Multi-Purpose AI Password Analysis Tool

By:

Osama Khalid– Team Member 1
27971
Sardar M. Zeeshan khan – Team Member 2
27969
Salman Ali – Team Member 3
27667

Supervised by:
Sir Osamah Ahmed

Faculty of Computing
Riphah International University, Islamabad
Spring/Fall 20xx

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 Software Engineering

Faculty of Computing
Riphah International University, Islamabad

Date: [date of final presentation]

Final Approval

This is to certify that we have read the report submitted by ***name of student(s) (CMS #)***, for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Software Engineering (BSSE). 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 Software Engineering (BSSE).

Committee:

1 _____

[Name Supervisor]
(Supervisor)

2 _____

[Name of HOD/chairman]
(Head of Department/chairman)

Declaration

We hereby declare that this document "**[Project Title]**" 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 accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisor **[insert name of Supervisor(s)]**. 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

Insert dedication here…

Acknowledgement

First of all we are obliged to Allah 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, **Mr. Osama Ahmad**, for his 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. Tajamul 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

This project proposes the development of a multi-purpose AI password analysis tool offering both offensive and defensive capabilities. It leverages AI to analyze and crack passwords, assess their security strength, and identify breached credentials. This tool empowers users to test the resilience of their own passwords and proactively identify vulnerabilities.

Table of Contents

List of Figures

List of Tables

Chapter 1: Heading (30-Point Size, Times New Roman, Bold and Right aligned)

Table of Contents

Chapter 1: Introduction

Project Title: Multi-Purpose AI Password Analysis Tool

1.1 Introduction & Background

1.1.1 Introduction

In today's world, where everything is becoming digital and online threats are on the rise, having strong password security is absolutely 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 tool 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.

1.1.2 Background

Traditional password cracking tools have relied on brute force techniques or precomputed dictionaries to decipher passwords. While these methods have been effective to some extent, they suffer from several limitations and challenges.

1. **Limited Efficiency**: Brute force attacks iterate through all possible combinations of characters, making them time-consuming and resource-intensive, especially for complex passwords.
2. **Dependence on Dictionaries**: Dictionary attacks rely on precompiled lists of commonly used passwords or words found in dictionaries. However, these lists may not encompass all possible variations, especially when users employ complex or unique passwords.
3. **Lack of Adaptability**: Traditional tools often struggle with adaptive password generation techniques, such as adding special characters or changing letter cases, making them less effective against modern password practices.
4. **Inability to Detect Breached Credentials**: Many password cracking tools do not have built-in mechanisms to cross-reference passwords with known breaches, leaving users unaware if their passwords have already been compromised.
5. **Scalability Issues**: As passwords become longer and more complex to enhance security, traditional methods face scalability challenges in terms of processing power and memory requirements.

1.2 Opportunity & Stakeholders

1.2.1 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.

1.2.2 Stakeholders

- **Businesses:** Companies aiming 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 Existing System

1.3.1 John the Ripper

Limitations:

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

Problems:

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

1.3.2 Brutus

Limitations:

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

Problems:

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

1.3.3 Wfuzz

Limitations:

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

Problems:

