Multi-Purpose AI Password Analysis Tool

By:

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

Supervised by:
Dr. Muhammad Mansoor Alam

Faculty of Computing
Riphah International University, Islamabad

Spring 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

Bachelor of Science in Cyber Security

Faculty of Computing
Riphah International University, Islamabad

Date: 12/11/2024

Final Approval

This is to certify that we have read the report submitted by ***Osama Khalid 27971, Sardar Muhammad Zeeshan Khan 27969, 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 Bachelor of Science in Cyber Security (BSCY).

Committee:

1 ______________________________

Dr. Muhammad Mansoor Alam
(Supervisor)

2 ______________________________

Dr. Jawaid Iqbal
(In-charge of Cyber Security)

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 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. Additionally, we dedicate this work to our colleagues and the academic community, whose guidance and knowledge have been instrumental in shaping our professional growth. Their commitment to excellence has inspired us to push the boundaries of our potential. Thank you for being our pillars of support and for always believing in 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 Supervisor:** We are particularly grateful to our project supervisor, **Dr. Mansoor Alam**, for their invaluable expertise, insightful feedback, and unwavering encouragement throughout the entire project development process. His dedication and support played a pivotal role in shaping the direction and success of our work.
- **Faculty Advisors:** We extend our sincere thanks to our faculty advisors, **Dr. Muhammad Mansoor Alam**, whose passion for AI inspired us to pursue this project, and **Dr. Jawaid Iqbal**, whose knowledge and guidance were invaluable throughout the research phase.
- **Technical Support Staff:** We appreciate the assistance provided by **Mr. Haseeb Ahmed**, **Mr. Ihtesham Ullah**, and **Mr. Tajammul Hussain**. Their technical expertise and willingness to help were crucial in overcoming technical challenges.
- **Project Convener:** We acknowledge the leadership of **Mr. Muhammad Ahmad Nawaz** as the project convener, whose guidance ensured the project's smooth execution within the academic framework.
- **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 an increasingly digitized world, the security of personal and organizational data is paramount. This project presents a comprehensive solution in the form of a multi-purpose AI-driven password analysis tool. By harnessing the power of artificial intelligence, this tool offers dual functionality, serving both offensive and defensive purposes.

On the offensive front, it employs advanced algorithms to analyze and crack passwords, providing insights into their vulnerabilities and potential points of exploitation. This capability enables security professionals to understand the weaknesses inherent in various password configurations, thereby facilitating the development of more robust defense strategies.

Simultaneously, the tool acts as a guardian on the defensive front, evaluating the strength of passwords and identifying potential breaches through sophisticated pattern recognition and analysis. By proactively identifying compromised credentials, it empowers users to take preemptive action, mitigating the risks associated with data breaches and unauthorized access.

Through its intuitive interface and customizable features, this AI-powered tool becomes an indispensable asset for individuals and organizations seeking to fortify their digital security posture. By enabling users to test the resilience of their passwords and stay ahead of emerging threats, it serves as a proactive safeguard in an ever-evolving cybersecurity landscape.

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

1.2 Opportunities and Stakeholders

1.3 Motivations and Challenges

1.4 Goals and Objectives

1.5 Solution Overview

1.6 Report Outline

Chapter 1: Introduction

Project Title: Multi-Purpose AI Password Analysis Tool

1.1 Introduction

In today's world, where everything is becoming digital and online threats are on the rise, having strong password security is crucial. That's where the AI Multipurpose Password Analysis Tool comes in. It's a game-changing software that's here to revolutionize how we manage passwords. By combining cutting-edge artificial intelligence with traditional password analysis tools, this tool offers a complete solution for keeping our digital accounts safe. Whether it's cracking passwords or assessing password's strength, this tool covers multiple aspects of password security. And it doesn't stop there - it seamlessly works with popular tools like Hashcat, making it even more powerful. So, as we step into the future of password management, the AI Multipurpose Password Analysis Tool is leading the way, providing both analysis and cracking abilities to protect our sensitive information from the ever-evolving threats online.

1.2 Opportunity & Stakeholders

The key opportunities and primary stakeholders associated with this project are below.

Opportunities

- **Enhanced Security Assessments:** Businesses and individuals can leverage the tool for penetration testing and vulnerability assessments, identifying potential weaknesses in their password security policies and practices.
- **AI-driven Offensive and Defensive Capabilities:** Utilizing AI for both offensive (password cracking) and defensive (password strength assessment) purposes provides a comprehensive approach to password security management.

Stakeholders

Businesses: Companies aim to improve their overall cybersecurity posture by identifying and mitigating password-related vulnerabilities.

- **Security Professionals:** Penetration testers and security consultants who can use the tool for vulnerability assessments and ethical hacking activities.

- **Law Enforcement:** Agencies might leverage the tool for lawful investigations adhering to court orders and legal guidelines.

1.3 Problem Statement

With increasing password complexity requirements, brute-force attacks become significantly slower and less efficient.

