F-scores of the selected features in isolation. The results shown in Table 20, 21 are performed on SM-GEN training dataset with ten cross-validation folds.
Table 20. Single Feature Evaluation on SM-GEN

<table>
<thead>
<tr>
<th>Feature</th>
<th>Number of features</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LR</td>
</tr>
<tr>
<td>U</td>
<td>200</td>
<td>0.56</td>
</tr>
<tr>
<td>B</td>
<td>125</td>
<td>0.56</td>
</tr>
<tr>
<td>U-B</td>
<td>325</td>
<td>0.59</td>
</tr>
<tr>
<td>TU</td>
<td>200</td>
<td>0.58</td>
</tr>
<tr>
<td>TB</td>
<td>125</td>
<td>0.55</td>
</tr>
<tr>
<td>TU-TB</td>
<td>325</td>
<td>0.59</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0.68</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>0.61</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
<td>0.65</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>0.68</td>
</tr>
<tr>
<td>Q</td>
<td>1</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Figure 20. Performance evaluation of single features
Table 21. Combination of features evaluation on SM-GEN

<table>
<thead>
<tr>
<th>Feature Combination</th>
<th>F-measure</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LR</td>
<td>NB</td>
<td>RF</td>
<td>SVM</td>
</tr>
<tr>
<td>D-S</td>
<td>0.72</td>
<td>0.72</td>
<td>0.68</td>
<td>0.67</td>
</tr>
<tr>
<td>D-P</td>
<td>0.74</td>
<td>0.73</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>D-L</td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>S-P</td>
<td>0.64</td>
<td>0.64</td>
<td>0.62</td>
<td>0.60</td>
</tr>
<tr>
<td>S-L</td>
<td>0.69</td>
<td>0.69</td>
<td>0.72</td>
<td>0.71</td>
</tr>
<tr>
<td>P-L</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
</tr>
<tr>
<td>D-S-P</td>
<td>0.74</td>
<td>0.74</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>D-P-L</td>
<td>0.78</td>
<td>0.78</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>S-P-L</td>
<td>0.68</td>
<td>0.69</td>
<td>0.70</td>
<td>0.69</td>
</tr>
<tr>
<td>D-S-P-L</td>
<td>0.81</td>
<td>0.81</td>
<td>0.82</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Figure 21. Performance evaluation of different combination of features on SM-GEN
The experiments reveal that the lexical (U, B, TU, TB) and Q features fail for the classification task while the features D, S, P, and L have a sizable impact on discriminating cheaters, victims, and neutral documents. The lexical features U, B, TU, TB and D all depend on the words present in a given document for the classification task. We choose the feature D as it outperforms all other lexical features. We test the combination of these selected features and finally choose the combination that performs better.

We have tested all of the combinations of the selected features with four classifiers, and the classification f-measures of these classifiers are in Table 21. The highest classification f-measure of 0.816 is achieved with the RF classifier using D-S-P-L as the feature combination.

4.5. Performance Analysis

The performance of a classification model can be assessed using a confusion matrix. We try to explain the background working of the selected features. The results shown in Figures 22, 23, 24, 25 are performed on SM-GEN training dataset with ten cross-validation folds. The confusion matrix of feature D and the corresponding data distribution plot are shown in Figure 22. From the confusion matrix, it is evident that feature D has substantial significance in discriminating the cheaters from victims while failing at identifying neutral cases. For all of the classifiers, we have obtained a classification F-measure of 0.68.
The confusion matrix of feature S and the corresponding data distribution plot are shown in Figure 23. Feature S is similar to feature D and has substantial significance in discriminating the cheaters from victims while failing at identifying neutral cases. For all of the classifiers, we have obtained a classification F-measure of 0.61 for this feature.
The confusion matrix of feature P and the corresponding data distribution plot are shown in Figure 24. The F-measure of feature P is slightly dropped when compared to feature D, but the notable point is, from the confusion matrix it is evident that feature P can discriminate neutral content far better than prior features. For all of the classifiers, we have obtained a classification F-measure of 0.65 for this feature.

