This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Improving Deep Learning Based Password Guessing Models Using Pre-processing

Yuxuan Wu[1], Ding Wang[2(✉)], Yunkai Zou[2] and Ziyi Huang[3]

[1] College of Computer Science, Nankai University, Tianjing 300381, China
[2] College of Cyber Science, Nankai University, Tianjing 300381, China
[3] College of Software, Nankai University, Tianjing 300457, China
wangding@nankai.edu.cn

Abstract. Passwords are the most widely used authentication method and play an important role in users' digital lives. Password guessing models are generally used to understand password security, yet statistic-based password models (like the Markov model and probabilistic context-free grammars (PCFG)) are subject to the inherent limitations of overfitting and sparsity. With the improvement of computing power, deep-learning based models with higher crack rates are emerging. Since neural networks are generally used as black boxes for learning password features, a key challenge for deep-learning based password guessing models is to *choose the appropriate preprocessing methods to learn more effective features.*

To fill the gap, this paper explores three new preprocessing methods and makes an attempt to apply them to two promising deep-learning networks, i.e., Long Short-Term Memory (LSTM) neural networks and Generative Adversarial Networks (GAN). First, we propose a character-feature based method for encoding to replace the canonical one-hot encoding. Second, we add so far the most comprehensive recognition rules of words, keyboard patterns, years, and website names into the basic PCFG, and find that the frequency distribution of extracted segments follows the Zipf's law. Third, we adopt Xu et al.'s PCFG improvement with chunk segmentation at CCS'21, and study the performance of the Chunk+PCFG preprocessing method when applied to LSTM and GAN.

Extensive experiments on six large real-world password datasets show the effectiveness of our preprocessing methods. Results show that within 50 million guesses: 1) When we apply the PCFG preprocessing method to PassGAN (a GAN-based password model proposed by Hitja et al. at ACNS'19), 13.83%∼38.81% (26.79% on average) more passwords can be cracked; 2) Our LSTM based model using PCFG for preprocessing (short for PL) outperforms Wang et al.'s original PL model by 0.35%∼3.94% (1.36% on average). Overall, our preprocessing methods can improve the attacking rates in four over seven tested cases. We believe this work provides new feasible directions for guessing optimization, and contributes to a better understanding of deep-learning based models.

Keywords: Password · Deep learning · Preprocessing · Generative adversarial networks · Long short-term memory neural networks

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

1 Introduction

Although passwords have some security problems and a variety of new authentication methods are constantly proposed, passwords promise to be the dominant authentication method in the foreseeable future due to their simplicity to deploy, easiness to change [2,3]. Thus, it is of great importance to understand password security, and a number of guessing algorithms have successively been proposed, such as statistical-based ones (i.e., probabilistic context free grammars (short for PCFG) [18] and Markov [10,12]) and deep-learning based ones (i.e., PassGAN [5] and FLA [11]). Password guessing algorithms understand password security from the perspective of attackers who focus on the vulnerability of passwords, and they in turn can be used to build protection countermeasures, such as constructing the password strength meter (PSM) to evaluate the strength of users' passwords.

Password guessing attacks can be divided into targeted guessing attacks and trawling guessing attacks [16]. The former is to crack the password of a given user as quickly as possible [10], and the latter is to crack as many passwords as possible in a given password set under the limited guess number [16, 19]. This paper focuses on trawling guessing attacks. A key challenge for trawling guessing attacks is to extract the password features effectively, and data preprocessing is a feasible method to improve the effect of the deep learning based models.

1.1 Related work

Major Password guessing models based on statistical probability include PCFG [18] and the Markov model [10, 12]. The main idea of PCFG is to divide a password into several segments according to the character types, and these segments can be regarded as the password features. For instance, the password `abc123` is parsed into the letter segment "abc" and digit segment "123". PCFG can also be integrated with other guessing models as a data preprocessing method [9, 17, 20]. The Markov model records the frequency of different characters after the password substring in the training phase, and then generates the guessing password character by character according to the frequency distribution. With various improvements made on the base of PCFG, Xie et al. [19] focused on the targeted guessing attack and added the recognition of special dates and names. Wang et al. [15] added the recognition of Chinese pinyin and the six-digit dates (e.g., 201862) to further exploit the features of Chinese passwords. Houshmand et al. [6] added the recognition of keyboard patterns. Yang et al. [21] also studied keyboard patterns and analyzed the frequency distribution of the keyboard patterns. However, these PCFG-based methods are not optimal because most of these added recognition rules only consider the targeted guessing, and are not comprehensively considered in trawling guessing. In this paper, we focus on the password features, and add the most comprehensive recognition rules of keyboard patterns, words, website names, years for trawling guessing