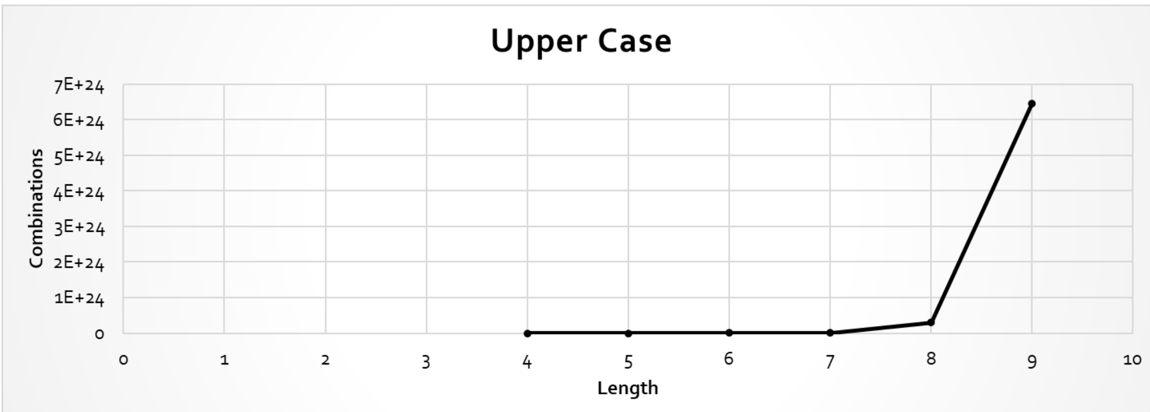


Figure 1: Upper Case

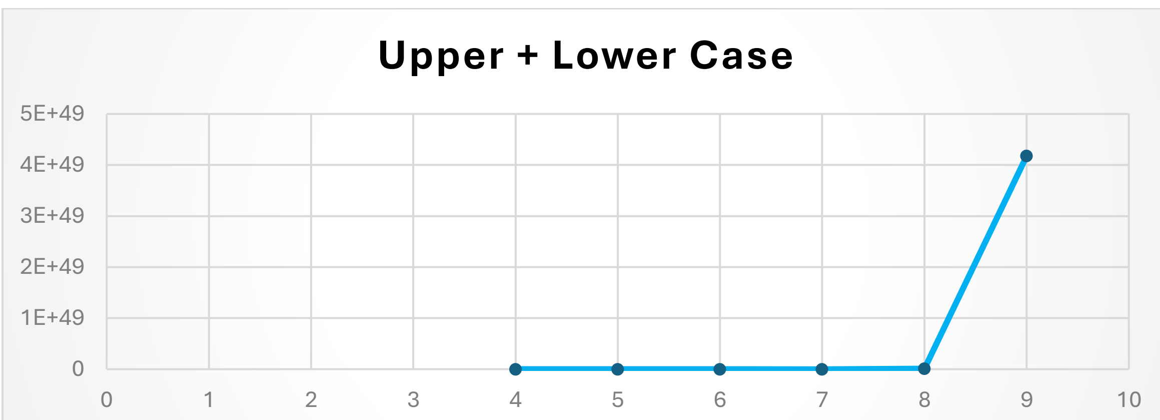


Figure 2 :Upper and Lowercase

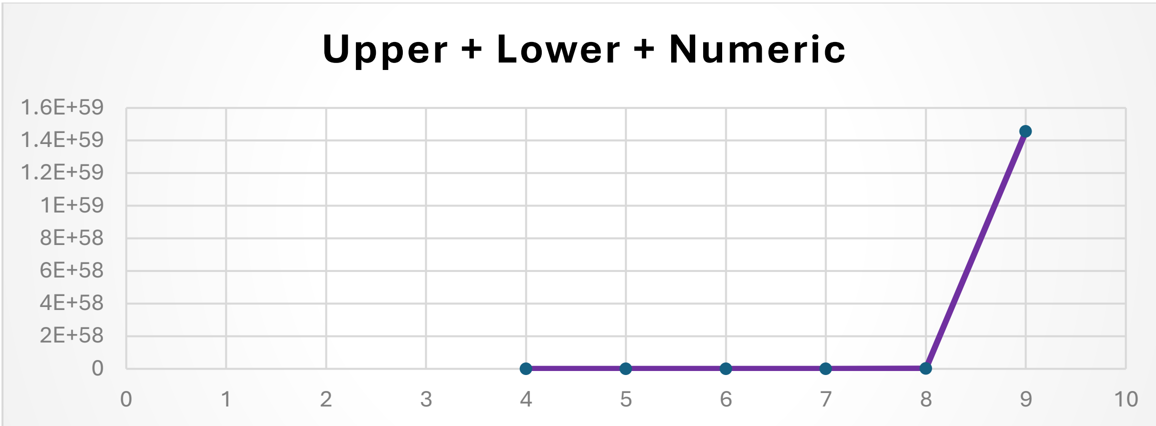


Figure 3: Upper, Lower and Numeric

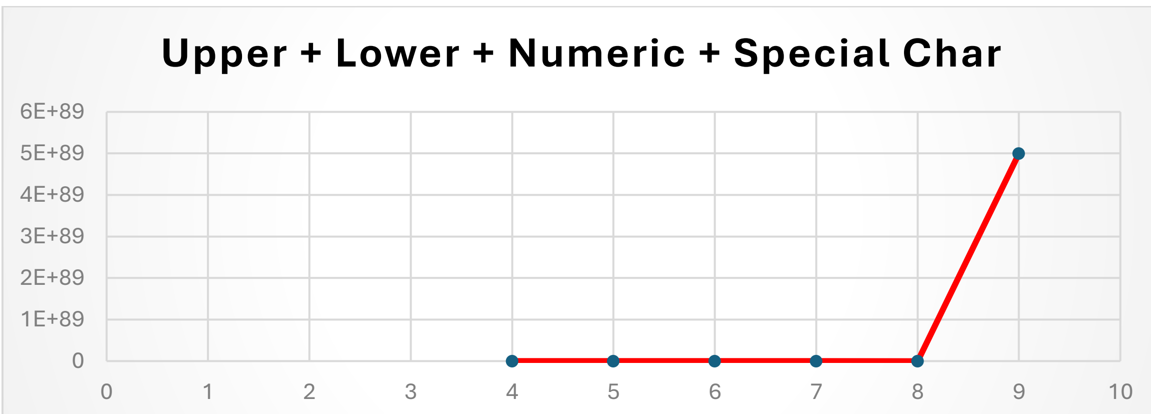


Figure 4: Upper, Lower, Numeric, and Special Char

Users often employ longer and more complex passwords (e.g., 8+ characters, combination of uppercase, lowercase, symbols) making traditional brute-force methods impractical.

LENGTH	COMBINATION
4^{26+26}	2220446049250313080847263336181640625
$4^{26+26+10}$	21267647932558653966460912964485513216
5^{26+26}	29098125988731506183153025616435306561536
$5^{26+26+10}$	21684043449710088680149056017398834228515625

Table 1: **No. of iterations for each length of passwords**

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

Table 2: **Password Combinations**

1.4 Motivation and Challenges

Outlined below are the motivations and challenges associated with our AI-driven password analysis tool.

Motivation

In the digital age, robust password security is more critical than ever. The need for an advanced, AI-driven password analysis tool arises from several key motivations.

Increasing Complexity of Passwords

- As cyber threats evolve, users are encouraged to create more complex passwords to enhance security. However, the complexity often leads to users adopting predictable patterns, which can be exploited.
- Our tool aims to address this by leveraging AI to recognize and exploit these patterns, thereby improving both offensive and defensive security measures.

Advancements in AI and Machine Learning

- The development of sophisticated AI algorithms offers new possibilities for password analysis and cracking. By integrating AI, our tool can stay ahead of traditional methods, providing more efficient and effective solutions.
- Utilizing state-of-the-art AI techniques ensures that our tool can handle the increasing complexity and diversity of modern passwords.

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.

- This dual functionality supports thorough security assessments, identifying vulnerabilities and testing the robustness of password policies.

Growing Number of Data Breaches

- The frequency and scale of data breaches are increasing, exposing millions of passwords. Our tool can analyze these breaches to understand common patterns and weaknesses, enhancing its cracking capabilities and providing insights into better password practices.
- By incorporating breach-checking features, our tool can alert users if their passwords have been compromised, enabling proactive security measures.

Enhancing Cybersecurity Awareness

- Educating users about the importance of strong passwords and the risks of weak ones is crucial. Our tool can demonstrate the ease with which weak passwords can be cracked, promoting better password practices.
- Security professionals can use our tool to raise awareness and provide tangible evidence of the need for robust password policies.

Challenges

Data Collection and Quality

- Obtaining a diverse and representative dataset of passwords is crucial for training the AI model. However, ethical and legal considerations must be adhered to when sourcing data from online leaks and breaches.
- Ensuring the dataset is comprehensive and free from sensitive or personally identifiable information requires rigorous data cleaning and preprocessing.

Balancing Offensive and Defensive Capabilities

- Integrating both offensive (password cracking) and defensive (password strength analysis) functionalities in a single tool presents a significant challenge. The tool must be designed to operate ethically, with proper authorization and adherence to legal guidelines.

- Ensuring that the tool's offensive capabilities do not compromise its defensive functionalities, and vice versa, requires careful design and implementation.

Adapting to Evolving Password Practices

- Password patterns and user behaviors are continuously evolving. Traditional methods often struggle to keep up with these changes, and our AI-driven approach must be flexible and adaptive to remain effective.
- Continuous updates and retraining of the AI model are necessary to handle new password trends and behaviors.

Efficiency and Scalability

- As passwords become more complex, traditional brute-force methods become impractical due to their time and resource-intensive nature. Our AI-powered tool must significantly reduce the search space and improve efficiency.
- Ensuring that the tool can scale effectively to handle large datasets and complex passwords without compromising performance is a critical challenge.

Security and Privacy Concerns

- Implementing features such as breach checking involves integrating with online databases, which must be done securely to prevent exposure to sensitive information.
- Ensuring that the tool itself is secure and does not become a vector for attacks is paramount. This includes safeguarding against misuse by malicious actors.

1.5 Goals and Objectives

Goals

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

Objectives

- Focus cracking efforts on statistically probable password patterns.

- Analyze password strength based on complexity metrics and AI-driven pattern recognition.
- Reduced cracking time.
- Improved vulnerability assessment.

1.6 Solution Overview

Our proposed AI-powered tool enhances password security by leveraging advanced algorithms and human-like password data. It focuses on both offensive and defensive capabilities:

Focused Cracking Efforts: Utilizes AI to identify statistically probable password patterns, reducing the search space and accelerating the cracking process compared to traditional brute-force methods.

Password Strength Analysis: Employs AI-driven pattern recognition and complexity metrics to analyze and identify weaknesses in complex passwords, aiding in targeted cracking attempts.

Reduced Cracking Time: Enhances efficiency by concentrating on likely password patterns, significantly reducing the time required for cracking.

Improved Vulnerability Assessment: Provides comprehensive assessments of password vulnerabilities, helping to identify and address weak points proactively.

Scope of the Project:

Defensive capabilities:

- **Password strength assessment:** Analyze password complexity, identify dictionary words, and check for patterns using AI.

  <u>Optional</u>

- **Breach checking:** Integrate with online databases to verify if entered passwords and email addresses have been exposed in known data breaches.

Offensive capabilities: (For Ethical purposes only):

AI-powered password cracking: Utilize correlation of password data and advanced algorithms to efficiently crack passwords within a specified scope and with proper authorization.