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

1.3.4 Comparison

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

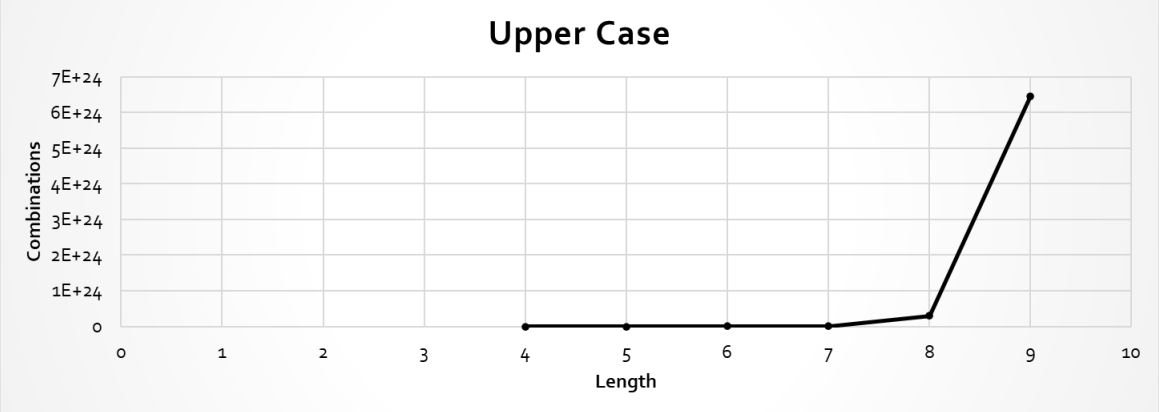
Comparison Table 1.1

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

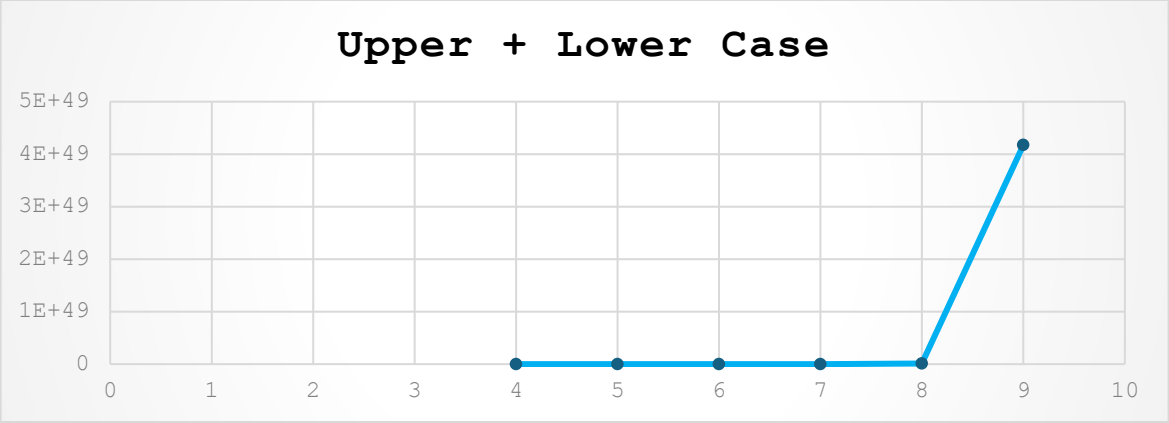
Comparison Table 1.2

1.4 Problem Statement

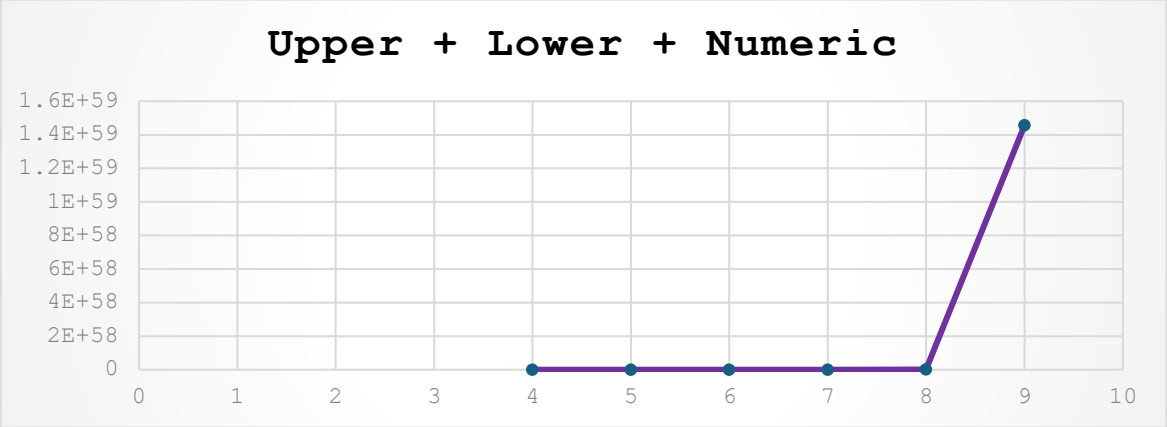
With increasing password complexity requirements, brute-force attacks become significantly slower and less efficient.



