Multi-Purpose AI Password Analysis Tool

By:

Osama Khalid
27971
Sardar M. Zeeshan khan
27969
Salman Ali
27667

Supervised by:
Dr. Muhammad Mansoor Alam

Co-Supervised by:
Mr. Osamah Ahmed

Faculty of Computing
Riphah International University, Islamabad

Fall 2024

A Dissertation Submitted To

Faculty of Computing,

Riphah International University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of

the Degree of

Bachelors of Science in Cyber Security

Faculty of Computing
Riphah International University, Islamabad

Date: 12th November 2024

Final Approval

This is to certify that we have read the report submitted by ***Osama Khalid 27971, Sardar Muhammad Zeeshan Khan 27969, and Salman Ali 27667***, for the partial fulfillment of the requirements for the degree of the Bachelor of Science in Cyber Security (BSCY). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Cyber Security (BSCY).

Committee:

1 ____________________________

Dr. Muhammad Mansoor Alam
(Supervisor)

2 ____________________________

Mr. Osamah Ahmed
(Co-Supervisor)

3 ____________________________

Dr. Musharraf Ahmed
(Head of Department)

Declaration

We hereby declare that this document **“Multi-Purpose AI Password Analysis Tool”** neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanying report entirely based on our personal efforts, under the proficient guidance of our teachers, especially our supervisor **Dr. Mansoor Alam.** If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

Osama Khalid
27971

Sardar M. Zeeshan khan

27969

Salman Ali

27667

Dedication

This project “ Multi-Purpose AI password Analysis Tool” is dedicated to our friends, mentors, and educators, whose unwavering support and encouragement have been a constant source of strength throughout this journey. Their belief in our capabilities has fueled our determination to succeed. Moreover, the work presented here is also dedicated to all our colleagues and the entire academic environment that has contributed significantly to the theoretical and practical development of our professional activities. This determination to achieve the best for their clients has challenged us to strive for the best outcome. We appreciate you for standing with us and for always being there for us.

Acknowledgement

First of all, we are obliged to Allah the Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project. We would like to express our sincere gratitude to the following individuals whose guidance and support were instrumental in the successful completion of this project:

- **Project Supervisors:** We are grateful to our project supervisors **Dr. Muhammad Mansoor Alam** and **Mr. Osamah Ahmed** for their encouragement, comments, and support while projecting this formulation. We owe it to them for supporting and steering the work that we undertook together in the right path.
- **Faculty Advisors:** We would also like the express gratitude to our faculty advisors for their contribution throughout this project: **Dr. Muhammad Mansoor Alam**, whose enthusiasm for the topic ignited this project, and **Dr. Jawaid Iqbal**, whose expert insight was critical during the research process.
- **Colleagues and Friends:** Finally, we would like to thank our colleagues, **Mr. Osama Raza** and **Mr. Awais Nawaz**, for their encouragement, collaboration, and support throughout the course of this project. Their positive spirit and willingness to assist contributed significantly to our progress.

Osama Khalid
27971

Sardar M. Zeeshan khan
27969

Salman Ali
27667

Abstract

In the growing use of novel technology, it is essential to have secure information and data of individuals and organizations. The presented solution here by this project is in the form of a multi-purpose AI-driven password analysis tool. AI integration in this tool was specifically for the purpose of offense as well as defense in a strategic battle. On the offense side, it uses sophisticated techniques of way on cracking passwords and determining their strength and weakness. This capability gives the security professionals a clear perspective of some of the flaws that are present when it comes to different password settings, hence, helping them to build tougher protective mechanisms. At the same time, the tool operates as a protective guardian of the organizational perimeter, assessing password compliance and searching for breaches by utilizing higher patterns and analytical results. Through detecting exposed credentials, it allowed users to factor control and prevent problems related to data leakage and unauthorized access. Due to ease of use and availability of functional options, this AI tool develops into a valuable resource for people and companies who want to enhance their protection in cyberspace. In allowing the users to try out the strength of their passwords and also make efforts to come up with ways of combating new threats, it is a preventive measure in the ever-changing world of cyber threats.

Keywords: Multi-Purpose, Password Analysis Tool, AI-Driven, Artificial Intelligence, Cracking Passwords, Vulnerabilities, Advanced Algorithms.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Chapter 1: Introduction

1.1 Introduction

Password security is critical nowadays where everything is going digital and more so with the ever increasing cases of cyber threats. Enter the AI Multipurpose Password Analysis Tool that will help you out. It is an innovative software which is actually designed to create a change in the way people deal with passwords. By integrating high-end artificial intelligence with the old school password cracking methods, this tool provides the entire package for securing our online identities. In spite of the fact of cracking passwords or getting an idea about password's strength, this instrument has many facets of password safety. But wait, there is more - it integrates right into well-known tools such as Hashcat, thereby increasing its versatility even further. Thus, with the entering of password management in the future, the AI Multipurpose Password Analysis Tool is the forthcoming one with both features as analysis and cracking one that will guard our important data from threats in the virtual world.

1.2 Opportunity & Stakeholders

This project's potential and main stakeholders are outlined as follows.

1.2.1 Opportunities

- **Enhanced Security Assessments:** Organizations and individuals can use the tool for penetration testing and vulnerability scan for their weak points when it comes to password security policies and standards.
- **AI-driven Offensive and Defensive Capabilities:** Using AI to break passwords is thus a powerful tool for a password attack, while using the same technique to check on password strength will go a long way in enhancing password integrity.

1.2.2 Stakeholders

- **Businesses:** Organizations strive to enhance the overall cybersecurity status of their organizations by looking for weaknesses in password usage.

- **Security Professionals:** Vulnerability assessors and ethical hackers working in an organization, security consultants who use the tool for their work.
- **Law Enforcement:** The agency may use the tool to conduct legal investigations following the court order and legal remands.

1.3 Problem Statement

However, as the increase of complexity of passwords steadily raised as the standard and security measures improved, brute force attacks get slower and ineffective.

The Figure 1.1: Upper Case, Figure 1.2: Upper & Lower Case, Figure 1.3: Upper, Lower & Numeric, Figure 1.4: Upper, Lower, Numeric & Char show the no. of complexity in terms of combination when increased from 4 onwards with mixture of Uppercase, lowercase, special characters and numbers.

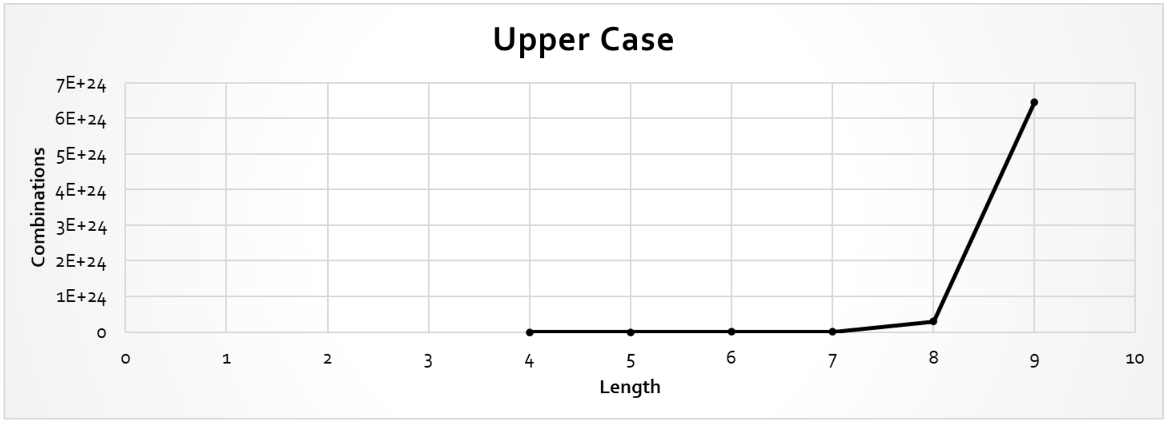


Figure 1.1: Upper Case

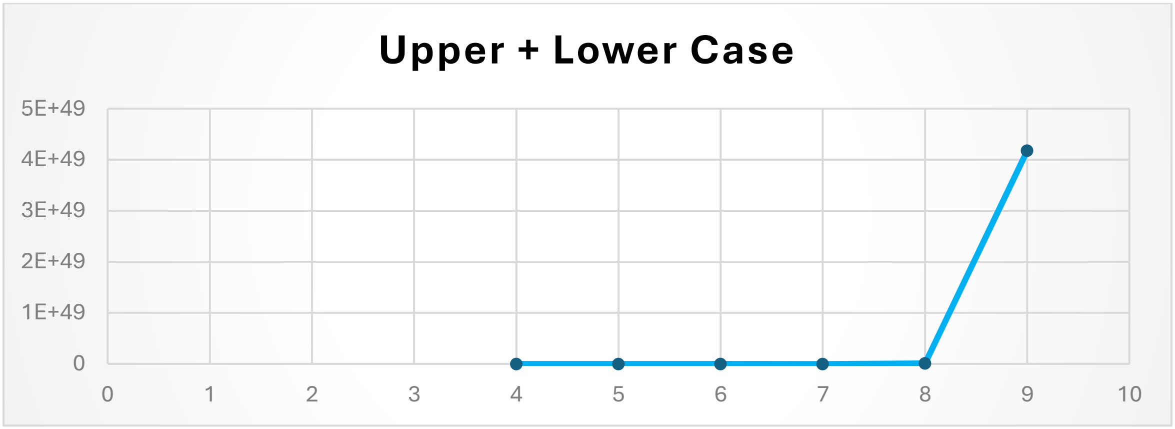


Figure 1.2: Upper & Lower Case

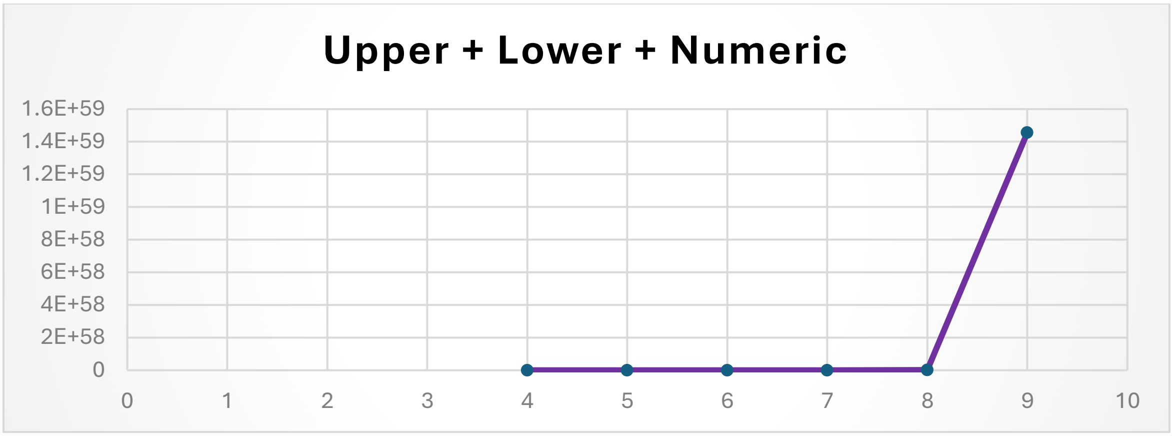


Figure 1.3: Upper, Lower & Numeric

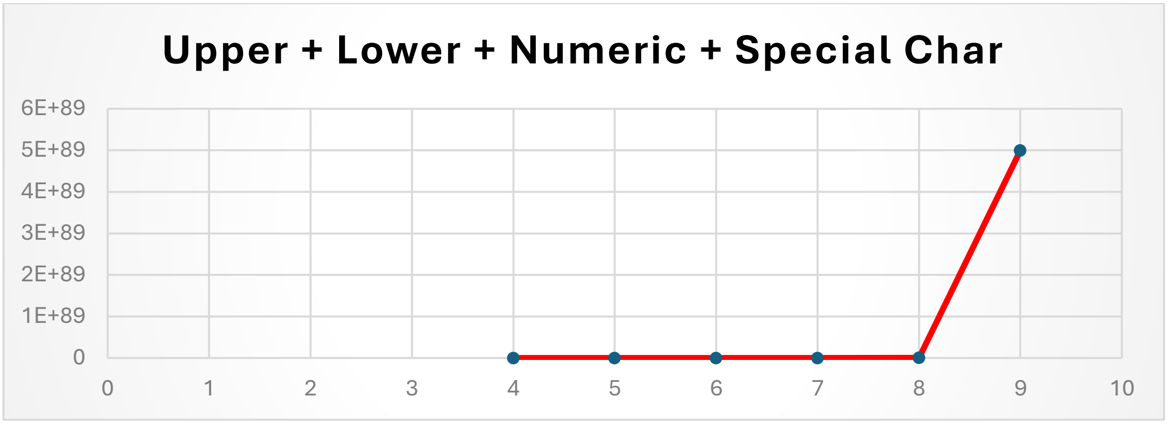


Figure 1.4: Upper, Lower, Numeric & Char

Nowadays, users create passwords that are longer and more complex e.g. combining uppercase lowercase symbols and number which makes brute force technique unsuitable. The tables Table 1.1: Length and their respective No. of combinations, Table 1.2: Passwords Length and their Respective Combinations up to 9th. below show the no. of iterations for each length of passwords.

Table 1.1: Length and their respective No. of combinations

Length	**Combination**
4^{26+26}	22204460492503130808472633361816 40625
$4^{26+26+10}$	21267647932558653966460912964485513216
5^{26+26}	2909812598873150618315302561643530 6561536
$5^{26+26+10}$	216840434497100886801490560173988342 28515625

Table 1.2: Passwords Length and their Respective Combinations up to 9th.

Length	Upper Case	Upper + Lower Case	Upper + Lower + Numeric	Upper + Lower + Numeric + Special Char
4	4.5036E+15	2.02824E+31	2.12676E+37	3.92319E+56
5	1.49012E+18	2.22045E+36	2.1684E+43	5.04871E+65
6	1.70582E+20	2.90981E+40	1.75945E+48	1.40029E+73
7	9.38748E+21	8.81248E+43	2.48931E+52	2.74926E+79
8	3.02231E+23	9.13439E+46	9.80797E+55	7.77068E+84
9	6.46108E+24	4.17456E+49	1.45558E+59	4.998E+89

1.4 Motivation and Challenges

Below are the reasons for developing the tool and potential difficulties when using Artificial Intelligence in the password analysis process.

1.4.1 Motivation

You only need to open a newspaper or switch on the news to find tales of password breaches, and so the security of passwords today cannot be overemphasized. Several motivations explain why there is a necessity to develop an enhanced AI-informed password analysis tool.

- **Increasing Complexity of Passwords**
  - The clearly emerging types and levels of cyber threats require persons using computers to often set better passwords to secure their ordeals. But due to this complexity the traffic tends to fall into a pattern that can be easily exploited.
  - This is where our tool comes in since through applying AI techniques the tool is able to identify such patterning and hence enhance both attacking and defending security postures.
- **Advancements in AI and Machine Learning**
  - Advanced AI algorithms create new opportunities in password solving due to the complexity of AI approach. It will be possible to maintain competitiveness against conventional techniques by building the AI into the tool.

    - Modern passwords are complex and diverse to create, which is why it is possible to implement the tool using the latest AI methods.
- **Comprehensive Security Assessments**
    - Businesses and security professionals need comprehensive tools that can provide both offensive and defensive capabilities. By integrating password strength analysis with AI-driven password cracking, our tool offers a holistic approach to password security.
    - The use of security assessments to both audit security measures and to test password's strength ensures improved vulnerability detection.
- **Growing Number of Data Breaches**
    - Major events are getting more and more frequent and large-scale, leaking millions of passwords. From these breaches our tool can learn general patterns and weaknesses that exist, improving the cracking functions and offering insight on proper password use.
    - Incorporated with breach-checking features, users can be notified of breached passwords to make the necessary changes a lot earlier.
- **Enhancing Cybersecurity Awareness**
    - Continue to educate people, through training programs and activities, on how weak passwords can get them into trouble and expose their personal data. That is why our tool will be useful: it can show how quickly one can crack weak passwords and encourage people to choose better ones.
    - This way security professionals can make use of our tool to educate the community while presenting them with proof about the requirements for strong password policies.

1.4.2 Challenges

- **Data Collection and Quality**
    - To achieve this, it needs to be emphasized for the dataset of human passwords to be as broad as possible containing as many sample passwords as possible. But legal and ethical consideration has to be followed when collecting data from leaks and breaches which are mostly sourced from the internet.

    - The next step of data preprocessing is to clean data and wash out all missing values or the values containing highly confidential and personal information.
- **Balancing Offensive and Defensive Capabilities**
    - Evaluating password strength (defensive side) and password cracking (offensive side) within the same tool will remain the major problem. It must be created as an ethically functioning tool with needed authorization of its work and legal compliance.
    - The defense capabilities should not be affected by the attack features of this tool, and vice versa, poses another significant challenge when designing and implementing it.
- **Adapting to Evolving Password Practices**
    - Password pattern and the conduct of the users do not form structural constant. These changes may not be evident during normal visits, and thus the new strategy we are employing using AI must constantly remain in a flexible state so as to maintain optimality.
    - This is due to the fact that the model is required to be updated fairly often to ensure that the passwords and their behavior are take into account the new techniques that may be coming out on the market.
- **Efficiency and Scalability**
    - Password cracking is a well-known problem whereas passwords get complex, traditional method of attacking is not practical because of its time and resource consumptions. Our goal is that the combination of the AI tool should dramatically decrease the search space and increase efficiency.
    - One of the main difficulties is to decide how to make the tool easily scalable to deal with large amount of data and long passwords without losing too much time.
- **Security and Privacy Concerns**
    - Adding check features like the one described in this case presupposes one’s ability to work with online databases of breaches, which have to be achieved without compromising users’ data.

- It is essential, therefore, to make certain not only that the tool is secure but that it cannot become a route by which attacks are made. This involves protecting users from various and sundry malicious actors seeking to take advantage of the tool.

1.5 Goals and Objectives

1.5.1 Goals

- Enhance Password Security
- Efficient Password Cracking
- Advanced Password Strength Analysis
- Comprehensive Vulnerability Assessments

1.5.2 Objectives

- Direct attention to the cracking attempts to those numerical passwords which are numerically feasible.
- Evaluate passwords from the point of view of such factors as length, charset, character types and frequency, and usage of AI algorithms.
- Reduced cracking time.
- Improved vulnerability assessment.

1.6 Solution Overview

Our proposed tool is a intelligent system that improves password security through intensive methods and password data. It focuses on both offensive and defensive capabilities.

- **Focused Cracking Efforts:** You get statistically the most likely password combinations and decrease the search space to crack down as compared to brute force.
- **Password Strength Analysis:** Solves AI-aided pattern recognition of the complexity and other associated parameters of intricate passwords to help design weaknesses in such passes to enable cracking them easily.
- **Reduced Cracking Time:** It makes the work quicker through focusing on probable password sequences which shrinks the cracking session time.

- **Improved Vulnerability Assessment:** Offers detailed password risk analyses that enables one to find out areas of weakness and work on them before any attacker goes for the kill.

1.6.1 Scope of the Project

- **Defensive capabilities:**
  - **Password Strength Assessment:** Automatic password strength, perceived words identification, and patterns finding with AI.

<u>Optional</u>

  - **Breach Checking:** Link up with data breach checking sites to check whether the passwords and email addresses that have been entered have been compromised.
- **Offensive capabilities: (For Ethical purposes only):**
  - **AI-powered password cracking:** Make use of password comparison and top tiered algorithms to best get over passwords within a range and only when permitted.

1.7 Report Outline

The report of this project, termed "Multi-Purpose AI Password Analysis Tool," features an introduction section that provides a comprehensive background into password security and the motivators for developing this tool. It offers insights into the advantages of improved password analysis and identifies those who will benefit from it. The report also presents an overview of previous password analysis tools, including John the Ripper, Brutus, and the Web application Wfuzz, before addressing the problem statement and solution. Our proposed solution leverages AI to identify statistically likely password characters and complexity values, aiming to decrease cracking time and enhance vulnerability identification. The project goals are clearly outlined, highlighting defensive aspects such as password strength assessment and breach detection, as well as ethical offensive capabilities like AI-powered password cracking.

The assessment plan focuses on diversifying assessment data, which includes information from tainted breaches, password dictionaries' datasets, and synthetic datasets. Development and training subsections delve into how datasets for the application are obtained and pre-processed, the model is tuned and optimized, and the most suitable algorithms are selected. Implementation emphasizes password strength evaluation, breach identification, and a password update suggestion function, along with an optional automatic password cracking feature. Additionally, the report explains the design and development of the user interface, ensuring it works harmoniously with the AI model and its functionalities.

To demonstrate the tool's efficiency and utility, the report details testing and improvement procedures. It includes references and a list of figures to supplement the content and enhance readability, making the report both thorough and user-friendly.

Chapter 2:
Market Survey

Chapter 2: Market Survey

2.1 Introduction

The usual approach of password cracking comprised only a brute force and dictionary attacks with precomputed tables purposefully created for password cracking. Such approaches that are useful for specific purposes show severe problems. Brute force attack for instance is very costly in terms of time and most of the time cannot be effectuated for complex passwords owing to their many possibilities. On the other hand, dictionary attacks are constrained in the sense that attackers work from lists of words and phrases and may not contain specific, large and particular passwords developed by users. Furthermore, as password protection measures are changing, traditional approaches are not elastic and cannot cope with such approaches as the combination of lowercase and uppercase letters and special signs. Lastly, it has become a problem with new long and complicated passwords since the old tools fail to handle the increased number of computations and data storage.

2.2 Market Survey/Technologies & Products Overview

The essentials of password protection are rooted by protection against unauthorized access, which is often performed by encrypting methods and hashing. In the past few years, different tools and methods have been invented that are used to analyze and penetrate passwords, which have their distinctive qualities.

2.2.1 Existing Systems

The Existing systems which are in the market are not earlier than a decade of existence and are described as follows,

John the Ripper

- **Limitations**
  - Leaks depend on conventional methods of password cracking such as the dictionary attack, brute force attack and rainbow table attack.