1.7 Report Outline

The project report for the "Multi-Purpose AI Password Analysis Tool" begins with an introduction, providing a comprehensive background on password security and the impetus for developing this tool. It identifies key opportunities and stakeholders who will benefit from enhanced password analysis. The report then reviews existing password analysis tools such as John the Ripper, Brutus, and Wfuzz, highlighting their limitations and setting the context for the problem statement and proposed solution. Our proposed solution leverages AI to focus on statistically probable password patterns and complexity metrics, aiming to reduce cracking time and improve vulnerability assessment. The objectives and scope of the project are clearly defined, emphasizing defensive capabilities like password strength assessment and breach checking, as well as ethical offensive capabilities like AI-powered password cracking. The evaluation plan outlines a comprehensive assessment using diverse datasets from real-world breaches, password dictionaries, and synthetic data. The development and training section details the process of acquiring and preparing relevant datasets, training and optimizing the AI model, and selecting appropriate algorithms. Implementation focuses on core functionalities for password strength analysis and breach checking, with optional automated password cracking. The report also discusses the design and development of a user-friendly interface, ensuring seamless integration with the AI model and functionalities. Thorough testing and refinement processes are documented to ensure the tool's effectiveness and usability. Finally, the report includes references and a list of figures to support the content and provide visual clarity.

Chapter 2:
Market Survey

Chapter 2: Market Survey

2.1 Introduction

2.2 Literature Review/Technologies & Products Overview

2.3 Comparative Analysis

2.4 Research Gaps

2.5 Problem Statement

2.6 Summary

Chapter 2: Market Survey

2.1 Introduction

Traditional password-cracking methods have primarily relied on brute force and precomputed dictionaries to attempt to decipher passwords. These methods, while effective for certain use cases, face significant limitations. Brute force attacks, for example, are highly resource-intensive and often impractical for complex passwords due to the sheer number of combinations to test. Meanwhile, dictionary attacks are limited by their reliance on precompiled lists, which may not encompass unique or complex passwords created by users. Additionally, as password security practices evolve, traditional methods lack adaptability and cannot handle nuanced techniques like mixed character cases and special characters. Lastly, scalability has become a pressing issue with modern, complex passwords, as traditional tools struggle to meet the increased processing and memory demands.

2.2 Market Survey/Technologies & Products Overview

The foundation of password security lies in its ability to resist unauthorized access, primarily achieved through encryption and hashing techniques. Over the years, several tools and methodologies have been developed to analyze and crack passwords, each with its unique strengths and weaknesses.

Existing Systems

The following are the Existing systems that exist in the market for about more than a decade.

John the Ripper

Limitations

- Relies on traditional methods of password cracking such as dictionary attacks, brute-force attacks, and rainbow tables.
- While powerful, it may not effectively adapt to evolving password patterns and behaviors without continuous updates and modifications.

Problems

- Lack of advanced AI integration for targeted password list generation based on learned patterns.
- Limited defensive capabilities in analyzing password strength beyond basic dictionary checks.

Brutus

Limitations

- Primarily designed for online password cracking through network protocols like HTTP, FTP, SMB, etc.
- May lack advanced AI-driven capabilities for generating targeted password lists.

Problems

- Limited applicability for offline password analysis and cracking scenarios.
- May not integrate well with the defensive aspects of the proposed tool, such as analyzing password strength and checking for breached credentials.

Wfuzz

Limitations

- Primarily focuses on web application security testing, including fuzzing and brute-forcing directories and files.
- May lack specialized features for password cracking and analysis.

Problems

- Not specifically tailored for password analysis and cracking tasks, thus requiring significant adaptation to fit into the proposed project's scope.
- Limited or no integration with AI-driven password analysis and cracking techniques.

2.3 Comparative Analysis

The existing tools analyzed above, such as John the Ripper, RainbowCrack, OphCrack, L0phtCrack, and Aircrack-ng, each have unique features suited to specific contexts of password or network security. However, the major drawbacks include limited adaptability to evolving password complexities, lack of AI-driven password generation techniques, and minimal integration of defensive capabilities such as breach detection. Tools like Brutus and Wfuzz further illustrate this, as they lack the flexibility and AI-driven features necessary for advanced password analysis.

Comparison Tables

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 3: Comparsion Tables of tools

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 Testi

Table 4: Comparison Table

2.4 Research Gaps

Several gaps exist in the current password-cracking tools landscape:

- **Limited AI Integration:** Many tools lack AI-driven methods for generating adaptive and targeted password lists based on learned patterns from real-world data, reducing their effectiveness.
- **Defensive Shortcomings:** Current tools often do not include mechanisms to assess password strength comprehensively or to cross-reference with breached credentials, leaving users vulnerable.
- **Scalability Constraints:** As passwords increase in length and complexity, traditional tools face performance issues that make scaling difficult without substantial computational resources.
- **Adaptability:** Tools often fail to account for unique and complex password structures, such as those with mixed character types or special symbols, reducing their applicability to modern password practices.

2.5 Problem Statement

With increasing password complexity requirements, brute-force attacks become significantly slower and less efficient.

