eISSN 2783-6851

VILNIUS TECH
Vilnius Gediminas Technical University

NEW TRENDS in COMPUTER SCIENCES

2023
Volume 1
Issue 1
Pages 1–18
https://doi.org/10.3846/ntcs.2023.17639

MULTI-PURPOSE PASSWORD DATASET GENERATION AND ITS APPLICATION IN DECISION MAKING FOR PASSWORD CRACKING THROUGH MACHINE LEARNING

Mark VAINER ✉

Vilnius Gediminas Technical University, Vilnius, Lithuania[♦]

Article History:
- received 20 September 2022
- accepted 23 February 2023

Abstract. This article proposes a method for multi-purpose password dataset generation suitable for use in further machine learning and other research related, directly or indirectly, to passwords. Currently, password datasets are not suitable for machine learning or decision-driven password cracking. Most password datasets are just any old password dictionaries that contain only leaked and common passwords and no other information. Other password datasets are small and include only weak passwords that have previously been leaked. The literature is rich in terms of methods used for password cracking based on password datasets. Those methods are mainly focused on generating more password candidates like the ones included in the training dataset. The proposed method exploits statistical analysis of leaked passwords and randomness to ensure diversity in the dataset. An experiment with the generated dataset has shown significant improvement in time when performing dictionary attack but not when performing brute-force attack.

Keywords: passwords, password cracking, password dataset, password strength, machine learning.

✉Corresponding author. E-mail: ***mark.vainer@vilniustech.lt***

Introduction

Password is the most common and widely used method of authentication. This is a memorized secret composed of a sequence of characters which can be lowercase letters, uppercase letters, digits and/or special symbols. According to the most common password lists of the recent years, passwords are short, can be easily guessed or brute-forced very quickly. Those passwords are extremely limited in the variety of character types used. Passwords with such characteristics are called weak passwords. In contrast, strong passwords are more secure due to their length and inclusion of wide range of character types. Those two factors are the reason why password cracking can be difficult and time consuming.

Many researchers suggest generating even more password candidates using machine learning and statistical models with raw password dictionaries as the training dataset, Weir et al. (2009) proposed one such method based on probabilistic context-free grammars. Other methods exploit the patterns of common leaked passwords (Deng et al., 2019; Devi & Arumugam, 2019; Li et al., 2019; Tatli, 2015).

Most of those methods are quite effective with high success rates. For example, probabilistic context-free grammars (Weir et al., 2009) cracked 28 to 129 percent more passwords than the built-in ruleset utilized by John the Ripper. However, there is a cost of processing

[♦]This paper was written based on the author's master thesis which was written while being a student at this university.

time when using bigger dictionaries for dictionary attack. The bigger the dictionary the longer it takes to crack a password.

When having some knowledge about the target password, it is possible to crack it a little bit faster. The knowledge can be gained through knowing the person or through social engineering techniques, for example shoulder surfing. In some cases, shoulder surfing can supply all the information and in other cases it can provide very little due to some limitations.

Machine Learning (ML) can help to close the gap between what we know about the password and the actual password and help in decision making during the password cracking process. For the machine to learn effectively a dataset is needed. The current state of password datasets is not suitable for learning by machines and for password cracking driven by decision making. Currently, majority of password datasets are just any old password dictionary containing only leaked and common passwords and no other information is given, RockYou (Bowes, 2008) is one such example. Other password datasets are small, based on passwords which were leaked in the past and are weak, it is possible to find such datasets in Kaggle.

In this paper, a method for multi-purpose password dataset generation suitable for machine learning use and other applications is proposed and evaluated for decision making in password cracking process. This is to raise awareness on the security of passwords, how they are typed in, and the physical environment around us when the password is typed in.

The structure of the paper is as follows: In Section 1, a literature review is provided, and classic and data-driven password cracking methods are discussed, Section 2 presents a review of existing password datasets, Section 3 presents the proposed method, in Section 4 the results of the experimental investigation are presented and finally, conclusions are provided in the last Section.

1. Related works

1.1. About passwords

According to NIST a password is a memorized secret used for authentication of a user to access information in a computer system (Grassi et al., 2017). This memorized secret is a sequence of characters. It is recommended that the password will contain variation of characters like numbers, letters (lowercase and uppercase) and other special symbols on the keyboard.

It is not clear when and who invented the passwords, but they have been used since ancient times. The first usage of computer passwords was in the mid-1960s when researchers at MIT built a massive time-sharing computer called CTSS. This computer allowed multiple users to access a computer at the same time, but this also introduced a problem which allowed users to interrupt each other's work. Fernando Corbato, the head of the CTSS project, suggested to isolate each account by assigning a password to each individual user which kept their files and programs hidden from each other but since the MIT researchers did not care about security, passwords were insecure (McMillan, 2012) and this has not changed since then, even today people create passwords which are easy for hackers to figure out and to easily crack. Most passwords include words from a dictionary, meaningful patterns like dates and names of family members or pets (NordPass, 2021).

Ur et al. (2015) conducted an experiment which showed how human-made passwords are poorly created. The participants of the experiment were asked to create passwords for three different fake websites that were developed for the purpose of the experiment, The three websites were a news website, a banking website, and an email website. The results show that most participants have procedures to create their passwords. While some of these procedures consistently provide secure passwords, others frequently produce passwords that are easy to guess. These procedures followed similar structures to other passwords created for other websites which makes it easy to guess as knowing the procedure and the person can be significant for hackers.