    - It could be very cost effective yet maybe not very successful to identify and manage new password trends and change for wanted and unwanted accesses without much update and change.
- **Problems**
    - No incorporation with an AI element that would create passwords which are specifically fired out as a result of learned patterns with the software.
    - An absence of dynamic option in checking the passwords' strength excluding dictionary check.

Brutus

- **Limitations**
    - Primarily design for online password guessing through Network protocols such as HTTP, FTP, SMB and many others.
    - May not have the ability to potentially incorporate more specific AI processing related password lists of target values.
- **Problems**
    - Limited permission for cracking in distinct password and analysis in offline modality.
    - May not be compatible with the other defense features of the proposed tool such as the password quality analyzer and the credential breach finder.

Wfuzz

- **Limitations**
    - Its main purpose is web application security testing in terms of directory and file fuzzing and brute-force checking.
    - It can be devoid of extra utilities such as password decoders and password parsers.
- **Problems**
    - Considered for the present project not as the main tool developed for password analysis and cracking operations but for the given project.
    - Lack of integration with only the password handling and cracking algorithms of AI.

2.3 Comparative Analysis

Thus, the tools presented above, in this paper, such as John the Ripper, RainbowCrack, OphCrack, L0phtCrack, and Aircrack-ng have special characteristics which correspond to certain situations of password or network security. However, there are significant disadvantages, which are the inability to scale up the rating of password complexity, no use of artificial intelligence in password generation, and very low incorporation of defensive features such as breach discovery. Another example includes the absence of flexibility and AI integrated capabilities required in performing advanced password analysis of screens such as Brutus and Wfuzz.

Comparison Table

The Table 2.1: Tools currently in Market with their specifications (Continued) and Table 2.2: Tools currently in Market shows the comparison between the Tools that are being currently used in the market.

Table 2.1: Tools currently in Market with their specifications (Continued)

Tool	**John the Ripper**	**RainbowCrack**	**OphCrack**
Type	Password Cracker	Password Cracker	Password Cracker
Supported Platforms	Windows, Linux, macOS	Windows, Linux	Windows, Linux
Password Hashes Supported	Various (Unix, Windows, etc.)	LM, NTLM, MD5, SHA1, SHA256, SHA512	LM, NTLM
Attack Methods	Dictionary, Brute Force, Hybrid	Precomputed Hash Tables	Rainbow Tables, Brute Force
Speed	Fast	Depends on Rainbow Table size	Moderate
User Interface	Command Line	Command Line	GUI
License	Open Source	Freeware	Open Source
Usage	Penetration Testing, Password Auditing	Password Cracking	Password Recovery

Table 2.2: Tools currently in Market

Tool	L0phtCrack	Aircrack-ng
Type	Password Cracker	Wi-Fi Network Security Tool
Supported Platforms	Windows	Linux, macOS
Password Hashes Supported	LM, NTLM	WEP, WPA, WPA2
Attack Methods	Dictionary, Brute Force	Dictionary, Brute Force, WPS PIN
Speed	Fast	Depends on hardware and complexity of the password
User Interface	GUI	Command Line
License	Commercial	Open Source
Usage	Password Cracking	Wi-Fi Network Security Testing

2.4 Research Gaps

Below are the several gaps that exist in the current password cracking tools market.

2.4.1 **Limited AI Integration:** Most of the existing tools do not incorporate machine learning approaches to automatically generate reliable list of passwords using actual patterns observed from the real world data.

2.4.2 **Defensive Shortcomings:** Current tools don’t offer some of the core features, such as a feature that evaluates passwords rigorously or the feature that allows intersecting passwords with breached ones.

2.4.3 **Scalability Constraints:** When there is a relative raise in password length and complexity traditional tools experience performance issues which makes scaling challenging and computationally expensive.

2.4.4 **Adaptability:** Most of the tools are not able to handle unique and complex password formation such as combination passwords of upper and lower case letters, numbers and symbols which are commonly used nowadays.

2.5 Problem Statement

As password complexity continues to intensify, the efficiency of brute force drops drastically, it cannot work like a charm.

Below Figures Figure 2.1: Upper Case, Figure 2.2: Upper & Lower case, Figure 2.3: Upper, Lower & Numeric, Figure 2.4: Upper, Lower, Numeric & Special Characters show the no. of complexity in terms of combination when increased from 4 onwards with mixture of Uppercase, lowercase, special characters and numbers.

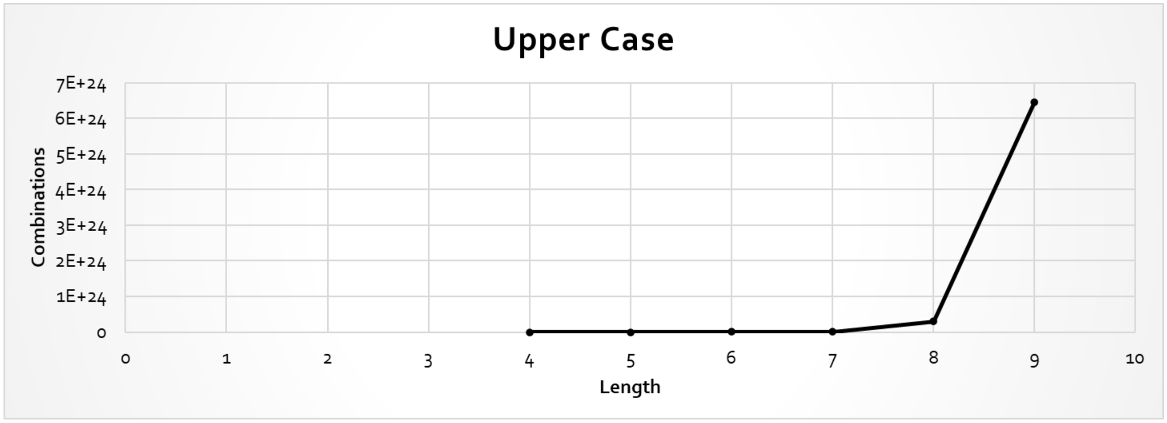


Figure 2.1: Upper Case

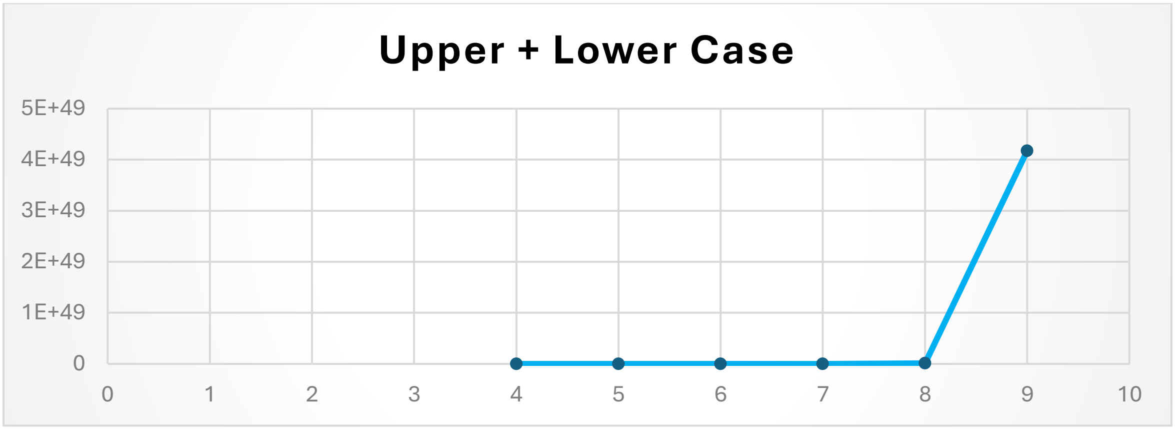


Figure 2.2: Upper & Lower case

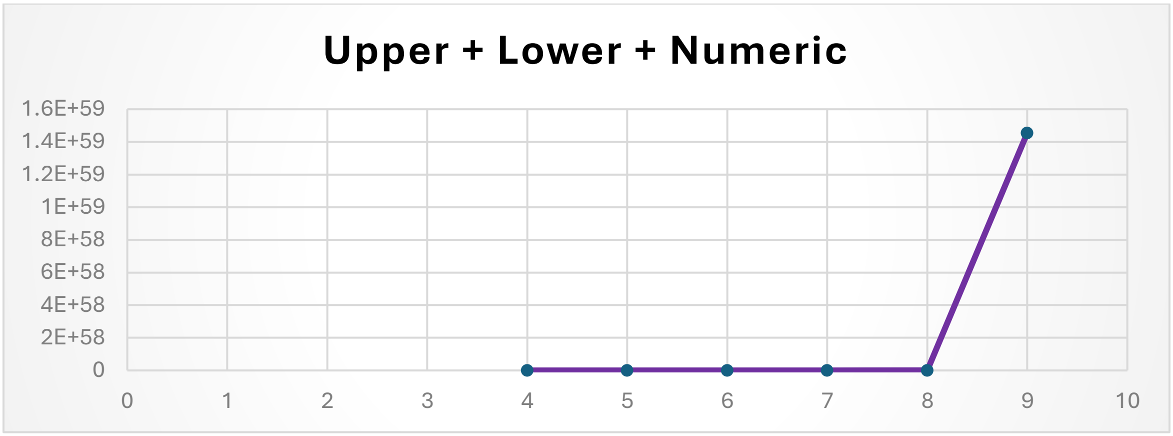


Figure 2.3: Upper, Lower & Numeric

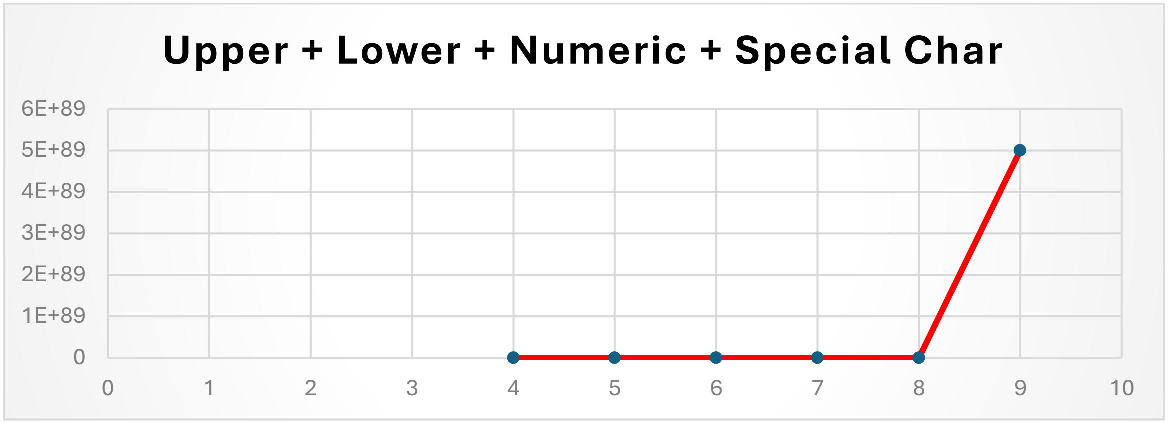


Figure 2.4: Upper, Lower, Numeric & Special Characters

Nowadays, users create passwords that are longer and more complex e.g. combining uppercase lowercase symbols and number which makes brute force technique unsuitable. The table below shows the no. of iterations for each length of passwords.

Table 2.3: Passwords Length with their no. of combinations

Length	Combination
4^{26+26}	22204460492503130808472633361816640625
$4^{26+26+10}$	21267647932558653966460912964485513216
5^{26+26}	2909812598873150618315302561643530656 1536
$5^{26+26+10}$	2168404344971008868014905601739883422 8515625

Table 2.4: Passwords Length with their no. of combinations upto 9th.

Length	**Upper Case**	**Upper + Lower Case**	**Upper + Lower + Numeric**	**Upper + Lower + Numeric + Special Char**
4	4.5036E+15	2.02824E+31	2.12676E+37	3.92319E+56
5	1.49012E+18	2.22045E+36	2.1684E+43	5.04871E+65
6	1.70582E+20	2.90981E+40	1.75945E+48	1.40029E+73
7	9.38748E+21	8.81248E+43	2.48931E+52	2.74926E+79
8	3.02231E+23	9.13439E+46	9.80797E+55	7.77068E+84
9	6.46108E+24	4.17456E+49	1.45558E+59	4.998E+89

2.6 Summary

Password security technology advancements show how the fight between cybersecurity experts and hackers goes on. John the Ripper tool, Brutus, Wfuzz are essential in password cracking and testing to secure systems. However, these methods are increasingly limited by the complexity of present day passwords generated by end users which include structures and special characters. This type of password shows weakness to traditional attacks including brute force and dictionary attacks since such kind of passwords takes so much time and depends a lot on precompiled lists that might not be complete in changes. Besides, they can also be less scalable and adjusted to the modern concept of password protection at the same time.

Advanced AI and machine learning in passwords is a revolutionizing move as it sources data-informed password cracking patterns and models. The capability set to be offered by this innovation is more advanced and complex than the previous solution and includes human-like password data generation, intelligent password strength analysis, and breach detection that helps protect the system better from contemporary cyber threats. The weakness of tools currently used, including AI not being used for generating a list of passwords, lack of strong defense mechanisms, and mutability, make the problem of password security vitally important.

This way, the present study identifies the technological shortcomings that can be further used to create a more extensive AI-based password analysis tool. It would have features of efficiency, flexibility, and security and would allow providing reliable protection from threats emanating from computer systems in a world that becomes more and more computerized.

Chapter 3:
Requirements and System Design

Chapter 3: Requirements and System Design

3.1 Introduction

The “Multi-Purpose AI Password Analysis Tool” presented here is an attempt to increase passwords’ safety by utilizing AI methods. This section presents the overall structure of the system as well as the functional and non-functional requirements, as well as the design needs required for building an efficient tool for password analysis and management.

3.2 System Architecture

The system architecture has several components listed below:

- **Graphical User Interface (GUI):** A user-friendly graphical interface for operations and to get interacted with the tool.
- **AI Model:** The core component utilizing RNNs for password generation and analysis.
- **Data Management Module:** Handles the collection, cleaning, and preprocessing of password datasets.
- **Security Module:** Performs a scan check and password assessment, a security breach check.
- **Integration Layer:** Integrates with external tools like Hashcat for multiple purposes and operations.

3.3 Functional Requirements

- **Password Generation:** The tool should generate secure passwords based on user-defined criteria.
- **Password Strength Analysis:** Analyze the complexity and strength of user-provided passwords.
- **Breach Checking:** Verify if passwords have been exposed in known data breaches.
- **Reporting:** Generate reports on password strength assessments and breach checks.

3.4 Non-Functional Requirements

- **Performance:** The tool should efficiently handle large datasets and provide quick responses for password analysis.
- **Scalability:** The system must scale to accommodate increasing numbers of users and datasets.
- **Security:** Ensure that user data and passwords are stored securely and that the tool adheres to best practices in cybersecurity.
- **Usability:** The interface should be intuitive and easy to navigate for users of varying technical expertise.
- **Reliability:** The system should be robust, with minimal downtime and effective error handling.

3.5 Design Diagrams

- **System Architecture Diagram:** The Figure 3.1: Design Architecture of the Tool. Illustrates the overall architecture, including components and their interactions.
- **Data Flow Diagram:** Figure 3.2: Methodology shows how data moves through the system, from input to processing and output.

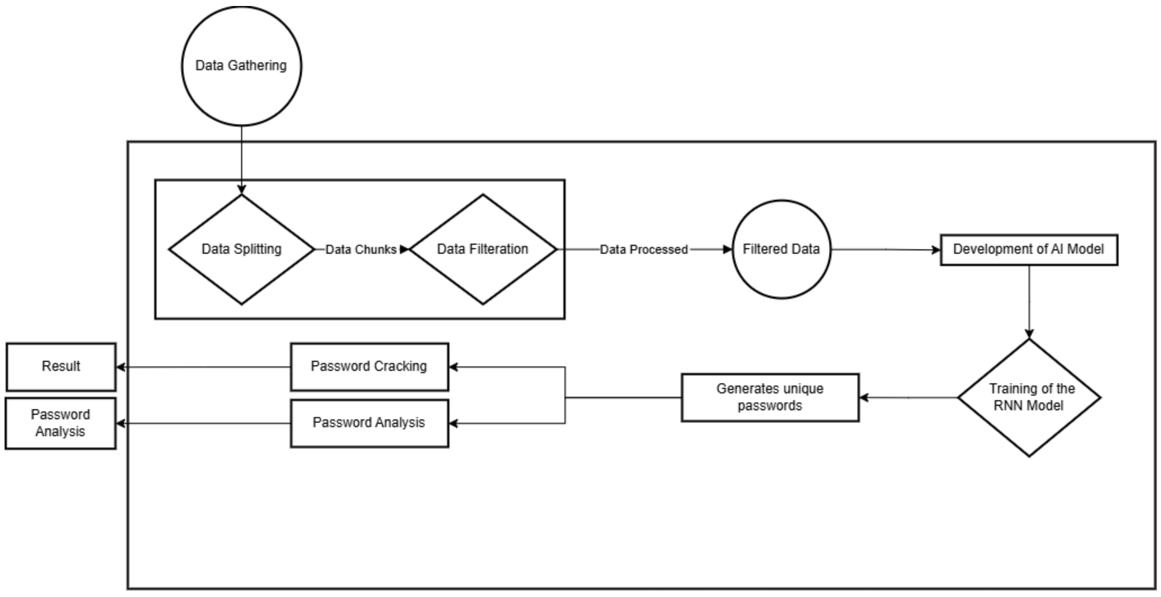


Figure 3.1: Design Architecture of the Tool.

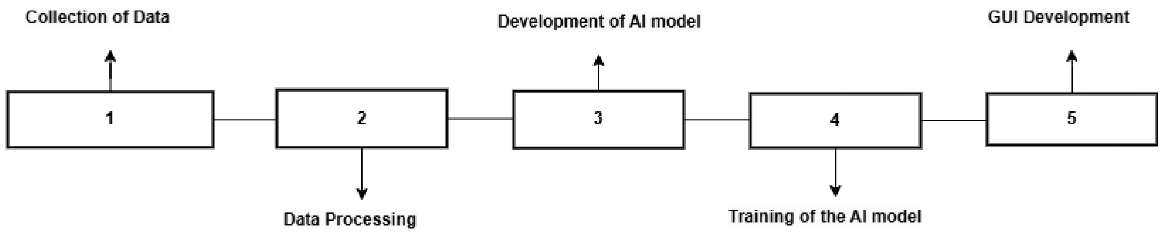


Figure 3.2: Methodology

3.6 Hardware and Software Requirements

3.6.1 Hardware Requirements

- **Workstation: <u>HP Z840</u>:** The HP Z840 is a high-performance desktop implemented for high profile activities like AI and machine learning model training. It provides provision for high compute capability, high memory allowance and superior graphical computations which are suitable for high level computations.
- **Processor: <u>Intel Xeon E5-2680 V4 Dual</u>:** It is a processor for server, it has 14 cores and 28 threads its base frequency is 2.4 GHz, and its turbo frequency is 3.3 GHz. While two of these processors enhance 28 cores and 56 threads performance that considerably improves multithreaded calculations important for training deep neural networks.
- **RAM: <u>64GB DDR4 ECC</u>:** 64GB of DDR4 Error-Correcting Code (ECC) RAM, we obtain high data integrity and excellent performance reducing data corruption amid critical calculations. Such an amount of memory allows multiple AI models and applications to be operative without affecting one another's efficiency.
- **GPU: <u>RTX 3070TI 8GB GDDR6x</u>:** Integrated with 8G DDR6x memory, supports parallel computing and. NVIDIA GeForce RTX 3070TI 8GB GDDR6: The graphics card features NVIDIA GeForce RTX 3070TI GPU, 8GB GDDR6x memory, offering powerful parallel processing performance for deep learning Its CUDA cores and Tensor cores are defined for AI applications, thus providing incredibly faster training and performing aspects of neural networks.
- **SSD: <u>512GB</u>:** Fast Interface Data Transfer with a 512GB SSD then assisting quick booting and loading times and a responsive user interface. The SSD is solely

employed for OS/boot, application, and project file storage because the SSD enabling fast data access.

- **CUDA Toolkit: <u>Version 12.4</u>:** A development kit that is released by NVIDIA for the programming of applications that will utilize a NVIDIA GPU.

3.6.2 Software Requirements

- **Operating System: <u>Windows 11 64bit</u>**: This is a new windows operating system that comes with a new look, high security measures ad support for new hardware. Of course, the 64- bit is essential for achieving the potential of the workstation and each component, for example, the RAM, the dual processor, and others.
- **Integrated Development Environment (IDE):** <u>Visual Studio</u>: It is an Integrated Development Environment, for many programming languages and technologies which is developed by Microsoft. It supports several programming languages, and works with various frameworks and libraries; therefore it suits the top two requirements when the AI based password cracking program is being developed and tested.
- **Programming Language: <u>Python 3.12</u>:** Python 3.12 is the latest stable release of the Python programming language, known for its simplicity, readability, and extensive library support. It is widely used in AI and machine learning projects due to its powerful frameworks and community support. Python serves as the primary language for developing the AI models and integrating different components of the tool.
- **Machine Learning Framework: <u>TensorFlow</u>:** Machine learning is a sub branch of artificial intelligence where a computer is trained that how it will learn in the given data set and which models are ready to used when new data come in. It supports Deep learning and neural network training, it has very high performance even on CPU as too as on GPU. When selecting the parameters such as flexibility and scalability TensorFlow has become one of the most used kit by developers that use it for developing AI intensive applications.
- **Deep Learning Library: PyTorch:** Deep Learning Library named PyTorch which is developed in Facebook Artificial Intelligence Research laboratory and people

like it because PyTorch builds dynamic computation graph. It incorporated broadly in research and production in AI and can provide a basic but reliable acceleration with GPUs in addition to heterogeneous support with other ML software. This password analysis tool is used and also experimented on artificial intelligent models with PyTorch.