Recently, deep learning technology provides a new way for password attacking, and the model based on supervised learning was first used. In 2016, Melicher et al. [11] used Recurrent Neural Network (RNN) to build a password guessing model (i.e., FLA) which can be considered as a character-level model because the

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

smallest unit it handles is each character in the password. In 2018, Liu et al. [9] proposed a multi-source PCFG+LSTM model (The LSTM based model using PCFG for preprocessing, short for PL) with the adversarial generation which can maintain high accuracy for different datasets. The password guessing model using PCFG for preprocessing can be regarded as a segment-level model, which divides passwords into different segments. In 2021, Wang et al. [17] found that the PL model could crack more passwords than PCFG and Markov within 50 million guesses. However, they only use the original PCFG which can not comprehensively extract password features. Xu et al. [20] proposed a new preprocessing method based on the Byte-Pair-Encoding (BPE) algorithm to divide passwords into chunks that consist of frequently occurring characters, and then built three models: the Markov based model using the chunk based preprocessing method, the model using the Chunk+PCFG preprocessing method, and the LSTM based model using chunk based preprocessing method. Password guessing models using the chunk-based preprocessing method are considered chunk-level. In this paper, we integrate the Chunk+PCFG preprocessing method with neural networks.

Unsupervised learning methods are also used in password cracking. In 2019, Hitaj et al. [5] first proposed a password guessing model based on Generative Adversarial Networks (GAN) and named it PassGAN. However, the cracking result of PassGAN is not ideal, and even lower than the traditional methods. We apply three preprocessing methods to PassGAN model, and find that using the basic PCFG for preprocessing can dramatically improve the cracking rate.

1.2 Our contributions

In this work, we make the following key contributions:

(1) **Character feature based encoding method**. Character-level models usually adopt the one-hot encoding method which can not fully utilize the character features. Therefore, we propose a new approach based on the type of characters and the corresponding keyboard positions, where each character is represented as a 4-dimensional vector: (character type, character serial number, keyboard row number, keyboard column number). Although this encoding method does not improve the effect, it still provides a new feasible direction for password guessing.

(2) **Refined PCFG**. Existing PCFG [6] divides passwords into four segments (letters, digits, special characters, and keyboard). We propose a refined PCFG based preprocessing method which adds the recognition rules of words, website names, and years to enable a more comprehensive password feature extraction. Inspired by Wang et al.'s work that the distribution of passwords follows the Zipf's law [14], we find that the frequency distribution of extracted segments also follows PDF-Zipf.

(3) **An extensive evaluation**. We perform a series of experiments on 9 models, including two baseline ones (i.e., the LSTM based model using one-hot encoding method in Wang's work [17] and the original PassGAN model in Hitaj's work [5]), and seven models using preprocessing(i.e., the LSTM based model using our new encoding method, the LSTM based

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

model using basic PCFG for preprocessing [17], the LSTM based model using our refined PCFG for preprocessing, the LSTM based model using Chunk+PCFG for preprocessing, the PassGAN model using basic PCFG for preprocessing, the PassGAN model using our refined PCFG for preprocessing, and the PassGAN model using Chunk+PCFG for preprocessing). Our empirical results show that character-level models can improve their effect by using PCFG based preprocessing methods. In particular, the PassGAN model using PCFG for preprocessing can improve the success rates drastically (avg. 26.79 %) compared to the original PassGAN model.

2 Background

In this section, we briefly introduce the background on deep learning based password guessing models (i.e., LSTM based models and GAN based models).