Passwords are the most used way of authentication. They provide the first line of defense against unauthorized access to your computer, web services or other personal information. The stronger the password is, the more protected your computer will be from hackers and malicious software. But people usually do not follow these requirements as they want a password which is easy to remember, but most of the time it is a very weak password. They simply do not know how to create a strong password. This makes the passwords less secure and hence, less important. This leads many information security specialists to believe in the idea of "password(s) are/is dead". This idea means that passwords are no longer good for securing information in computer systems, and it is time to look for alternatives like biometrics, one-time passwords, visual passwords, and many others (Bachmann, 2014) but even though passwords are not ideal, they are still far more practical than the alternatives (Potter, 2005). Even nowadays when more people from the IT security industry talk about passwords alternatives it will take time until a new authentication method will be adapted.

1.2. Password cracking and classic password cracking methods

Password cracking is the general procedure of recovering the original password from an unreadable form called hash value. The hash value does not give any information about the original password, and it is generated using a hash function. The most common hash functions are Message Digest 5 (MD5), and Secure Hashing Algorithm (SHA) 1 and 2.

The traditional way to retrieve the password from the hash is by hashing different password guesses and comparing the resulting hash with the target hash, if the hash matches then the password is cracked. Otherwise, the password remains unknown, it is also possible to apply online password guessing to find the password. Classic password cracking includes three methods, those are the brute-force attack, the dictionary attack, and the rainbow table attack.

Brute-Force Attack. In the context of password cracking, brute-force is a technique to crack the password by testing all the combinations of possible characters. The right password will eventually be discovered by brute-force attack, but the time required is unpredictable. It is impossible to search through every element in an input space that is extremely large but for short passwords this method is highly effective (Kaspersky, n.d.; Yu & Huang, 2015).

To make it harder to perform a brute-force attack passwords must be longer and must contain a variety of characters. A long password and high characters diversity in a password leads to a larger search space which requires a lot of computing power and a lot of time.

Dictionary Attack. People tend to use normal words as their passwords to avoid remembering complex passwords. This makes dictionary attack a highly effective approach. The idea is to use a wordlist, a list of common passwords and dictionary words to gain access to a computer system by comparing the hash of every password candidate in the wordlist with the target hash (Yu & Huang, 2015). It is also possible to apply some word mangling rules to the dictionary to generate more password combinations to test.

Rainbow Table Attack. A rainbow table is a precomputed table for caching the output of cryptographic hash functions, usually for cracking password hashes. It is a practical example of a space-time trade-off, using less computer processing time and more storage which is significantly lower compared to brute-force attack which would require 4TB storage space when trying to crack an 8-character long password containing lowercase letters and hashed with MD5 hashing algorithm (Yu & Huang, 2015). Rainbow table attack was proposed by Oechslin (2003) which is an improvement of the method proposed by Hellman (1980).

By using a large amount of memory, rainbow tables allow to reduce the difficulty of cracking a password. The passwords are arranged in chains, and only the first and last elements of each chain are stored in a rainbow table structure. Hashing and reduction operations are chained together to create rainbow tables. When cracking a password, the password hash is passed through the chain link by link and compared to the hashes that have been previously stored. Rainbow tables then retrieves the plaintext that generated the hash if there is a match.

1.3. Data-driven password cracking methods

According to the scientific literature, several methods are proposed for enhanced password cracking which are based on datasets of leaked passwords. One such method is called probabilistic context-free grammars (Weir et al., 2009) and is viewed as an extension for dictionary attacks which uses existing dictionary to automatically generate word-mangling rules and then using these rules and the dictionary to further derive password guesses in decreasing probability order, this is done to maximize the likelihood of cracking the target passwords in a minimal number of guesses. The process of generating the password guesses is very similar to the way grammar is generated in a classic probabilistic context-free grammars. The classic context-free grammars (Sipser, 2012, p. 104) are formally defined as a tuple with 4 elements: $G = (V, \Sigma, R, S)$. In this definition, V denotes the set of variables, Σ denotes the set of terminal symbols which are unchangeable, R is the set of production rules and, finally, S is a symbol from V which is the starting symbol. In the specific case of the application of probabilistic context-free grammars in password cracking, V contains the variables: S, L_n, D_n, S_n, where S is the starting symbol and L_n, D_n, S_n are respectively a sequence of letters, digits, and special symbols of length n for $n > 0$. Σ contains all possible characters that can be included in a password. Each production rule is assigned a probability based on distribution of different sequences of characters in a dictionary. The conducted experiments with probabilistic context-free grammars showed an improvement in success rate of 28% to 129% more passwords compared to John the Ripper.

Devi and Arumugam (2019) utilized conjunctive grammars which are a generalization of context-free grammars which allow the use of an explicit intersection operator in the production rules to generate password guesses. In general, the process of identifying the base structure of each password in the training dataset and the association of probabilities to each production rule is the same as in the probabilistic context-free grammar approach. The author mentions that the performance of the password cracker based on conjunctive grammars is better than the one based on the context-free grammar, but he does not mention numerical results.

Deng et al. (2019) proposed a different method based on the concept of word segmentation from natural language processing to teach a machine learning model about the structure of passwords and generate the passwords list based on those structures. In this approach, the passwords in the training dataset are broken down into segments using a password segmentation model. Then, a model based on n-grams and Markov chain approach is used to learn about the labelled segments. While guessing the passwords, the model predicts the next segment based on previous segments and the learned password structure and concatenates those segments together to form a password guess. The password segmentation model is based on Conditional Random Field (CRF) approach which is proven to be one of the most accurate methods in word segmentation. The suggested password segmentation model uses the following methodology to produce password segments for each training password. The suggested method divides a given character string into numerous words, each of which could be regarded as a segment of the string, by labeling each character as the "beginning", "middle", or "end" of a word and based on those labels it can segment the password to words which are considered a segment of the password. Each segment is labeled as one the entity labels: name, date, common words, common passwords, letter patterns and mixed-characters patterns. After the labelling, an N-gram Markov model is used to learn about the password structure based on the entity labels and then the passwords are generated for password guessing. It is obvious that the passwords generated by this method are only effective for human generated passwords, it can guess the passwords with better results compared to other methods. For more complex passwords, the attack might not be as effective.