- **GPU-Accelerated Library:** cuDNN: cuDNN or CUDA Deep Neural Network library is a high level library, carried out by Nvidia, intended to improve efficacy of deep learning libraries. It has specific and fast running standard operation such as forward & backward convolution and pooling & normalization & activation layers, on the other hand cuDNN is very crucial to improve TensorFlow & PyTorch efficiency in RTX 3070TI GPU.
- **Windows Software Development Kit (SDK):** Version 10.0.22621.3233: Windows SDK for Windows 10.0.22621.3233: Windows Developer Pack including tools, frameworks, includes and docs required by applications and C++ projects for developing Windows apps.
- **Windows Visual Studio Installer: Version 17.5.0:** For a program that assists with dealing with installing and updating of Microsoft Visual Studio, this is the new version 17.5.
- **Python Modules Requirements:**

Below are the modules in Table 3.1: Python Modules that were used & Installed during this project. which we installed that was used and needed for the project.

Table 3.1: Python Modules that were used & Installed during this project.

Library Name	**Version**	**Library Name**	**Version**	**Library Name**	**Version**
asttokens	2.4.1	comm	0.2.2	pywin32	308
colorama	0.4.6	debugpy	1.8.7	PyQt5_sip	12.15.0
executing	2.1.0	decorator	5.1.1	PyQt5	5.15.11
filelock	3.13.1	fsspec	2024.2.0	py-cpuinfo	9.0.0
ipython	8.28.0	ipykernel	6.29.5	psutil	6.1.0
jedi	0.19.1	Jinja2	3.1.3	platformdirs	4.3.6
joblib	1.4.2	jupyter_core	5.7.2	parso	0.8.4
MarkupSafe	2.1.5	matplotlib-inline	0.1.7	packaging	24.1

torch	2.4.1+cu124	wcwidth	0.2.13	networkx	3.2.1
typing_extensions	4.9.0	traitlets	5.14.3	Pygments	2.18.0
tqdm	4.66.5	tornado	6.4.1	pure_eval	0.2.3
mpmath	1.3.0	torchvision	0.19.1+cu124	prompt_toolkit	3.0.48
torchaudio	2.4.1+cu124	sympy	1.12	pillow	10.2.0
stack-data	0.6.3	six	1.16.0	nest-asyncio	1.6.0
setuptools	70.0.0	pyzmq	26.2.0	numpy	1.26.3
PyQt5Qt5	5.15.2	python-dateutil	2.9.0.post0	Pandas	2.2.3
Scikit-learn	1.5.2	Seaborn	0.13.2	Matplotlib	3.9.2

3.7 Threat Scenarios

- **Malicious Use:** The application that is being wrongly used in some depraved activity like an unethical win or hacking a password. Remarkably it is necessary to mention that both the necessary functionalities of Dictionary Attack as well as Password Analysis may be potentially used for the criminal purposes in case of non-presence of the corresponding control forms. This goes a long way to show that there is a need to come up with some measures of putting into prevention measures that will deny the intruders access to the system.
- **Dictionary Attack:** This is a method used to break passwords by typing in every word in a directory or a list of resources that has the most probable passwords. The Dictionary files local to the attackers contain password, variations and combination attempts that the attackers by experience know will grant them the access sought.
- **Password Analysis:** This ranges from analyzing password strength which in most instances are accomplished by checking them against vulnerability or repetitiveness. This can be used to strengthen up on Security plus it can also be used to make an effortless guess on the passwords of that specific account which has been scheduled for some malicious purposes.

3.7.1 Malicious Use Cases

- **Unauthorized Access:** These come in handy if an attacker wishes to gain unauthorized access to a user account, system or network. Thus using weak passwords, they are capable of gaining unauthorized access, corrupting information, or interfering with services.

- **Credential Stuffing:** If Hale's team get hold of a list of usernames and passwords from one breach, they can attempt a dictionary attack using those same usernames and passwords on other websites, getting those users, who unfortunately use the same password in different websites, every time.
- **Data Theft:** The result of insider's access is information theft that may comprise of personal data, financial information or even organization's secrets with detrimental consequences for clients, shareholders, investors and employers.
- **Ransomware Deployment:** One can achieve remote access to a system and proceed to locking down the files by using ransomware in exchange for a payment.

3.7.2 Ethical Use Cases

- **Security Testing:** These tools are used by ethical hackers and security professionals in detailing the passwords and generally the overall systems. This assist organizations to strengthen their security in as much as passwords which can be exploited if not amended where made weak.
- **User Education:** Password strength alone can be an informative tool for organizations to help the users improve on creating better passwords, hence improving security.
- **Compliance:** There are many industries that have sounded security policies that call for periodic security audits. These regulations can be complied with by the proper use of password analysis tools.

3.7.3 Importance of Control Measures

To reduce the risk of unethical use, it is important to implement some control measures,

- **Access Control:** Password analysis should be kept away from personnel who have no business dealing with it. This way, the use of these tools will be limited to ethical hackers or any professional involved in a security organization.
- **Monitoring and Logging:** The use of password analysis tools should therefore be monitored through the put in place monitoring systems. This can serve to alert whenever there is an unauthorized attempt to gain access to restricted systems.

- **User Education:** Inform users about the importance of strong passwords and dangers of having weak passwords. Promoting the use of password managers in order to come up with better & harder to guess passwords.
- **Legal & Ethical Guidelines:** Since the application of these tools is widely accepted in various professions, set principles on their proper utilization. It is important to observe integrity in all security testing to avoid crossing the legal line without legal approval.

3.8 Summary

The concept of Multi-Purpose AI Password Analysis Tool is much powerful to secure the password in an environment requiring more cyber security than ever before. This chapter details the system architecture, which comprises several essential components: Rich and intuitive GUI for easy password handling, password and model generation service utilizing RNN, the integrated form for effective dataset management, security and breach checking module, integration with the Hashcat and other programs.

Functional requirements are well stated Such are capacities for initial password production, rigorous password assessment for strengths, checking against prevalent weaknesses, and detailed account on security tests. Performance related requirement guarantee of sufficiency and effectiveness of the tool, scalability and security related requirements guarantee the tool's ability to perform in context with current and future requirements while usability and reliability related requirement guarantee its compatibility with users' needs.

In summary, this chapter offers a detailed discussion of the pertinent design factors, without which it is impossible to create a sound password analysis tool. Thus, it is designed to become a useful and indispensable tool for individuals and organizations planning to improve their passwords and cybersecurity.

Chapter 4: Proposed Solution

Chapter 4: Proposed Solution

4.1 Introduction

In particular, the proposed solution to the "Multi-Purpose AI Password Analysis Tool" is discussed regarding the usage of the complicated AI-based methods for improving password security. This chapter gives an introduction into the Prognostic Model and its design, data acquisition, methods of data pre-processing as well as the methods used in creating this tool. Thus, the goal is to approach the job both as a problem of high performance password analysis and generation of secure passwords and as a system optimized for the current challenges of modern password security systems.

4.2 Proposed Model

Quantitative and experimental research is playing a central role while proposing the model for the "Multi-Purpose AI Password Analysis Tool". This section documents why specific methodologies have been used and how they make the tool efficient.

4.2.1 Quantitative Approach

The quantitative aspect of the model uses number and statistics where data is numerated and analyzed or generated in the form of passwords.

- **Data-Driven Analysis**
  - The model is based on a big sample of passwords that is acquired from different sources as breaches and password lists. This data is then quantitatively analyzed to determine if there are similarities and differences and if there are emergent beats in passwords.
  - Nominal estimators are used in order to evaluate the passwords' strength based on the complexity indicators such as number of characters, presence of upper-case, lower-case, numeral and non-alphanumerical signs, Shannon entropy.

- **Performance Metrics**

The effectiveness of the password generation and analysis features is measured using quantitative metrics.

  - **Accuracy:** The percentage of correctly identified password strengths.
  - **Efficiency:** The time taken to generate passwords or analyze their strength.
  - **Success Rate:** The percentage of passwords successfully cracked as tested amongst the existent passwords.

4.2.2 Experimental Approach

The experimental concern of the model lies in the repeated evaluation of the effectiveness of the introduced AI algorithms involved in password analysis and generation.

- **Model Training and Optimization**
  - The password datasets are then used to train Recurrent Neural Network (RNN) model. This involves using various architectures from (LSTM, GRU) to tuning on hyperparameters from learning rate and batch size among others.
  - The training process can be assumed as experimental, as the number of iterations has to be increased to achieve a better accuracy of the model and its predictability in terms of the password strength as well as in view is psychological patterns of human behavior to provide effective inspiration to generate human-like passwords.

- **Validation and Testing**

Consequently, concerning the different validation datasets, that denote the testing of the model, an assessment of its performance is made after being trained. This includes:

  - **Cross-Validation:** Computing the passwords that are being generated in an attempt to compare the best passwords with our training set in the endeavor of evaluating the effectiveness of the model and generate passwords, which possess characteristics used while training the model.
  - **A/B Testing:** In other words how to select and compare the different structural models to determine which format promotes better passwords generation/analysis.

4.2.3 Qualitative Insights

The major emphasis is put on quantitative and experimental simulations, while quantitative results are used to complement the improvement of the model.

- **Expert Reviews:** Seeking advice of cybersecurity specialists to review the model's usage strategy and efficiency. Their input is useful when it comes to making minor changes to the algorithm and make sure that the tool in question follows most of the standards in password security.

4.3 Data Collection

The collected datasets of various types of passwords from different sources are mentioned below.

- **Online Leaks and Breaches:** Using known breaches to gather data ethically to get the best sort of password types.
- **Password Dictionaries:** Leaving aside that existing password lists are available that contain passwords with different length, complexity and contents.

Collected Datasets Table

Below Table 4.1: Downloaded Passwords Dumps from various online resources. is the dumps we collected during the requirement analysis for the training of the model.

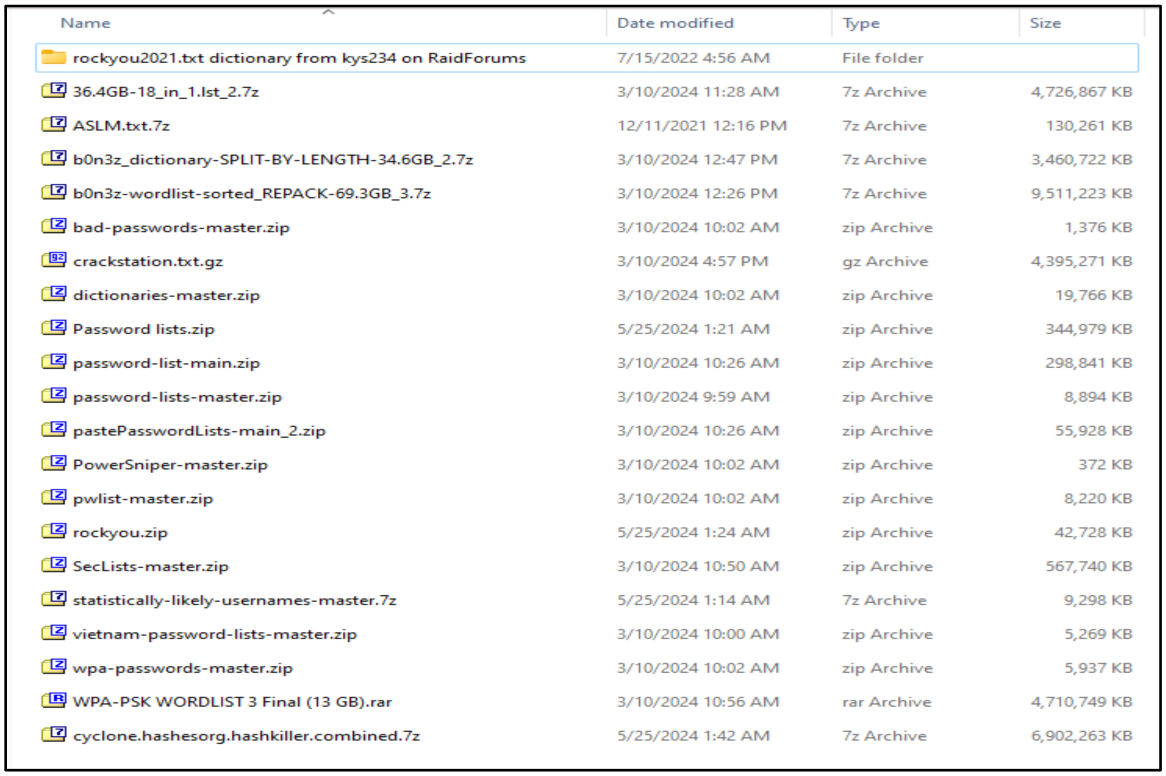
Table 4.1: Downloaded Passwords Dumps from various online resources.

No	Name of the list	Size
1	rockyou2021.txt dictionary from kys234 on RaidForums	12.7 GB
2	36.4GB-18_in_1.lst_2.7z	4.50 GB
3	ASLM.txt.7z	127 MB
4	b0n3z_dictionary-SPLIT-BY-LENGTH-34.6GB_2.7z	3.29 GB
5	b0n3z-wordlist-sorted_REPACK-69.3GB_3.7z	9.07 GB
6	bad-passwords-master.zip	1.34 MB
7	crackstation.txt.gz	4.19 GB
8	dictionaries-master.zip	19.3 MB
9	Password lists.zip	336 MB
10	password-list-main.zip	291 MB

11	password-lists-master.zip	8.86 MB
12	pastePasswordLists-main_2.zip	54.6 MB
13	PowerSniper-master.zip	0.3 MB
14	pwlist-master.zip	8.02 MB
15	rockyou.zip	41.7 MB
16	SecLists-master.zip	554 MB
17	statistically-likely-usernames-master.7z	9.07 MB
18	vietnam-password-lists-master.zip	5.14 MB
19	wpa-passwords-master.zip	5.79 MB
20	WPA-PSK WORDLIST 3 Final (13 GB).rar	4.49 GB
21	cyclone.hashesorg.hashkiller.combined.7z	6.58GB

Dataset Downloaded Screen Shot

The Figure 4.1: Screenshot of Datasets Dumps below shows screenshot of the data dumps in our system.



Name	Date modified	Type	Size
rockyou2021.txt dictionary from kys234 on RaidForums	7/15/2022 4:56 AM	File folder	
36.4GB-18_in_1.lst_2.7z	3/10/2024 11:28 AM	7z Archive	4,726,867 KB
ASLM.txt.7z	12/11/2021 12:16 PM	7z Archive	130,261 KB
b0n3z_dictionary-SPLIT-BY-LENGTH-34.6GB_2.7z	3/10/2024 12:47 PM	7z Archive	3,460,722 KB
b0n3z-wordlist-sorted_REPACK-69.3GB_3.7z	3/10/2024 12:26 PM	7z Archive	9,511,223 KB
bad-passwords-master.zip	3/10/2024 10:02 AM	zip Archive	1,376 KB
crackstation.txt.gz	3/10/2024 4:57 PM	gz Archive	4,395,271 KB
dictionaries-master.zip	3/10/2024 10:02 AM	zip Archive	19,766 KB
Password lists.zip	5/25/2024 1:21 AM	zip Archive	344,979 KB
password-list-main.zip	3/10/2024 10:26 AM	zip Archive	298,841 KB
password-lists-master.zip	3/10/2024 9:59 AM	zip Archive	8,894 KB
pastePasswordLists-main_2.zip	3/10/2024 10:26 AM	zip Archive	55,928 KB
PowerSniper-master.zip	3/10/2024 10:02 AM	zip Archive	372 KB
pwlist-master.zip	3/10/2024 10:02 AM	zip Archive	8,220 KB
rockyou.zip	5/25/2024 1:24 AM	zip Archive	42,728 KB
SecLists-master.zip	3/10/2024 10:50 AM	zip Archive	567,740 KB
statistically-likely-usernames-master.7z	5/25/2024 1:14 AM	7z Archive	9,298 KB
vietnam-password-lists-master.zip	3/10/2024 10:00 AM	zip Archive	5,269 KB
wpa-passwords-master.zip	3/10/2024 10:02 AM	zip Archive	5,937 KB
WPA-PSK WORDLIST 3 Final (13 GB).rar	3/10/2024 10:56 AM	rar Archive	4,710,749 KB
cyclone.hashesorg.hashkiller.combined.7z	5/25/2024 1:42 AM	7z Archive	6,902,263 KB

Figure 4.1: Screenshot of Datasets Dumps

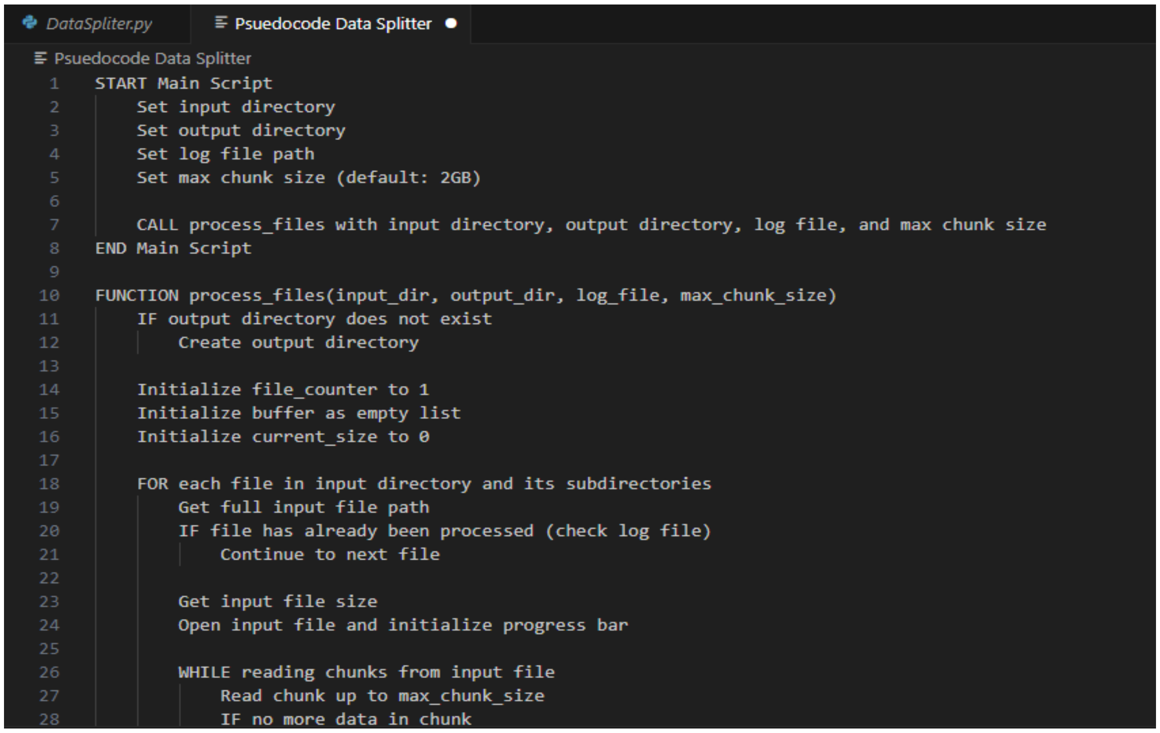
4.4 Data Pre-processing

The evaluation shows that data preprocessing is a critical and efficient component needed to prepare good data sets,

- **Data Splitting:** Dividing large datasets into manageable chunks (approximately 2 GB each) for efficient processing.
- **Data Cleaning:** Filtering passwords based on length (5 to 8 characters) and removing corrupted entries.

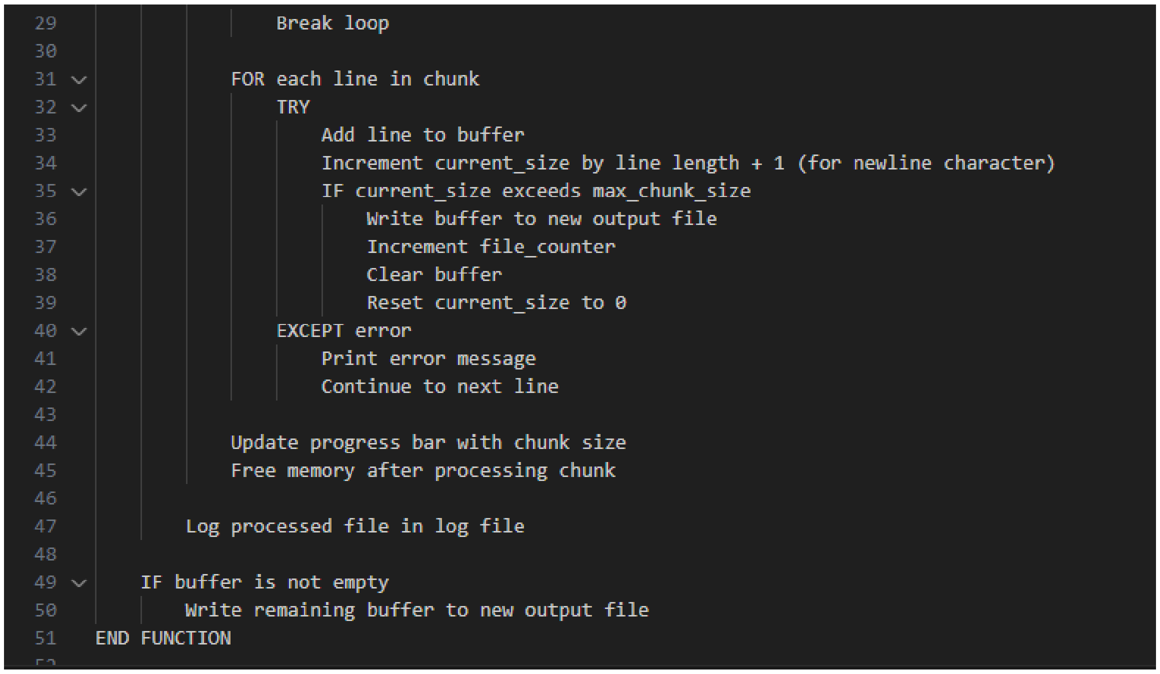
4.4.1 Data Splitting

We divided our dataset into multiple 2 GB files for efficient processing and management. Our approach involves processing these chunks independently, which helps in handling large datasets without the need for significant memory resources. Each chunk is processed, and the results are aggregated to ensure comprehensive analysis. This method allows us to train, validate, and test our model effectively while managing data size constraints.
Figure 4.2: Data Splitting Pseudocode, Figure 4.3: Data Splitting Pseudocode (Continued), Figure 4.4: Data Splitting Pseudocode (Continued) shows Pseudocode for Data Splitting.



