Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS EU 2025 - Selected Papers from the 12th Annual Digital Forensics Research Conference Europe

PaSSw0rdVib3s!: AI-assisted password recognition for digital forensic investigations

Romke van Dijk[a,b,*], Judith van de Wetering[b,**], Ranieri Argentini[b], Leonie Gorka[b], Anne Fleur van Luenen[b], Sieds Minnema[b], Edwin Rijgersberg[b], Mattijs Ugen[b], Zoltán Ádám Mann[a,c], Zeno Geradts[a,b]

[a] *University of Amsterdam, Science Park 904, Amsterdam, 1098 XH, the Netherlands*
[b] *Netherlands Forensic Institute, Laan van Ypenburg 6, The Hague, 2497 GB, the Netherlands*
[c] *University of Halle-Wittenberg, Von-Seckendorff-Platz 1, Halle, 06108, Germany*

ARTICLE INFO

Keywords:
Password recognition
Machine learning
AI-Assisted digital forensics

ABSTRACT

In digital forensic investigations, the ability to identify passwords in cleartext within digital evidence is often essential for the acquisition of data from encrypted devices. Passwords may be stored in cleartext, knowingly or accidentally, in various locations within a device, e.g., in text messages, notes, or system log files. Finding those passwords is a challenging task, as devices typically contain a substantial amount and a wide variety of textual data. This paper explores the performance of several different types of machine learning models trained to distinguish passwords from non-passwords, and ranks them according to their likelihood of being a human-generated password. Three deep learning models (PassGPT, CodeBERT and DistilBERT) were fine-tuned, and two traditional machine learning models (a feature-based XGBoost and a TF/IDF-based XGBoost) were trained. These were compared to the existing state-of-the-art technology, a password recognition model based on probabilistic context-free grammars. Our research shows that the fine-tuned PassGPT model outperforms the other models. We show that the combination of multiple different types of training datasets, carefully chosen based on the context, is needed to achieve good results. In particular, it is important to train not only on dictionary words and leaked credentials, but also on data scraped from chats and websites. Our approach was evaluated with realistic hardware that could fit inside an investigator's workstation. The evaluation was conducted on the publicly available RockYou and MyHeritage leaks, but also on a dataset derived from real casework, showing that these innovations can indeed be used in a real forensic context.

1. Introduction

Passwords remain one of the main methods for securing sensitive information. In recent years, devices have been considerably strengthened against password recovery attacks with improved software and hardware. This is especially true for what are often referred to as 'secure phones'. These phones have modified software and hardware with high security standards, making them attractive to large-scale organized crime groups for secure communication, preventing law enforcement from intercepting their communication (Europol (2021)). Traditional password recovery methods, such as brute-force attacks or dictionary attacks, may no longer be feasible on these devices. This requires forensic investigators to embrace new techniques.

A digital investigation often spans multiple devices. Some will be directly accessible for investigation, while others will be secured with an unknown password. Analyzing the available data from already unlocked devices can provide useful leads to find password information. Passwords may be stored, either knowingly or accidentally, in cleartext in various locations on a device. For example, in text messages, notes, or system log files. Finding these passwords may provide critical information on the password that can be used to unlock secure phones.

Finding those cleartext passwords is a challenging task. In theory, an investigator could extract all possible text (strings) from the already available data and then try all those strings as a potential password on the secure phone. However, the list of strings from a typical smartphone is too large to be tried in succession on a secure phone. Secure phones

* Corresponding author. University of Amsterdam, Science Park 904, Amsterdam, 1098 XH, the Netherlands.
** Corresponding author.
E-mail addresses: romke.van.dijk@nfi.nl (R. van Dijk), j.van.de.wetering@nfi.nl (J. van de Wetering).

https://doi.org/10.1016/j.fsidi.2025.301870

Available online 24 March 2025

have modified software and hardware, protecting themselves with techniques such as hardware-bound keys and memory-hard encryption algorithms.

To overcome this difficulty, a ranking algorithm would be needed that can rank all strings according to the likelihood that they are passwords, with the most promising potential passwords at the top of the list. This would allow investigators to attempt the most likely passwords first.

The search and ranking of potential passwords is a problem that has not received the attention it deserves so far. After early work by Houshmand et al. (2015), the research question lay dormant for years as published research (see Section 2) focused on related but different problems. The research field focused on developing more efficient password recovery models. The only similar research found was done by Feng et al. (2022). They attempted to discover accidentally disclosed passwords and API keys in GitHub repositories. This is a task that is superficially similar to recognizing passwords stored on devices, but benefits from the uniformity of the application domain (computer code repositories). However, data from a device lacks uniformity, as it consists of various types of known and unknown file formats.

Our work aims to rank text strings, acquired from cleartext of previously unlocked devices, according to their likelihood of being a human-generated password, taking into account the specifics of data typically found on such devices. To create the ranking, we address the following subproblem: given a string acquired from a device, determine the likelihood of it being a human-generated password. We formulate this as a machine learning problem and apply modern machine learning methods to solve this problem.

Our two main research questions are the following:

1. What mix of different kinds of training data yields the best results for this task?
2. What model architecture performs best, in terms of machine learning performance metrics and computational performance (evaluations per second)?

In the first question, our results show that expanding the training set beyond dictionary words and leaked credentials is critical. A wide variety of non-password data should be used to ensure the model is able to perform well when confronted with the variety of data that are stored on modern devices. In the second question, we find that modern machine learning techniques outperform traditional models and, while the modern techniques are slower, they are definitely useable in forensic investigations.