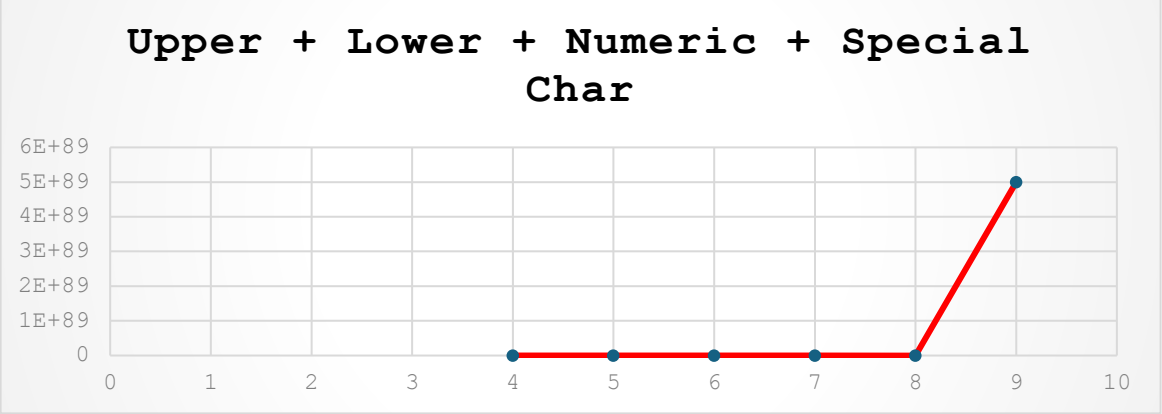
Upper Case Graph 1.1



Upper and Lower Case Graph 1.2



Upper, Lower and Numeric Graph 1.3



Upper, Lower, Numeric and Special Char Graph 1.4

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}	**22204460492503130808472633361816406 25**

$4^{26+26+10}$	**212676479325586539664609129644855132 16**
5^{26+26}	**290981259887315061831530256164353065615 36**
$5^{26+26+10}$	**216840434497100886801490560173988342285156 25**

Password Combination Table 1.2

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

Password Combination Table 1.2

1.5 Proposed Solution

Our AI-powered tool leverages advanced algorithms and Human like passwords data to:

- **Focus cracking efforts on statistically probable password patterns:** This reduces the search space and accelerates the cracking process compared to traditional brute-force methods.
- **Analyze password strength based on complexity metrics and AI-driven pattern recognition.** This identifies potential weaknesses in complex passwords, aiding in targeted cracking attempts.

1.6 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.

1.7 Scope of the Project

- **Defensive capabilities:**

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

  Optional

  - **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.

1.8 Evaluation Plan

Aligned with the objectives of our project, we will conduct a comprehensive assessment of the AI Multipurpose Password Analysis Tool's performance using diverse password datasets sourced from various sources, including real-world breaches, password dictionaries, and synthetic data generation. This dataset will serve as the cornerstone for evaluating the tool's effectiveness in analyzing password strength and facilitating password cracking across different scenarios.

Chapter 2: Develop and Train the AI Model

2.1 Research and select appropriate AI algorithms.

List of AI algorithms suitable for training on a password dataset and generating a dictionary file.