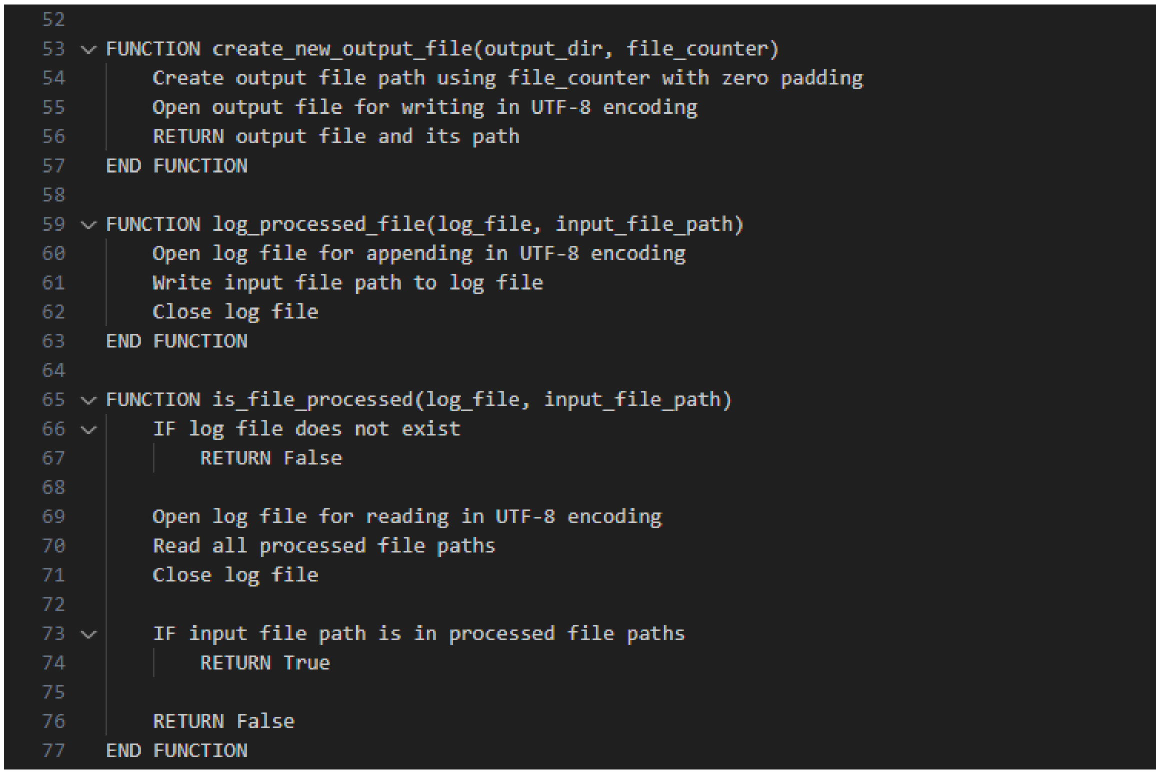
```
START Main Script
    Set input directory
    Set output directory
    Set log file path
    Set max chunk size (default: 2GB)

    CALL process_files with input directory, output directory, log file, and max chunk size
END Main Script

FUNCTION process_files(input_dir, output_dir, log_file, max_chunk_size)
    IF output directory does not exist
        Create output directory

    Initialize file_counter to 1
    Initialize buffer as empty list
    Initialize current_size to 0

    FOR each file in input directory and its subdirectories
        Get full input file path
        IF file has already been processed (check log file)
            Continue to next file

        Get input file size
        Open input file and initialize progress bar

        WHILE reading chunks from input file
            Read chunk up to max_chunk_size
            IF no more data in chunk
```

Figure 4.2: Data Splitting Pseudocode



```
                Break loop

            FOR each line in chunk
                TRY
                    Add line to buffer
                    Increment current_size by line length + 1 (for newline character)
                    IF current_size exceeds max_chunk_size
                        Write buffer to new output file
                        Increment file_counter
                        Clear buffer
                        Reset current_size to 0
                EXCEPT error
                    Print error message
                    Continue to next line

            Update progress bar with chunk size
            Free memory after processing chunk

        Log processed file in log file

    IF buffer is not empty
        Write remaining buffer to new output file
END FUNCTION
```

Figure 4.3: Data Splitting Pseudocode (Continued)



```

FUNCTION create_new_output_file(output_dir, file_counter)
    Create output file path using file_counter with zero padding
    Open output file for writing in UTF-8 encoding
    RETURN output file and its path
END FUNCTION

FUNCTION log_processed_file(log_file, input_file_path)
    Open log file for appending in UTF-8 encoding
    Write input file path to log file
    Close log file
END FUNCTION

FUNCTION is_file_processed(log_file, input_file_path)
    IF log file does not exist
        RETURN False

    Open log file for reading in UTF-8 encoding
    Read all processed file paths
    Close log file

    IF input file path is in processed file paths
        RETURN True

    RETURN False
END FUNCTION
```

Figure 4.4: Data Splitting Pseudocode (Continued)

Figure 4.5: Flowchart of Data Splitting shows the flowchart of the Data Splitting method.

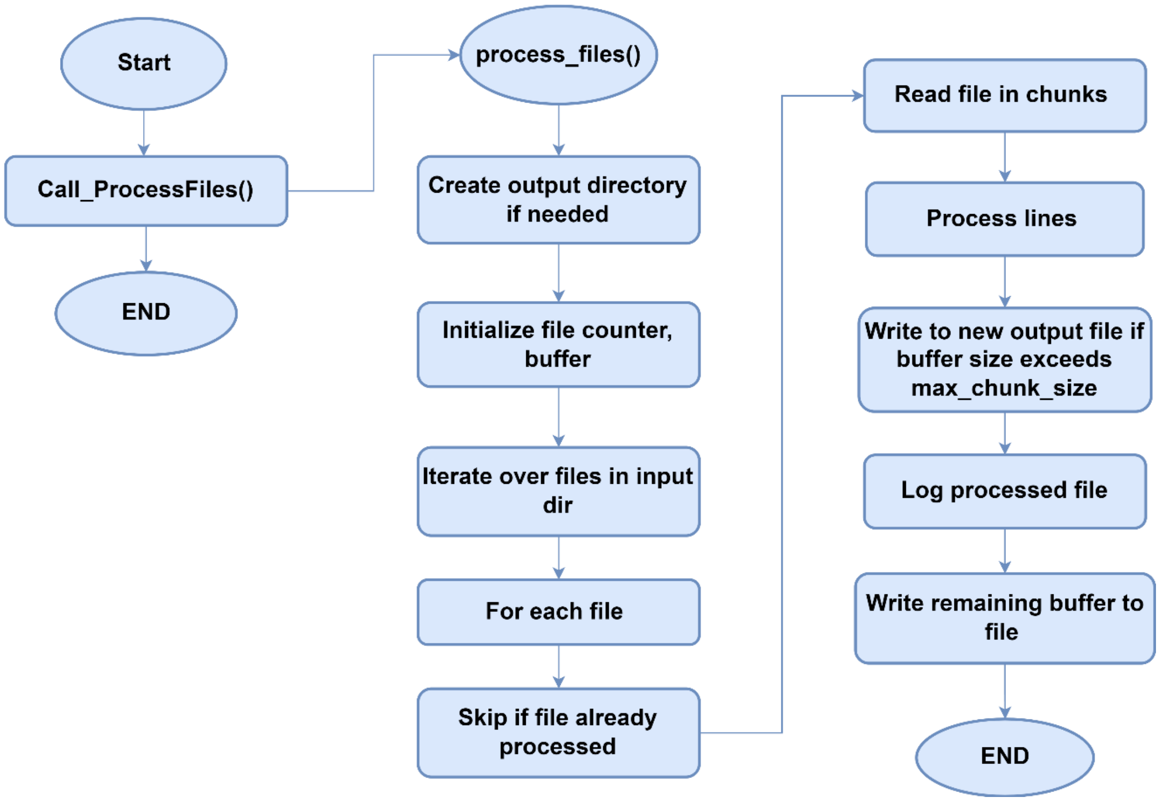

Start
Call_ProcessFiles()
END
process_files()
Create output directory if needed
Initialize file counter, buffer
Iterate over files in input dir
For each file
Skip if file already processed
Read file in chunks
Process lines
Write to new output file if buffer size exceeds max_chunk_size
Log processed file
Write remaining buffer to file
END

Figure 4.5: Flowchart of Data Splitting

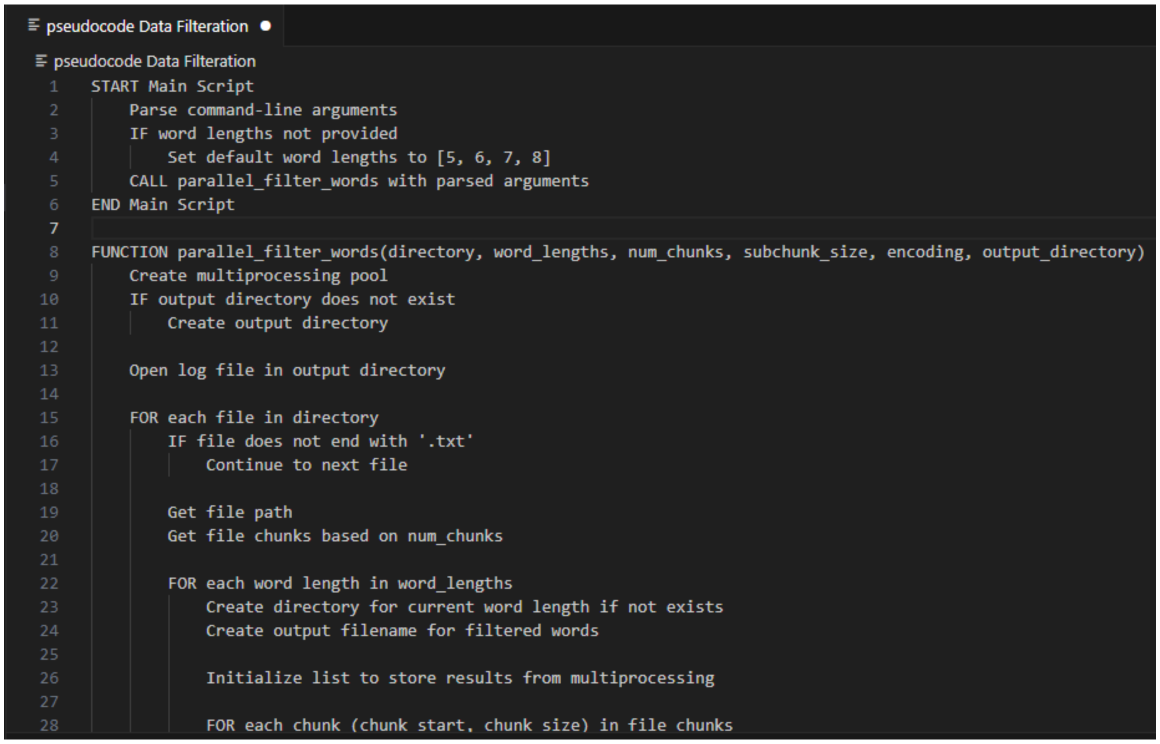
- **Saving the Pre-processed Dataset**

As the files contain datasets of huge volumes i.e. 120GB, 90GB, 100GB. We couldn't run these files. So, we split each file into multiples files which were approx. 2 GB and saved them as txt files and used the Data filtration technique to filter the passwords for training.

4.4.2 Data Cleaning

- Filter the password of length of 5 to 8 to keep the datasets clean and manageable.
- Get rid of any weird or corrupted entries that could mess up training.

Figure 4.6: Pseudocode of Data Filtering, Figure 4.7: Pseudocode of Data Filtering (Continued), Figure 4.8: Pseudocode of Data Filtering (Continued), Figure 4.9: Pseudocode of Data Filtering (Continued) shows the Pseudocode of Data Filtration

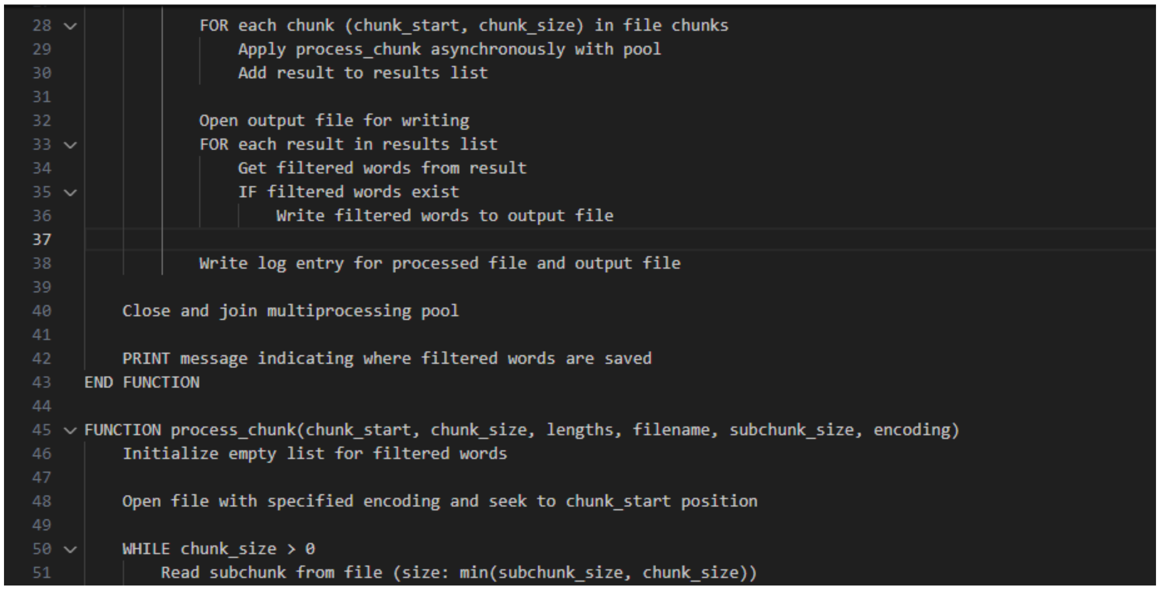


pseudocode Data Filteration

pseudocode Data Filteration

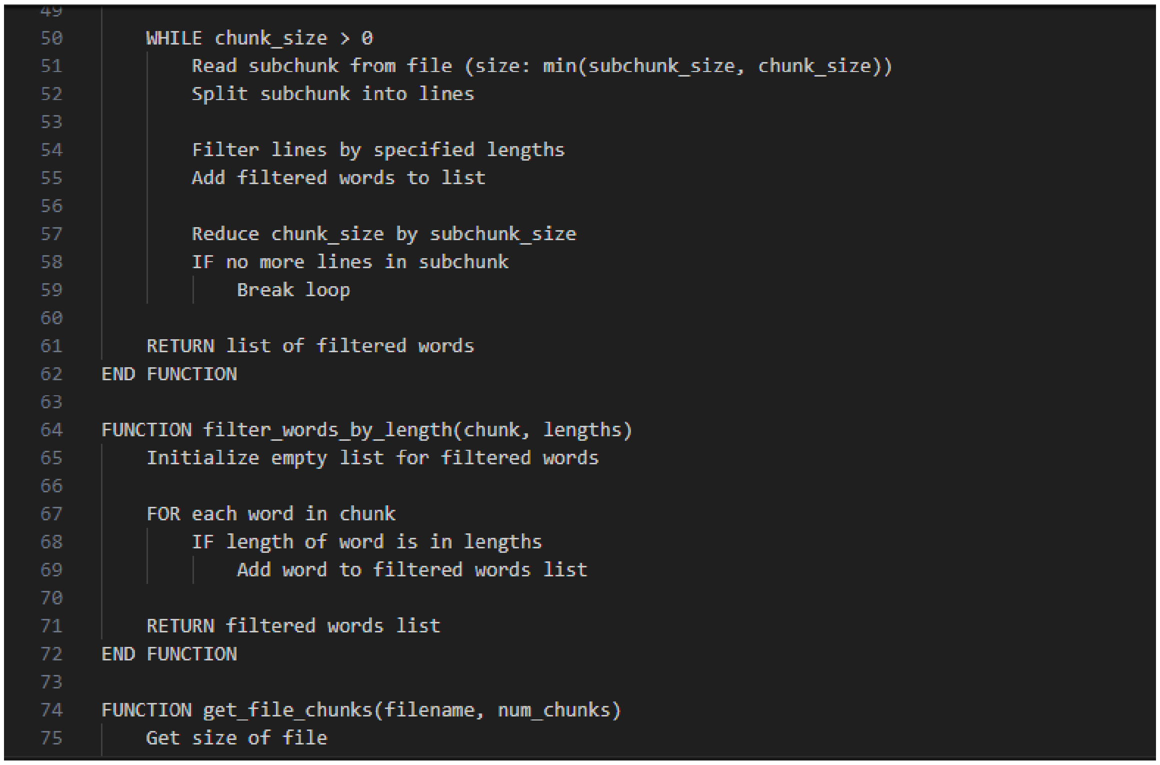
```
START Main Script
    Parse command-line arguments
    IF word lengths not provided
        Set default word lengths to [5, 6, 7, 8]
    CALL parallel_filter_words with parsed arguments
END Main Script

FUNCTION parallel_filter_words(directory, word_lengths, num_chunks, subchunk_size, encoding, output_directory)
    Create multiprocessing pool
    IF output directory does not exist
        Create output directory

    Open log file in output directory

    FOR each file in directory
        IF file does not end with '.txt'
            Continue to next file

        Get file path
        Get file chunks based on num_chunks

        FOR each word length in word_lengths
            Create directory for current word length if not exists
            Create output filename for filtered words

            Initialize list to store results from multiprocessing

            FOR each chunk (chunk_start, chunk_size) in file chunks
```

Figure 4.6: Pseudocode of Data Filtering



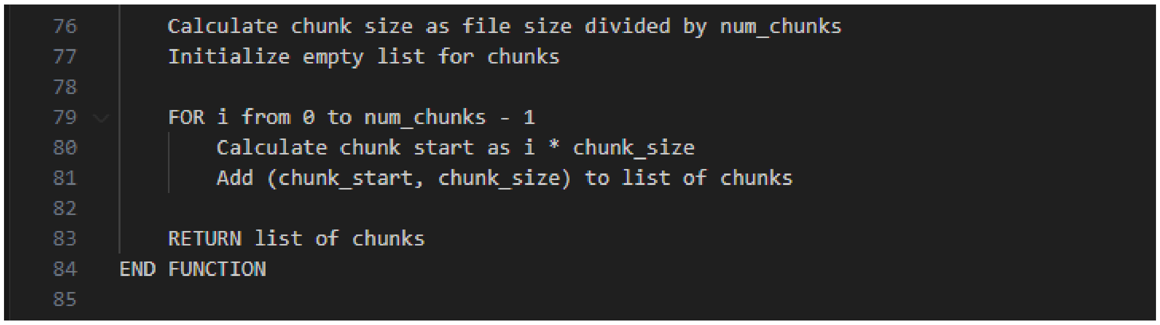
```
            FOR each chunk (chunk_start, chunk_size) in file chunks
                Apply process_chunk asynchronously with pool
                Add result to results list

            Open output file for writing
            FOR each result in results list
                Get filtered words from result
                IF filtered words exist
                    Write filtered words to output file

            Write log entry for processed file and output file

    Close and join multiprocessing pool

    PRINT message indicating where filtered words are saved
END FUNCTION

FUNCTION process_chunk(chunk_start, chunk_size, lengths, filename, subchunk_size, encoding)
    Initialize empty list for filtered words

    Open file with specified encoding and seek to chunk_start position

    WHILE chunk_size > 0
        Read subchunk from file (size: min(subchunk_size, chunk_size))
```

Figure 4.7: Pseudocode of Data Filtering (Continued)



```
    WHILE chunk_size > 0
        Read subchunk from file (size: min(subchunk_size, chunk_size))
        Split subchunk into lines

        Filter lines by specified lengths
        Add filtered words to list

        Reduce chunk_size by subchunk_size
        IF no more lines in subchunk
            Break loop

    RETURN list of filtered words
END FUNCTION

FUNCTION filter_words_by_length(chunk, lengths)
    Initialize empty list for filtered words

    FOR each word in chunk
        IF length of word is in lengths
            Add word to filtered words list

    RETURN filtered words list
END FUNCTION

FUNCTION get_file_chunks(filename, num_chunks)
    Get size of file
```

Figure 4.8: Pseudocode of Data Filtering (Continued)



```
    Calculate chunk size as file size divided by num_chunks
    Initialize empty list for chunks

    FOR i from 0 to num_chunks - 1
        Calculate chunk start as i * chunk_size
        Add (chunk_start, chunk_size) to list of chunks

    RETURN list of chunks
END FUNCTION
```

Figure 4.9: Pseudocode of Data Filtering (Continued)

The Figure 4.10: Data Filtering Flowchart below shows the flowchart of the Data Filtration method.