2.1 LSTM based models

Recurrent Neural Network (RNN) and its Variants, such as Long Short-Term Memory (LSTM), can all be used in password guessing models [11,17]. To avoid gradient vanishing problems [8], we use LSTM instead of RNN. The one-hot encoding is usually performed on each character to convert a password string to a matrix. Moreover, it is necessary to construct the corresponding label y for the input password x due to the supervised learning method. We use an example to illustrate this training process (see in Fig. 1). Suppose the character set that contains all the characters appearing in the dataset is {a, b, c, Bos, Eos}, where Bos represents the beginning of a password and Eos represents the end of a password. Then the password `abc` is converted to a matrix: [[0 , 0 , 0 , 1 , 0], [1 , 0 , 0 , 0 ,0], [0, 1, 0, 0, 0], [0 , 0 , 1 , 0 , 0]]. The corresponding label y is: [[1 , 0 , 0 , 0 , 0], [0 , 1 , 0 , 0 , 0], [0 , 0 , 1 , 0 , 0], [0 , 0 , 0 , 0 , 1]].

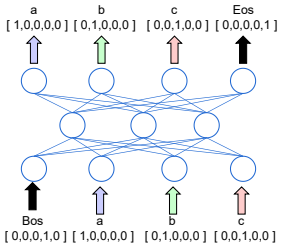


Fig. 1. The training process of LSTM based models using one-hot encoding method. The character set is {a, b, c, Bos, Eos}, where Bos represents the beginning of a password and Eos repreasents the end of a password.

The LSTM based model is a probability model which assigns probabilities to the guessing passwords. The probability of the next character is obtained by entering the prefix of the password string into LSTM. For example, a password generation process could be: `B`→`Ba`→`Bab`→ `Babc`→`BabcE`, where each time we select the character with the highest probability as the next character. The LSTM based model with this training and generating method is character-level.

2.2 PassGAN

PassGAN, proposed by hitaj [5], is based on WGAN-GP [4] due to the difficulty of training original GAN. PassGAN consists of a generator and a discriminator. The generator captures the real data distribution by building a mapping function from prior noise distribution to real data space and generate fake samples. The discriminator learns to determine whether a sample is from the generator

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

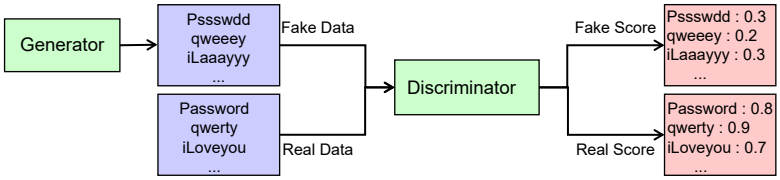


Fig. 2. The training process of PassGAN model. Passwords are first converted into matrices and then input to the discriminator. The score is the output of the discriminator which will be used to calculate the gradient.

distribution or the real data distribution. PassGAN are trained adversarially in this way until the discriminator cannot identify the source of the data. The main structure of the PassGAN model is shown in Fig. 2.

Original PassGAN is character-level, but not as effective as LSTM based character-level models [17]. Another problem of original PassGAN is that it can not assign probabilities to the generated guessing passwords, so we can not obtain the priority of different guessing passwords. These problems will all be solved by using preprocessing methods in this paper.

3 Preliminaries

In this section, we first explicate the datasets used in this paper, including the basic information, the length distribution, the character composition, and the top-10 passwords. In addition, since passwords are confidential data related to personal privacy, ethical considerations are also explained.

3.1 Datasets

We compare the password guessing models with different preprocessing methods based on six large password datasets (see in Table 1) with a total of 57 million passwords. These datasets are different in terms of service, size, user localization, and language, which suggests that our models can be used to well characterize different user-chosen passwords. We name each dataset according to its website's domain name. The first three datasets, namely CSDN, YueJunYou, Renren, are all from Chinese websites. CSDN is a well-known community website of Chinese programmers, founded in 1999. Renren is a real-name social networking platform, founded in 2005. YueJunYou is a website for making friends and traveling, founded in 2014. The last three datasets, namely Rockyou, Yahoo, Youporn, are all from English websites. Rockyou is a game website that contains 320 million passwords, which is the largest dataset among six datasets. Yahoo is a famous internet portal site in the United States, founded in 1995. Youporn is a video website only for adults, founded in 2006.