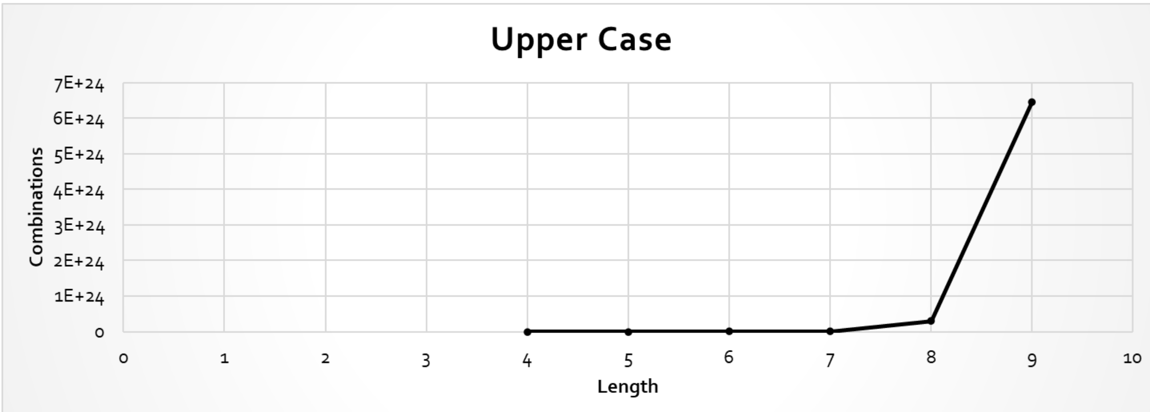


Figure 5: Upper Case

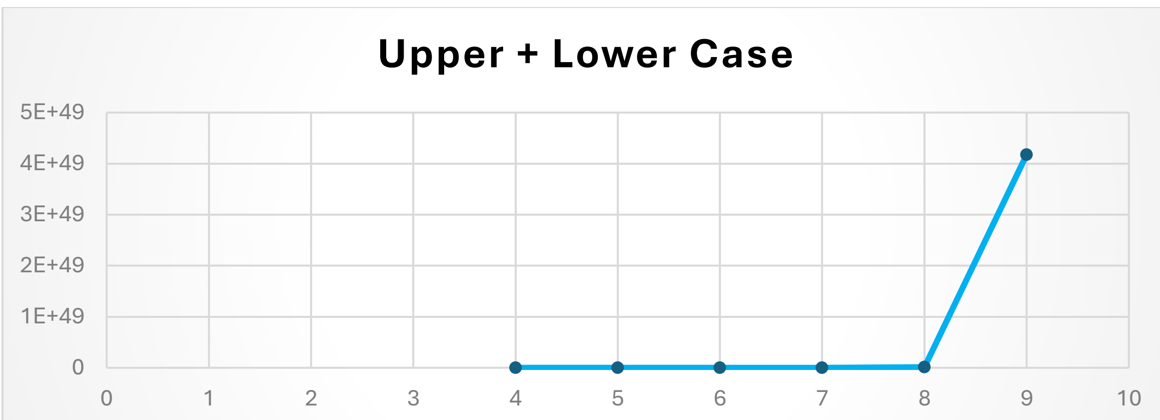


Figure 6 :Upper and Lowercase

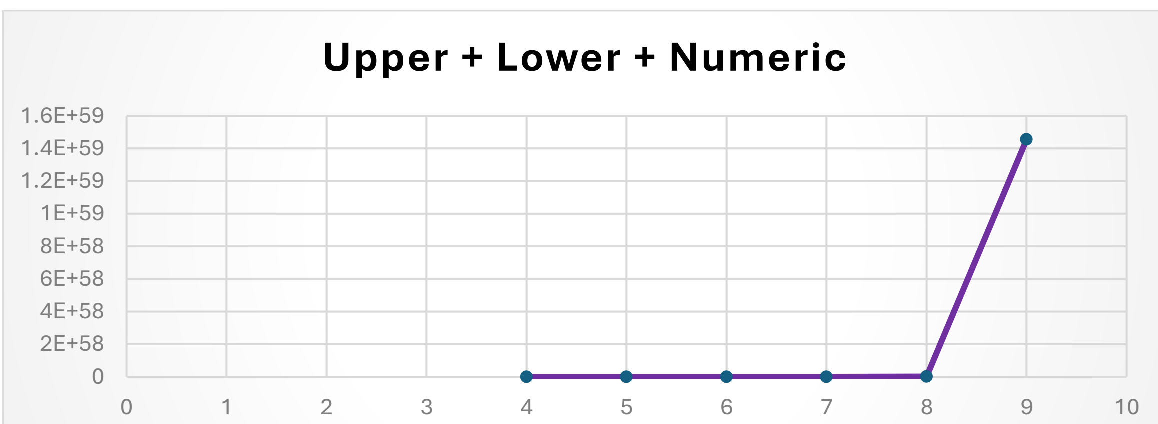


Figure 7: Upper, Lower and Numeric

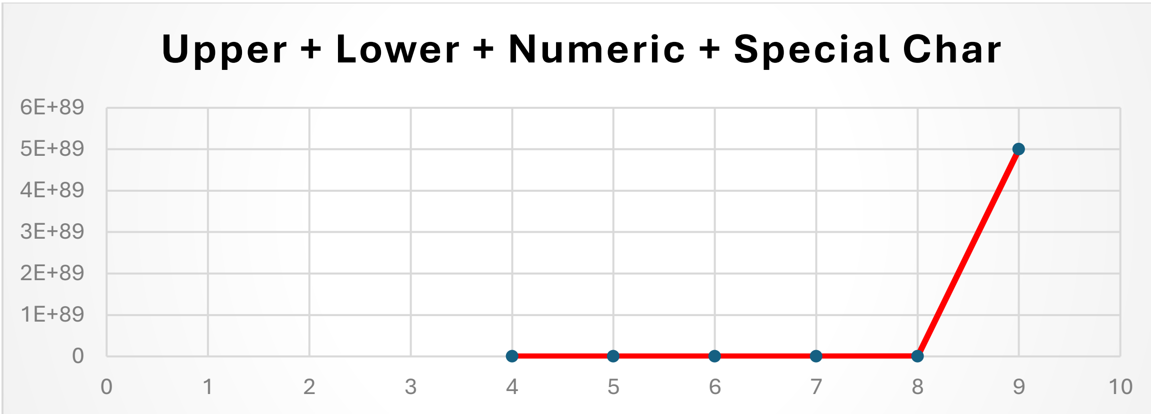


Figure 8: Upper, Lower, Numeric, and Special Char

Users often employ longer and more complex passwords (e.g., 8+ characters, combination of uppercase, lowercase, symbols) making traditional brute-force methods impractical.

Length	Combination
4^{26+26}	2220446049250313080847263336181640625
$4^{26+26+10}$	21267647932558653966460912964485513216
5^{26+26}	29098125988731506183153025616435306561536
$5^{26+26+10}$	21684043449710088680149056017398834228515625

Table 5: **No. of iterations for each length of passwords**

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

Table 6: **Password Combinations**

2.6 Summary

The evolution of password security technologies underscores the ongoing battle between cybersecurity professionals and cybercriminals. Traditional tools like John the Ripper, Brutus, and Wfuzz continue to be pivotal in password cracking and security testing. However, these methods are increasingly challenged by the rise of complex, user-generated passwords that incorporate unique structures and special characters. Traditional approaches, such as brute force and dictionary attacks, are often inefficient for handling such complex passwords due to their time-intensive nature and reliance on precompiled lists that may not cover all variations. Moreover, these tools often struggle with scalability and adaptability to modern password security practices.

The integration of AI and machine learning offers a transformative shift in password security, enabling adaptive, targeted password cracking based on learned password patterns and real-world data. This innovation introduces enhanced capabilities, including human-like password data generation, intelligent password strength analysis, and breach detection, providing a more effective defense against evolving cyber threats. The limitations of existing tools, such as the absence of AI-driven password list generation, limited defensive capabilities, and scalability issues, highlight a critical need for advanced solutions in password security.

This literature review emphasizes the potential of AI-powered tools to enhance password security, addressing the gaps present in traditional methods. By understanding the strengths and weaknesses of existing technologies, this research sets the foundation for developing a comprehensive AI-driven password analysis tool. Such a tool would combine efficiency, adaptability, and defensive capabilities, offering robust protection of sensitive data in an increasingly digital world.

Chapter 3:
Requirements and System Design

Chapter 3: Requirements and System Design

Chapter 3: Requirements and System Design

3.1 Introduction

The "Multi-Purpose AI Password Analysis Tool" aims to enhance password security through advanced AI-driven techniques. This section outlines the system's architecture, functional and non-functional requirements, and the design considerations necessary for developing a robust and effective tool for password analysis and management.

3.2 System Architecture

The system architecture consists of several key components:

- **Graphical User Interface (GUI):** A user-friendly graphical interface for interacting 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:** Implements breach checking and password strength analysis.
- **Integration Layer:** Connects with external tools (e.g., Hashcat) for enhanced functionality.

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:** Illustrates the overall architecture, including components and their interactions.
- **Data Flow Diagram:** Shows how data moves through the system, from input to processing and output.

3.6 Hardware and Software Requirements

Hardware Requirements

- **Workstation: HP Z840:** The HP Z840 workstation is a high-performance desktop designed for demanding tasks such as AI and machine learning model training. It offers robust processing power, extensive memory capacity, and advanced graphics capabilities, making it ideal for computationally intensive applications.
- **Processor: Intel Xeon E5-2680 V4 Dual:** The Intel Xeon E5-2680 V4 is a high-end server processor with 14 cores and 28 threads, offering a base clock speed of 2.4 GHz and a turbo boost up to 3.3 GHz. Utilizing two of these processors in a dual configuration provides a total of 28 cores and 56 threads, significantly enhancing parallel processing capabilities crucial for training complex AI models.
- **RAM: 64GB DDR4 ECC:** 64GB of DDR4 Error-Correcting Code (ECC) RAM ensures high data integrity and system stability, which is essential for handling large datasets and preventing data corruption during intensive computations. This amount of memory supports the simultaneous operation of multiple AI models and applications without performance degradation.

- **GPU: <u>RTX 3070TI 8GB GDDR6x</u>:** The NVIDIA RTX 3070TI graphics card, with 8GB of GDDR6x memory, provides powerful parallel processing capabilities and accelerated performance for deep learning tasks. Its CUDA cores and Tensor cores are optimized for AI workloads, significantly speeding up the training and inference processes of neural networks.
- **SSD: <u>512GB</u>:** A 512GB Solid-State Drive (SSD) offers fast read/write speeds and quick access to data, reducing loading times and improving overall system responsiveness. The SSD is used for storing the operating system, software applications, and project files, ensuring efficient data retrieval and storage.

<u>Software Requirements</u>

- **Operating System: <u>Windows 11 64bit</u>:** Windows 11 is the latest version of Microsoft's operating system, providing a modern interface, enhanced security features, and support for the latest hardware and software technologies. The 64-bit version is necessary to leverage the full potential of the workstation's hardware, especially the large amount of RAM and dual processors.
- **Integrated Development Environment (IDE): <u>Visual Studio</u>:** Visual Studio is a comprehensive development environment from Microsoft, offering tools and features for coding, debugging, and deploying applications. It supports multiple programming languages and integrates well with various frameworks and libraries, making it suitable for developing and testing the AI password analysis tool.
- **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>:** TensorFlow is an open-source machine learning framework developed by Google, offering comprehensive tools for building and deploying machine learning models. It supports deep learning and neural network training, providing high performance on both CPUs and GPUs. TensorFlow's flexibility and scalability make it a preferred choice for developing complex AI applications.

- **Deep Learning Library: PyTorch:** PyTorch is an open-source deep learning library developed by Facebook's AI Research lab, known for its dynamic computational graph and ease of use. It is widely adopted for research and production in AI, offering strong support for GPU acceleration and integration with other machine learning tools. PyTorch is used for experimenting with and deploying AI models in the password analysis tool.
- **GPU-Accelerated Library: cuDNN:** cuDNN (CUDA Deep Neural Network library) is a GPU-accelerated library developed by NVIDIA, designed to enhance the performance of deep learning frameworks. It provides highly optimized implementations of standard routines such as forward and backward convolution, pooling, normalization, and activation layers. cuDNN is essential for maximizing the efficiency of TensorFlow and PyTorch models on the RTX 3070TI GPU.

3.7 Threat Scenarios

- **Denial of Service (DoS):** Overloading the system to disrupt service availability.
- **Malicious Use:** The tool being used for unethical hacking or unauthorized password cracking.