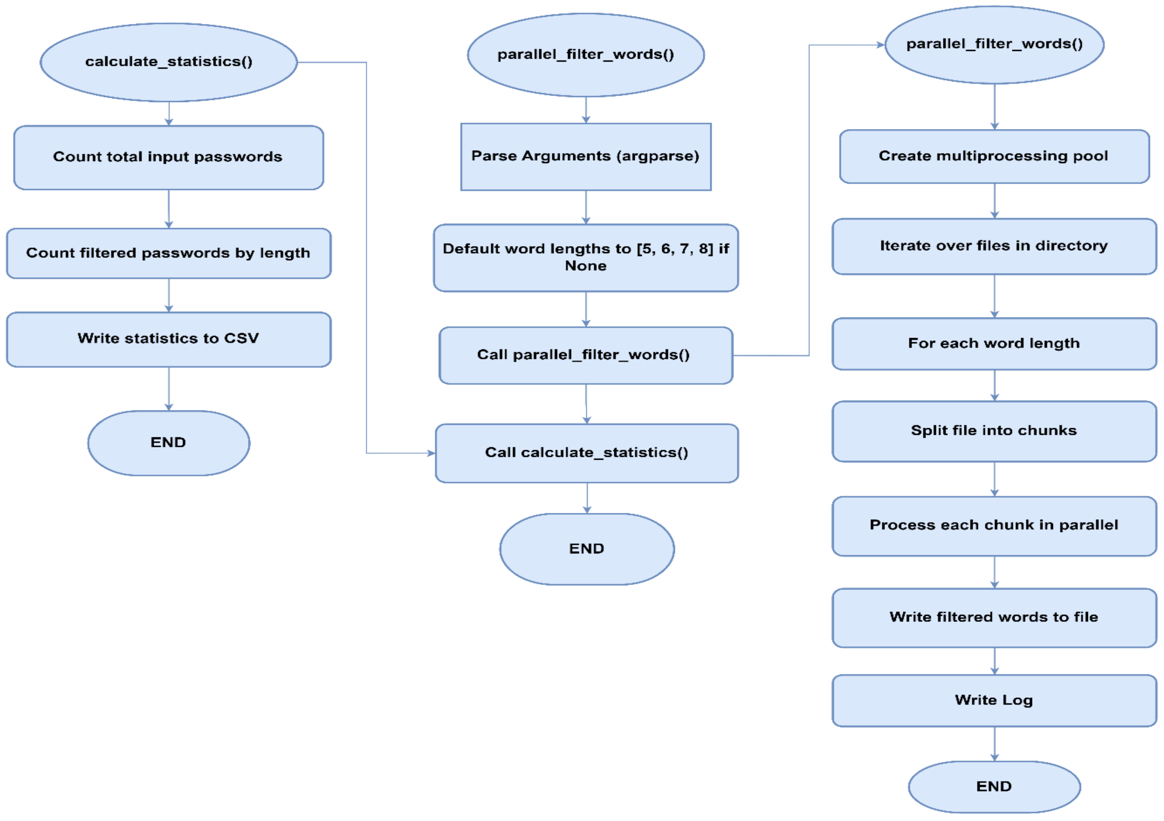

calculate_statistics()
Count total input passwords
Count filtered passwords by length
Write statistics to CSV
END
parallel_filter_words()
Parse Arguments (argparse)
Default word lengths to [5, 6, 7, 8] if None
Call parallel_filter_words()
Call calculate_statistics()
END
parallel_filter_words()
Create multiprocessing pool
Iterate over files in directory
For each word length
Split file into chunks
Process each chunk in parallel
Write filtered words to file
Write Log
END

Figure 4.10: Data Filtering Flowchart

The Table 4.2: Total no. of filtered password w.r.t to their lengths. below shows the total no. of filtered password that were produced after splitting and filtering of the data.

Table 4.2: Total no. of filtered password w.r.t to their lengths.

Total number of Input passwords	**8459063135**
Length	**Total Count of Filtered Passwords**
5	10,751,871
6	563,608,354
7	465,597,114
8	1,357,729,013
TOTAL	2,397,686,352

4.5 Tools and Techniques

The development of the tool employs various tools and techniques.

4.5.1 Tools

- **Programming Languages:** Python 3.12 for implementing AI models and data processing.
- **Machine Learning Frameworks:** To design and train the RNN model will use TensorFlow and PyTorch.
- **Data Management Tools:** Scrubbing, pre-processing, data selection, and data partitioning scripts for our AI model development.
- **Visualization Tools:** Diagrams, flowcharts, tables and pseudocodes to show processing of data and model training methods.
- **Integrated Development Environment (IDE): Visual Studio:** Programming Environment Used for writing, testing and for running the application.
- **Password Cracking Tool: Hashcat:** used in order to improve the effectiveness in cracking passwords.
- **CUDA Toolkit Version 12.4:** The CUDA Toolkit is a developer application programming interface, which is issued by NVIDIA Corporation for the development of applications capable of utilizing NVIDIA GPUs. It comprises libraries, debuggers and optimizers, a compiler and other utilities needed when programming a GPU.
- **Windows Software Development Kit (SDK) Version 10.0.22621.3233:** The Windows SDK is a set of tools, libraries, and sundry files and better still, the documentation that developers need when writing applications that will run Windows OS's. Helps the developers get all the tools they require to develop, construct, test and even tweak for Windows.
- **Windows Visual Studio Installer Version 17.5.0:** The Visual Studio Installer is an overlaid utility that aids in the administration of the installation and update of Microsoft Visual Studio. It enables a user to install only what he requires to develop his software through workloads and other components.

4.5.2 Techniques

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are other types of neural networks which involve a feed back of previous computation by the aid of hidden state which enables the network to understand past inputs in a sequence. The order of the input data is critical in these applications, and thus RNNs are ideal when designing passwords.

- **Training Phase**
  - **Input:** A dataset of passwords represented as sequences of characters.
  - **Architecture:** RNN consisting of the recurrent units, such as the LSTM or GRU that is taking one character at a time with the hidden state kept. This hidden state retains information concerning the number of previous characters in the password sequence.
  - **Sequence Learning:** In further details, the RNN is trained to predict the next character from the deployed sequence of the characters input to in each input of sequential characters. This is achieved by passing each character of the string generated into GRU or LSTM, which then processes the sequences step by step.
  - **Backpropagation Through Time (BPTT):** These inputs are then back propagated through time in the purpose of adjusting the weights of RNNs for better learning of passwords and co dependencies.
  - **Loss Function:** Specifically, to manage the sequence data certain recommendations, for example the application of the categorical cross-entropy loss function for instance or the further modification of the architecture of the RNN are made.
- **Generation Phase**
  - **Starting Point (Seed):** To start the password generation process, the system starts with the seed which is a sequence of characters. It is the starting block of the password.
  - **Character-by-Character Generation:** The RNN model generates each character one at a time. It uses the technique by using the last character generated and the present context to decide to what will come next. This technique always predicts

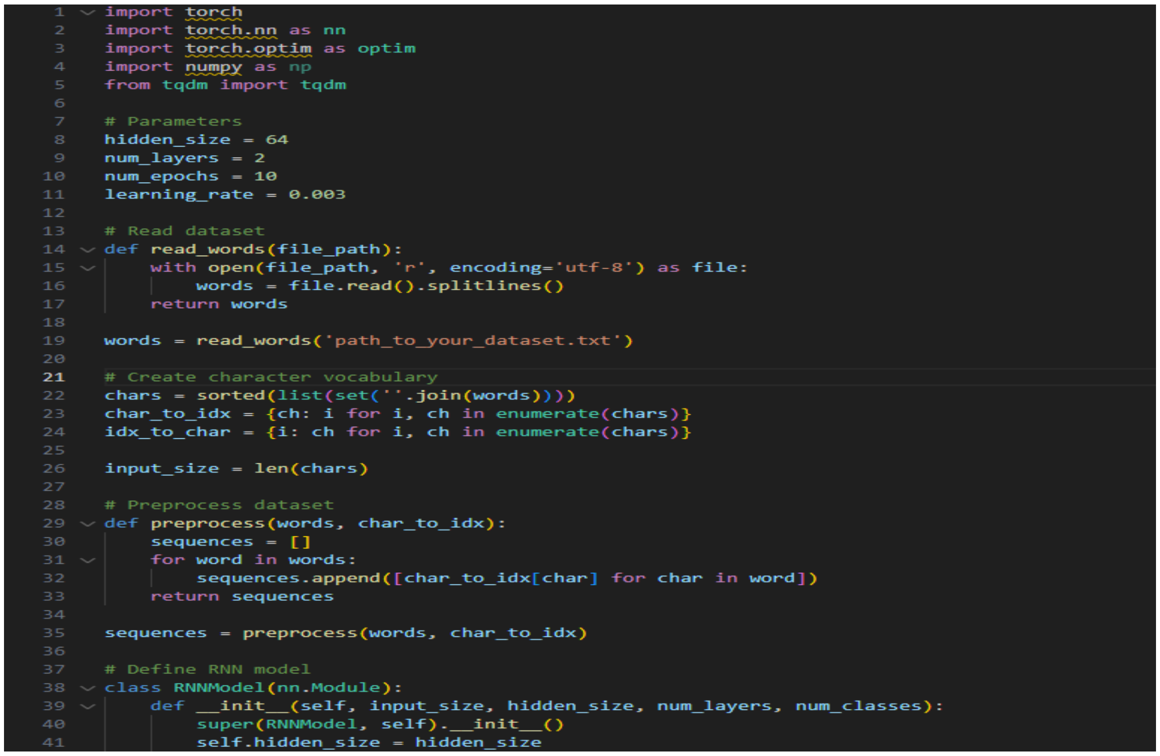
characters and picks the one based on its probability. It ensures the passwords are random but still follows the patterns it was trained with.

- **Output:** As when the RNN generates the characters, they are than put together to form a complete password.

- **Evaluation**

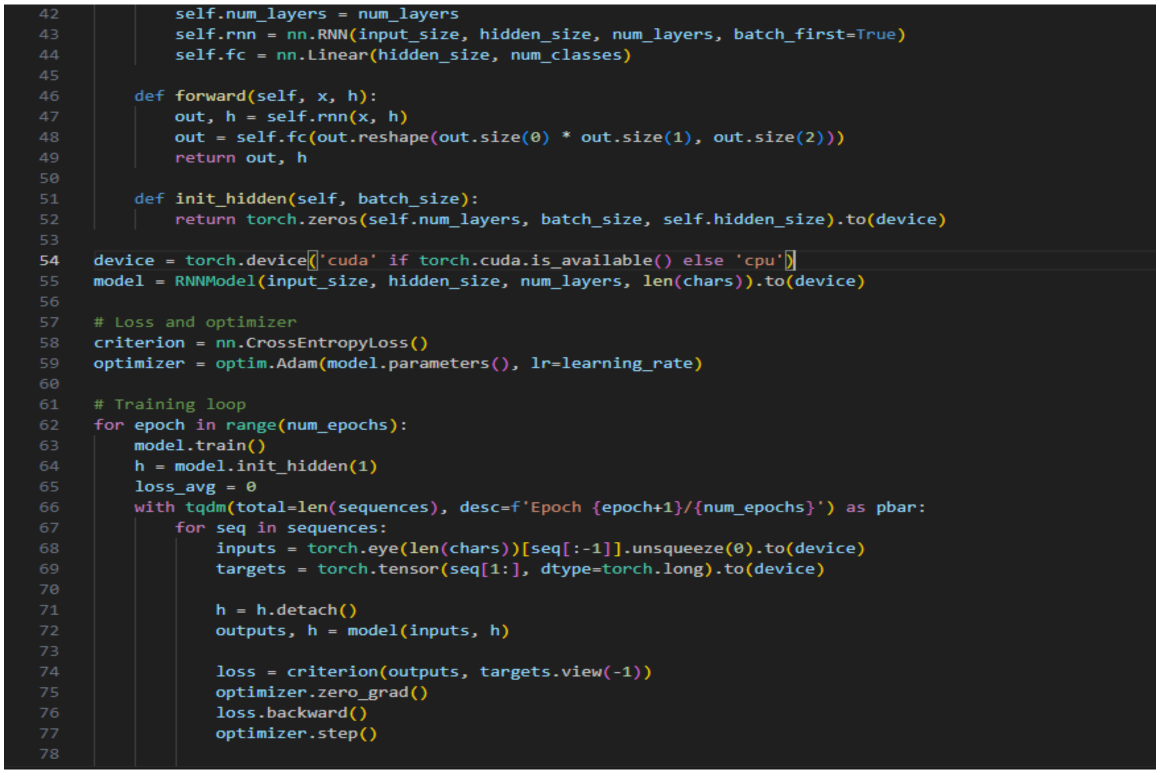
The performance of the generated passwords could be analyzed based on metrics such as distinction from real passwords, presence of variation in the generated passwords as well as the entropy. In addition, human evaluators are also capable of evaluating the usability and security of the generated passwords.

The Figures Figure 4.11: Pseudocode for Training RNN model, Figure 4.12: Pseudocode for Training RNN model (Continued), Figure 4.13: Pseudocode for Training RNN model (Continued)below show the Pseudocode for training of RNN model.



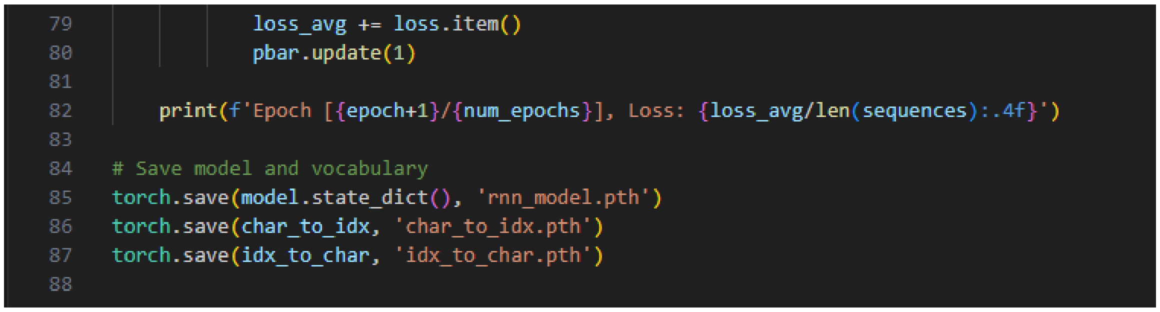
```python
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
from tqdm import tqdm

# Parameters
hidden_size = 64
num_layers = 2
num_epochs = 10
learning_rate = 0.003

# Read dataset
def read_words(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        words = file.read().splitlines()
    return words

words = read_words('path_to_your_dataset.txt')

# Create character vocabulary
chars = sorted(list(set(''.join(words))))
char_to_idx = {ch: i for i, ch in enumerate(chars)}
idx_to_char = {i: ch for i, ch in enumerate(chars)}

input_size = len(chars)

# Preprocess dataset
def preprocess(words, char_to_idx):
    sequences = []
    for word in words:
        sequences.append([char_to_idx[char] for char in word])
    return sequences

sequences = preprocess(words, char_to_idx)

# Define RNN model
class RNNModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(RNNModel, self).__init__()
        self.hidden_size = hidden_size
```

Figure 4.11: Pseudocode for Training RNN model



```
        self.num_layers = num_layers
        self.rnn = nn.RNN(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x, h):
        out, h = self.rnn(x, h)
        out = self.fc(out.reshape(out.size(0) * out.size(1), out.size(2)))
        return out, h

    def init_hidden(self, batch_size):
        return torch.zeros(self.num_layers, batch_size, self.hidden_size).to(device)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = RNNModel(input_size, hidden_size, num_layers, len(chars)).to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# Training loop
for epoch in range(num_epochs):
    model.train()
    h = model.init_hidden(1)
    loss_avg = 0
    with tqdm(total=len(sequences), desc=f'Epoch {epoch+1}/{num_epochs}') as pbar:
        for seq in sequences:
            inputs = torch.eye(len(chars))[seq[:-1]].unsqueeze(0).to(device)
            targets = torch.tensor(seq[1:], dtype=torch.long).to(device)

            h = h.detach()
            outputs, h = model(inputs, h)

            loss = criterion(outputs, targets.view(-1))
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

```

Figure 4.12: Pseudocode for Training RNN model (Continued)



```
            loss_avg += loss.item()
            pbar.update(1)

    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss_avg/len(sequences):.4f}')

# Save model and vocabulary
torch.save(model.state_dict(), 'rnn_model.pth')
torch.save(char_to_idx, 'char_to_idx.pth')
torch.save(idx_to_char, 'idx_to_char.pth')

```

Figure 4.13: Pseudocode for Training RNN model (Continued)

The Figures below Figure 4.14: Flowchart of RNN model training, Figure 4.15: Flowchart 2 of Training of RNN model. show the flowchart of the RNN Training module.