Datasets cleaning. We note that these original datasets contain some abnormal passwords that are either too long (> 40) or too short (< 4), which are unlikely to be user-chosen passwords or simply junk information. Thus, we launch the work of dataset cleaning before any experiment. We first remove the passwords that contain symbols beyond the 95 printable ASCII characters, and then we also

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Table 1. Basic information about six web services.

Dataset	Web service	Language	When leaked	Original PWs	After cleaning
CSDN	Programmer forum	Chinese	Dec.,2011	6,428,277	6,427,538
YueJunYou	Social forum	Chinese	May.,2006	5,365,338	5,286,494
Renren	Social forum	Chinese	Oct.,2011	4,733,366	4,662,654
Rockyou	Gaming	English	Dec.,2009	32,581,870	32,573,986
Yahoo	Email	English	Jul.,2012	5,737,797	5,605,985
Youporn	Video	English	Oct.,2017	2,677,951	2,105,452

remove the passwords with length < 4 or length > 30, because these passwords do not comply with the password policy of most websites or may not be considered by the attackers who care about cracking efficiency [1]. Generally, the removed passwords are less than 1% for each dataset.

Here we also provide a concrete grasp of user-chosen passwords: 1) The length of most passwords is between 6 and 9, accounting for 62.08%∼83.84% of each web service (see details in Table 7 of Appendix A); 2) Chinese users love to use digits (avg. 54.64%) and this figure for English users is 18.62%, while English users love to use characters (avg. 42.79%) and this figure for Chinese users is 15.43% (see details in Table 8); 3) Top-10 passwords account for 7.18% 10.43% of Chinese users, and this figure for English users is 2.05% 5.29%, indicating Chinese passwords are more concentrated, as found in [15] (see in Table 9).

3.2 Ethical considerations

Although these datasets are widely used in the literature [5,10,11,20,21], they are still private data. Therefore, we only report the aggregated statistical information and treat each individual account as confidential, so that using them in our research will not increase the risk to the corresponding victim. Furthermore, these datasets may be utilized by attackers as cracking dictionaries, while our use is both beneficial for the academic community to understand the strength of users' password choices, and for security administrators to secure their passwords. In addition, we have consulted privacy experts a number of times and got approval from our center's IRB for this evaluation. Since our datasets are all available from the Internet, the results in this work are reproducible.

4 Preprocessing methods

In this section, we describe different preprocessing methods to improve the effect of the deep-learning based password guessing models.

4.1 Important abbreviations

To facilitate the reading process, we introduce the important abbreviations used in this article. GAN means Generative Adversarial Networks; PassGAN is short for the password guessing model based on GAN; LSTM is short for Long Short-Term Memory neural networks; PCFG is short for probabilistic context-free grammars; LSTM/PassGAN+X means the LSTM/PassGAN based password gurssing model using X for preprocessing.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

4.2 Character feature based encoding method

Canonical one-hot encoding method as used in [11,17,20] only classifies different characters, but can not fully reflect other character features. Moreover, the matrices converted by the one-hot encoding method are sparse [13]. The main challenge for character encoding is how to distinguish characters and reflect their features with as little space as possible. Therefore, we comprehensively consider different kinds of character features and then propose a new character encoding method that greatly reduces the occupied space.

The password character has two important features, one is the type, and the other one is the keyboard location since keyboard pattern is also a popular way in password creation [16]. Thus, we represent each character in four dimensions. The first dimension represents the type of characters, where we use 1, 2, 3, and 4 to represent digits, uppercase letters, lowercase letters, and special characters. The second dimension is the serial number of the characters in each type. For example, a∼z can be represented by 1∼26 according to the dictionary order, and the digits can be represented by themselves. The third dimension represents the keyboard row number, and the fourth dimension represents the keyboard column number. The row number of the keyboard increases from top to bottom, and the column number increases from left to right. For example, the string "1234567890-=" is in the first row, and the string "1qaz" is in the first column. Using our new encoding method, the password `1234` can be converted to a matrix [[1 , 1 , 1 , 1], [1 , 2 , 1 , 2], [1 , 3 , 1 , 3], [1 , 4 , 1 , 4]].

4.3 Refined PCFG

The basic PCFG [18] only divides passwords into letters, digits, and special characters, which may destroy the integrity of some segments and ignore user's habit of creating passwords. For example, the string "1!2@3#" should be considered as a complete segment due to the adjacency of characters on the keyboard, while it would be converted to $D_1S_1D_1S_1D_1S_1$ in the basic PCFG. Therefore, we add the recognition rules of 4 important password features.