3.8 Threat Modeling Techniques

STRIDE: Assessing threats based on Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege.

3.9 Threat Resistance Model

- **Access Control:** Ensuring that only authorized users can access sensitive functionalities.
- **Data Encryption:** Encrypting sensitive data both at rest and in transit to prevent unauthorized access.
- **Regular Security Audits:** Conducting periodic assessments to identify and mitigate vulnerabilities.

3.10 Summary

The "Multi-Purpose AI Password Analysis Tool" is designed to address the growing need for robust password security in an increasingly digital world. By outlining the system architecture, functional and non-functional requirements, and potential threats, this section provides a comprehensive overview of the design considerations necessary for developing an effective and secure password analysis tool. The integration of advanced AI techniques, along with a focus on usability and security, positions this tool as a valuable asset for individuals and organizations seeking to enhance their password management practices.

Chapter 4:
Proposed Solution

Chapter 4: Proposed Solution

4.1 Introduction

4.2 Proposed Model

4.3 Data Collection

4.4 Data Pre-processing

4.5 Tools and techniques

4.6 Summary

Chapter 4: Proposed Solution

4.1 Introduction

The proposed solution for the "Multi-Purpose AI Password Analysis Tool" focuses on leveraging advanced AI techniques to enhance password security. This chapter outlines the model architecture, data collection methods, preprocessing steps, and the tools and techniques employed in the development of the tool. The goal is to create a robust system capable of both analyzing and generating secure passwords while addressing the challenges posed by modern password complexities.

4.2 Proposed Model

The proposed model for the "Multi-Purpose AI Password Analysis Tool" is primarily based on a quantitative and experimental approach. This section outlines the rationale behind the chosen methodologies and how they contribute to the overall effectiveness of the tool.

Quantitative Approach

The quantitative aspect of the model focuses on the use of numerical data and statistical methods to analyze and generate passwords.

Data-Driven Analysis

- The model utilizes large datasets of passwords collected from various sources, including online breaches and password dictionaries. This data is analyzed quantitatively to identify patterns, trends, and common characteristics of passwords.
- Statistical techniques are employed to evaluate password strength based on complexity metrics, such as length, character variety (uppercase, lowercase, numeric, special characters), and 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 proportion of successfully cracked passwords during testing.

Experimental Approach

The experimental aspect of the model involves the iterative testing and refinement of the AI algorithms used for password analysis and generation.

Model Training and Optimization

- The Recurrent Neural Network (RNN) model is trained on the collected password datasets. This involves experimenting with different architectures (e.g., LSTM, GRU) and hyperparameters (e.g., learning rate, batch size) to optimize performance.
- The training process is experimental in nature, as it requires multiple iterations to fine-tune the model for better accuracy and efficiency in predicting password strength and generating secure passwords.

Validation and Testing

After training, the model undergoes rigorous testing using separate validation datasets to assess its performance. This includes:

- **Cross-Validation:** Splitting the dataset into training and validation sets to ensure the model generalizes well to unseen data.
- **A/B Testing:** Comparing different versions of the model to determine which configuration yields better results in terms of password generation and analysis.

Qualitative Insights

While the primary focus is on quantitative and experimental methods, qualitative insights are also gathered to enhance the model's effectiveness.

User Feedback: Engaging with users during the development process to gather qualitative feedback on the usability and functionality of the tool. This feedback helps identify areas for improvement and ensures that the tool meets user expectations.

Expert Reviews: Consulting cybersecurity experts to evaluate the model's approach and effectiveness. Their insights contribute to refining the algorithms and ensuring that the tool adheres to best practices in password security.

4.3 Data Collection

Data collection involves gathering a diverse set of password datasets from various sources.

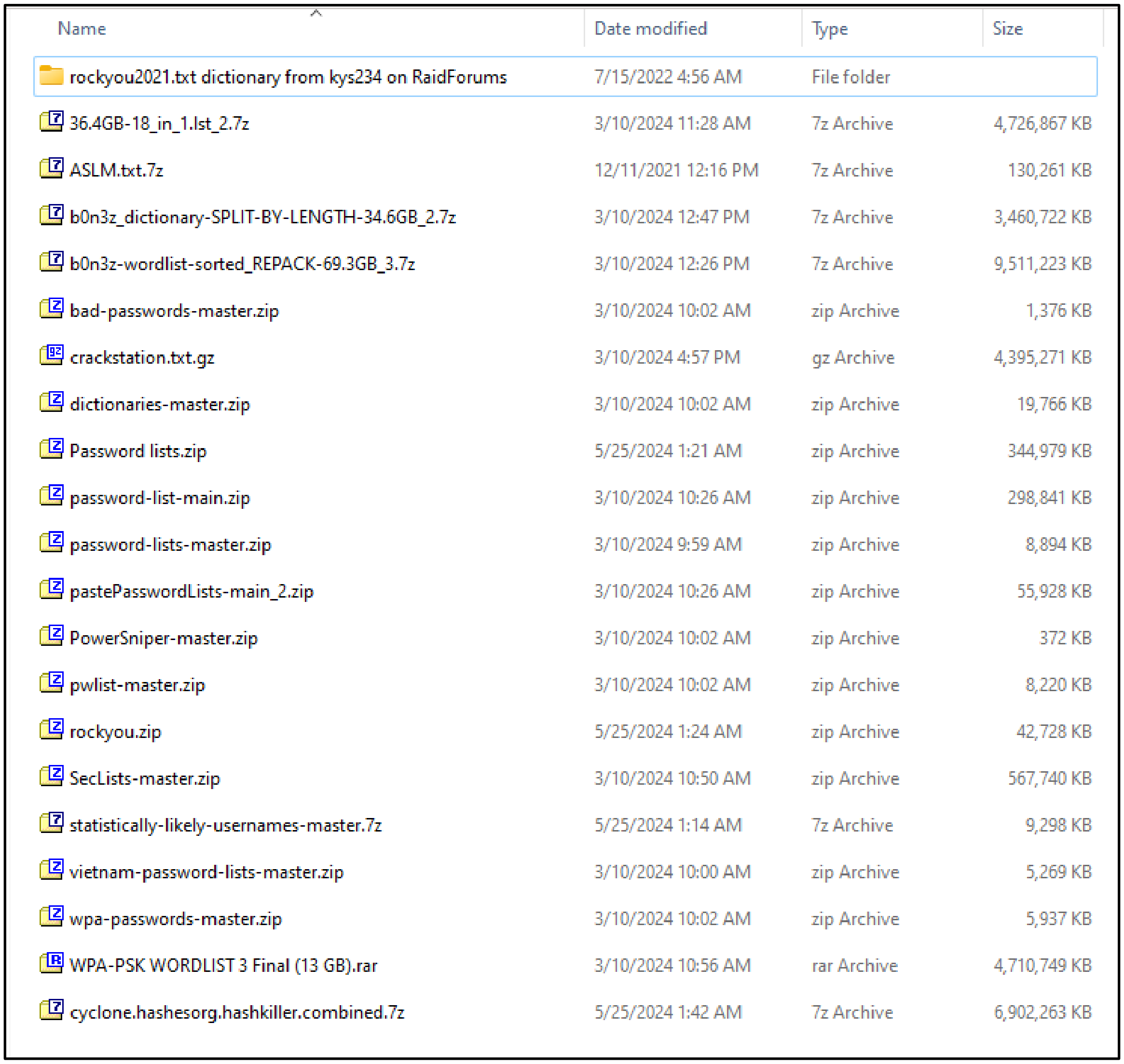
- **Online Leaks and Breaches:** Ethical sourcing of data from known breaches to ensure a comprehensive representation of password types.
- **Password Dictionaries:** Utilizing existing password lists that cover a wide range of lengths, complexities, and compositions.

Collected Datasets Table

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

Table 7: Datasets Table

Dataset Downloaded Screen Shot



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 9: **Datasets Screenshots**

4.4 Data Pre-processing

Data preprocessing is crucial for ensuring the quality and relevance of the datasets. Key steps include:

- **Data Cleaning:** Filtering passwords based on length (4 to 10 characters) and removing corrupted entries.
- **Data Splitting:** Dividing large datasets into manageable chunks (approximately 2 GB each) for efficient processing.

Data Cleaning

- Filter the password of length of 4 to 10 to keep the datasets clean and manageable.
- Get rid of any weird or corrupted entries that could mess up training.