Another method proposed by Li et al. (2019) suggest guessing passwords through big data. The password guessing method is broken into two phases: In the first phase, a password guessing attack dictionary composed of popular and high frequency passwords is created.

The password guessing utilizing personal information passwords is performed in the second phase. As part of an experiment, the authors performed a proactive password check based on the password requirement in the China's train ticketing 12306 website: the length of the password string is 6–20 digits, and the optional characters include two or more of letters, numerals, and underline. This requirement corresponds to approximately 31 bits of entropy which was used as a threshold value. A dataset of 1.4 billion passwords which was sorted by frequency and combined with common passwords lists was cleaned from duplicates and passwords which did not meet the proactive check requirement, those which are less than 31 bits of entropy, and formed different attack dictionaries of different sizes which were used to guess the passwords in the Chinese ticketing websites leaked password dataset. The results show that the number of successfully guessed passwords increases as the password count in attack dictionary increases. The success rate of this approach for the attack dictionary of size 100 was 4.84% and with the largest one of 1000 it was 6.05%. The success rates are not very high, and they will only get higher with an even larger dictionary, which is a desirable goal but will require more time as the dictionary gets bigger.

Tatli (2015) went a few steps further for increasing the success rate of password guessing. The author's method is based on analyzing leaked real-life user passwords and identification of common patterns which are used by many users to create a complex and strong password from a dictionary word. After analyzing several different patterns, a pattern-based password dictionary consisting of 2.3 billion passwords was generated and used as an input dictionary for dictionary attack. With the pattern-based dictionary, the author could crack 150% more passwords compared to the RockYou dictionary which could crack only 18,500 password hashes. It is important to note that success rate and dictionary size are often strongly correlated but dictionary size is also strongly correlated with cracking time.

Another popular password guessing method is based on Generative Adversarial Network (GAN) and is called PassGAN (Hitaj et al., 2017). This is a deep learning model capable of generating password guesses which are of the same style as the passwords it was trained on. When trained on a subset of the RockYou dictionary, PassGAN was able to match 34.2% of the passwords in the testing set. Additionally, when PassGAN was trained on the RockYou password dictionary, it was possible to match 21.9% of the passwords in the LinkedIn dictionary.

All the proposed methods in this literature review present good results in terms of performance and success rates but achieving a high success rate takes a significant amount of time which is strongly related to the size of the dictionary being generated. The password cracking process can and should be more focused and target specific, this way the attack can be performed in less time.

2. Review of existing password datasets

Password datasets can be in one of two forms, single-column datasets, also known as dictionaries, or multi-column datasets containing additional information about the passwords. In the previous chapter, the exclusive type of dataset which was used throughout all the data-driven password cracking methods is the single-column dataset.

A common and well-known dataset of the single-column dataset type is called RockYou dictionary, and it is widely used in dictionary attacks and analysis. RockYou is a rather large dictionary containing 14 341 564 unique passwords which were used in 32 603 388 accounts. The passwords were leaked by exploiting a 10-year-old SQL vulnerability while the passwords were stored in the database as plaintext.

Another example is the dictionary containing leaked passwords from Myspace data breach which became publicly known in May 2016. It is not clear when the breach happened, but it was estimated to happen around 2008 or 2009 and it affected almost 360 million Myspace users of which 37 144 unique passwords are included in the dictionary.

Dictionaries are not necessarily have to be originated from a specific data breach, there are also general dictionaries containing common passwords and words in different languages, but they are all essentially just a simple, potentially long list of passwords with no additional information or features and thus wrongly referred to as a dataset.

The second type of datasets, the multi-column datasets, is less common but a few examples still exist and can be found mainly in dataset repositories like Kaggle. The first dataset of this category that could be found is called *Password Strength Classifier Dataset* (B. Bansal, 2019) and is based on a data breach in 000webhost. The dataset itself consist of 669643 unique passwords and two columns, the password in plaintext and a strength value between 0 and 2 – 0 for weak, 1 for medium and 2 for strong. Obviously, this dataset is made specifically for password strength classification and because the passwords originate from previous leaks, they are weak already.

Another multi-column dataset is called *10000 Most Common Passwords* (S. Bansal, 2021). The dataset consists of 9 998 unique passwords and nine columns which include the password in plaintext, length, number of symbols, uppercase and lowercase letters, digits, vowels, and syllables. This dataset is an improvement from the *Password Strength Classifier Dataset* in terms of inclusion of useful features, but it is still based on common and previously leaked passwords.

At this point, there are no password datasets which are suitable for data-driven password cracking or for a password cracking process which is based on machine learning. Existing datasets are not sufficient and are only helpful for cracking specific passwords. Those other passwords are longer, more complex and require more time for cracking but with some information on those passwords we could crack a wider range of passwords than we could before.

3. The proposed method for multi-purpose password dataset generation

The proposed method is based on randomness and statistical analysis of previously leaked passwords. Although randomness might not seem to be the best choice for password dataset generation it allows for better diversity and more flexibility in the dataset which are the main requirements for the proposed method. The proposed method itself for password dataset generation is a two-step process:

Step 1. Generate large collection of passwords.

Step 2. Calculate features for each password.

3.1. Password collection generation

The passwords in the dataset were generated randomly according to four different methods which allow for strong passwords as well as weak passwords that look like humans created them. In addition, common passwords were added to the dataset as well for increased usability. All approaches for password collection generation were implemented with python 2 in a Jupyter notebook environment. In addition, pandas library was used for combining all passwords and their corresponding features in one CSV file.

Random Password Collection. The first method uses a random number generator to generate passwords of various lengths with characters chosen independently of each other at random.

Pool-Based Password Collection. The second method is a variation of the first method with small difference. Only the most common characters are used and can be included in a password.