In general, our work gives forensic investigators a new approach in combating the acquisition of secure mobile devices.

2. Related work

2.1. Password research

From the moment passwords were introduced, research on them also started. Morris and Thompson (1979) were the first to show that human-generated passwords were often based on dictionary words. These kinds of weaknesses can be exploited using tools such as Hashcat (Steube (2009)) and JohnTheRipper (Openwall (1996)). In recent years, new advanced methods have been developed. Weir et al. (2009) developed a method that uses Probabilistic Context Free Grammar (PCFG). They developed a model that trained on existing passwords and learned managing rules to improve the efficiency of password recovery. Machine learning techniques were also used for password strength meters. Melicher et al. (2016) developed a lightweight neural network (written in JavaScript) that determines the strength of the password locally in the user's browser. In addition to password strength meters, Hitaj et al. (2019) advanced the field using machine learning techniques to improve the efficiency of password recovery using Generative Adversarial Network (GAN). More recently, Rando et al. (2023) developed a password recovery technology that uses a Generative Pre-trained Transformer (GPT) model.

Dell'Amico et al. (2010) performed an analysis on a large-scale public dataset and researched the characteristics (e.g., length and composition) of passwords and their resilience against existing attacks. A more recent analysis (Kanta et al. (2021)) analyzed an even larger public dataset, which contains 3.9 billion accounts. In addition to analyzing the characteristics of the passwords, they also classified fragments according to their semantic meaning.

The above-mentioned attack methods and analysis results can help a forensic investigator towards more effective password recovery attempts. These methods focus primarily on improving the effectiveness of password recovery. Another potential weakness that may be exploited is password reuse. Florencio and Herley (2007) showed that people tend to reuse passwords. So, finding passwords that a user has used for other accounts or devices can result in an additional attack strategy. However, this type of approach has received very little research attention so far.

2.2. Password recognition

Even though there has been a lot of research on the effectiveness of password recovery and on general analysis of password characteristics, there is limited research on the topic of password recognition, i.e. how to decide if a string is a password. There are open-source tools (Gitleaks, TruffleSecurity, Awslabs, Microsoft) that search for leaked credentials in, for example, GitHub source code repositories. These tools use rules for classifying passwords (e.g., does a string contain both special characters and alphabet characters) and take into consideration the location where the given string was found (for example, searching for the word 'password' inside a configuration file). Such tools rely on a predictable text structure, which is usually not the case in forensically acquired data from seized devices. On a device, passwords may not only be stored in configuration files but also in notes, chat messages, or hidden in unstructured binary data. Thus, existing approaches for finding passwords in source code repositories are not applicable to the problem of finding passwords in data acquired from devices in forensic investigations.

Password strength meters might be used to recognize human-generated passwords. An example of a metric used by password strength meters is entropy. NIST and Aroms (2012) described that the entropy of a password would be higher than that of normal text. However, Kelley et al. (2012) showed that the entropy of passwords is close to that of normal text. Entropy-based approaches are not very accurate as they also mark machine-generated strings like hex strings as potential passwords, as the entropy of such strings can be very high. Therefore, using entropy is not a good solution for recognizing human-generated passwords.

Houshmand et al. (2015) uses PCFG for finding passwords on disk images. However, the framework is limited to disk images and focuses on strings inside text documents (such as Word documents). Their work also investigated potential methods for ranking passwords. PCFG assigns high probability scores to frequently used passwords, such as '12345678' because of their common occurrence in password datasets. Thus, the approach is good at recognizing weak passwords. This weakness was mitigated by implementing filters. They found that a filter that removed all lines only containing alphabet characters results in a better performing algorithm. This relies on the assumption that password policies would require passwords to have characters from multiple classes (e.g., both lower-case letters and special characters). However, mobile devices can have a different password policy. For example, modern Android devices do not require passwords to contain characters from multiple classes (Google). This might result in users using weak passwords, such as passwords that contain only alphabetic characters. Such passwords might provide critical information about the possible password for a secure device as they might be based on a weak password (for example, adding a special character to make it fit the passwords policy).

Feng et al. (2022) developed a method to recognize passwords using a Text Convolution Neural Network (TEXTCNN). Their paper was published in 2022; at that time it was already known that transformers outperformed TEXTCNN based models (Vaswani (2017)). Feng et al. (2022) focused on finding passwords in GitHub code repositories. As discussed previously, code repositories are characterized by a well-structured form of text compared to unstructured text extraction from mobile devices.

2.3. Our contribution

Most of the existing related work focuses on increasing the efficiency of password recovery attacks or on researching the characteristics of passwords. There is very limited research on the recognition of passwords. Existing research on password recognition used outdated machine learning technology. Previous research has also not analyzed the impact of different training sets on the performance of machine learning models. And no previous research has analyzed which machine learning model performs best in ranking strings on the likelihood of being a human-generated password.

Our work advances the field in three ways. First, we show the impact of different training sets on the performance of the models. Second, we compare state-of-the-art machine learning models with each other and with the current state-of-the-art password recognition approach. Third, we analyze the suitability and practical applicability of our approach, showing that with this research, we provide forensic investigators with new capabilities to combat secure phones.

3. Password recognition

Our aim is to create a (deep learning) model that can determine for a text string, extracted from a device, the likelihood that the string is a human-generated password. Fig. 1 shows how such a model could be used in the context of a forensic investigation, starting with the extraction of strings from a device, leading to a ranked list of text strings, ordered according to their likelihood of being a human-generated password. This paper focuses on the model itself.