Pseudocode: 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 10: Pseudocode DataFilteration-1

```
            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 11: Pseudocode DataFilteration-2

```
    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 12: Pseudocode DataFileration-3

```
    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 13: Pseudocode DataFilteration-4

Figure 14: DataFilteration Flowchart

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

Table 8: No. of Password as per Length

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.

Pseudocode: Data Splitting

DataSpliter.py | Psuedocode Data Splitter

Psuedocode Data Splitter

```
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 15: Pseudocode DataSplitting-1

```
                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 16: Pseudocode DataSplitting-2

```
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 17: Pseudocode Data Splitting-3

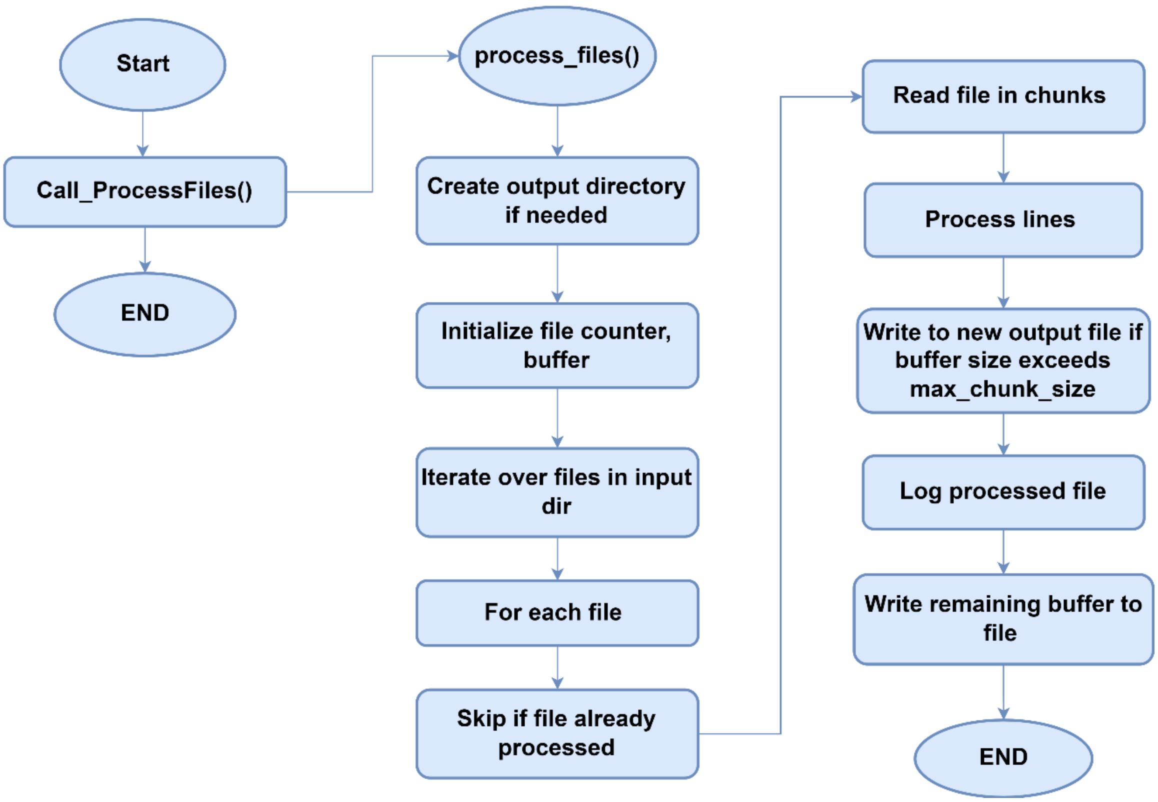


Figure 18: DataSplitting Flowchart

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.

4.5 Tools and Techniques

The development of the tool employs various tools and techniques.

Tools

- **Programming Languages:** Python 3.12 for implementing AI models and data processing.
- **Machine Learning Frameworks:** TensorFlow and PyTorch for building and training the RNN model.
- **Data Management Tools:** Custom scripts for data cleaning, filtering, splitting and training of our AI model.

- **Visualization Tools:** Flowcharts and pseudocode to illustrate data processing and model training steps.
- **Integrated Development Environment (IDE): Visual Studio:** Used for coding, debugging, and deploying the application.
- **Password Cracking Tool: Hashcat:** Utilized for enhanced password cracking capabilities.

Techniques

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data by maintaining a hidden state that captures information about previous inputs in the sequence. RNNs are well-suited for tasks where the input data's temporal order is important, making them a suitable choice for password generation, where the order of characters matters.

Training Phase

- **Input:** A dataset of passwords represented as sequences of characters.
- **Architecture:** The RNN consists of recurrent units (such as LSTM - Long Short-Term Memory, or GRU - Gated Recurrent Unit) that process one character at a time while maintaining a hidden state. This hidden state captures information about previous characters in the password sequence.
- **Sequence Learning:** The RNN is trained to predict the next character in the sequence given the previous characters. This is done by feeding each character in the sequence into the RNN and comparing the predicted next character with the actual next character in the training data.
- **Backpropagation Through Time (BPTT):** The errors are backpropagated through time to update the weights of the RNN, allowing it to learn the patterns and dependencies present in the password dataset.
- **Loss Function:** The RNN is trained to minimize a loss function, such as categorical cross-entropy, which measures the difference between the predicted and actual characters in the sequence.

Generation Phase

- **Seed:** During the generation phase, a seed sequence is provided as input to the RNN to initiate the generation process. This seed sequence can be randomly chosen or predefined.
- **Character-by-Character Generation:** RNN generates new characters one at a time by feeding the previous character and the current hidden state back into the network. The output character is sampled from the predicted probability distribution over the character vocabulary.
- **Output:** The generated characters are concatenated to form a new password sequence.

Evaluation

The quality of the generated passwords can be evaluated using metrics such as similarity to real passwords, diversity, and entropy. Human evaluators can also assess the usability and security of the generated passwords.

Pseudocode (For Generating Passwords)

Pseudocode RNN Generate > RNNModel > init_hidden

```python
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 19: Pseudocode RNN for Generating Passwords

```python
    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 20: Pseudocode RNN for Generating Passwords-1

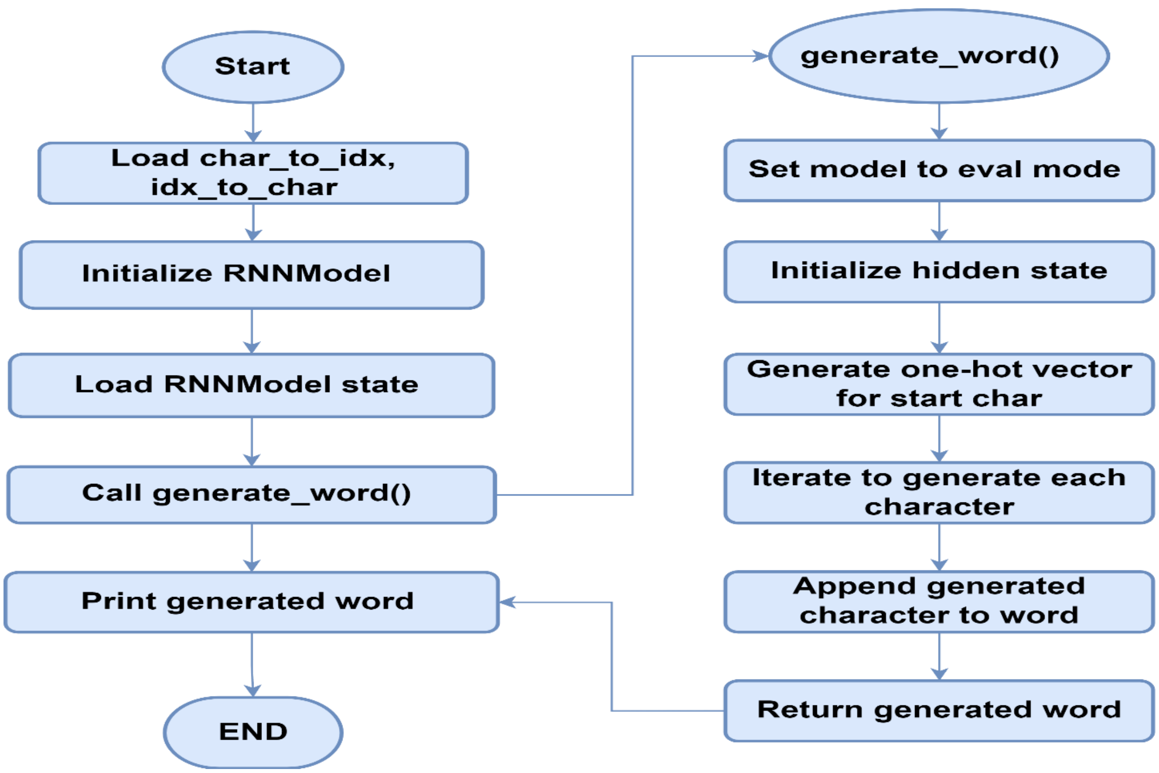


Figure 21: flowchart of RNN for Generating Passwords

Pseudocode (For Training)

```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 22: Pseudocode for RNN's Training