Position-Based Password Collection. The third method assumes that a specific character is more common in one position than in another position. At each step of the generation process, a character is generated from a distribution of characters for a specific position in the password.

Next-Char Password Collection. The fourth, and final, method is like the previous method but, unlike the previous method, in this one each character starting from the second character is generated from a distribution of characters that includes characters with higher probability of appearance after the previous character.

3.2. Features

The dataset contains sixteen features, five qualitative (categorical) features and eleven quantitative (numeric) features describing each password. Each feature (or combination of features) has an intended purpose.

Similar Chars. The similar chars feature presents characters which are like the ones used in the password itself. The characters listed under this feature are similar in the sense that they can be substituted according to the L33t substitution rules (Craenen, n.d.). This feature is helpful for generation of other passwords.

Hash. The hash feature is simply the SHA-256 hashing algorithm applied to the generated password. This feature is intended to be used for cracking passwords. In other words, when attempting to crack a password, the dataset can be used as a lookup table for the hash to find the correct password. SHA-256 was chosen as it is the most popular and widely used hashing algorithm.

Mask. The mask feature is a string representing the general structure of the password without considering specific characters. Namely, each character is replaced with one of the following letters: 'L','U','D' or 'S' for lowercase, uppercase, digit, or symbol, respectively. For example, the password "Passw()Rd1" will result in "ULLLLSSULD". The mask was then simplified in such a way that every substring of repeated letter is replaced with one. The above-mentioned mask will, after simplification, become "ULSULD". This process was performed mainly to avoid an unnecessary increase in the file size as well as to eliminate information which is already presented as a different feature. This feature is useful in cases when we know the mask (or structure) of the password but do not want to go through all the possibilities of each character in the charset like it is done in John the Ripper or Hashcat. We can simply query the dataset for passwords that match the specific structure and compare their hash with the known hash.

Password Scheme. The password scheme is a description of the type of characters used in the password. This is a categorical feature with 4 different possible options: Alpha, Numeric, Alphanumeric and Non-Alphanumeric. This feature is another feature that can be predicted based on other features and used for reducing the search space of possible password candidates when performing dictionary attack or brute-force attack. This feature is like the mask feature in terms of meaning but in some cases can be used easier than the mask feature, for example when making prediction password scheme is easier as it has fewer categories compared to the mask which contains numerous.

Password Entropy. Password entropy is one of the heuristic-based approaches that can be used for password strength estimation. It is defined as the measure of the difficulty in guessing or determining a password. The idea of password entropy is based on works by Claude Shannon (Shannon, 1948) who defined the term entropy in information theory as a measure of a message's information content, which is a measure of the message's ability to reduce uncertainty. The original Shannon entropy formula is expressed mathematically in Equation (1) and is written in terms of the probability of a random variable *X*.

$$H\left(X\right) = -\sum_{i=1}^{n} P\left(x_i\right)\log_2 P\left(x_i\right). \tag{1}$$

Password entropy is a specific application of Shannon entropy when considering the set of all passwords of length *L* with elements taken from the set of *R* possible characters, there are obviously R^L possible such passwords each equally likely with probability of $\frac{1}{R^L}$ (Szczepanek, 2021). When substituting these values in the formula for Shannon entropy in Equation (1) we can quickly derive the formula for password entropy in Equation (2).

$$E = \log_2\left(R^L\right), \tag{2}$$

where: *R* is the size of the pool of unique characters from which the password was created; *L* is the password length, the number of characters in the password.

Strength. The strength is the class (or label) feature which is the feature to be predicted. This is a categorical variable which can be one of the five following strength levels: Very Strong, Strong, Medium, Weak or Very Weak. Each password is assigned a strength level based on the password entropy calculation. Password entropy is considered by many as a good estimation of password strength, and it indicates the number of guesses an attacker will have to make to guess the password correctly, it is measured in terms of number of bits. The entropy value is not easily understood and can be misinterpreted by an ordinary person. To solve this problem, the entropy values are split into five ranges and each of them is assigned a meaningful label representing the strength. The password entropy ranges, and their corresponding strength level are presented in Table 1. This scheme of mapping entropy to strength was used by a few people.

Table 1. Strength level for each entropy range (source: Pleacher, n.d.)

Password Entropy Range	Strength Level
$E < 28$	Very Weak
$28 \le E \le 35$	Weak
$36 \le E \le 59$	Medium
$60 \le E \le 127$	Strong
$E \ge 128$	Very Strong

Estimated Crack Time (log). The estimated crack time (log) is calculated as the log of the time required to crack the password in seconds. The actual time is calculated based on the formula presented in the paper by Kim et al. (2021) and is expressed as:

$$T = 2^{E-1} \times 0.0001, \tag{3}$$

where E is the entropy of the password. The number 0.0001 represents the number of seconds per one attempt and depends on the attacker's computational power.

Charset Size. The charset size feature is a numerical variable representing the size of the set of characters used for the creation of the password. Each password can be composed of characters from 4 different sets: lowercase (containing 26 letters), uppercase (containing 26 letters), digits (containing 10 digits) and special symbols (containing 33 non-alphanumeric symbols). The special symbols category could contain more symbols from Unicode but those are less common in passwords creation. To compute the charset size, first, it is required to identify which character type (charset) out of the 4 mentioned above is used in the password. Then the sizes of charsets are summed up. The resulting value is the charset size. Charset size can be understood as a measure of the password complexity and can be used as an additional feature for password strength prediction.

Total Repeats. The total repeats feature is a numerical variable indicating the number of total character repetitions in the password. A big number indicates that the password has repeated characters and hence can be cracked faster by reusing a certain character more than once.

Password Length. The password length feature corresponds to the number of characters in the password. This is one of the most important features in the dataset as the longer the password the stronger it is.