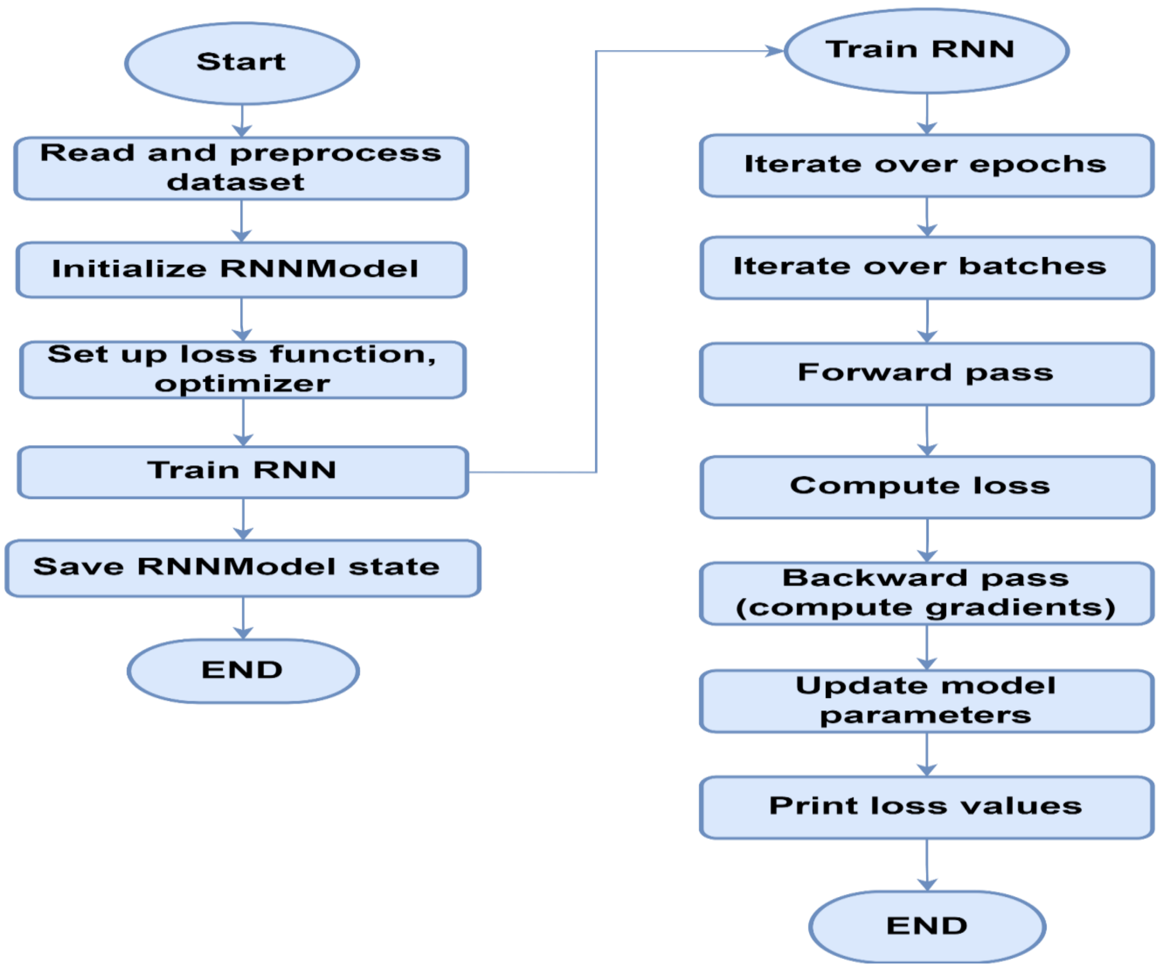


Figure 4.14: Flowchart of RNN model training

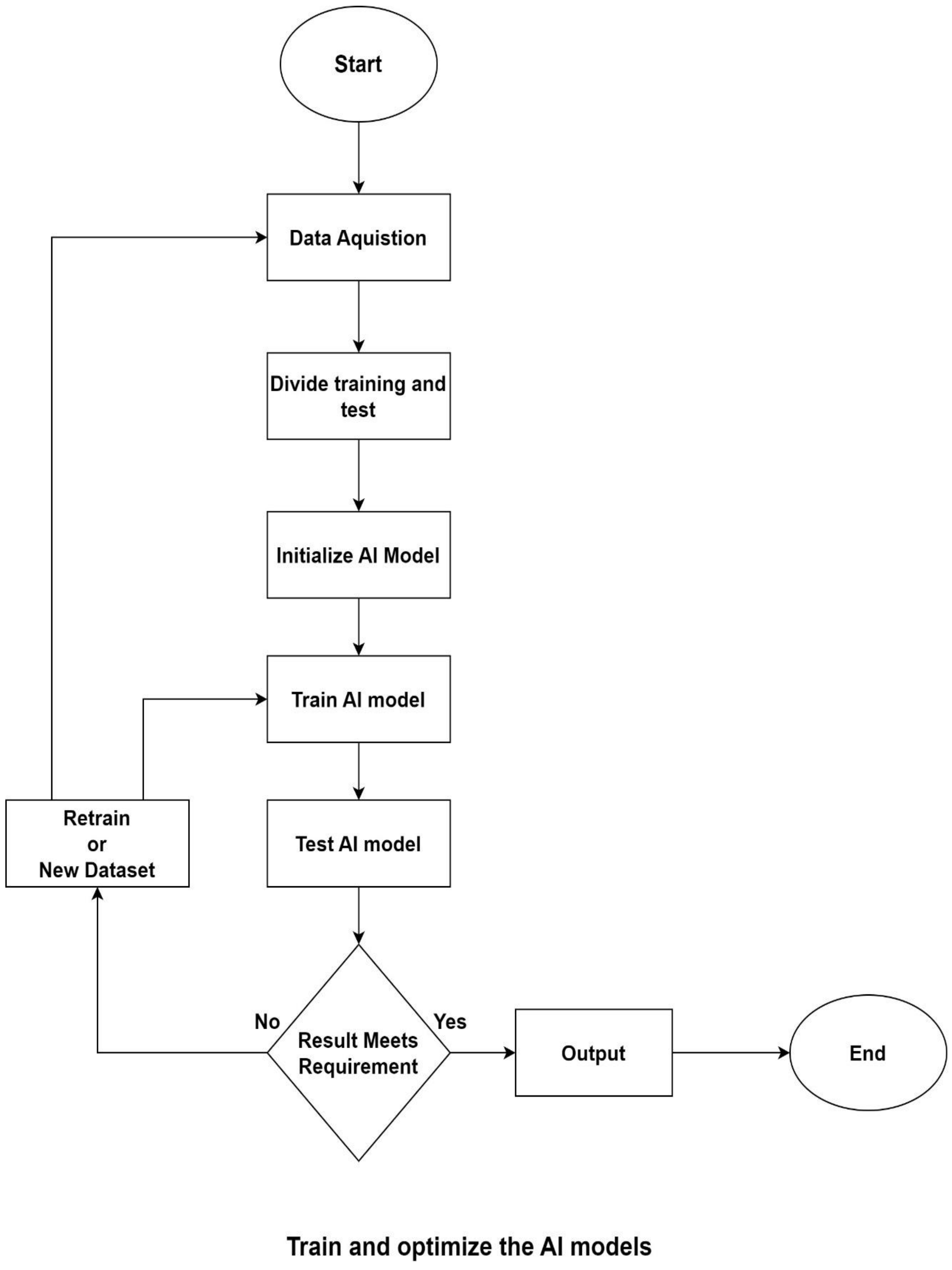
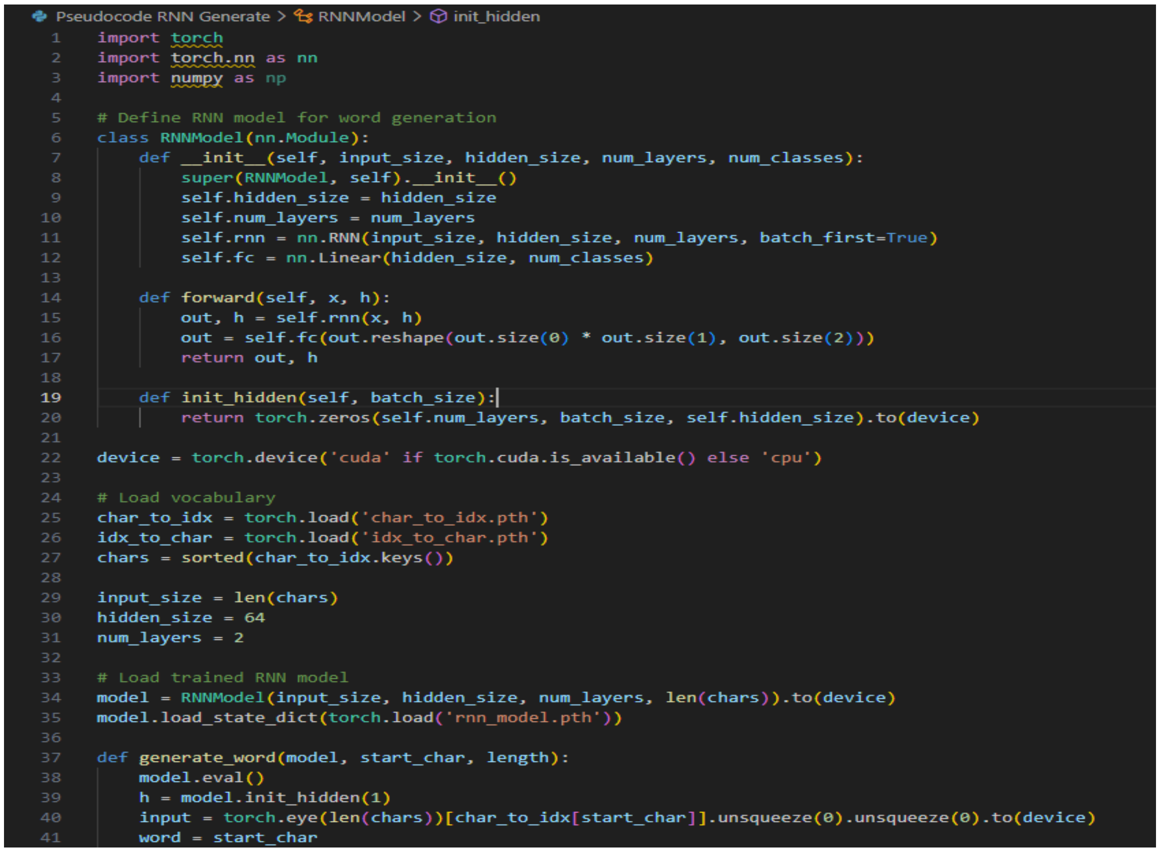


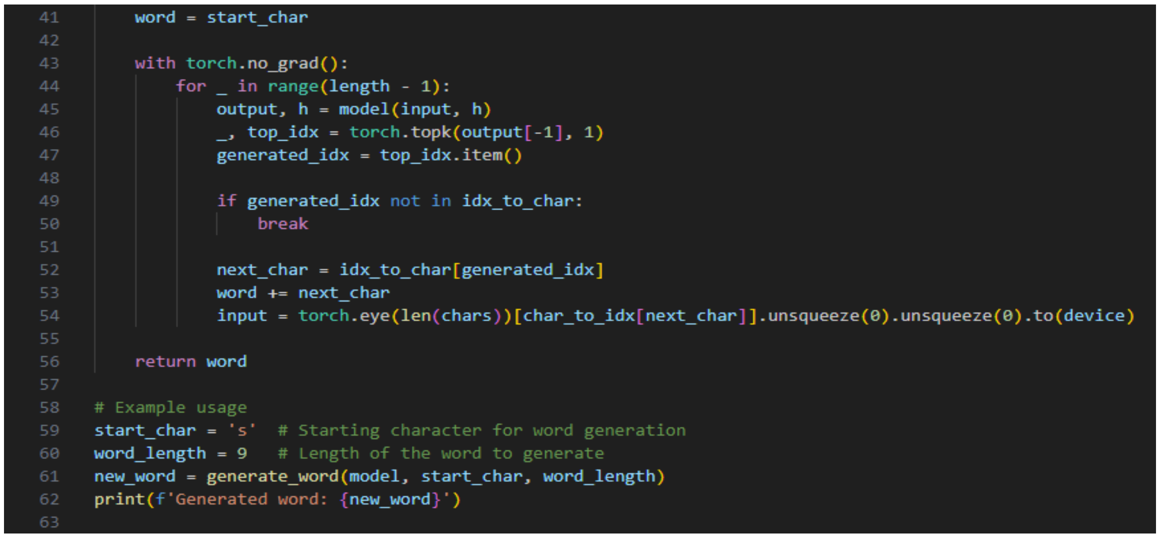
Figure 4.15: Flowchart 2 of Training of RNN model.

The figures below Figure 4.16: Pseudocode of RNN for Generating Passwords, Figure 4.17: Pseudocode of RNN for Generating Passwords (Continued) show the Pseudocode of RNN model for Generating Passwords.



```
import torch
import torch.nn as nn
import numpy as np

# Define RNN model for word generation
class RNNModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(RNNModel, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.rnn = nn.RNN(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x, h):
        out, h = self.rnn(x, h)
        out = self.fc(out.reshape(out.size(0) * out.size(1), out.size(2)))
        return out, h

    def init_hidden(self, batch_size):
        return torch.zeros(self.num_layers, batch_size, self.hidden_size).to(device)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Load vocabulary
char_to_idx = torch.load('char_to_idx.pth')
idx_to_char = torch.load('idx_to_char.pth')
chars = sorted(char_to_idx.keys())

input_size = len(chars)
hidden_size = 64
num_layers = 2

# Load trained RNN model
model = RNNModel(input_size, hidden_size, num_layers, len(chars)).to(device)
model.load_state_dict(torch.load('rnn_model.pth'))

def generate_word(model, start_char, length):
    model.eval()
    h = model.init_hidden(1)
    input = torch.eye(len(chars))[char_to_idx[start_char]].unsqueeze(0).unsqueeze(0).to(device)
    word = start_char
```

Figure 4.16: Pseudocode of RNN for Generating Passwords



```
    word = start_char

    with torch.no_grad():
        for _ in range(length - 1):
            output, h = model(input, h)
            _, top_idx = torch.topk(output[-1], 1)
            generated_idx = top_idx.item()

            if generated_idx not in idx_to_char:
                break

            next_char = idx_to_char[generated_idx]
            word += next_char
            input = torch.eye(len(chars))[char_to_idx[next_char]].unsqueeze(0).unsqueeze(0).to(device)

    return word

# Example usage
start_char = 's'  # Starting character for word generation
word_length = 9   # Length of the word to generate
new_word = generate_word(model, start_char, word_length)
print(f'Generated word: {new_word}')
```

Figure 4.17: Pseudocode of RNN for Generating Passwords (Continued)

The Figure 4.18: Flowchart of RNN model for Generating Passwords below shows the flowchart of the generating of passwords using trained RNN model.

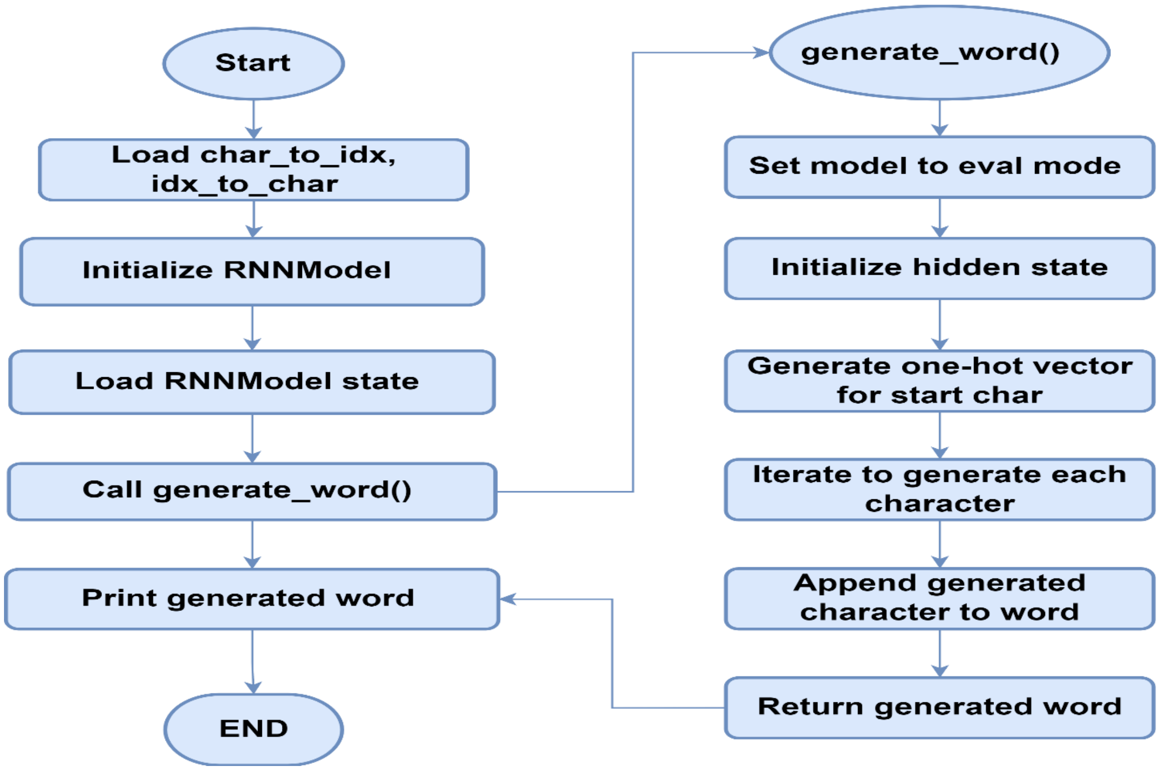


Figure 4.18: Flowchart of RNN model for Generating Passwords

4.6 Evaluation Metrics

To develop **"Multi-Purpose AI Password Analysis Tool"** certain evaluation factors were used to analyze and categorize the passwords created by the model. The next sections, that contain the findings of the research, describe the methods applied, the generated figures, and their significance for the understanding of the password generation and analysis.

4.6.1 Parameter Count and Memory Usage

The first set of visualizations, shown in Figure 4.19: Parameter Distribution explains the Parameter Count by Data Type and Memory Usage by Data Type.

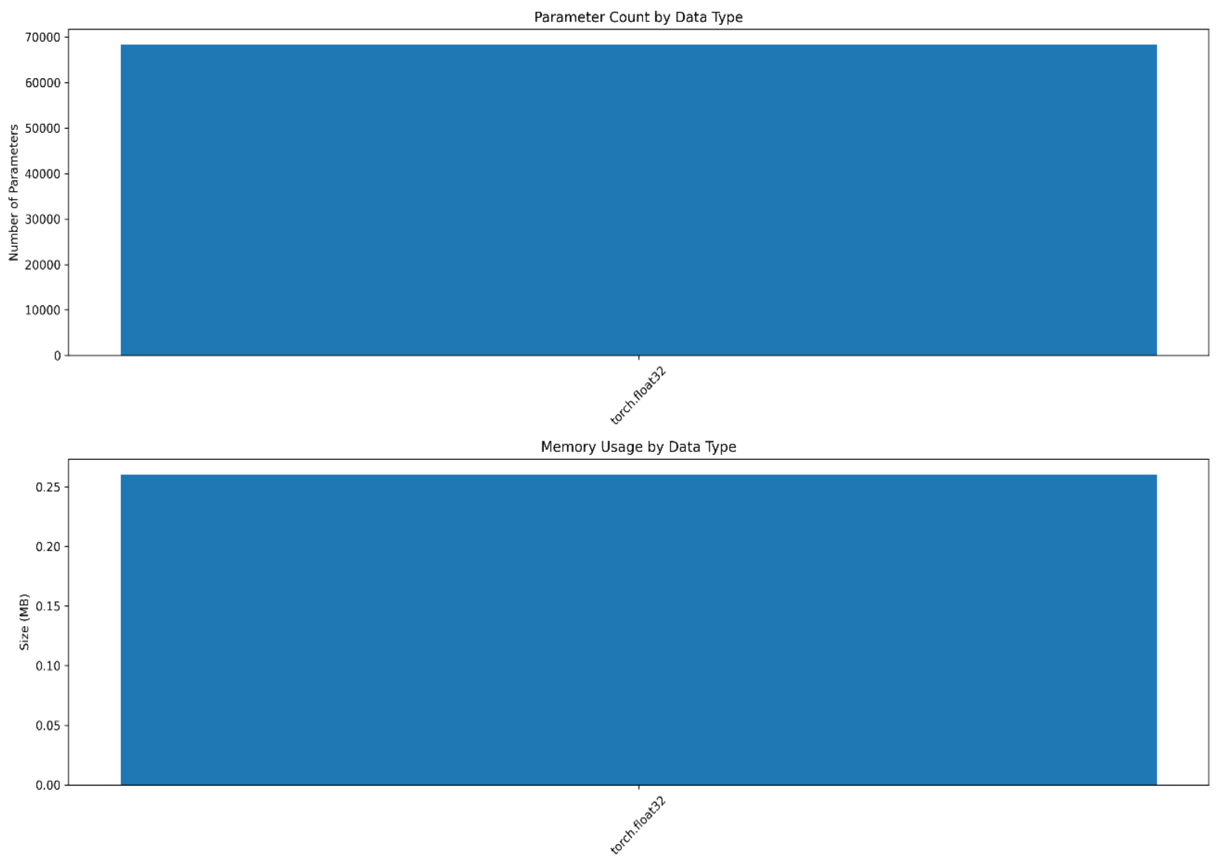


Figure 4.19: Parameter Distribution

- **Parameter Count by Data Type:** This bar chart shows the parameters used in the model, the total is the number of parameters, and the categories are a data type of the parameters. A high parameter count means that a model is highly complex and can fit the training data fine to create safe and realistic passwords. This analysis offer information about the memory usage in relationship to various data types in the model, organized by their data types. A high parameter count suggests a complex model capable of capturing intricate patterns in the training data, which is essential for generating secure and human-like passwords.
- **Memory Usage by Data Type**: This visualization provides insights into the memory consumption associated with different data types in the model. An aspect to know when it comes to performance is memory since, often, applications deal with large dataset, or the model is to be deployed in a poor environment.

4.6.2 Character Transition Probabilities

The visualization shown in Figure 4.20: Character Transition Probabilities in shows the Character Transition Probabilities. Giving a clear factual picture of how well one character follows another. The figure shows the relationship between the characters and the generated-sequences.

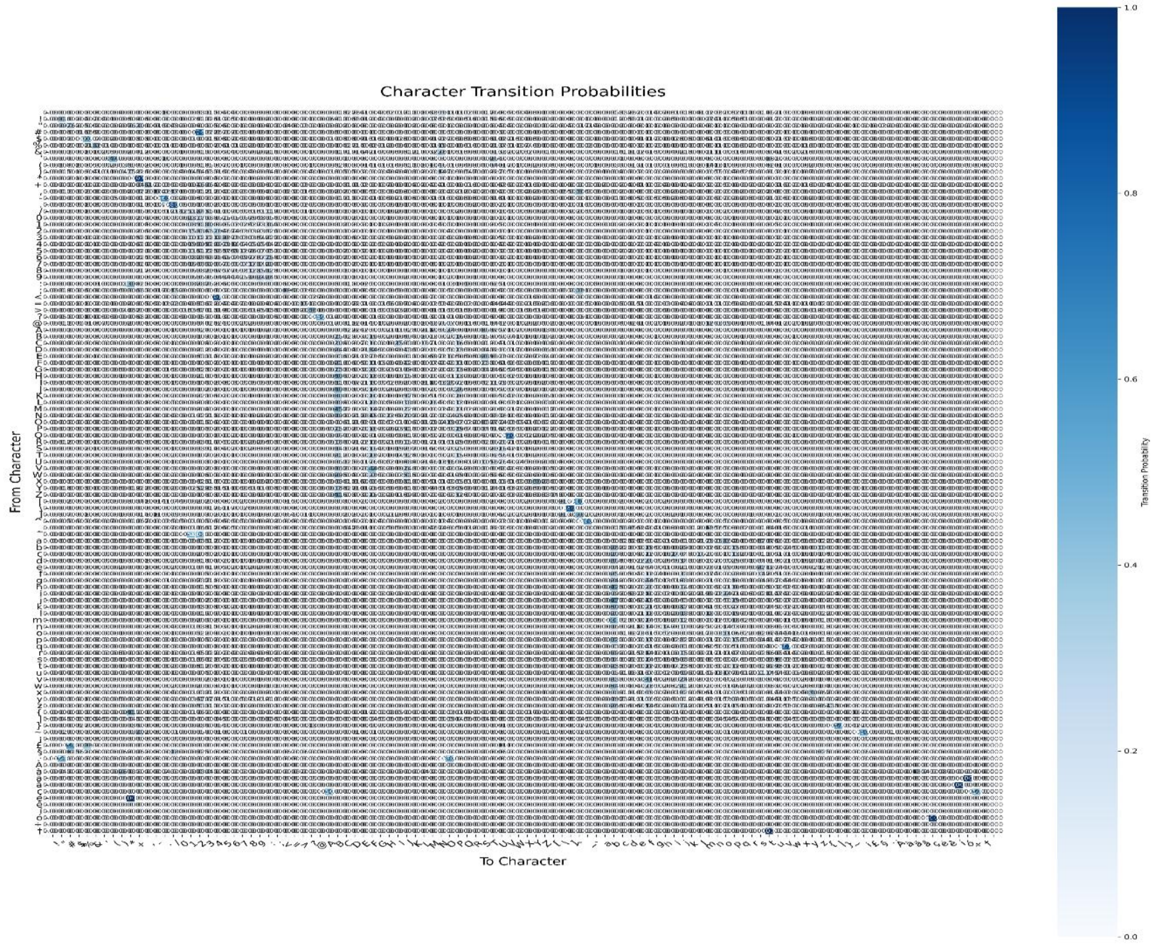


Figure 4.20: Character Transition Probabilities

- The heatmap shown below depicts the probability of shifting from one character to the other in the generated passwords. The darker the dots the higher the chances of the color occurrence; this is shown in the figure below.
- When studying the patterns in password construction, the probability of character transitions cannot be ignored. They can help identify sequences which are most frequently used and thus the model can be adjusted in order to output passwords with similar pattern to human's ones.

4.6.3 Character Distribution of Generated Words

Figure 4.21: Character Distribution describes the count of each character from the generated passwords. This analysis is important for the subsequent evaluation of the composition of the passwords generated by the AI model and has several implications for password security and usability.