To create a high-performance model, the right set of labeled training data is crucial. We need both positive training samples (passwords) and negative training samples (non-passwords). In addition, we need to decide on the type and architecture of the deep learning model.

In this section, we first discuss our efforts in collecting the necessary training data, both for positive and negative examples. An overview of all training data can be found in Table 1. We then explain the different models we trained, both existing and our new deep learning models, to compare the performance of different techniques.

3.1. Training data – Positives

For the positives, we have one category of data: passwords. We have used BreachCompilation (2017), a compilation of 1.4 billion leaked credentials. We removed the usernames and/or e-mail addresses using demeuk (NFI (2021)). From this compilation, we used the 100 million most frequently occurring passwords. This number was used to have an acceptable training set. We believe that the most used passwords are more likely to be human-generated passwords. When training the PCFG model, we used the 10 million most common passwords. This number was selected based on the developer documentation of the PCFG cracker, more passwords would not result in better performance but would slow down the generation speed of PCFG.

3.2. Training data – Negatives

Next to examples of passwords, we also need examples of non-passwords. More specifically, to achieve good classification performance, we need training samples that represent the variety of strings that can occur on a device, but are not passwords. To achieve this coverage, we collected such negative samples from a large variety of sources, which are described in the following sections. Besides English

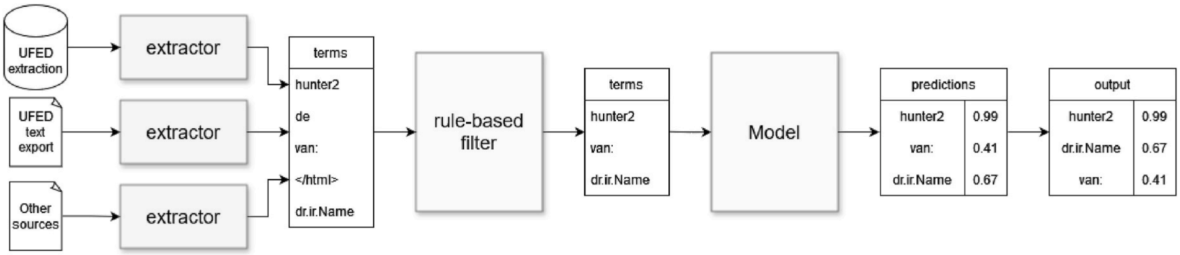

Fig. 1. Overview of the process of how the model created in this work could be used. The process starts with the extraction of text from a device, and ends with a ranked list of strings sorted according to their likelihood of being a human-generated password. The research described in this paper focuses on the last three blocks: the model, predictions and output.

Table 1
Overview of all training data used in this research, both positives (passwords) and negatives (non-passwords).

Name	Category	Language	Nr. of unique strings
Passwords	Passwords	–	100,000,000
DWYL	Wordlist	English	466,551
Open Taal	Wordlist	Dutch	413,938
Urban Dictionary	Wordlist	Mostly English	1,482,214
CulturaX	Crawls and Chats	English	10,000,005
CulturaX	Crawls and Chats	Dutch	10,000,052
Whatsapp Corpus	Crawls and Chats	Dutch	31,434
Telegram	Crawls and Chats	Dutch and English	3,734,120
Pixel 3a - Custom Dictionary	Carves	English	2146
Pixel 3a - In-house tooling	Carves	English	1,530,078
iPhone 12 - Custom Dictionary	Carves	English	1948
iPhone 12 - In-house tooling	Carves	English	9,657,983
base64	Encodings	–	10,000
hexadecimal	Encodings	–	10,000

Table 2
Evaluation of the impact of including different kinds of training sets on performance, measured with the precision@100 metric. A set of 1000 passwords were selected from the RockYou and a case dataset and combined with 234,524 non-password strings. In each column, the best value is marked in bold.

precision@100								
Chats	Carves	Hex & Filtering	feature_xgb (case data)	feature_xgb (rockyou)	tfidf_xgb (case data)	tfidf_xgb (rockyou)	PassGPT (case data)	PassGPT (rockyou)
✅	✅	✅	0.36	0.36	**0.77**	**0.80**	**0.9**	**0.96**
✅	✅	❌	0.68	**0.72**	0.67	0.64	0.83	0.83
✅	❌	❌	**0.71**	0.67	0.61	0.53	0.77	0.78
❌	✅	❌	0.18	0.56	0.1	0.1	0.03	0.07
❌	❌	❌	0.16	0.56	0.01	0.07	0.06	0.13

and language-independent strings, we added training data in Dutch where applicable, as this is the most common language in our casework. In Section 4.3, we experiment with the effect of adding or removing different categories of data to the training set to investigate their effect on the resulting performance.

3.2.1. Wordlists

The first category is wordlists, which we added to train the model to adequately recognize ‘normal’ text. We used two language-specific sets: the ‘DWYL’ English words list and the ‘Opentaal Woordenlijst Nederlands’ Dutch word list. Both these lists contain different verb conjugations. The final list we added is the Urban Dictionary word list. This list is a lot more varied in language use and spelling than the very clean English and Dutch lists. We only used entries that do not contain white space.