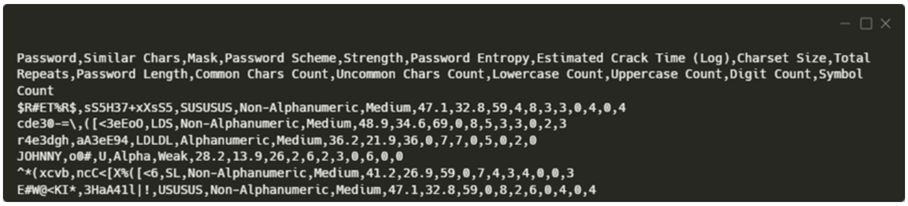
Common and Uncommon Characters. This pair of features represents the counts of the common and uncommon characters. A character is common if it appears in the top half of a character frequency table, sorted from largest to smallest. This feature is helpful because if a password has many common characters, an attacker can concentrate on those passwords in a dictionary with more common characters.

Counts of Lowercase, Uppercase, Digits and Special Symbols. The counts of several types of characters are also an important aspect of password strength. Inclusion of characters other than lowercase and digits makes it possible to achieve a stronger and more secure password.

3.3. Password dataset view and size

The password dataset was generated according to the proposed method in four different versions: small with 12 781 instances, medium with 999 471 instances, large with 20 888 563 instances, and huge containing 60 million instances. Each version weighs 1.75 MB, 183 MB, 3.79 GB and 10 GB respectively when stored on disk. The decision to generate four versions of the dataset comes from the understanding that working with big dataset for quick analysis will require more time and computational memory but in other cases a bigger dataset will be more effective and will provide more statistical power. Figure 1 shows the main features of the dataset and a few instances.

The diversity requirement was presented in chapter three. This requirement means that the dataset should contain variety of different types of passwords, this includes different



```
Password,Similar Chars,Mask,Password Scheme,Strength,Password Entropy,Estimated Crack Time (Log),Charset Size,Total
Repeats,Password Length,Common Chars Count,Uncommon Chars Count,Lowercase Count,Uppercase Count,Digit Count,Symbol
Count
$R#ET%R$,sS5H37+xXsS5,SUSUSUS,Non-Alphanumeric,Medium,47.1,32.8,59,4,8,3,3,0,4,0,4
cde30-=\,([<3eEoO,LDS,Non-Alphanumeric,Medium,48.9,34.6,69,0,8,5,3,3,0,2,3
r4e3dgh,aA3eE94,LDLDL,Alphanumeric,Medium,36.2,21.9,36,0,7,7,0,5,0,2,0
JOHNNY,o0#,U,Alpha,Weak,28.2,13.9,26,2,6,2,3,0,6,0,0
^*(xcvb,ncC<[X%([<6,SL,Non-Alphanumeric,Medium,41.2,26.9,59,0,7,4,3,4,0,0,3
E#W@<KI*,3HaA41l|!,USUSUS,Non-Alphanumeric,Medium,47.1,32.8,59,0,8,2,6,0,4,0,4
```

Figure 1. A look into the password dataset CSV file (compiled by the author)

strength levels, different character types and different lengths. In this regards, Table 2 presents the distribution of strength levels in each version of the dataset.

Table 2. Distribution of strength levels in the four versions of the password dataset (compiled by the author)

Strength	Small Dataset	Medium Dataset	Large Dataset	Huge Dataset
Very Weak	965	62 017	837 100	668 691
Weak	1276	32 105	668 430	3 559 635
Medium	9984	125 962	2 415 902	19 536 379
Strong	261	291 163	6 106 492	14 309 398
Very Strong	295	488 224	1 0860 638	21 925 897
Total	12 781	999 471	20 888 563	60 000 000

4. Experimental investigation

For the experimental investigation, a password dataset was generated according to the proposed method presented in chapter three. The dataset was used to build a classification and regression models for password strength evaluation and prediction of password length, the results of which were used for smarter and more target-specific password cracking.

Machine learning (ML) models were trained on known password features to predict additional features which are helpful when performing password cracking with a password cracking tool like John the Ripper or Hashcat. The experimental investigation was conducted in two phases. In the first phase, common classification and regression models were chosen and used for training. Each model was evaluated and the best one was chosen for use in making prediction for use in the second phase where the prediction was used for dictionary attack and brute-force attack the results were compared to cracking passwords normally (see Figure 2).

For building and training models, Scikit-learn 0.24.2 Python library was used.

It includes various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and more. The parameters that were used for training the machine learning models during experiments were mostly the default parameters.

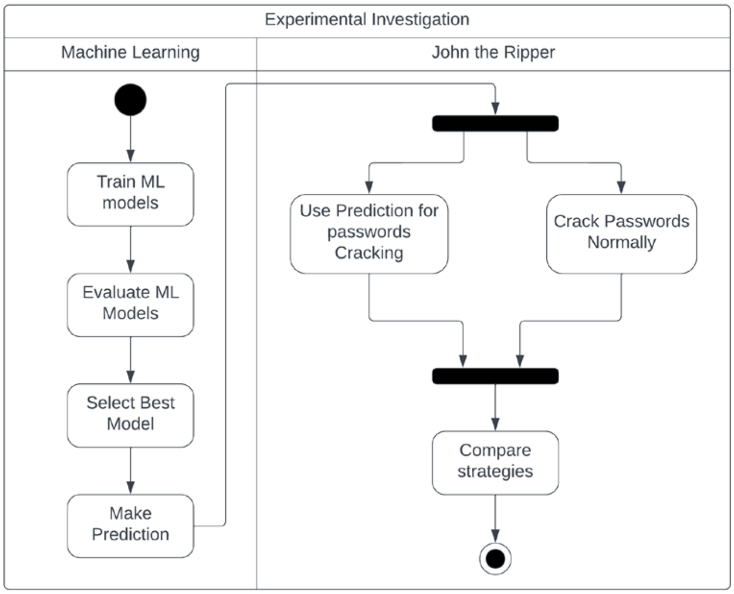


Figure 2. Schematic representation of the experimental investigation (compiled by the author)