- Markov Models
- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)
- Recurrent Neural Networks (RNNs)
- Transformer Models
- Probabilistic Context-Free Grammars (PCFGs)

2.1.1 Algorithms Overview

2.1.1.1 Markov Models

Markov Models are statistical models used to describe the probabilistic transitions between different states in a sequence of data. In the context of password generation, a Markov

Model can be used to learn the statistical patterns and transitions between characters or character sequences in passwords.

1. **Training Phase**
    a. **Input**
    A dataset of passwords.
    b. **Preprocessing**
    The dataset is preprocessed to extract relevant features, such as character sequences or n-grams (sequences of n characters).
    c. **Model Construction**
    The Markov Model is constructed based on the observed transitions between characters or character sequences in the dataset.
    d. **Transition Probabilities**
    The model calculates transition probabilities between characters or character sequences based on their frequencies in the dataset. For example, the probability of transitioning from the character 'a' to 'b' or from the sequence 'ab' to 'cd' is estimated.

2. **Generation Phase**
    a. **Seed**
    A starting point (seed) is chosen to initiate the generation process. This can be a randomly chosen character or character sequence or a predefined starting point.
    b. **Random Walk**
    The model performs a random walk through the states (characters or character sequences) based on the learned transition probabilities. At each step, it selects the next state probabilistically according to the transition probabilities from the current state.
    c. **Generation Stop Criterion**
    Generation continues until a predefined stopping criterion is met, such as reaching a maximum password length or generating a certain number of passwords

3. **Output**
    The generated passwords are outputted as the result of the Markov Model

2.1.1.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a class of deep learning models consisting of two neural networks, the generator and the discriminator, which are trained simultaneously through a competitive process. GANs have been widely used for generating synthetic data that closely resembles real data distributions. In the context of password generation, GANs can be trained on a dataset of passwords to learn the underlying distribution of passwords and generate new passwords that mimic the characteristics of real passwords.

1. **Training Phase**
    a. **Input**
    A dataset of passwords.
    b. **Generator Network**
    The generator takes random noise as input and learns to generate synthetic passwords. Initially, the generator produces random noise that bears no resemblance to real passwords.
    c. **Discriminator Network**
    The discriminator is trained to distinguish between real passwords from the dataset and fake passwords generated by the generator. It learns to differentiate between real and synthetic passwords.
    d. **Adversarial Training**
    The generator and discriminator are trained simultaneously in a competitive manner. The generator aims to produce synthetic passwords that are indistinguishable from real passwords, while the discriminator aims to correctly classify real and fake passwords.
    e. **Backpropagation**
    The errors from the discriminator are backpropagated to update the weights of both the generator and discriminator networks using techniques such as stochastic gradient descent (SGD) or variants like Adam.

2. **Generation Phase**
    a. **Random Noise Input**
    During the generation phase, the generator takes random noise vectors as input.
    b. **Synthetic Password Generation**
    The generator generates synthetic passwords by transforming the random noise vectors into meaningful password representations.
    c. **Output**
    The generated passwords are outputted as the result of the GAN.

3. **Evaluation**
    The quality of the generated passwords can be evaluated using metrics such as similarity to real passwords, diversity, and entropy. Additionally, human evaluators can assess the usability and security of the generated passwords.

2.1.1.3 Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) are a type of generative model that learns to encode input data into a lower-dimensional latent space and decode it back into the original data distribution. VAEs are trained to capture the underlying structure of the data distribution, allowing them to generate new samples that closely resemble the input data. In the context of password generation, VAEs can be trained on a dataset of passwords to learn the distribution of passwords and generate new passwords that exhibit similar characteristics.