Figure 24. Confusion matrix and data distribution plot for feature P on SM-GEN train dataset

Figure 25. Confusion matrix and plot for feature L on SM-GEN train dataset
The confusion matrix of feature L and the corresponding data distribution plot are shown in Figure 25. By comparing confusion matrices of feature P and feature L, the latter can discriminate cheaters and victims more efficiently than the former, and the accuracy in distinguishing neutral content is same for both features. For all of the classifiers, we have obtained a classification F-measure of 0.68 for this feature.

Figure 26. Confusion matrices of different classifiers starting from top left LR, NB, RF, SVM for feature combination D-S-P-L
The confusion matrices of the D-S-P-L combination of features for LR, NB, RF and SVM are shown in Figure 26. This combination of features has overall better classification f-measure and performs better in classifying neutral documents as well as discriminating between cheaters and victims. Hence we have selected this combination of features. The highest classification F-measure of 0.816 is obtained with the random forest classifier for this combination of features.

Figure 27. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on SM-GEN test data set
Table 22. F-scores of classifiers on SM-GEN test dataset

<table>
<thead>
<tr>
<th>Classifier</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.79</td>
</tr>
<tr>
<td>NB</td>
<td>0.79</td>
</tr>
<tr>
<td>RF</td>
<td>0.78</td>
</tr>
<tr>
<td>SVM</td>
<td>0.76</td>
</tr>
</tbody>
</table>

The final features selected (D-S-P-L) are then extracted from SM-GEN test dataset, and the classifiers trained are now tested on this dataset. The output of these classifiers is represented using confusion matrices and are shown in Figure 27. The highest classification f-measure of 0.79 is obtained with the LR and NB classifiers.

We have now extracted the features D-S-P-L for the SM-CSGO dataset, which contains 500 tweets. The classifiers trained are now tested on this dataset. The output of these classifiers is represented using confusion matrices and are shown in figure 28. The highest classification f-measure of 0.77 is obtained with the LR and NB classifiers.
Figure 28. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on SM-CSGO data set

<table>
<thead>
<tr>
<th>Classifier</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.77</td>
</tr>
<tr>
<td>NB</td>
<td>0.76</td>
</tr>
<tr>
<td>RF</td>
<td>0.74</td>
</tr>
<tr>
<td>SVM</td>
<td>0.74</td>
</tr>
</tbody>
</table>

We have now extracted the location of players who are classified as a cheater or victim in the SM-CSGO data set and obtained the locations of these users. As the tweets considered were English language only, Figure 29 indicates the English speaking countries where cheating is predominant. From Figure 29, it is evident that cheaters are more prevailing in the United States of America. Figure 30 shows the locations inside the United States of America, indicated by circles, where cheating is predominant. The larger the size of the circle the more the number of the cheaters.

The classifiers built on the SM-GEN training data set are now tested on the MC-TF2C dataset, which contains 175 logs suspected of involving cheating. Features D-S-P-L
are extracted from the dataset, and while extracting P and L features for a message posted by a player ‘p’ belonging to a team ‘t’, we have considered all of the names of the opposite team players in the third person pronouns list (refer to section 3.5 to see how P and L features are extracted). By doing this, features P and L can perform well, and we can achieve higher accuracies. In this dataset, we are interested in identifying whether or not cheating is involved in a particular game. For this, if classifier classifies a message as cheater/victim, or in other words if any of the team’s messages indicates use of cheats/hacks by themselves or opponents in that game, the output will be “YES” and if all of the messages in the chat indicate neutral then the output will be “NO”. The f-measures of the classifiers are shown in Table 24.