3.2.2. Chats and crawls

The second category are text strings from web crawls and chats. The main difference between this set and the wordlists is that these webcrawls and chats are full texts, which we have split on whitespace. Because of this, some of the entries contain punctuation (for example, periods, commas, or question marks at the end of a word, or opening brackets at the start) and capitalization which we both specifically maintain. It is important to train our models on this type of data as it is expected that otherwise the model would quickly associate words including punctuation marks with passwords if we do not also train the model on non-password strings that include such special characters. These sets also contain a wider variety of conjugations, spelling, and other variations. We used the English and Dutch sections of the CulturaX web crawl (Nguyen et al. (2023)), the WhatsApp corpus (Spooren et al. (2018)), and a proprietary capture of Telegram group chats.

3.2.3. Carves

The model will eventually be used mostly on carved data. Carved data contains a lot of ‘system language’ (an example of this are HTML tags) which is very different from the other types of data we train on. Thus, it is also useful to add such strings to the negative training set. We carved the full file-system copies of two mobile phones, a Google Pixel 3a and an Apple iPhone 12, both in factory-reset state. The carved data were made in two different ways: using the ‘Custom Dictionary’ functionality of Cellebrite Physical Analyzer version 7.61 and using in-house software.

3.2.4. Hex & filtering

The final category in our set of negative training samples consists of randomly generated base64 and hexadecimal strings. Initially, we did not include these strings in our training dataset, but in early testing we noticed that the model associated these types of strings with a high likelihood of being passwords; therefore, we added them to our training negatives. We refer to the combination of base64 strings and hex strings as the hex dataset.

Because this dataset was added later in our research, we had already implemented some basic filtering in our training process. That means that passwords were removed from the training set that contained more than 32 characters and passwords that only consisted of digits (since we are not interested in PIN codes). Thus, the increase in performance when adding the hex dataset is not only caused by the dataset itself, but also by additional filtering.

3.3. Machine learning models

We have established what data we have available. The next step is to train a variety of different models in order to compare their performance. We grouped the models into three different categories. First, we will look at the work by Houshmand et al. (2015), who used Probabilistic Context Free Grammars (PCFG) to identify passwords. Next, we will look at two ways to turn passwords into numeric vectors in order to train a traditional machine learning model on these vectors. Finally, we introduce three deep learning models, based on PassGPT, CodeBERT, and DistilBERT.

3.3.1. PCFG

As mentioned in section 2, PCFG can also be used for ranking strings on their likelihood of being human-generated passwords. PCFG is a well-known model in the password recovery community. One of the main drawbacks of PCFG for password recognition is that it is unable to deal with out-of-vocabulary passwords. If a substring of a password candidate was not included in the original training data, PCFG will always give the password candidate a score of 0.

3.3.2. Baseline machine learning

Before deep learning entered the stage, ‘traditional’ machine learning methodologies were employed, where a list of numbers that capture the essential aspects of the data (features) are collected into a feature vector. The assembly of this set of features is a critical part of the implementation of a traditional machine learning system. As a baseline, two sets of features were implemented. These methods also act as a fair replacement for simple rule-based filtering systems (for example, based on regular expressions).

The Character N-gram TF/IDF model utilizes TF/IDF (Term Frequency-Inverse Document Frequency, see Leskovec et al. (2020) chapter 1.3.1) to assess the significance of a character sequence (n-gram) occurring within a string, compared to other strings.

Term Frequency (TF) measures how often a specific n-gram appears within a string, while Inverse Document Frequency (IDF) reduces the importance of n-grams that are common across many strings. The words 'term' and 'document' hint at the origin of the method as a ranking function used in information retrieval, but this method is still frequently used for turning a text (in our case candidate passwords) into a numerical vector representation.

Another way to assemble a feature vector is to score a number of predetermined properties of the text input. This is known as feature engineering. For this model, we used the following features:

1. Length of the string in characters
2. Number of lowercase letter characters
3. Fraction of lowercase letter characters
4. Number of uppercase letter characters
5. Fraction of uppercase letter characters
6. Number of numeric characters
7. Fraction of numeric characters
8. Number of special (non-letter non-numeric) characters
9. The number of character sets used (lowercase, uppercase, numeric, special)
10. The number of transitions between consecutive character sets

One of the main advantages of this method is that applying this model is generally a lot faster than other methods, which is a factor in casework.

The resulting feature vectors are then used to train a classification or regression model using a machine learning algorithm. The gradient boosting algorithm XGBoost (Chen and Guestrin (2016)) was selected for both feature-based models based on the results of preliminary research. XGBoost outputs a score between 0 and 1, instead of a binary classification.

In the rest of our work we will refer to the model obtained by using XGBoost on the Character N-gram TF/IDF features as tfidf_xgb and the model obtained by using XGBoost on the engineered features as feature_xgb.

3.3.3. Deep learning

The introduction of the transformer architecture by Vaswani (2017) and the subsequent release of pretrained models such as BERT by Devlin et al. (2019) meant a large leap in what was possible in the field of Natural Language Processing (NLP).

Transformer models operate on sequences of tokens. A token is an individual, fundamental unit of data. Tokens are generated by running the input (in our case, text strings representing potential passwords) through a piece of software called a tokenizer. A token can be one or multiple letters or even a word or a phrase depending on the tokenizer used. A tokenizer is generally specific to the transformer model that will be used to process the tokens.

We chose three different pre-trained models to fine-tune on how well they ranked strings being a human-generated password. The first model is PassGPT, by Rando et al. (2023), a 66M-parameter generative model based on GPT2, which was trained on the RockYou dataset. The rationale behind choosing this pre-trained model is that it contains information on what passwords generally look like. We use the standard practice of adding a *classification head* to the model in order to turn it from a generative model into a classification model. PassGPT uses a simple single-character-based tokenizer.