Table 2. The proportion of top-10 mostly used years (1900-2100) for each web service∗.

Dataset	CSDN	Renren	YunJunYou	Rockyou	Yahoo	Youporn
Proportion	58.68%	54.74%	64.16%	33.94%	26.19%	32.68%
Unique†	201	201	201	201	201	201

∗ We record the proportion of top-10 year segments in all year segments.
† Unique represents the number of unique years in the web service.

Year recognition: Years have been found popular in passwords [7, 15]. We count the number of the years (from 1900 to 2100, a total of 201), and record the proportion of top-10 most widely used years in Table 2. Results show that some years occupy a large proportion, which indicates users may focus on some years. However, the basic PCFG would extract years into digits. Thus, we extract digit segments with length of four and value between 1900-2100 from the passwords.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Table 3. The proportion of top-10 mostly used websites for each web service.

Dataset	CSDN	Renren	YunJunYou	Rockyou	Yahoo	Youporn
Proportion	15.15%	20.93%	25.25%	15.35%	17.23%	25.44%
Unique†	3,348	3,080	202	6,415	1,109	255

† Unique represents the number of unique websites in the web service.

Website name recognition: We for the first time count the number of the website names in each dataset, and record the proportion of top-10 most widely used website names in Table 3. Although the website names are less concentrated than the years, some website names still account for a large proportion, and the passwords with website name segments may be considered strong passwords in the basic PCFG. For instance, the password "csdn.net" is converted to the base structure $L_4S_1L_3$, and may be assigned a low probability of being cracked by basic PCFG. To address this problem, we add the recognition of website names. First, common website name suffixes, such as ".com", ".net", are used to construct a suffix list, and then the complete website segments are extracted from the passwords according to the suffix list. For example, the website name segment "csdn.net" is extracted from the password "123csdn.net123".

Keyboard pattern recognition. Since keyboard pattern is also a popular way in password creation [6, 16], we add the corresponding recognition rule. Furthermore, keyboard patterns with only one character type can be extracted completely by basic PCFG, so our refined PCFG (i.e., basic PCFG with the additional recognition rules) focus on the keyboard patterns with multiple character types. For example, the password `q1w2e3` is converted to the base structure $L_1D_1L_1D_1L_1D_1$ by basic PCFG, which destroys the integrity of the keyboard pattern, while it is converted to K_6 by our refined PCFG (k represents the keyboard pattern and 6 represents the length of the segment).

Word recognition: The keyboard pattern has inherent limitations that may extract wrong segments [21]. For instance, the segment "password" should be regarded as a complete segment, but "assw" would be recognized as a keyboard pattern. Although our keyboard pattern recognition method avoids most of the error cases, wrong results may still occur. Our solution is to use a dictionary that contains common words, and extract segments that appear in the dictionary from the passwords before the keyboard pattern extraction. However, the effect of this method depends on the quality of the dictionary. Once the dictionary does not contain the corresponding word, keyboard pattern extraction will still cause errors. Therefore, we choose to construct the word dictionary through the training set. This process is described in Algorithm 1.

Frequency distribution: Wang et al. [14] found that the distribution of popular passwords follows PDF-Zipf:

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Algorithm 1: Dictionary contruction algorithm.

Input: Password set $\mathcal{S}$
Output: Word dictionary $\mathcal{D}$.

```
for pwd in S do
    letters_list = extract_letters(pwd);/* extract letter segments in pwd and store
      them in a list. */
    for seg in letters_list do
        if len(seg) > len_min then
            /* len_min is the minimum word length. */
            D[seg]+ = 1;/* D is the initial dictionary to record the frequency of
              different words. */
for seg in D do
    if D[seg] < threshold then
        /* threshold is the minimum word frequency */
        delete D[seg];
return D;
```

$$f_r = \frac{C}{r^s} \tag{1}$$

where f_r is the frequency of the password, r is the rank of the password, C and s are constants depending on the datasets. To verify that the data conforms to this distribution, we use the following equation:

$$log(f_r) = logC - s \cdot log(r) \tag{2}$$

where $log(f_r)$ and $log(r)$ are linearly related. We verify that the extracted segment meets the Zipf distribution based on equation 2. The extracted results are sorted in descending order of frequency on six datasets and the $log(f_r)$-$log(r)$ graphs are shown in Fig. 3. Moreover, all the coefficients of determination (R^2) which can measure the fitting degree of the regression line to the sample data are shown in Table 4. The closer the determination coefficient is to 1, the better the fitting effect is. The results indicate that the frequency distribution of the keyboard patterns, words, website names, and years with a frequency of more than five can meet the PDF-Zipf model well.