1. **Training Phase**

a. **Input**
   A dataset of passwords.
b. **Encoder Network**
   The encoder takes input passwords and maps them to a lower-dimensional latent space, where each point represents a compressed representation of a password.
c. **Decoder Network**
   The decoder takes points from the latent space and reconstructs them into passwords. The decoder aims to reconstruct passwords that are similar to the input passwords.
d. **Variational Inference**
   VAEs incorporate variational inference, which involves learning a probabilistic distribution over the latent space. During training, the encoder learns to approximate this distribution, while the decoder learns to reconstruct passwords from samples drawn from this distribution..
e. **Reconstruction Loss**
   The VAE is trained to minimize the reconstruction loss, which measures the difference between the original passwords and the reconstructed passwords.
f. **Regularization**
   VAEs also incorporate a regularization term, such as the Kullback-Leibler (KL) divergence, to encourage the learned latent space to follow a prior distribution (e.g., Gaussian distribution).

2. **Generation Phase**
   a. **Sampling**
      During the generation phase, random points are sampled from the latent space.
   b. **Decoder Output**
      The decoder takes these sampled points and generates new passwords by reconstructing them into the password space.
   c. **Output**
      The generated passwords are outputted as the result of the VAE.

3. **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.

2.1.1.4 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.

1. **Training Phase**

a. **Input**
   A dataset of passwords represented as sequences of characters.
b. **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.
c. **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.
d. **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.
e. **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.

2. **Generation Phase**
   a. **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.
   b. **Character-by-Character Generation**
      The 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.
   c. **Output**
      The generated characters are concatenated to form a new password sequence.

3. **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.

2.1.1.5 Transformer Models

Transformer models are a type of neural network architecture that has gained significant popularity in natural language processing tasks due to its ability to capture long-range dependencies in sequential data efficiently. Transformer models, such as OpenAI's GPT (Generative Pre-trained Transformer) and Google's BERT (Bidirectional Encoder

Representations from Transformers), are powerful language models capable of generating coherent text based on input sequences. In the context of password generation, transformer models can be fine-tuned on a dataset of passwords to learn the distribution of passwords and generate new passwords that exhibit similar characteristics.

1. **Training Phase**
    a. **Input**
    A dataset of passwords represented as sequences of characters.
    b. **Architecture**
    The transformer model consists of encoder and decoder layers, each composed of self-attention mechanisms and feed-forward neural networks. The encoder processes the input sequence, while the decoder generates the output sequence.
    c. **Pre-Training**
    The transformer model is pre-trained on a large corpus of text data using unsupervised learning objectives, such as language modeling or masked language modeling. This pre-training step allows the model to learn general language patterns and representations.
    d. **Fine-Tuning**
    The pre-trained transformer model is fine-tuned on the password dataset using supervised learning. During fine-tuning, the model learns to generate passwords by minimizing a loss function that measures the difference between the predicted and actual passwords in the dataset.
    e. **Tokenization**
    The input passwords are tokenized into subword or character-level tokens to represent them as numerical inputs to the transformer model.

2. **Generation Phase**
    a. **Seed**
    During the generation phase, a seed sequence is provided as input to the transformer model to initiate the generation process. This seed sequence can be randomly chosen or predefined.
    b. **Autoregressive Generation**
    The transformer model generates new characters or tokens autoregressively, meaning that it predicts each token based on the previously generated tokens. At each step, the model outputs a probability distribution over the vocabulary of characters or tokens, from which the next token is sampled.
    c. **Output**
    The generated characters or tokens are concatenated to form a new password sequence.

3. **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.

2.1.1.6 Probabilistic Context-Free Grammars (PCFGs)

Probabilistic Context-Free Grammars (PCFGs) are formal grammars used to model the hierarchical structure of sequences, where each production rule is associated with a probability indicating its likelihood of being applied. PCFGs are commonly used in natural language processing and computational linguistics for tasks such as syntax parsing and language generation. In the context of password generation, PCFGs can be used to model the structural patterns and dependencies present in passwords and generate new passwords that adhere to these patterns.