The second model is a 135M-parameter multilingual distilBERT model by Sanh et al. (2019). The 'distil' of distilBERT stands for distillation, the process of turning a larger model into a smaller, more efficient model with minimal loss of accuracy performance. DistilBERT is a general-purpose model trained on natural language. Our hope for this model is that it will be able to incorporate some semantic information that will enable it to better generalize from the examples provided in the training corpus. For example, if London123 and Mercedes2012 are included in the training set, the model should be able to recognize Manchester123 and Audi2012 as passwords because the pre-trained model has learned about cities in the UK and car manufacturers. DistilBERT uses a trained tokenizer that generally tokenizes words into multi-character tokens.

Our final model is an 84M parameter CodeBERT model (Feng et al. (2020)). The model is pre-trained on a large collection of open source code and associated documentation. Our rationale for choosing this model is that it will have had a lot more exposure to special characters, and the context passwords might be found in. CodeBERT uses a tokenizer that was trained on code, causing it to more efficiently tokenize code strings into multi-character tokens.

4. Evaluation

We evaluated our approach, with its variants described in Section 3, to answer the following research questions:

RQ1: What mix of training data yields the best results for the task of finding human-generated passwords in data acquired from a device?
RQ2: What type of machine learning model performs best for the given task?

For evaluating the performance of different variants of the approach (for both RQ1 and RQ2), we use standard metrics typically used to evaluate machine learning models; see below. In addition, for RQ2, we also assess the computational performance (execution time of the inference process) of the different models.[1]

The best metrics to use when evaluating a model are the ones that reflect the intended use case of the system. In our case, the output of the model is used to sort the list of strings by their score, with the highest scores at the top. We therefore use the precision@k metric, which is very useful when scoring models on their ability to rank data. The precision@k metric is calculated by dividing the number of passwords in the top k results by score, by k. We use $k = 100$ and $k = 1000$ to reflect different situations.

Additionally, we used a commonly used metric in Machine Learning research, especially when it is not possible to determine an adequate threshold; this is *ROC/AUC*. The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC). This plots the true positive rate against the false positive rate at various threshold settings and quantifies the overall ability of the model to discriminate between classes.

Graphically, we use *inverse recall* graphs to show the model performance. These graphs show the number of passwords found as a function of the number of strings processed.

4.1. Evaluation datasets

When evaluating models, an 80/20 split is often used, where the model is trained on 80 % of the dataset and then evaluated on the remaining 20 % of the dataset. We decided not to do this. Instead, we collected an evaluation set of password and non-password strings that closely mirrors the intended real-world usage. This shows that a model can be trained on publicly available data and then be used in real forensic investigations. This results in a sharable model, as no confidential data was used in its training.

Furthermore, for training on leaked password lists the 80/20 split has an important drawback. Passwords from a website like MyHeritage will include the term 'heritage' more often than passwords for the RockYou website. Thus, running the evaluation on the same training set might result in better performance than when using different evaluation and training sets.

[1] Computational performance is not assessed for RQ1, since the choice of training data does not influence the execution time of the inference process.

We evaluated the trained model on three different datasets. The first dataset, Case data, was obtained during the course of a criminal investigation. This dataset was provided by the Dutch Police. This dataset cannot be shared due to legislation that governs case data. The second dataset is RockYou (Brannondorsey (2009)), which is widely used in an academic context. RockYou was leaked in 2009 and, therefore, is not necessarily representative of modern password usage. The final dataset is MyHeritage (hashmob (2017)) which was breached in 2017, when security standards were raised considerably compared to 2009. MyHeritage contains password hashes, the evaluations can therefore only be done on the cracked passwords. This is not the case for the Case data and RockYou as these are cleartext passwordsets.

For all three datasets the positive (password) examples in the evaluation set are a random weighted sample of 1000 passwords: The list contains only unique passwords, but passwords that occur more often in the dataset have a higher likelihood of being chosen. The negative examples (i.e., the non-password strings) of the evaluation dataset are obtained by running the Custom Dictionary feature of Cellebrite Physical Analyser version 7.61 on seven devices from the following manufacturers and models:

- Apple iPhone 6s plus
- Apple iPhone 7
- Apple iPhone 11
- Huawei P smart
- Motorola Moto G9 plus
- Samsung Galaxy J7
- Samsung S20FE

These devices are part of a mock-up case used for investigator training purposes and contain a significant amount of traces of normal user activity. The passwords of the device users are known. To avoid polluting the set of negative examples, these passwords were removed from the list prior to conducting the evaluation.

The total evaluation set consists of 234,524 non-passwords and 1000 passwords (0.4 %) which we consider to be a good balance between a realistic scenario (only a fraction of the total set is a password) and having a large enough set of passwords to reduce noise in the evaluation.

4.2. *Evaluation environment*

All of the experiments were ran with the following hardware:

- CPU: Intel Xeon E5-1650 v3 3.5 GHz
- GPU: NVIDIA RTX 4500 Ada Generation (24 GB GDDR6)
- Memory: 32GiB

4.3. *Evaluation results*

4.3.1. *RQ1 – Which combination of training data yields the best results?*

This research question is evaluated on the feature-based models and the PassGPT-based model (as a representative of the deep learning models). The training dataset always contains the positives described in Section 3.1 and the wordlists as negatives, described in Section 3.2. For the other three types of negative samples (Chats and crawls, Carves and Hex, described in Sections 3.2–3.2), we test different combinations of which datasets are included in the training set and which ones are not.