4.1. Password strength classification

In machine learning, the most common classification algorithms include logistic regression, support vector machines (SVM), decision trees, multilayer perceptron, and *k*-nearest neighbors (*k*-NN). For this reason, those are the models that were chosen for the experiments. For evaluation purposes, the dataset was split 20% for testing and 80% for training the model. Each model was trained on 80% of the data and afterwards tested with the remaining data. The output of each trained model is one of 5 possible strength labels – very weak, weak, medium, strong, and very strong. Each trained model was evaluated by calculating its accuracy using the following formula (scikit-learn, n.d.):

$$accuracy = \frac{1}{n}\sum_{i=1}^{n} 1(\hat{y}_i = y_i), \qquad (4)$$

where: n is the size of the testing dataset; y_i and $\hat{y}_i$ are the true and predicted value respectively; $1(\cdot)$ is the indicator function which returns 1 if the true and predicted values are equal, otherwise it returns 0.

For the classification of password strength, the most relevant features were selected, those features are password length, lowercase count, uppercase count, digit count and symbol count. Table 3 presents the accuracy score of each classification model when it is provided with the generated dataset.

Table 3. Accuracy of strength classification models (compiled by the author)

Model Name	Accuracy
Logistic Regression	92.18%
Support Vector Machine (SVM)	96.56%
Decision Tree	100%
Multilayer Perceptron (MLP)	99.65%
k-Nearest Neighbors (*k*-NN)	99.22%

All the popular classification models provide excellent performance with accuracy above 90%. The decision tree model is a highly performing model with accuracy of 100%. The models were also cross-validated to check if a model can be generalized to yet-unseen data. Table 4 shows the worst, the best and the average score of each iteration of *k*-fold cross validation for $k = 10$.

Table 4. Minimum, maximum, and average scores for strength classification models (compiled by the author)

Model Name	Minimum Score	Maximum Score	Average Score
Logistic Regression	90.46%	93.2%	92.18%
Support Vector Machine (SVM)	95.47%	97.5%	96.81%
Decision Tree	99.76%	100%	99.86%
Multilayer Perceptron (MLP)	99.3%	99.84%	99.56%
k-Nearest Neighbors (*k*-NN)	99.06%	99.69%	99.39%

Based on Table 3 and Table 4, decision tree was the best model for strength classification and hence was chosen for the next phase of the experimental investigation.

In the next step of the experiment, five passwords were selected from RockYou dictionary each from a different strength level, according to the password strength classification model. The same model was used to split the RockYou dictionary into five separate dictionaries each containing passwords that belong to the same strength level. Those lists and the full RockYou dictionary were provided one by one to John the Ripper to crack the five different passwords which were hashed with four different hashing algorithms: SHA-256, MD5, bcrypt and argon2. A comparison between a dictionary attack performed with a full RockYou dictionary and with its five strength subsets is shown in Table 5. The five strength subsets are denoted as VW, W, M, S, VS for very weak, weak, medium, strong, and very strong respectively.

The experiment shows that a strength-oriented dictionary attack powered by ML helps in reducing the time it takes to crack a password using a dictionary attack especially when the hash comes from a slow hashing algorithm. For a medium strength password there is no significant difference between using the full dictionary and a subset of medium strength passwords, but this could depend on the dictionary size and the strength of passwords included in it. In any case applying machine learning for strength classification is very helpful if the attacker knows about the strength of the password. Manual strength classification might not achieve good accuracy.

Table 5. Comparison of dictionary attack performed with full size RockYou and strength subsets (compiled by the author)

	Hashing Algorithm	Full RockYou	Splitted RockYou (by Strength)				
			VW	W	M	S	VS
Very Weak	argon2	6 m 19 s	25 s	1 m 30 s	5 m 17 s	40 s	0 s
	bcrypt	56 m 42 s	3 m 59 s	15 m 44 s	43 m 53 s	6 m 13 s	6 s
	MD5	2 s	1 s	1 s	2 s	1 s	1 s
	SHA-256	6 s	3 s	4 s	5 s	4 s	3 s
Weak	argon2	2 m 36 s	49 s	32 s	4 m 20 s	37 s	0 s
	bcrypt	26 m	8 m 26 s	5 m 23 s	43 m 18 s	5 m 50 s	6 s
	MD5	1 s	1 s	1 s	2 s	1 s	1 s
	SHA-256	5 s	3 s	3 s	4 s	3 s	3 s
Medium	argon2	0 s	46 s	1 m 27 s	0 s	36 s	0 s
	bcrypt	3 s	7 m 16 s	13 m 4 s	1 s	5 m 47 s	6 s
	MD5	1 s	1 s	1 s	1 s	1 s	1 s
	SHA-256	4 s	3 s	4 s	3 s	3 s	4 s
Strong	argon2	5 m 20 s	52 s	1 m 36 s	4 m 28 s	19 s	0 s
	bcrypt	37 m 56 s	7 m 44 s	14 m 29 s	42 m 23 s	3 m 7 s	6 s
	MD5	2 s	1 s	1 s	2 s	1 s	1 s
	SHA-256	4 s	3 s	3 s	5 s	3 s	3 s
Very Strong	argon2	3 m 39 s	48 s	1 m 33 s	4 m 21 s	38 s	0 s
	bcrypt	34 m 55 s	8 m 57 s	15 m 5 s	44 m 2 s	6 m 17 s	3 s
	MD5	1 s	1 s	1 s	2 s	1 s	0 s
	SHA-256	5 s	3 s	4 s	6 s	4 s	4 s

Note: because it is not clear how John the Ripper measures the cracking time duration the values in the table are likely to be inaccurate and are due to the limitations of the tool used but they are sufficient to show the performance.