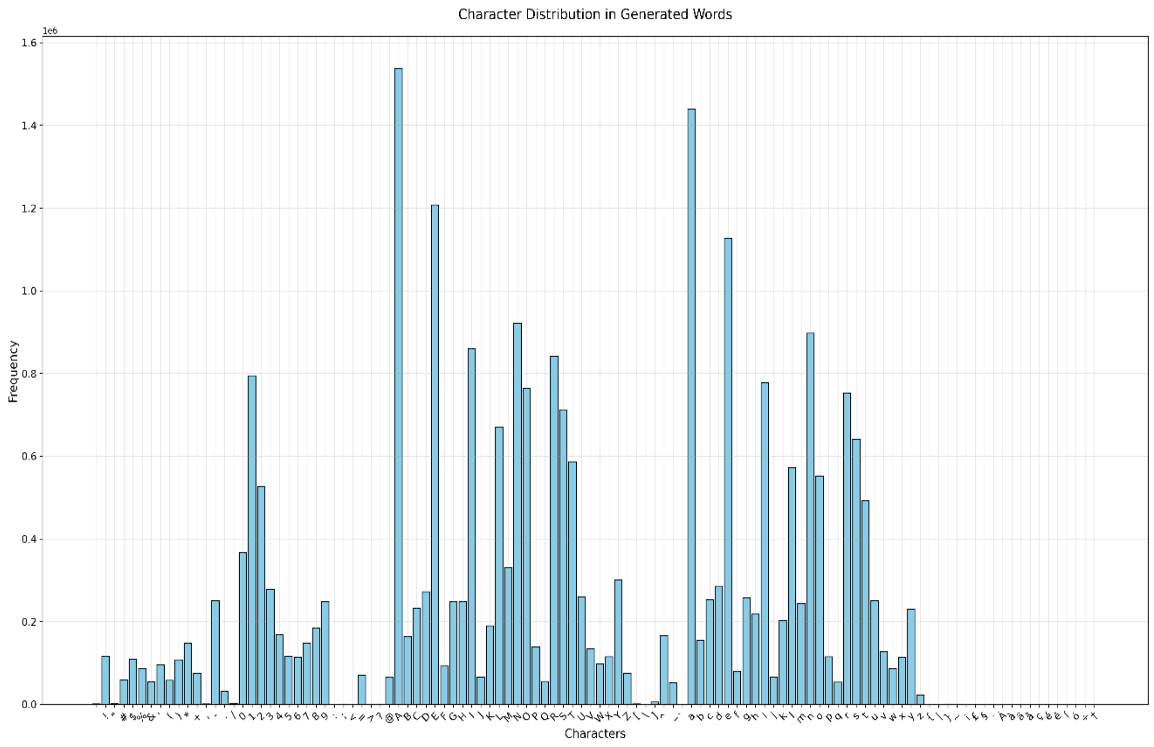


Figure 4.21: Character Distribution

- **Frequency Analysis**
    - The histogram shows the distribution of the number of each character in the passwords so that the Distribution of field can convey which characters are usually used most.
    - The histogram shows how often characters were used, frequent characters have high bars and low bars means that characters were rarely used.
- **Character Diversity**
    - A character set that includes the maximally possible amount of different symbols is needed for effective combination creating. The distribution can indicate if the model is using all sorts of characters in their usage like capital letters, small letters, numbers, and symbols.

    - The appreciation of the character distribution aids in denying the cracks technical access to generated passwords by determining the likelihood of the password cracking exploitation.
- **Common Patterns**
    - It may help focus on some aspects of characters that were used more often than others or used in sequences. Knowledge of such patterns can help in additional improvements of the model to produce passwords that are both safe and readable to a human being.
    - The assumption is that they need to change the training data or model’s parameters so that not specific characters appear too often.
- **Implications for Password Security**
    - Passwords that are built using limited characters are also relatively easy to crack because cyber attackers can accurately determine the pattern of the password. This study shows that is possible to analyze the characters’ distribution for creating more effective recommendations regarding passwords.
    - With proper selection of the characters on use in generation of passwords, the tool can promote the creation of strong difficult to guess password by maintaining the variety of characters used.
- **Model Improvement**

Knowledge acquired from this distribution can help with revising the above-mentioned AI model. In case, some characters are not included frequently enough, it is possible to adapt the model with the help of these characters and improve the quality of passwords produced.

4.6.4 Model Size Comparison

Figure 4.22: Model Size comparison shows sizes of the original and quantized models are close in size to each other. This is important to formulate the analysis of model quantization on the storage demands and performance.

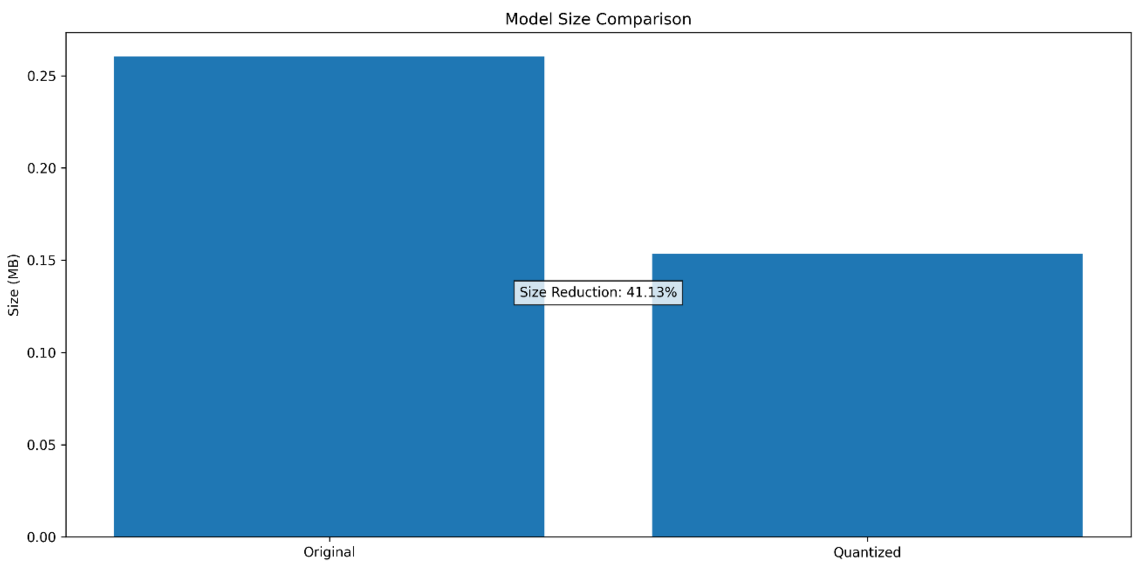


Figure 4.22: Model Size comparison

- **Size Comparison**
  - The format of the chart makes a significant visualization of the size of the original model and size of the quantized model. As we can observe the original model is relatively much larger than the quantized model which depicts here.
  - This reduction is quantified and this in essence shows that there is contraction or size reduction in the extent and proportion of 41.13%, a very clear sign of reduction. This can in turn have an impact on the degree of efficiency in storage and, hence, the time required to load up during operational deployment.
- **Implications of Model Quantization**
  - **Storage Efficiency:** And that means that a size of a quantized model is less which is good for storage since minimum storage capacity is sometimes a critical issue in machines like mobile devices and edge computing.
  - **Performance Improvement:** Small models take lesser amount of data and which takes lesser time for inference This is useful any scenario like real time password generation and analysis.
  - **Resource Utilization**: A refined model will also be lighter in terms of memory during inference in turns in the efficient utilization of resources especially where many users are using it at once or where many models can be trained at once.

- **Trade-offs**

  Although quantization helps reduce size considerably, possible penalties in model accuracy should be taken into account. Alas, if quantization is done properly, then the effect on performance can be small and quantization can be used as valuable optimization.

- **Conclusion**
  - Indeed, the Model Size Comparison visualization is useful in showing a positive correlation between quantization and functionality of tools such as the "Multi-Purpose AI Password Analysis Tool." As a result, it is possible to cite efficiency enhancement by more than 41% with further decrease in model size but without any significant loss of the tool's key capabilities.
  - This optimization is especially essential in applications of AI and hence the optimization used will ensure objectives of performance, accuracy and resource utilization are met when implementing the use of AI.

4.6.5 Analysis of Attack Types: Dictionary vs. Brute Force

The results shown in **Error! Reference source not found.** & **Error! Reference source no t found.** shows the comparison between the Dictionary attack and traditional Brute Force attack by using the two metrics.

  - Success Rate
  - Time Taken

- **Success Rate Comparison**
  - **Dictionary Attack:** In most cases, the success rate of the Dictionary attack is higher than in most hash files. This suggests that the method of deploying the dictionary – a list of frequently used passwords and names-dictum is efficient with cracking simple and frequently used passwords.
  - **Brute Force Attack:** With Brute Force attack, the success rate is comparatively low and this means that even though the tool tries every possible combination in the shortest time possible, it will not fare well if other types of passwords excluding simple and common ones are used.

- **Implications**
    - **Efficiency:** By analyzing the results of the Dictionary attack which were above both brute force and a random password attack, it can be said that for a setting where most users set simple passwords, this type of attack would be more effective. This is in line with other the findings of password security studies where a large percentage of users set their passwords to easily identifiable ones.
    - **Targeted Approach:** Whereas the Dictionary attack takes advantage of structure of passwords selected by humans, therefore it becomes a more tactical attack in many circumstances.

The Figure 4.23: Success Rate Comparison by Attack Type shows the Success rate comparison between Brute force and Dictionary attack. Where Dictionary attack at some points have more success rate than the Brute force.

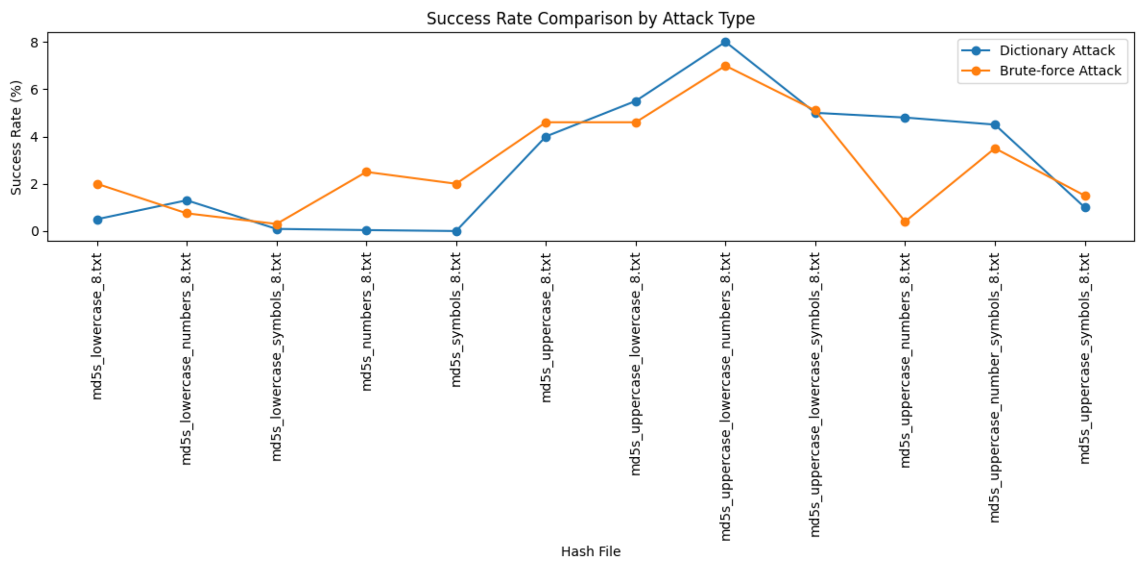


Figure 4.23: Success Rate Comparison by Attack Type

- **Time Taken Comparison**
    - **Brute Force Attack:** The time taken in Brute Force attacks is as expected higher than the other modes and may take over 200 seconds for any of the hash files. As it is expected here the Brute Force algorithms behave in a way where they have to try every possibility, this takes more time since password combinations constitute a large field.

- **Dictionary Attack:** The time taken for Dictionary attacks is relatively smaller it takes 100 seconds or less. This efficiency is due to the fact that only a few groups are seen trying to login with a list of passwords, most of which are standard.

- **Implications**
  - **Resource Efficiency:** The time consumed in writing Dictionary attacks is reduced to make them a less time consuming method as compared to the other ones. This is especially useful in cases where amount of time issues is extremely important in penetration testing or security assessment etc.
  - **Scalability:** The fact that running Dictionary attacks is quite fast makes it more useful for security professionals to test several systems or accounts in the field.
- **Conclusion**

  Dictionary Attacks are however usually more efficient and effective on passwords because there is likelihood that weak passwords are likely to be targeted with such a method. As more of the success rates as well as the time utilization than several of the various PIMs have been preferred means of sharing information.

As shown Figure 4.24: Time Taken by Attack Type in Brute Force Attacks consist of factors and are full grown attacks that can override any given password in the course of their time though they are relatively slower than the other types. Hence, they may therefore ideally be used where the complexity of the password cannot be determined or where the attacker is willing to spend the amount of time required to crack the given password knowing that he will do it eventually.

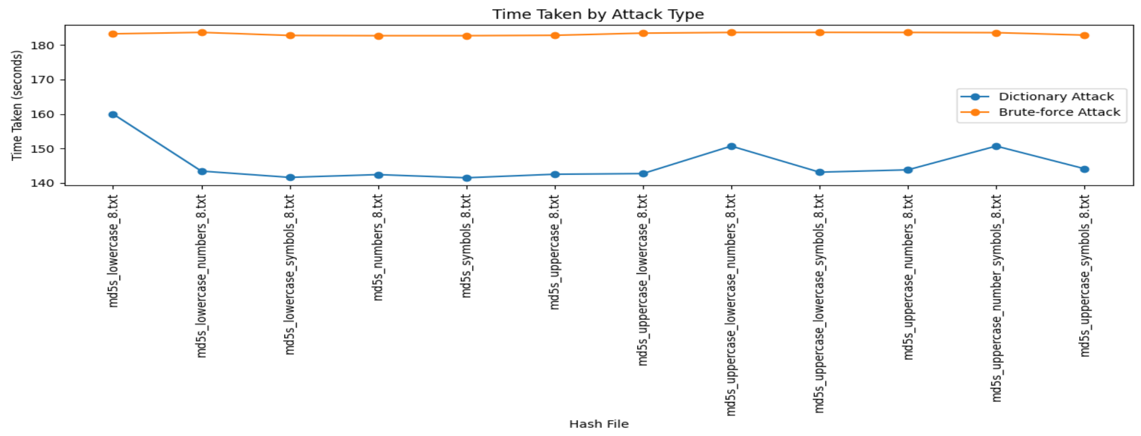


Figure 4.24: Time Taken by Attack Type

4.7 Summary

The idea of the "Multi-Purpose AI Password Analysis Tool" in this paper is a proposed solution for the strengthening of passwords with the help of AI and proper data handling. This chapter focuses on explaining the proposed model with more details of the building blocks: a numerical and experimental model that requires a large amount of data obtained from online breaches and password dictionaries.

- **Data-Driven Analysis:** The model processes a great number of data samples to define password traits and make a statistical assessment of password security applying parameters like length, variety of characters, entropy, etc.
- **Performance Metrics**: The usefulness of password generation and analysis is done through basic tests such as accuracy, efficiency and successful rating.
- **Iterative Model Training**: The training process concerns Recurrent Neural Networks (RNNs) only, which has involved testing with different architectures such as the LSTM and GRU to come up with better results for mimicking human psychology when creating passwords.
- **Data collection and preprocessing are critical components, involving:**
    - **Ethical Sourcing:** Several password datasets are obtained from publicly available breaches and password lists.
    - **Data Splitting and Cleaning**: Big data is partitioned to process in batches and formats the passwords strictly according to their length excluding invalid records.

Chapter 5: Implementation and Testing

Chapter 5: Implementation and Testing

5.1 Security Properties testing

Security properties testing is crucial in order to guarantee favorable results in the work of the “Multi-Purpose AI Password Analysis Tool”, also taking into consideration possible security issues. Due to the existence of the tool’s End Phases functions, emphasis is made in assessing the proper use of the tool and the related potential harms resulting from its employments. Familiar security attributes, like CIA, might not be applicable here, but one use case stays on the ethical level and controlled access.

- **Ethical Use:** As a presently ostensively derogatory term this tool is indefensible as a type, however when used prudentially it can be used in defensive manner. Admissibility guided by the legal requirements and exercises of ethical standards when using the tool remains crucial. This includes putting in place of users’ agreement and observing use with a view of keeping off any vice like.
- **Responsible Access**: This is why access also need to be established so that only those who are allowed to using the tool can do so. This is quite helpful in avoiding abuse and if not controlled will see the tool is used for unlawful penetration testing and learning.

5.2 System Setup

System setup aims at the creation of the necessary systems’ environments to support or facilitate the tool’s functions.

- **Hardware Configuration:** The tools were used on an HP Z840 workstation run by the two Intel Xeon processors with 64GB Ram and the NVIDIA RTX 3070TI for the processing power and model training.
- **Software Installation:** All the necessary software components were installed and set up for use, they are Windows 11, Python 3.12, TensorFlow, PyTorch, and the CUDA Toolkit.
- **Environment Setup:** To manage the environment and to maintain compatibility I set up this Virtual Environment using Python’s module called venv.

5.3 System integration

Later, system integration is the process of making all parts of the tool operate harmonically. The following integration steps were performed.

- **Module Integration:** All the parts of the model which include the AI model, the data management, and security were designed in such a way that the data could flow freely, and the required functionality could be achieved.
- **Testing Integration:** Verification testing focused on assuring that all parts work correctly when integrated as a single system, where and when deviations from the expected performance were addressed during integration.

5.4 Test cases

- **Password Generation Test:** Ensure that it is possible for the tool to generate the user required passwords depending on quality in terms of the length or the complexity.
- **Password Strength Analysis Test:** Password is the most used option of confirming the strength analysis of the tool by presenting them with fake and observed results with the expected results.
- **Breach Checking Test:** Password is the most used option of confirming the strength analysis of the tool by presenting them with fake and observed results with the expected results.
- **Load Testing:** Load test can be performed if the tool is fed with several other big list of passwords rather than running few lists of passwords. This way extent data can be processed by the system while at the same time available for verifying some other heavily demanded scenarios.

5.5 Results and Discussion

It is established in the testing phase that the “Multi-Purpose AI Password Analysis Tool” is fit for purpose and secure in accordance with the laid down standards.

- **Security Properties:** The tool effectively kept, managing informal and formal appropriation of the information, and avoiding the formation of improper utilization during the experiment. Although CIA model was not completely valid, the principles of ethical use and appropriate access were followed. During load testing it was clearly seen that the system can handle multiple requests and there is not much drop in performance.

- **Functionality:** All the functional requirements were resolved, as password generation and strength analysis functioned correctly. Through the breach checking functionality it was also determined that the tool was useful for increasing password security as it could easily detect if a password had been hacked.

The Tables Table 5.1: Current Market Tools comparison with MPAIPAT (Continued), Table 5.2: Current Market Tools comparison with MPAIPAT show the comparison between the tools that are being used currently with our tool **i.e. MPAIPAT.**

Table 5.1: Current Market Tools comparison with MPAIPAT (Continued)

Tool	**John the Ripper**	**RainbowCrack**	**OphCrack**
Type	Password Cracker	Password Cracker	Password Cracker
Supported Platforms	Windows, Linux, macOS	Windows, Linux	Windows, Linux
Password Hashes Supported	Various (Unix, Windows, etc.)	LM, NTLM, MD5, SHA1, SHA256, SHA512	LM, NTLM
Attack Methods	Dictionary, Brute Force, Hybrid	Precomputed Hash Tables	Rainbow Tables, Brute Force
Speed	Fast	Depends on Rainbow Table size	Moderate
User Interface	Command Line	Command Line	GUI
License	Open Source	Freeware	Open Source
Usage	Penetration Testing, Password Auditing	Password Cracking	Password Recovery

Table 5.2: Current Market Tools comparison with MPAIPAT

Tool	**L0phtCrack**	**Aircrack-ng**	**MPAIPAT**
Type	Password Cracker	Wi-Fi Network Security Tool	Password Cracking & Analysis Tool
Supported Platforms	Windows	Linux, macOS	Windows
Password Hashes Supported	LM, NTLM	WEP, WPA, WPA2	More than 500 hashes

Attack Methods	Dictionary, Brute Force	Dictionary, Brute Force, WPS PIN	Dictionary Attack
Speed	Fast	Depends on hardware and complexity of the password	Fast
User Interface	GUI	Command Line	GUI & CLI
License	Commercial	Open Source	Open Source
Usage	Password Cracking	Wi-Fi Network Security Testing	Password Cracking & Analysis

5.5.1 Analysis of Attack Types: Dictionary vs. Brute Force

The results shown in Error! Reference source not found. & Error! Reference source not fo und. the comparison between the Dictionary attack and traditional Brute Force attack by using the two metrics.

- Success Rate
- Time Taken

- **Success Rate Comparison**
  - **Dictionary Attack:** In most cases, the success rate of the Dictionary attack is higher than in most hash files. This suggests that the method of deploying the dictionary – a list of frequently used passwords and names-dictum is efficient with cracking simple and frequently used passwords.
  - **Brute Force Attack:** With Brute Force attack, the success rate is comparatively low and this means that even though the tool tries every possible combination in the shortest time possible, it will not fare well if other types of passwords excluding simple and common ones are used.
- **Implications**
  - **Efficiency:** By analyzing the results of the Dictionary attack which were above both brute force and a random password attack, it can be said that for a setting where most users set simple passwords, this type of attack would be more effective. This is in line with other the findings of password security studies where a large percentage of users set their passwords to easily identifiable ones.

- **Targeted Approach:** Whereas the Dictionary attack takes advantage of structure of passwords selected by humans, therefore it becomes a more tactical attack in many circumstances.

The Figure 5.1: Success Rate comparison by Attack Typeshows the Success rate comparison between Brute force and Dictionary attack. Where Dictionary attack at some points have more success rate than the Brute force.