Looking at the results (see Tables 2 and 3) of this experiment, it is clear that adding chats and crawls is very important for the model performance. For all models and both metrics, the performance increases considerably when adding the chats. The effect of adding carved data is less clear. When only carved data are added, performance generally goes down. However, in most cases, adding both chat and carved data leads to higher scores than adding only one of them. When adding the hex dataset and filtering, the performance dropped for the feature-based model, but for all of the other models it resulted in the

Table 4
Evaluation of the performance of different model architectures. The models were run on data extracted from mobile devices (234,524 items), combined with 1000 passwords from 3 datasets: case data, MyHeritage and RockYou.

Case data			
model	precision@100	precision@1000	roc_auc
distilBERT	0.89	0.326	0.9187
codeBERT	0.83	0.35	0.9331
passGPT	**0.90**	**0.468**	**0.9399**
feature_xgb	0.36	0.24	0.9225
tfidf_xgb	0.77	0.335	0.9191
PCFG	0.18	0.105	0.7607
MyHeritage			
model	precision@100	precision@1000	roc_auc
distilBERT	0.81	0.237	0.8926
codeBERT	0.75	0.297	0.9059
passGPT	**0.87**	0.388	**0.9141**
feature_xgb	0.32	0.188	0.9087
tfidf_xgb	0.53	**0.414**	**0.914**
PCFG	0.17	0.073	0.7245
RockYou			
model	precision@100	precision@1000	roc_auc
distilBERT	0.91	0.364	0.847
codeBERT	0.89	0.372	0.8553
passGPT	**0.96**	**0.487**	**0.8627**
feature_xgb	0.36	0.263	0.8454
tfidf_xgb	0.79	0.362	0.8472
PCFG	0.31	0.142	0.7506

Table 3
Same as Table 2, but with performance measured by the ROC AUC metric.

ROC AUC								
Chats	Carves	Hex & Filtering	feature_xgb (case data)	feature_xgb (rockyou)	tfidf_xgb (case data)	tfidf_xgb (rockyou)	PassGPT (case data)	PassGPT (rockyou)
✅	✅	✅	0.9225	0.8454	**0.9191**	0.8472	0.9399	0.8627
✅	✅	❌	**0.9225**	**0.9458**	0.9122	**0.9435**	0.9545	**0.9782**
✅	❌	❌	0.9213	0.9395	0.9087	0.9287	**0.9594**	0.9776
❌	✅	❌	0.789	0.8362	0.8306	0.8771	0.8108	0.8704
❌	❌	❌	0.8264	0.8596	0.8154	0.7849	0.8599	0.8925

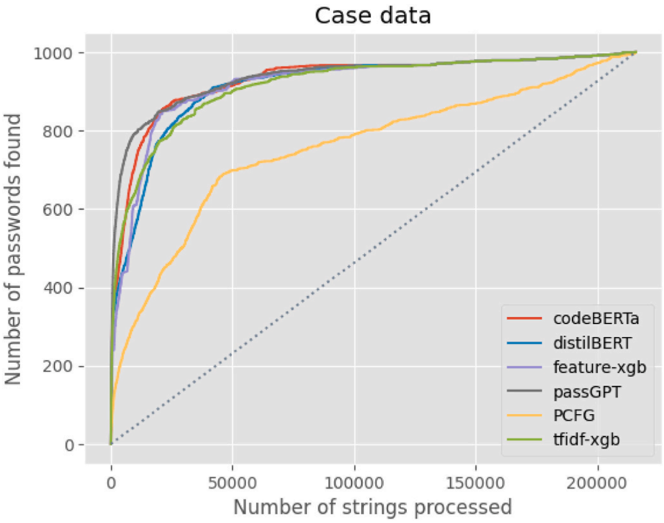

Fig. 2. Inverse recall of models evaluations on the Case dataset.

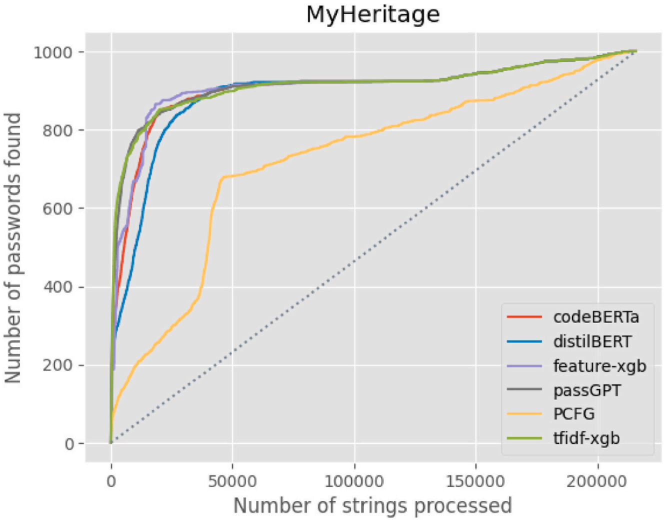

Fig. 3. Inverse recall of models evaluations on the MyHeritage dataset.