Our intention in testing the classifier trained on SM-GEN, on MC-TF2C is to experiment with how well the classifier performs on a dataset of a different domain, and from f-scores, it is evident that our classifier performs efficiently on the MC-TF2C dataset. Even though SM-GEN contains tweets and MC-TF2C contains multiplayer chat logs, the wording used by users is similar in both datasets. In both multiplayer chats and twitter posts players express their views in short texts. To provide precision and latency players are often connected to the servers that are near to them. The game developers can prioritize releasing patches to their servers in these locations.
Figure 29. Filled map showing geographic locations where cheating is predominant.

Figure 30. Map showing locations in the USA where cheating is dominant
Figure 31. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on MC-TF2C data set

<table>
<thead>
<tr>
<th>Classifier</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.89</td>
</tr>
<tr>
<td>NB</td>
<td>0.88</td>
</tr>
<tr>
<td>RF</td>
<td>0.83</td>
</tr>
<tr>
<td>SVM</td>
<td>0.82</td>
</tr>
</tbody>
</table>

Table 24. F-scores of classifiers on MC-TF2C dataset

We have identified the cheater teams and their chat IDs from the documents classified as cheaters by using the above classifier. Using these chat IDs we have pulled
the stats corresponding to chat IDs from the MC-TF2S dataset, and the distribution of results along with stats of games with no cheating/hacking are compared using a boxplot. In a boxplot, whiskers indicate the locations of maximum and minimum, the inner rectangle spans the first to third quartile, and the segment inside the rectangle shows the median. Outliers are three times more than the third quartile or three times below the first quartile. Suspected outliers are 1.5 times more than that of the third quartile or 1.5 times below the first quartile.

Figure 32 shows the boxplot of damage done by an entire team to the opposite team. Notice that the datasets have different ranges. The range of Normal Team starts at 0 while that of Cheater Team starts around 25,000. Most of the values of Normal Team are less than 50,000 while most of Cheater Team’s values are more than 50,000. From the box plot, it is evident that a cheater team which uses cheating/hacking tools normally can do more damage on the opposite team.
Figure 33. Box plot of kills

Figure 33 shows the boxplot of the number of kills performed by an entire team on the opposite team. Notice that the datasets have different ranges. The range of Normal Team starts at 0 while that of Cheater Team starts around 180. Most of the values of Normal Team are less than 150 while most of Cheater Team values are more than 150. From the box plot, it is evident that a cheater team which uses cheating/hacking tools can perform more kills on the opposite team.

Figure 34 shows the boxplot of final scores of a team. Notice that the datasets have almost the same ranges. The range of Normal Team starts at 0 while that of Cheater Team starts at 2. Most of the values of Normal Team are less than three while most of Cheater Team's values are more than three. From the box plot, it is evident that a cheater team which uses cheating/hacking tools can achieve high scores compared to the opposite team.
The classifiers trained on SM-GEN are now tested on RV-CSGO, which contains 685 reviews of the game Counter Strike Global Offensive. The confusion matrices of the output of classifiers are shown in figure 35. The classifiers perform poorly on this dataset, which is evident from Table 25. Reviews tend to be longer and have a lot of information whereas social media posts in Twitter and multiplayer chat messages are smaller in length and players exhibit their views in short sentences. In social media posts and multiplayer chat messages the players are precise and speak to the point, but in reviews, the player expresses a lot of his experiences, and thus the overall word count in a review is more. Hence, the features selected on short texts did not work efficiently on longer texts.
Figure 35. Resultant confusion matrices of different classifiers starting from top left LR, NB, RF, SVM when applied on RV-CSGO data set

Table 25. F-scores of classifiers on RV-CSGO dataset

<table>
<thead>
<tr>
<th>Classifier</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.42</td>
</tr>
<tr>
<td>NB</td>
<td>0.41</td>
</tr>
<tr>
<td>RF</td>
<td>0.41</td>
</tr>
<tr>
<td>SVM</td>
<td>0.36</td>
</tr>
</tbody>
</table>
5. Conclusion and Future Work