1. **Training Phase**
    a. **Input**
    A dataset of passwords represented as sequences of characters.
    b. **Grammar Construction**
    From the dataset, a PCFG is constructed by identifying common structural patterns and dependencies present in the passwords. Each production rule in the grammar represents a possible transformation or combination of characters.
    c. **Rule Probabilities**
    Each production rule in the PCFG is associated with a probability indicating its likelihood of being applied. These probabilities are estimated based on their frequencies in the training dataset.
    d. **Smoothing**
    To handle unseen or rare patterns, smoothing techniques may be applied to adjust the probabilities of production rules.

2. **Generation Phase**
    a. **Seed**
    During the generation phase, a seed symbol or starting point is provided as input to the PCFG to initiate the generation process. This seed symbol could represent a starting character or character sequence.
    b. **Probabilistic Expansion**
    The PCFG probabilistically expands the seed symbol into a sequence of characters by recursively applying production rules according to their probabilities. At each step, a production rule is chosen based on its probability, and the corresponding symbols are generated.
    c. **Output**
    The generated characters are concatenated to form a new password sequence.

3. **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.

2.2 Acquire or prepare relevant password datasets for training

Getting a dataset ready for password generation involves a series of steps aimed at making sure the data accurately represents different password characteristics and is suitable for training. Here's a simplified breakdown of what we need to do:

2.2.1 Data Collection

Get a mix of passwords from various sources, like:

- Online leaks and breaches (follow ethical and legal guidelines).
- Making sure our dataset covers a wide range of password types, including different lengths, complexities, and compositions.

2.2.2 Data Cleaning

Ensure the data doesn't contain any sensitive or personally identifiable information.
Remove duplicate passwords to keep the dataset clean and manageable.
Get rid of any weird or corrupted entries that could mess up your training.

2.2.3 Data Preprocessing

Get all the passwords formatted the same way (like making everything lowercase).
Break the passwords down into individual characters or sequences depending on how we are going to train our model.

- Making sure our dataset is balanced if certain types of passwords are overrepresented.

2.2.4 Splitting the Dataset

Divide our dataset into three parts: training, validation, and test sets.
Train our model on the training set, tweak its settings using the validation set, and then check how well it's doing with the test set.

2.2.5 Data Augmentation (Optional)

Add more variety to our dataset by making small changes to existing passwords.
We can switch out characters, add or remove some, or even create completely new passwords based on certain rules.