4.2. Password length prediction

For length prediction, regression models were used. Equivalent models to the classification models used in previous section were used for regression as well (except for logistic regression) with additional models which are specific for regression, those models include linear regression and lasso regression. For evaluation purposes, the dataset was split 20% for testing and 80% for training the model. Each model was trained on 80% of the data and afterwards tested with the remaining data. The output of the length prediction models is a numerical value representing the predicted length of the provided password. For evaluation of each regression model, the R^2 metric (or coefficient of determination) was used. It represents the proportion of variance that has been explained by the independent variables in the model. It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n} \left(y_i - \overline{y}\right)^2}, \tag{5}$$

where: y_i, $\hat{y}_i$ and $\overline{y}$ are the actual, predicted, and average value respectively; n is the size of the testing dataset.

Another metric that was used and is important in regression analysis is an error measure. A few such measures exist but for this research the mean squared error (MSE) was used. It is the mean of the square of the residuals which are the difference between the actual and predicted value.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2, \tag{6}$$

where: n is the size of the testing dataset; y_i and $\hat{y}_i$ are the true and predicted value respectively.

The features that were used for the regression models are the strength (encoded as numbers from 0 to 4), password entropy, charset size, lowercase count and digit count.

The R^2 score and the MSE for each regression model were calculated using Equation (5) and Equation (6).

Table 6. R^2 score and MSE for each regression model (compiled by the author)

Model Name	R^2 Score	MSE
Linear Regression	99.24%	0.0908
Lasso Regression	98.97%	0.1233
Support Vector Machine (SVM)	98.8%	0.1435
Decision Tree	99.92%	0.0094
Multilayer Perceptron (MLP)	99.88%	0.0148
k-Nearest Neighbors (*k*-NN)	99.92%	0.0095

The regression models provide high performance with R^2 score above 90%. According to Table 6, the regression models based on decision tree and *k*-Nearest Neighbors (*k*-NN) give the highest R^2 score but the model based on decision tree makes predictions with lower mean squared error. However, it is important to note that with a bigger dataset the results can improve even more.

In this specific case with a dataset of 12 781 instances, the decision tree model is the best model with 99.92% R^2 score and 0.0094 mean squared error which is the lowest among the tested models and hence was chosen for the second phase of the experiment.

The prediction for password length can be used for both dictionary attack and brute-force attack and hence two experiments were conducted.

Dictionary Attack. In the first experiment, a prediction was made based on known information about two different passwords. This prediction was used to select from the RockYou dictionary only those passwords who match the predicted length. The two dictionaries, RockYou and filtered RockYou, were given to John the Ripper to crack the password which was hashed with four different hashing algorithms: SHA-256, MD5, bcrypt and argon2. The results of the dictionary attack experiment are presented in Table 7.

Table 7. Comparison of original RockYou and filtered RockYou based on length for dictionary attack (compiled by the author)

Predicted Length	Hashing Algorithm	Full RockYou	Filtered RockYou	Cracked?	
				Full	Filtered
8	argon2	3 m 18 s	42 s	yes	yes
	bcrypt	30 m 57 s	6 m 43 s	yes	yes
	MD5	2 s	1 s	yes	yes
	SHA-256	5 s	3 s	yes	yes
10	argon2	6 m 11 s	42 s	yes	yes
	bcrypt	56 m 44 s	6 m 58 s	yes	yes
	MD5	2 s	1 s	yes	yes
	SHA-256	5 s	4 s	yes	yes
17	argon2	2 m 16 s	1 s	yes	no
	bcrypt	22 m 20 s	10 s	yes	no
	MD5	2 s	1 s	yes	no
	SHA-256	4 s	4 s	yes	no

Note: because it is not clear how John the Ripper measures the cracking time duration the values in the table are likely to be inaccurate and are due to the limitations of the tool used but they are sufficient to show the performance.

The results in Table 7 show that if an accurate length prediction is made, it is possible to eliminate irrelevant password candidates and make the dictionary attack a more focused and target-specific attack as well as reducing the time it takes to crack a password quickly even when slow hashing algorithm is used.

Brute-Force Attack. The second experiment is related to brute-force attack. In brute-force attack it is possible to restrict the length of the password to a specific range or a specific length. This, in theory, can reduce the amount of time required to wait for password recovery due to the reduction of the size of the search space containing all the possible character combinations. This experiment has shown no significant improvement, not even for the fast-hashing algorithms. This leads to a conclusion that restrictions on the length of the password are not sufficient for fast password cracking.

Conclusions

This paper aimed at proposing a method for multi-purpose password dataset generation suitable for machine learning and other research related to passwords either directly or indirectly.

A review of existing password datasets have shown that a few password datasets already exist, but they are very limited in size and features as well as they are, mostly, based on passwords which originated from leaked databases or passwords which are weak in general.

A review of the literature on the topic of password cracking methods based on password datasets have shown that there are several studies that present the applicability of password datasets during a password cracking attack, but they are all focused on generation of more password candidates based on patterns in leaked passwords. The researchers behind those methods concentrate on increasing password cracking effectiveness rather than decreasing password cracking time.

This paper suggested looking at the problem of password cracking from a different angle by generating a password dataset with meaningful features which are helpful to know if fast password cracking is what we want to achieve. However, the proposed method has noticeable limitation which is that it is based on randomness, but the randomness is what allows to easily introduce diversity to the dataset.

The experimental investigation has shown that if we have a partial information about the target password, it is possible to use machine learning with the generated password dataset as a training dataset to predict the missing information which can be used to narrow down the possibilities of password candidates to those which are more likely to be the password we are looking for. The use of machine learning in this way was found to be applicable only for dictionary attacks.