```python
        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 23: Pseudocode for RNN's Traning-2

```
            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 24: Pseudocode for RNN's Traning-3

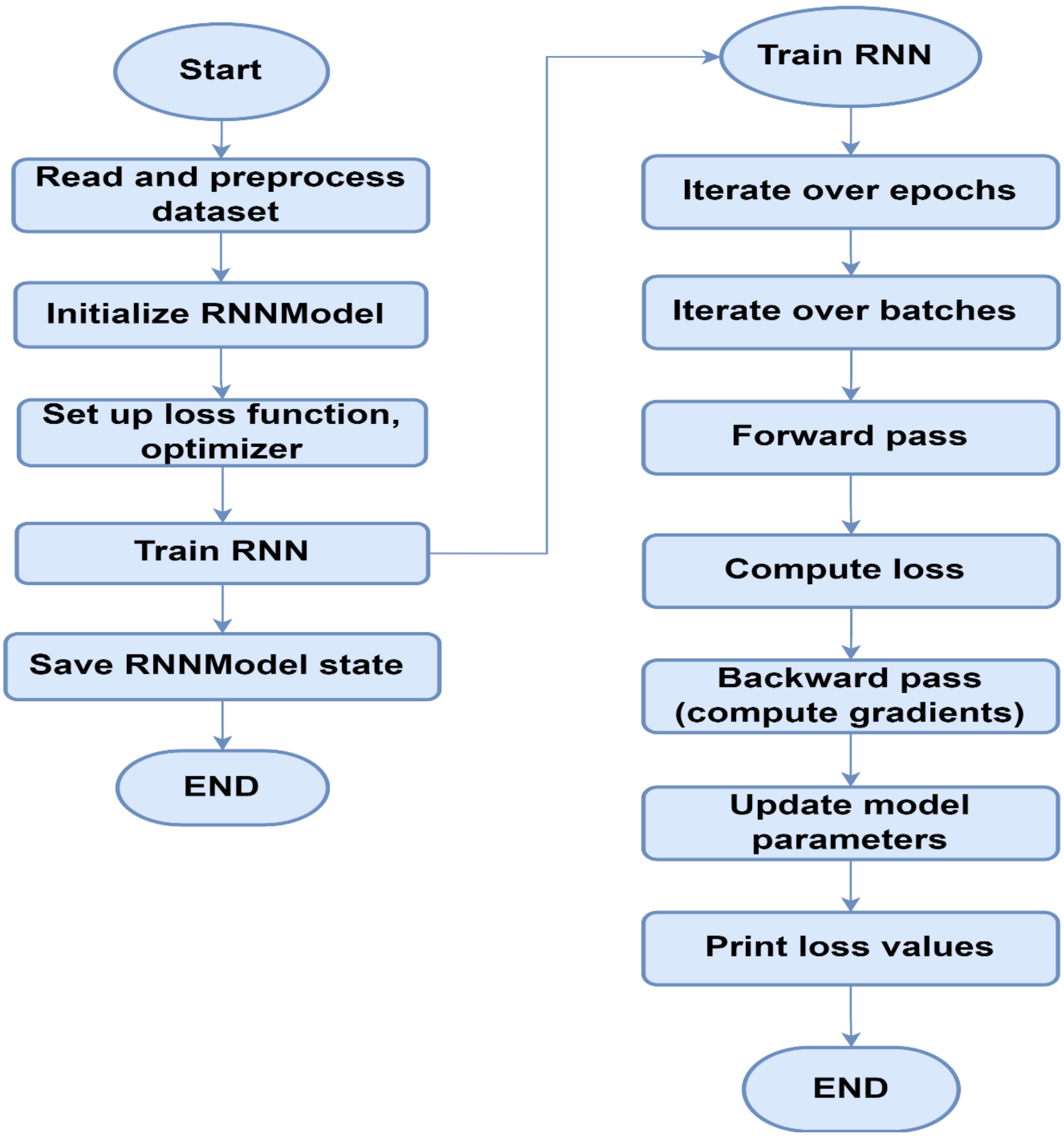


Figure 25: Flowchart of RNN's Training

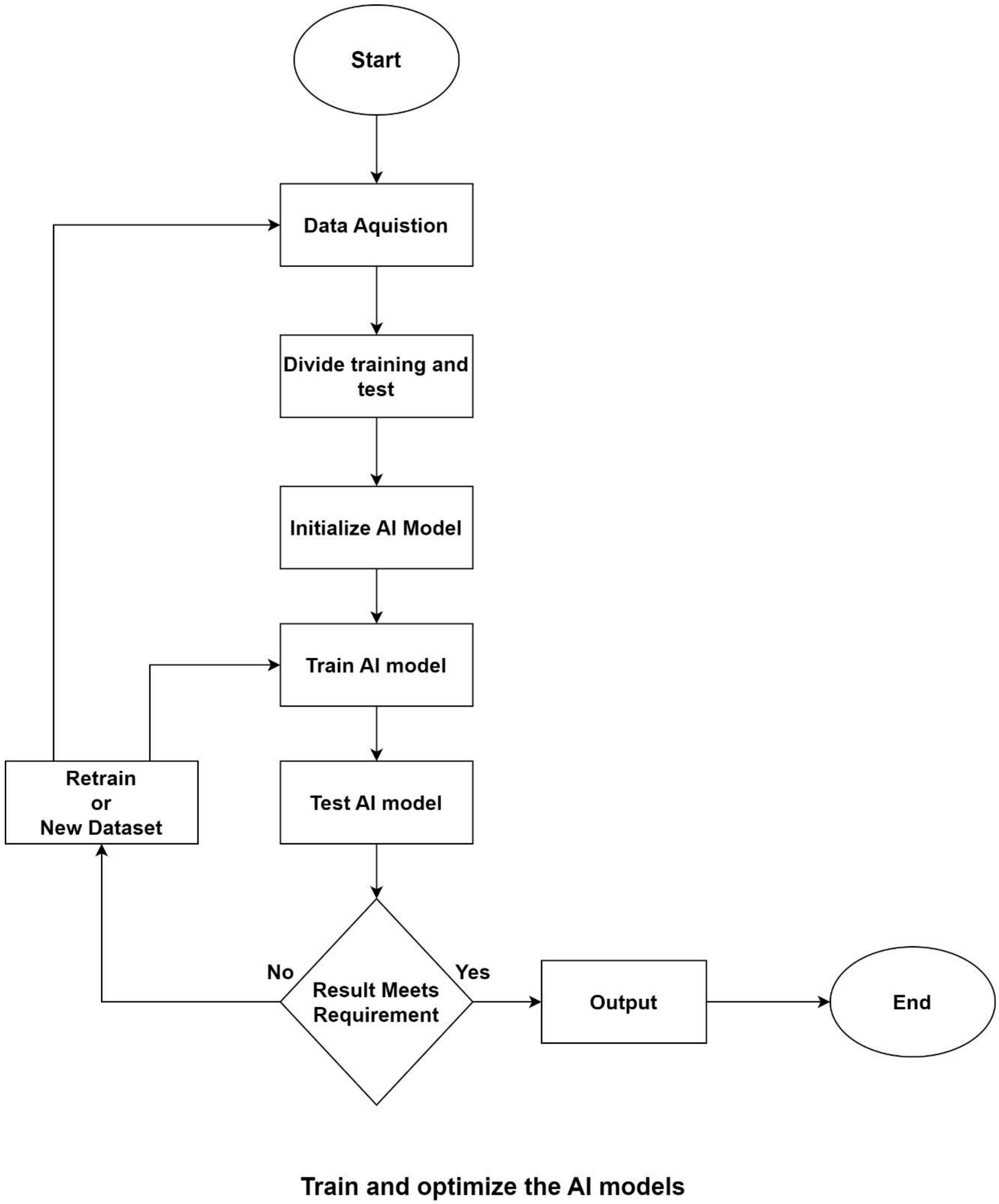


Figure 26: Training AI Model Flow Chart

4.6 Evaluation Metrics

4.7 Summary

The proposed solution for the "Multi-Purpose AI Password Analysis Tool" integrates advanced AI techniques with a robust data management strategy. By focusing on effective data collection, preprocessing, and the use of RNNs, the tool aims to enhance password security through comprehensive analysis and generation capabilities. This chapter sets the foundation for the subsequent development and training of the AI model, ensuring that the tool is equipped to address the challenges posed by modern password complexities.

Chapter 5:
Implementation and Testing

Chapter 5: Implementation and Testing

Chapter 5: Implementation and Testing

Chapter 6:
Conclusion & Future Work

Chapter 6: Conclusion & Future Work

Chapter 6: Conclusion & Future Work

6.1 Introduction

The "Multi-Purpose AI Password Analysis Tool" represents a significant advancement in password security management. By integrating artificial intelligence with traditional password analysis techniques, the tool provides both offensive and defensive capabilities. This chapter summarizes the achievements of the project, critically reviews its outcomes, and outlines future recommendations for further development and enhancement.

6.2 Achievements and Improvements

Comprehensive Functionality: The tool successfully combines password cracking and strength analysis, allowing users to evaluate the security of their passwords while also providing insights into potential vulnerabilities. This dual functionality is a significant improvement over existing tools that typically focus on one aspect.

AI Integration: By employing Recurrent Neural Networks (RNNs), the tool leverages machine learning to analyze vast datasets of passwords. This enables it to identify patterns and generate passwords that are not only secure but also reflective of real-world usage, enhancing its effectiveness in both cracking and generation tasks.

User-Centric Design: The development of a user-friendly graphical interface has made the tool accessible to a broader audience, including those with limited technical expertise. This focus on usability ensures that users can easily navigate the tool's features and functionalities.

Robust Testing Framework: The project implemented a rigorous testing framework, including security properties testing, system integration, and user acceptance testing. These efforts validated the tool's performance and reliability, ensuring it meets the needs of its users.

Community Engagement: The tool has been made available on GitHub, fostering community involvement and feedback. This open-source approach encourages collaboration and continuo us improvement, allowing other developers to contribute to its evolution.

6.3 Critical Review

Data Limitations: The effectiveness of the AI model is contingent upon the quality and diversity of the training data. While the project utilized extensive datasets, there is a need for ongoing efforts to expand and diversify these datasets to improve the model's adaptability to new password trends.

Ethical Implications: The tool's capabilities for both offensive and defensive purposes raise ethical concerns. It is crucial to establish clear guidelines and user agreements to prevent misuse and ensure that the tool is used responsibly and ethically.

Performance Limitations: Although the tool performs well with the current datasets, scalability remains a concern. As password complexity increases, the tool may require optimization to maintain efficiency and effectiveness.

User Education: While the tool provides valuable insights into password security, there is a need for ongoing user education to promote best practices in password management. Users must understand the importance of strong passwords and how to utilize the tool effectively.