2.2.6 Encoding Passwords

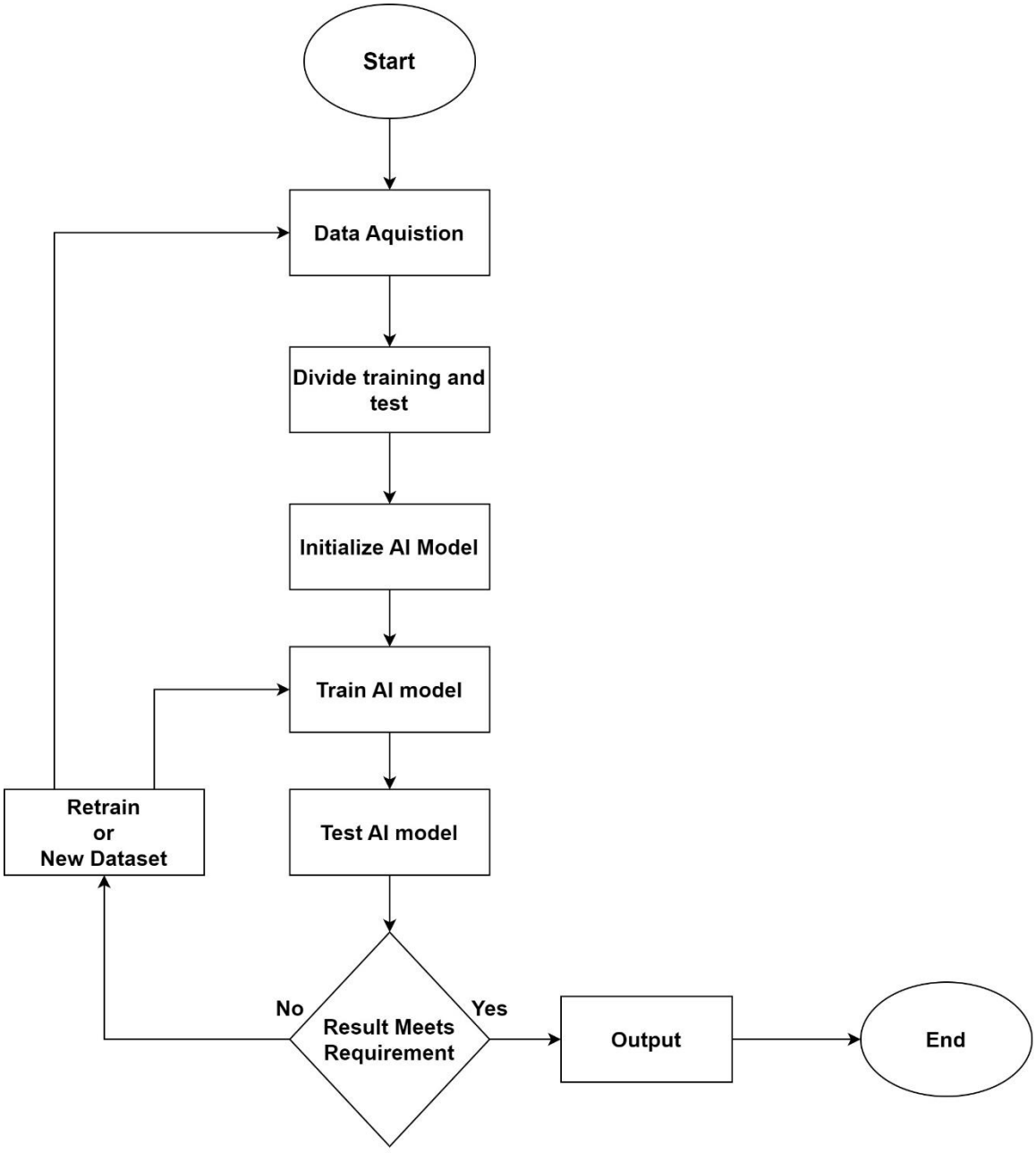
Change our passwords into numbers so our model can understand them better.
If we are working with characters, we might assign each one a different number or use a one-hot encoding.
For other types of models, we need to tokenize your passwords into smaller parts.

2.2.7 Save the Preprocessed Dataset

Once our data is all prepped and ready, save it in a format that works well for training models, like CSV or JSON.

Include all the info about our dataset so others can use it too, and to make it easier for us to come back to later.

2.3 Train and optimize the AI model with the selected datasets.



Train and optimize the AI models

Training AI Model Flow Chart

Chapter 3: Implement Core Functionalities

3.1 Design and develop functionalities for password strength analysis.

3.2 Design and develop functionalities for password strength analysis.

3.3 Integrate password breach checking with online databases (securely).

3.4 (Optional) Develop functionalities for automated password cracking (ethical considerations apply).

Chapter 4: Design and Develop User Interface

4.1 Design a user-friendly interface for both offensive (optional) and defensive functionalities.

4.2 Integrate the trained AI model and core functionalities into the user interface.

Chapter 5: Testing and Refinement

5.1 Conduct thorough testing of all functionalities with various password scenarios.

5.2 Refine the AI model, user interface, and functionalities based on testing results.

5.3 Document user manuals and instructions for the completed tool.

References

References:

- . Hitaj, Briland et al. “PassGAN: A Deep Learning Approach for Password Guessing.” *International Conference on Applied Cryptography and Network Security* (2017).

- Hitaj, Briland, et al. "Passgan: A deep learning approach for password guessing." *Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17*. Springer International Publishing, 2019.