As an example, a very week password hashed with bcrypt hashing algorithm was cracked with the full size RockYou dictionary in about 1 hour. Acquiring the information about the existence of different character types and the password length leads to knowing the password strength, this information allows to reduce the dictionary size and crack the password in about 4 minutes which is a reduction of 56 minutes which is significant. Similar results can be seen regarding passwords of other strengths as well.

The length of the password is known to be an important piece of information about the password because long passwords lead to stronger passwords. A password of length 10 hashed with bcrypt was cracked in about 1 hour using the full RockYou dictionary. Knowing the strength, entropy, charset size, lowercase count and digit count leads to accurate prediction of the password length and hence brings us closer to the actual password. Using the length prediction, the password was cracked in almost 7 minutes, but accurate prediction is not always achievable with the dataset size used in the experiment.

Another aspect in the length prediction is the choice of features. It is important that the features used for length prediction will be easily and, at least, partially achievable. The features used for length prediction in this paper are not all achievable but features like password mask and password scheme are. It is also important to note that the results in both experiments are based on a very law amount of test cases and extended research is required to verify the results obtained in this research.

References

Bachmann, M. (2014). Passwords are dead: Alternative authentication methods. In *2014 IEEE Joint Intelligence and Security Informatics Conference* (pp. 322–322). IEEE. https://doi.org/10.1109/JISIC.2014.67

Bansal, B. (2019). *Password strength classifier dataset*. Kaggle. https://www.kaggle.com/datasets/bhavikbb/password-strength-classifier-dataset

Bansal, S. (2021). *10000 most common passwords*. Kaggle. https://www.kaggle.com/datasets/shivamb/10000-most-common-passwords

Bowes, R. (2008). *Passwords – SkullSecurity*. https://wiki.skullsecurity.org/index.php/Passwords

Craenen, R. (n.d.). *Leet speak cheat sheet*. Retrieved August 21, 2022, from https://www.gamehouse.com/blog/leet-speak-cheat-sheet/

Deng, G., Yu, X., & Guo, H. (2019). Efficient password guessing based on a password segmentation approach. In *2019 IEEE Global Communications Conference (GLOBECOM)* (pp. 1–6). IEEE. https://doi.org/10.1109/GLOBECOM38437.2019.9013139

Devi, K. K., & Arumugam, S. (2019). Password cracking algorithm using probabilistic conjunctive grammar. In *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)* (pp. 1–4). IEEE. https://doi.org/10.1109/INCOS45849.2019.8951390

Grassi, P., Garcia, M., & Fenton, J. (2017). *Digital identity guidelines: Revision 3*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-63-3

Hellman, M. E. (1980). A cryptanalytic time – memory trade-off. *IEEE Transactions on Information Theory*, *26*(4), 401–406. https://doi.org/10.1109/TIT.1980.1056220

Hitaj, B., Gasti, P., Ateniese, G., & Perez-Cruz, F. (2017). *PassGAN: A deep learning approach for password guessing*. aXiv. https://doi.org/10.48550/arXiv.1709.00440

Kaspersky. (n.d.). *Brute force attack: Definition and examples*. Retrieved July 19, 2022, from https://www.kaspersky.com/resource-center/definitions/brute-force-attack

Kim, P., Lee, Y., Hong, Y.-S., & Kwon, T. (2021). A password meter without password exposure. *Sensors*, *21*(2), 345. https://doi.org/10.3390/s21020345

Li, Z., Li, T., & Zhu, F. (2019). An online password guessing method based on big data. In *Proceedings of the 2019 3rd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence* (pp. 59–62). https://doi.org/10.1145/3325773.3325779

McMillan, R. (2012). *The world's first computer password? It was useless too*. https://www.wired.com/2012/01/computer-password/

NordPass. (2021). *Top 200 most common password list 2021*. https://nordpass.com/most-common-passwords-list/

Oechslin, P. (2003). Making a faster cryptanalytic time-memory trade-off. In D. Boneh (Ed.), *Lecture notes in computer science: Vol. 2729. Advances in Cryptology – CRYPTO 2003*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45146-4_36

Pleacher, D. (n.d.). *Calculating password entropy*. Retrieved February 16, 2023, from https://www.pleacher.com/mp/mlessons/algebra/entropy.html

Potter, B. (2005). Are passwords dead? *Network Security*, *2005*(9), 7–8. https://doi.org/10.1016/S1353-4858(05)70280-4

scikit-learn. (n.d.). *Metrics and scoring: quantifying the quality of predictions – scikit-learn 1.2.1 documentation*. Retrieved February 16, 2023, from https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Sipser, M. (2012). *Introduction to the theory of computation*. Cengage Learning.

Szczepanek, A. (2021). *Password entropy calculator*. https://www.omnicalculator.com/other/password-entropy

Tatli, E. I. (2015). Cracking more password hashes with patterns. *IEEE Transactions on Information Forensics and Security*, *10*(8), 1656–1665. https://doi.org/10.1109/TIFS.2015.2422259

Ur, B., Noma, F., Bees, J., Segreti, S. M., Shay, R., Bauer, L., Christin, N., & Cranor, L. F. (2015). "I Added '!' at the End to Make It Secure": Observing password creation in the lab. In *SOUPS 2015 proceedings*. USENIX.

Weir, M., Aggarwal, S., Medeiros, B. de, & Glodek, B. (2009). Password cracking using probabilistic context-free grammars. In *2009 30th IEEE Symposium on Security and Privacy* (pp. 391–405). IEEE. https://doi.org/10.1109/SP.2009.8

Yu, F., & Huang, Y. (2015). An overview of study of passowrd cracking. In *2015 International Conference on Computer Science and Mechanical Automation (CSMA)* (pp. 25–29). IEEE. https://doi.org/10.1109/CSMA.2015.12