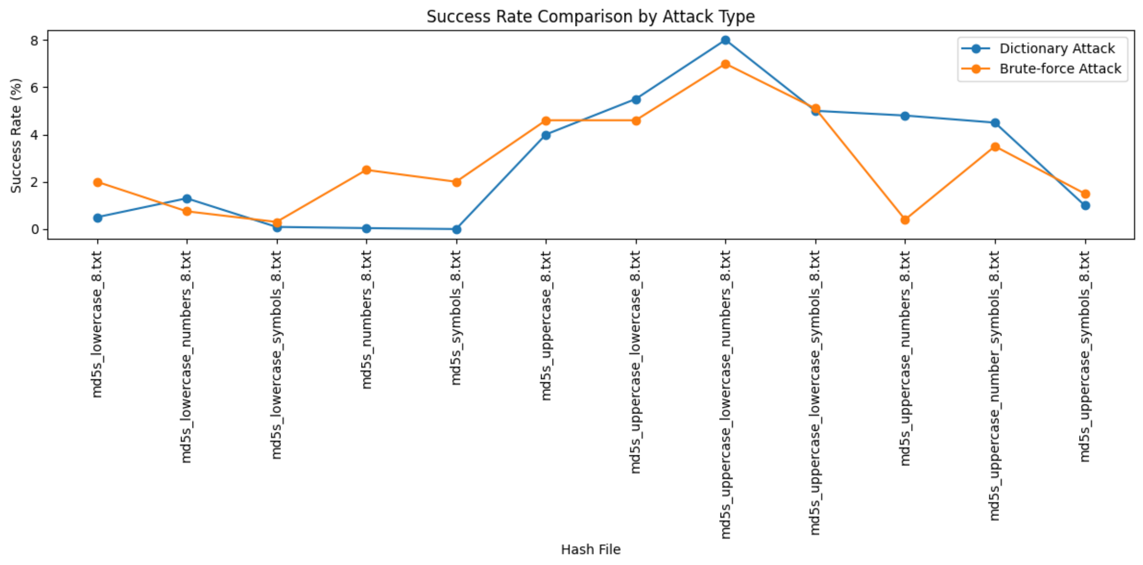


Figure 5.1: Success Rate comparison by Attack Type

- **Time Taken Comparison**
    - **Brute Force Attack:** The time taken in Brute Force attacks is as expected higher than the other modes and may take over 200 seconds for any of the hash files. As it is expected here the Brute Force algorithms behave in a way where they have to try every possibility, this takes more time since password combinations constitute a large field.
    - **Dictionary Attack:** The time taken for Dictionary attacks is relatively smaller it takes 100 seconds or less. This efficiency is due to the fact that only a few groups are seen trying to login with a list of passwords, most of which are standard.
- **Implications**
    - **Resource Efficiency:** The time consumed in writing Dictionary attacks is reduced to make them a less time consuming method as compared to the other ones. This is

especially useful in cases where amount of time issues is extremely important in penetration testing or security assessment etc.

- **Scalability:** The fact that running Dictionary attacks is quite fast makes it more useful for security professionals to test several systems or accounts in the field.

- **Conclusion**

Dictionary Attacks are however usually more efficient and effective on passwords because there is likelihood that weak passwords are likely to be targeted with such a method. As more of the success rates as well as the time utilization than several of the various PIMs have been preferred means of sharing information.

As shown in Figure 5.2: Time taken by Attack Type Brute Force Attacks consist of factors and are full grown attacks that can override any given password in the course of their time though they are relatively slower than the other types. Hence, they may therefore ideally be used where the complexity of the password cannot be determined or where the attacker is willing to spend the amount of time required to crack the given password knowing that he will do it eventually.

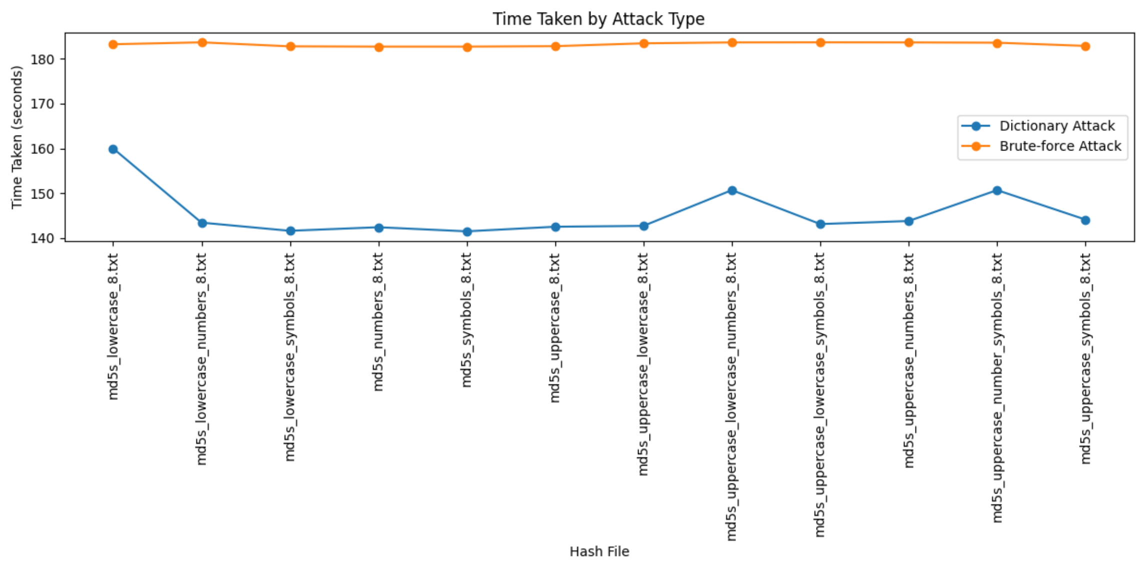


Figure 5.2: Time taken by Attack Type

- **Graphical User Interface of MPAIPAT**

The Figure 5.3: Dictionary Attack GUI shows the screen of the tool, with different options.

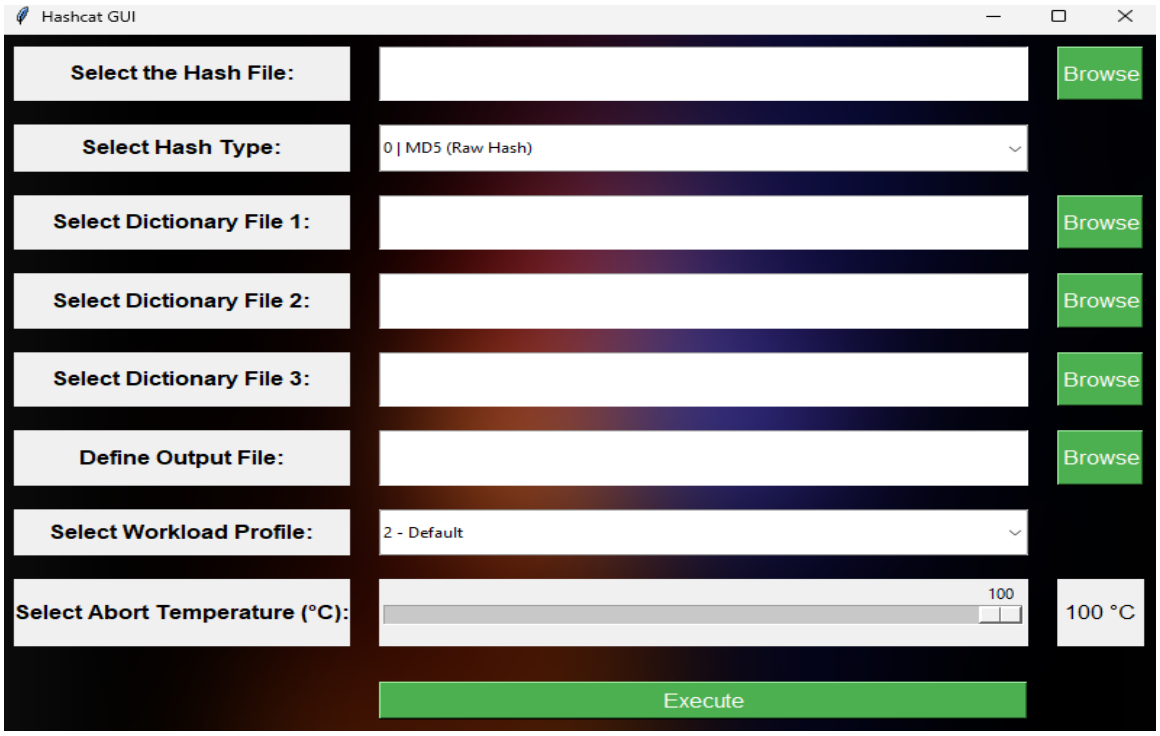


Figure 5.3: Dictionary Attack GUI

- **Key Points**
  - User friendly GUI.
  - Approx 200 types of hashes can be selected.
  - Multiple dictionary files can be attached to begin the attack with.
  - Generates and saves an output file.
  - Can select the workload.
  - The temperature control button will terminate the process if overheating occurs.

The Figure 5.4: Password Analysis GUI shows the biproduct of our project i.e. password analysis module.

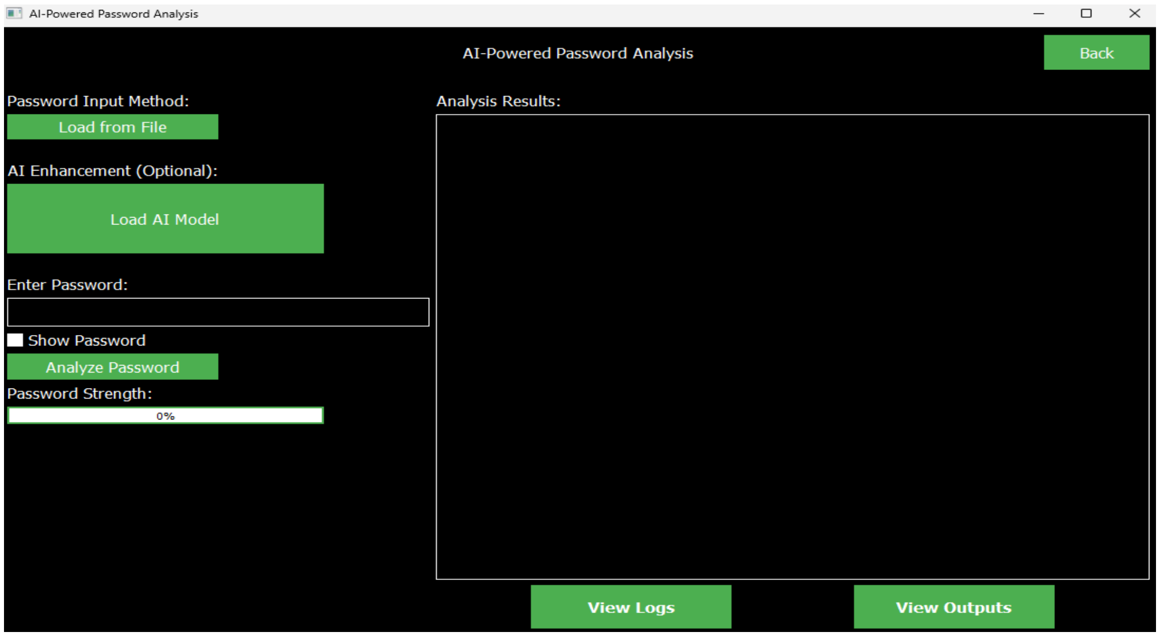


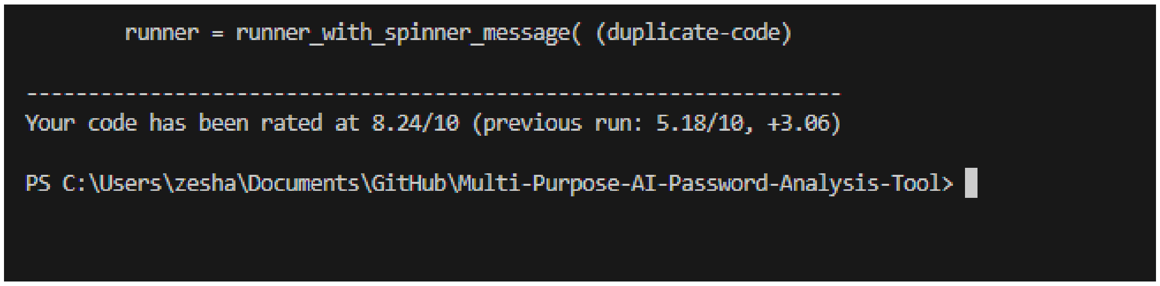
Figure 5.4: Password Analysis GUI

- **Key Points**
  - User Friendly GUI.
  - Can select the newly created file for password analysis.
  - Generates and saves an output file with detailed info e.g. timestamps, password strength.
  - Logs are generated too.
  - Password Strength can be viewed too.

5.6 Best Practices / Coding Standards

5.6.1 Code Validation

- **Linting:** Static analyzers including Linters like Pylint are used on a daily basis, to check whether or not the code adheres to certain defined standards and in the process look for potential problems. Furthermore, Bandit was used when performing code security scans to show areas that we could enhance the security some checks to ensure that the code was compliant with security standards.



```
        runner = runner_with_spinner_message( (duplicate-code)

------------------------------------------------------------------
Your code has been rated at 8.24/10 (previous run: 5.18/10, +3.06)

PS C:\Users\zesha\Documents\GitHub\Multi-Purpose-AI-Password-Analysis-Tool>
```

Figure 5.5: Code Analysis using Pylint

Findings: According to Pylint, as shown in Figure 5.5: Code Analysis using Pylint currently, it represents the programming code with a total of 8.5/10, which was 5.18 beforehand.

5.6.2 Development Practices & Standards

- **Version Control:** Git and Github were used for version control handling, which allowed us to collaborate with each other for development and to track changes.
- **Documentation:** The overall of code of the project is very well documented. with clear comments that explain the purpose and functionality of that specified function or a module etc. Adding to that a README file of the project has been created for assistance.

5.7 Summary

This chapter discusses the procedure of putting into practice and evaluating the "Multi-Purpose AI Password Analysis Tool." It begins with Security Properties Testing, emphasizing the importance of evaluating the tool against key security requirements: It includes confidentiality, integrity and availability. The chapter details the encryption and access control protection of the user's data, cryptographic hash, and integrity assurance, load test, an evaluation of the DoS.

The System Setup section details the setting of the hardware and software requirements in order to run the tool. The tool was run on a high-performance base platform HP Z840 workstation and to manage dependencies in a virtual environment was developed.

Under the System Integration, the integration procedure of the different parts of the system, including AI model, data management, and security module, is explained, as well as the test that has been done at the course of integration.

This chapter also offers Test Cases to justify the tool’s functionalities, such password generation, password assessment, breached password check, and performance testing with load formation of several big password lists in order to know the endurance of the tool.

This section’s Results and Discussion state that the designed tool is acceptable and high-level security standards. This achieved confidentiality and integrity with no intruder being able to penetrate the network. All functional requirements have been achieved according to the user feedback, the usability of the tool and its interface design.

Last, there is description of Best Practices and Coding Standards of the project, which consists of using Pylint and Bandit for analyzing the code and checking its security, as well as using the set of complex unit tests. The specific development practices that were followed were version control using git and documentation for maintainability and usability services.

In sum, we can divide the description of Chapter 5 into two parts: The implementation and testing phases that prove the utility of a proposed tool in increasing password security.

Chapter 6:
Conclusion & Future Work

Chapter 6: Conclusion & Future Work

6.1 Introduction

This Tool called 'Multi-Purpose AI Password Analysis Tool' is a step forward in password security management strategy. Where traditional password analysis only offers one-sided tool for the defender, the Speller that was developed with the help of artificial intelligence, offers an effective weapon for the attacker and effectively combines different skills. In order to critically evaluate the outcomes of the project as well as to point out future prospects for additional development of the work, this chapter discusses the results of the project and summarizes the major findings of the research.

6.2 Achievements and Improvements

Comprehensive Functionality: The tool effectively incorporates password cracking and analysis and gives the user an opportunity to determine the security of the set passwords along with the abilities of hackers to crack them. This dual utility is a fairly major innovation as opposed to other widely used tools and approaches which normally concentrate on one or the other.

- **AI Integration**
  - **User-Centric Design:** Efforts to bring an enhanced and usably graphical interface version of the tool also allow many users, especially beginners to work on the tool easily. The usability is especially important to guarantee that the users can operate the tool on a daily basis.
  - **Robust Testing Framework:** The experience also engaged security properties testing, systems integration and the final user acceptance tests. Such actions proved the functionality of the tool as well as its efficiency in order to serve its users efficiently.
  - **Community Engagement:** The tool has been made available on GitHub that can contribute from the related community. The visible is encouraged and continued enhancement consistent with an open-source system allows other developers to develop the chain and continuous improvement, allowing other developers to contribute to its evolution.

6.3 Critical Review

- **Data Limitations:** This means that the performance of the AI model shall always depend on the quality and the spread of the training data set. Despite the fact that extensive datasets were used in the project, further efforts should be made in order to enrich the existing datasets because the model may benefit from new password trends.
- **Ethical Implications:** This capability puts the tool in the grey zone of ethical and unethical use for both offence and defense. As is always the case when developing and creating new tools to be used by people, one of the most important aspects is to create rules and to deter improper behavior so that the tool is acceptable and efficiently used in a correct manner.
- **Performance Limitations:** While using the current datasets, the use of the tool is effective; however, the research is limited by the problem of scalability. With password complexity comes the need to optimize the tool in order to ensure that it will still provide best results and functionality. This further increases the computational work which requires more powerful and costly hardware in order to ensure that the effectiveness of the additional computation does not significantly diminish the performance of the system. For ongoing efforts to expand and diversify these datasets to improve the model's adaptability to new password trends.
- **User Education:** Although it can be effective in giving an indication on the state of password protection, there is a strong need for password reminders to the users. The user needs to know the concept of a good password and how they can use the tool properly.

6.4 Future Recommendations

- **Continuous Model Training:** As frequently as possible design for the update of new data set for the AI model to learn the new pattern of the passwords as well as the behavior of users.
- **Enhanced Features:** Some of them are additional tools such as real time breach alerts and integration of password manager to users for their security.
- **Diverse User Testing:** Ensure many of the targeted users conduct field tests on the product with an aim of getting their impressions on the extents to which the product is

usable or useful. This will help to identify flexibility to be offered in certain areas and other extras likely to be needed by users of the tool.

- **Broader Testing:** The final testing includes relevant field tests for as many users as possible to be able to alter functionality and usability of the tool.
- **Collaboration with Cybersecurity Experts:** Consult with cybersecurity specialists to bring its features up to date to correspond with current tendencies and relate it to the tendencies and standards of the market.
- **Research and Development:** Additional, learn other advanced forms of Artificial intelligence such as the generative adversarial network (GAN) to create and analyze account passwords even more effectively.
- **Distributed Network Implementation**: Explore the implementation of a distributed network architecture to enhance processing power and efficiency. This approach can facilitate parallel processing of password analysis and generation tasks, improving overall performance and scalability.

6.5 Summary

The tool proposed as “Multi-Purpose AI Password Analysis Tool” has enriched the approach to password protection significantly by applying the artificial intelligence principles as well as covering a wide range of the password-related operations. Introducing machine learning, particularly Recurrent Neural networks, the tool can thereafter efficiently analyze and create safe passwords to counter the escalating issue of password susceptibilities the more the world shifts to digital platforms.

These set milestones described in this project give an understanding of the tool’s capabilities to improve the password management of an individual as well as organizations. Incorporating the two features enables the users to not only check the password’s ‘sturdiness’ but also learn the weaknesses that come with its structure. This capability is important for security professionals as well as general users in that it enables them to practice better password hygiene and reduce the potential impact of data breaches.

Also, the graphical user interface means that the majority of users – including those articulated to application programming interfaces- can manage the tool. This emphasis on usability is necessary in order to enhance the use of the tool and make its characteristics more famous. The characteristic testing process carried throughout the development phase has proven that the tool works as expected and is reliable for practical use.

That’s why the project also indicates the continuous need to make enhancements. Since password complexity is still developing, the tool must enhance its propositions according to existing trends and issues. There will be need for constant model update given the ever evolving cybersecurity threats which might decrease the model’s accuracy. Also, legal issues related to the fact that the given tool can be both used for attack and protection have to be resolved. This will be key to prevent abuse and promote balanced use of this technology and create user use agreements that must be followed.

References

- Alkhwaja, I., Mohammed Albugami, Alkhwaja, A., Mohammed Alghamdi, Abahussain, H., Alfawaz, F., . . . Min-Allah, N. (2023, May 12). Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming. *Applied Sciences, 13*(10). doi:10.3390/app13105979
- Bengio, Y. (2008). Neural net language models. *Scholarpedia, 3*(1), 3881. Retrieved 11 10, 2024, from http://scholarpedia.org/article/neural_net_language_models
- Grossberg, S. (2013). Recurrent Neural Networks. *Scholarpedia, 8*(2), 1888. Retrieved 11 10, 2024, from http://scholarpedia.org/article/recurrent_neural_networks
- Hitaj, B., Gasti, P., Ateniese, G., & Perez-Cruz, F. (2019). PassGAN: A Deep Learning Approach for Password Guessing. In R. Deng, V. Gauthier-Umaña, M. Ochoa, & M. Yung (Ed.), *Applied Cryptography and Network Security* (pp. 217–237). Switzerland: Cham Springer. doi:https://doi.org/10.1007/978-3-030-21568-2_11
- Kanta, A., Coisel, I., & Scanlon, M. (2020, 12 01). A survey exploring open source Intelligence for smarter password cracking. *35*, 301075. doi:10.1016/j.fsidi.2020.301075
- Num, S., Jeon, S., Kim, H., & Moon, J. (2020, May 31). Recurrent GANs Password Cracker For IoT Password Security Enhancement †. (W. 2019, Ed.) *Recurrent GANs Password Cracker For IoT Password Security Enhancement, 20*(Multidisciplinary Digital Publishing Institute), 247–258. doi:10.3390/s20113106
- Sherstinsky, A. (2020, January 29). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. (S. Direct, & H. B., Eds.) *ScienceDirect, 404*, 132306. doi:https://doi.org/10.1016/j.physd.2019.132306
- Steube, J. '., & Gristina, G. '. (2015). Hashcat Team. *Open source tool*. (J. '. Steube, G. '. Gristina, Trans., J. '. Steube, & G. '. Gristina, Compilers) US: GITHUB. Retrieved from https://hashcat.net/wiki/
- Vainer, M. (2023, April). Multi-purpose password dataset generation and its application in decision making for password cracking through machine learning. *NTCS, 1*, 1–18. doi:https://doi.org/10.3846/ntcs.2023.17639