The main contribution of this thesis is a novel approach for detecting cheaters by classifying a textual document as cheater, victim or neutral. The experiments in this thesis were performed on five datasets (SM-GEN, SM-CSGO, MC-TF2C, MCTF2S, RV-CSGO) from three sources (Twitter, logs.tf, Steam). We have experimented with various combinations of features and noticed that D-S-P-L features are more efficient in player categorization. Furthermore, we have found out that the lexical features (unigrams, bigrams, term frequency and inverse document frequency) and isQuestion feature are less efficient for the classification task. Our work showed that the classifier trained on social media data could also be used for multiplayer chat data; as the wording used by players in both contexts is often the same. But, our proposed model fails miserably on the reviews data set; as the players tend to use long sentences in game reviews which contrast with the length of posts on Twitter, where users tend to express their views in short sentences, that are used to train the classifier.

By identifying the cheaters and victims of a game in twitter, we can identify the locations where cheating is predominant in that game and release of patches can be prioritized to the servers present in these locations. The boxplots (Figures 32, 33, 34) shows that players by using hacks are leading in a gameplay. By identifying the victims in a game,
game developers can give them small incentives like in-game items, so that they do not leave the game.

Our research reveals that almost every online game has some sort of hacks except those games that run entirely on the server side. The number of players affected by a player using hacks is huge if the game type is an MMO. Our investigation highlights the importance of game genre to identify the type of cheats available for a game, as the same genre games often have the same hacks. There are several websites where cheat developers sell their hacks for monthly subscriptions. Most common anti-cheat mechanisms used for detection and mitigation of cheating are studied in the section 2.4.3.

The game developers often rely on third party anti-cheat service providers like valve anti-cheat (VAC). There are several cases where VAC has falsely banned several players and even cases where players complain that they have seen cheaters wrecking the game even after the introduction of VAC. From the recorded instances, we can conclude that VAC is not optimal in detecting cheaters. Steam can use our proposed algorithm as one of the measurements, along with their standard VAC techniques, to improve the accuracy of their system.

As the topic of cheating in online gaming is so vast, one drawback of our proposed model is, it is not possible to look at every type of cheating in detail and provide a detection, mitigation, and prevention method for each. Our proposed system can classify a given document as cheater, victim or neutral. From a document classified as a cheater, it is easy to get the cheater’s identity. But from a document classified as a victim, it’s hard to identify
the cheater in the document, as the player can use second/third person pronouns or nicknames to refer to the cheater.

Despite the fact that the research ended up at this point, still, there are many areas where researchers can make their effort in the further development of the project. Some are discussed below.

Our work reveals that the detection of cheaters using textual data highly depends on the identification of keywords that can distinguish cheat relevant information from a document and advanced feature selection. Depending on the game, it might be possible to improve the classification accuracy by gaining in-depth knowledge about that game. For example, in Counter-Strike, an FPS game, smurfing is a kind of cheating where high ranking skilled players create new accounts to combat with low ranked players. One can come up with better keywords and features by gaining in-depth knowledge of the gaming, and thus the accuracy of the classifier can be improved.

False hack accusers are those who accuse exceptionally skilled players of using hacks even though they are not. Even though our system cannot discriminate these false accusations, higher accuracies can be achieved by combining our algorithm with other traditional cheat detection mechanisms.

Our model is limited to English language and can be extended to support multilingual content. Many of the modern multi-player games come with a built-in voice chat support. By extracting the features from the audio transcript associated with voice chat as text and further analyzing it, the performance of the classifier can be improved. Our model can be
extended to build a chatbot, that can track the multiplayer chat messages in real time and
can throw a player out of the game if the player is confirmed of using hacks.
Bibliography


Chai, K. M. A., Chieu, H. L., & Ng, H. T. (2002, August). Bayesian online classifiers for text classification and filtering. In *Proceedings of the 25th annual international...
ACM SIGIR conference on Research and development in information retrieval (pp. 97-104). ACM.


90