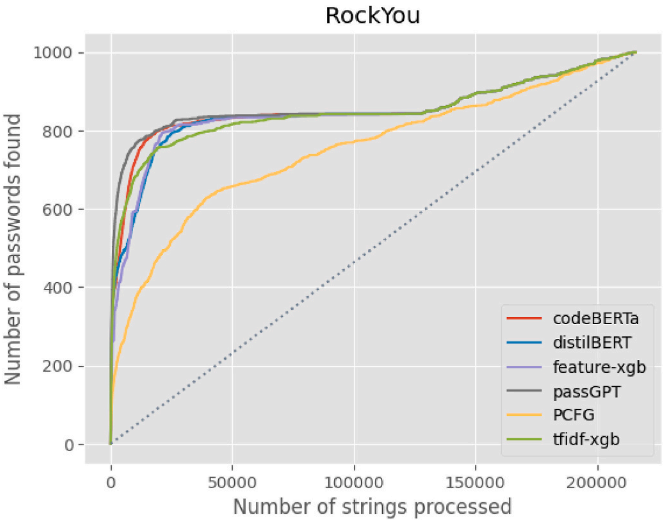

Fig. 4. Inverse recall of models evaluations on the RockYou dataset.

Table 5
Execution time of inference with different models in seconds, normalised with the fastest running model, and showing the precision@100 when running the model on the case data evaluation dataset. The models were run on data extracted from mobile devices (234,524 strings) combined with 1000 passwords.

Model	Seconds	Normalised	precision@100 for case data
distilBERT	21	1.50	0.89
codeBERT	20	1.43	0.83
passGPT	42	3.00	**0.90**
feature_xgb	**14**	**1.00**	0.36
tfidf_xgb	38	2.71	0.77
PCFG	50	3.57	0.18

highest precision@100 scores. Therefore, we decided to do further experiments for RQ2 on models trained on all data.

4.3.2. RQ2 – Which machine learning model performs best?

For this research question, we compare the performance of the different machine learning models described in Section 3.3. Each model is trained using the full training dataset described in Sections 3.1–3.2. Each model is evaluated on the three evaluation datasets described in Section 4.1. The results are summarized in Table 4. Figs. 2–4 show graphically how well the different models succeed in ranking real passwords.

From Table 4, we can see that the PassGPT-based model performs best for most metrics and evaluation datasets. For RockYou this is unsurprising, as PassGPT is trained on RockYou (and various other leaks), but this is not the case for the other two datasets. The only exception to the dominance of PassGPT is the MyHeritage dataset combined with the precision@1000 metric, where the tfidf_xgb model performs best (see also Fig. 3). This is quite surprising, as this model does not have any other standout performances. Furthermore, we see that the PCFG and feature_xgb models perform significantly worse on all metrics than the other models. For PCFG this is most likely due to its inability to properly score passwords with out-of-vocabulary sub-strings. The feature_xgb model is most likely not able to adequately capture the difference between passwords and non-passwords in limited features it uses.

The execution time of performing inference with the different models is compared in Table 5. We can see that the feature based model is the fastest, followed by the distilBERT and codeBERT models. PCFG and the PassGPT model are the slowest. There are clear differences in speed, but they are all within the same order of magnitude. When comparing both precision@100 and execution time, it can be derived that distilBERT and PassGPT have a similar precision@100 for the case dataset (0.89 vs 0.90). However, PassGPT requires twice the execution time.

When taking the precision@1000 for the case dataset, PassGPT clearly outperforms distilBERT (0.468 vs. 0.326). This shows that a forensic investigator can make a trade-off between execution time and accuracy. This suggests that distilBERT should be chosen when execution time is most important, or PassGPT when performance is more important.

5. Conclusion

Digital investigators face challenges in acquiring data from encrypted devices. An underexplored method is that of extracting strings from unsecured mobile devices and then identifying those strings that are most likely to be a human-generated password.

Our research shows that machine learning technology can be used to help rank strings extracted from a mobile device based on their likelihood of being human-generated passwords. Although the fastest deep learning model (distilBERT) runs 1.5 times slower than simpler machine learning technology (feature_xgb), it outperformed the same model 2.47 times (for the precision@100 metric, when running on the case dataset). When performance is more important than computational speed, the investigator should select PassGPT as it outperforms the performance of distilBERT. Furthermore, we showed that training with more than just dictionary words and passwords is essential to achieve good performance. Most of the investigated models benefited from adding chat, carved, and base64/hex data to the negative training set.

We limited our research to the part related to the ranking and did not focus on text extraction. Further investigation could explore how well different methods can extract strings from various sources. Additionally, the location where a string is found could also be taken into account when determining the likelihood of it being a human-generated password.

With this research, we show that the use of machine learning technology provides forensic investigators with new methods for combating encrypted devices.

Ethical consideration

The datasets used in this research have different origins. Some of the datasets were provided for research by the Dutch Police under Dutch law. Other datasets were already available publicly under different terms and conditions.

Some of the datasets we used are hacked datasets, datasets that are obtained through illicit means. Ienca and Vayena (2021) provide requirements for conducting research for such datasets. The hacked datasets that are used are: RockYou, MyHeritage, and BreachCompliation.

Those datasets are of significant importance for this research. Without them, this research would not be possible. Replacement data sets are not available. RockYou, despite being an old dataset, remains the benchmark for comparing different methods used in password recovery. BreachCompliation provides insights of password usage at scale, as it is one of the largest collections of passwords available. MyHeritage provides a more recent insight in password usages and is not included in BreachCompliation. No personal identifiable information from the datasets was used, only the password data.