6.4 Future Recommendations

Continuous Model Training: Implement a system for continuous learning where the AI model is regularly updated with new data to adapt to evolving password trends and user behaviors.

Enhanced Features: Consider adding features such as real-time breach alerts and integration with password managers to provide users with a comprehensive security solution.

Diverse User Testing: Conduct extensive field testing with a diverse range of users to gather feedback on usability and functionality. This will help identify areas for improvement and ensure the tool meets the needs of various user groups.

Broader Testing: Conduct extensive field testing with diverse user groups to gather feedback and improve the tool's functionality and user experience.

Collaboration with Cybersecurity Experts: Partner with cybersecurity professionals to refine the tool's capabilities and ensure it meets industry standards and best practices.

Research and Development: Invest in R&D to explore advanced AI techniques, such as generative adversarial networks (GANs), for even more effective password generation and analysis.

6.6 Summary

The "Multi-Purpose AI Password Analysis Tool" has made significant contributions to the field of password security through its innovative use of artificial intelligence and comprehensive functionality. By integrating advanced machine learning techniques, particularly Recurrent Neural Networks (RNNs), the tool effectively analyzes and generates secure passwords, addressing the growing concerns surrounding password vulnerabilities in an increasingly digital world.

The achievements outlined in this project highlight the tool's potential to enhance password management for both individuals and organizations. Its dual functionality allows users to not only assess the strength of their passwords but also to understand the vulnerabilities inherent in various password configurations. This capability is crucial for security professionals and everyday users alike, as it empowers them to adopt stronger password practices and mitigate risks associated with data breaches.

Moreover, the user-friendly graphical interface ensures that the tool is accessible to a broad audience, including those with limited technical expertise. This focus on usability is essential in promoting widespread adoption and effective utilization of the tool's features. The rigorous testing framework implemented throughout the development process has validated the tool's performance, ensuring reliability and effectiveness in real-world scenarios.

However, the project also recognizes the need for ongoing improvements. As password complexity continues to evolve, the tool must adapt to new trends and challenges. Continuous model training and updates will be necessary to maintain its effectiveness in a rapidly changing cybersecurity landscape. Additionally, ethical considerations surrounding the tool's dual capabilities for offensive and defensive purposes must be addressed. Establishing clear guidelines and user agreements will be vital in preventing misuse and ensuring responsible usage.

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

Appendix

Appendix A: Data Preparation Details

Table A.1: Collected Datasets

No	Name	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	4.7 GB

Figure A.1: Dataset Downloaded Screenshot

Appendix B: Data Filtration

Table A.2: Filtered Passwords Count

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

Appendix C: Data Splitting

Pseudocode: Data Splitting

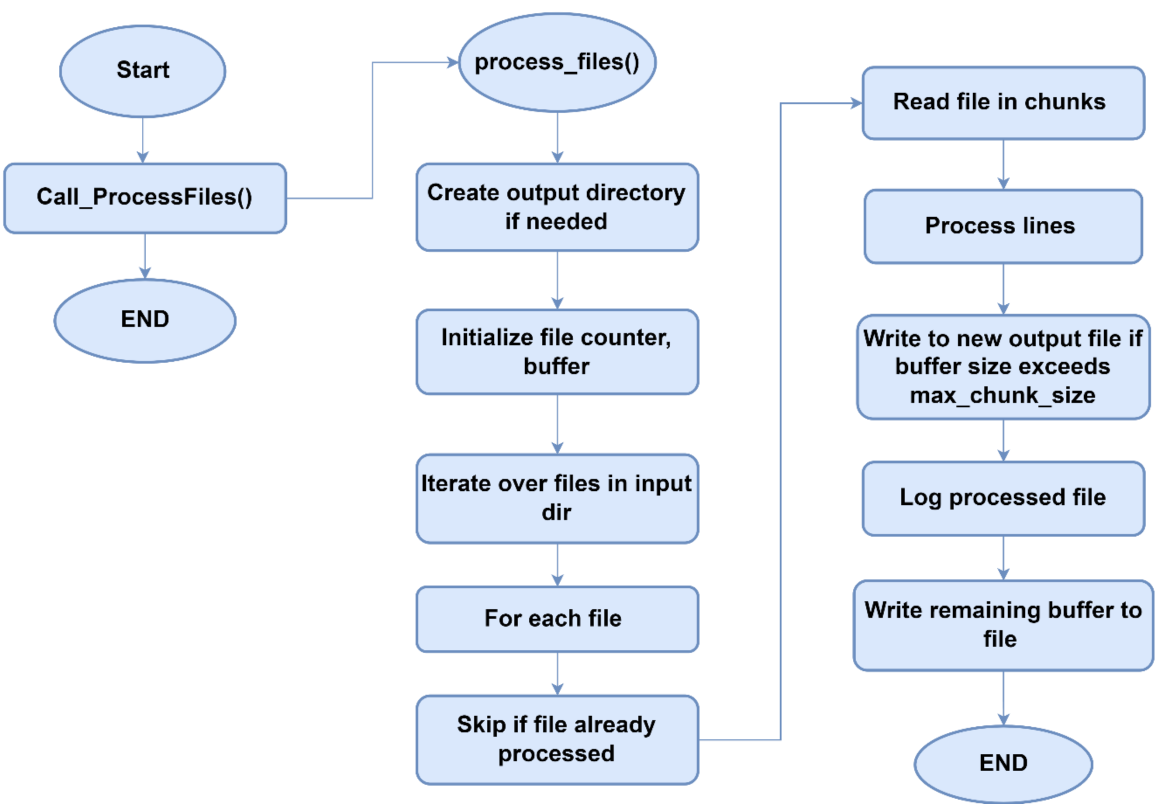
DataSpliter.py | Psuedocode Data Splitter

Psuedocode Data Splitter

```
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
                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
```

```
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
```

Flowchart: Data Splitting



Github

https://github.com/Zeshankhan03/Multi-Purpose-AI-Password-Analysis-Tool