This research provides forensic investigators with a new method for getting access to potentially critical information stored in encrypted data. In the end, this serves our society as a whole. Together with the minimal threat this research poses to individuals, we argue that our research has a favorable risk-benefits balance.

References

Awslabs, gitsecrets - prevents you from committing secrets and credentials into git repositories. https://github.com/awslabs/git-secrets. [Accessed 4-October-2024].

Brannondorsey, 2009. Rockyou. https://github.com/brannondorsey/naive-hashcat/releases. (Accessed 11 October 2024).

BreachCompilation, 2017. Breachcompilation magnet:?xt=urn:btih:7ffbcd8cee06aba2ce6561688cf68ce2addca0a3. (Accessed 13 September 2024).

Chen, T., Guestrin, C., 2016. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 785–794. https://doi.org/10.1145/2939672.2939785.

Dell'Amico, M., Michiardi, P., Roudier, Y., 2010. Password strength: an empirical analysis. In: 2010 Proceedings IEEE INFOCOM. IEEE, pp. 1–9.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Europol, 2021. New Major Interventions to Block Encrypted Communications of Criminal Networks. https://www.europol.europa.eu/media-press/newsroom/news/new-major-interventions-to-block-encrypted-communications-of-criminal-networks. (Accessed 4 October 2024).

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., et al., 2020. Codebert: a pre-trained model for programming and natural languages. arXiv preprint arXiv:2002.08155.

Feng, R., Yan, Z., Peng, S., Zhang, Y., 2022. Automated detection of password leakage from public github repositories. In: Proceedings of the 44th International Conference on Software Engineering, pp. 175–186.

Florencio, D., Herley, C., 2007. A large-scale study of web password habits. In: Proceedings of the 16th International Conference on World Wide Web, pp. 657–666.

Gitleaks, gitleaks - protect and discover secrets using. https://github.com/gitleaks/gitleaks. [Accessed 13-December-2024].

Google, Password requirements - android management api. https://developers.google.com/android/management/reference/rest/v1/PasswordRequirements#PasswordQuality. [Accessed 6-September-2024].

hashmob, 2017. Hashmob - myheritage. https://hashmob.net/hashlists/info/4362-myheritage. [Accessed 11-October-2024].

Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F., 2019. Passgan: a deep learning approach for password guessing. In: Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17. Springer, pp. 217–237.

Houshmand, S., Aggarwal, S., Karabiyik, U., 2015. Identifying passwords stored on disk. In: Advances in Digital Forensics XI: 11th IFIP WG 11.9 International Conference, Orlando, FL, USA, January 26-28, 2015, Revised Selected Papers 11. Springer, pp. 195–213.

Ienca, M., Vayena, E., 2021. Ethical requirements for responsible research with hacked data. Nat. Mach. Intell. 3, 744–748. https://doi.org/10.1038/s42256-021-00389-w. URL: https://www.nature.com/articles/s42256-021-00389-w.

Kanta, A., Coray, S., Coisel, I., Scanlon, M., 2021. How viable is password cracking in digital forensic investigation? analyzing the guessability of over 3.9 billion real-world accounts. Forensic Sci. Int.: Digit. Invest. 37, 301186.

Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., Lopez, J., 2012. Guess again (and again and again): measuring password strength by simulating password-cracking algorithms. In: 2012 IEEE Symposium on Security and Privacy, IEEE, pp. 523–537.

Leskovec, J., Rajaraman, A., Ullman, J.D., 2020. Mining of Massive Data Sets. Cambridge university press.

Melicher, W., Ur, B., Segreti, S.M., Komanduri, S., Bauer, L., Christin, N., Cranor, L.F., 2016. Fast, lean, and accurate: modeling password guessability using neural networks. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 175–191.

Microsoft, . Credscan - a static analysis tool to scan for credential leaks. https://secdevtools.azurewebsites.net/helpcredscan.html. [Accessed 4-October-2024].

Morris, R., Thompson, K., 1979. Password security: a case history. Commun. ACM 22, 594–597.

NFI, 2021. Demeuk. https://github.com/NetherlandsForensicInstitute/demeuk. (Accessed 13 September 2024).

NIST, Aroms, E., 2012. Nist special publication 800-63 electronic authentication guideline.

Nguyen, T., Nguyen, C.V., Lai, V.D., Man, H., Ngo, N.T., Dernoncourt, F., Rossi, R.A., Nguyen, T.H., 2023. Culturax: a cleaned, enormous, and multilingual dataset for large language models in 167 languages. arXiv:2309, 09400.

Openwall, 1996. John the ripper - password cracker. https://www.openwall.com/john/. (Accessed 4 October 2024).

Rando, J., Perez-Cruz, F., Hitaj, B., 2023. Passgpt: password modeling and (guided) generation with large language models. In: European Symposium on Research in Computer Security. Springer, pp. 164–183.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv abs/1910.01108.

Spooren, W., Verheijen, L., Komen, E., Hulsbosch, M., van den Heuvel, H., 2018. Whatsapp corpus verheijen. https://doi.org/10.17026/dans-2zm-hgka.

Steube, J., 2009. Hashcat - Advanced Password Recovery. https://hashcat.net/hashcat/. (Accessed 4 October 2024).

TruffleSecurity, trufflehog - find, verify, and analyze leaked credentials. https://github.com/trufflesecurity/trufflehog. [Accessed 13-December-2024].

Vaswani, A., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst.

Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B., 2009. Password cracking using probabilistic context-free grammars. In: 2009 30th IEEE Symposium on Security and Privacy, IEEE, pp. 391–405.