Table 4. The coefficient of determination (R^2) for fitting different extracted segments.

R^2	Keyboard pattern	Word	Website name	Year
CSDN	0.9667	0.9974	0.9791	0.9698
YueJunYou	0.9763	0.9895	0.9697	0.9650
Renren	0.9757	0.9935	0.9764	0.9772
Rockyou	0.9843	0.9973	0.9894	0.9358
Yahoo	0.9687	0.9932	0.9767	0.8710
Youporn	0.9795	0.9949	0.9661	0.9033

Recognition order: Some recognition rules may conflict with each other. For example, "989" in "1989" is recognized as a keyboard pattern, and "csdn" in "csdn.net" may be recognized as a word. Therefore, it is essential to set a

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

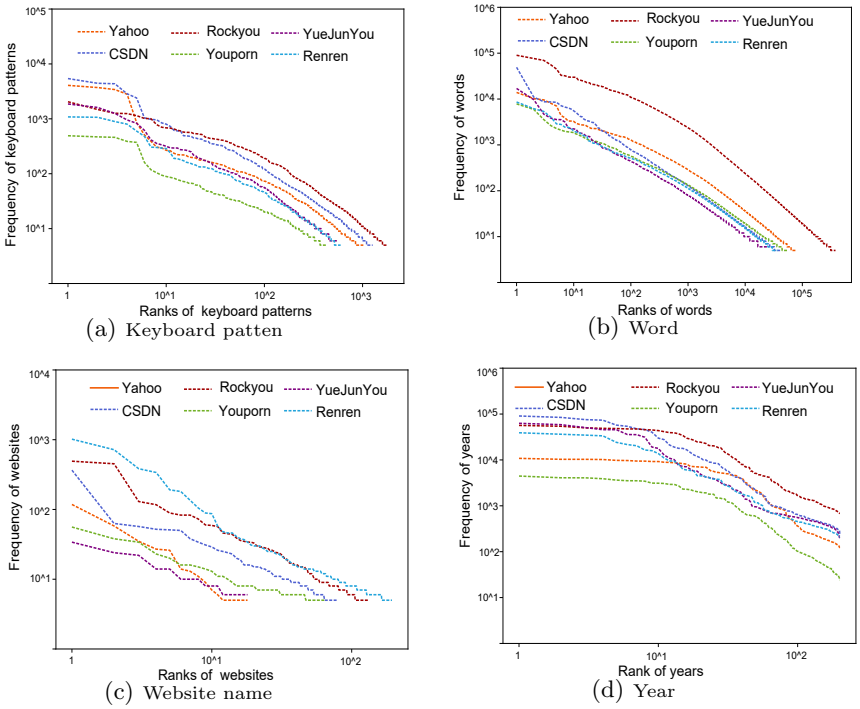


Fig. 3. Frequency distribution of different types of extracted segments.

reasonable extraction order and begin with the least conflict one. Since the year recognition may conflict with the keyboard pattern recognition, and the website name recognition may conflict with the word recognition. Our recognition order is: year→website name→word→keyboard pattern→basic PCFG recognition. Overall, our refined PCFG is different from the basic PCFG on the tags of base structures, where some tags are added, such as K (keyboard patterns), W (words), E (website names), and Y (years). The complete extraction process of our refined PCFG is described in Algorithm 2. The structure of the neural networks using the PCFG based preprocessing method is shown in Fig. 4.

4.4 PassGAN using PCFG for preprocessing

The result of PassGAN [5] is worse than the LSTM based models within 10^7 guessing passwords, so we infer that the ability of GAN to learn text features is weaker than LSTM. Character-level passwords are relatively complex for GAN due to the length and the multiple character types. Therefore, we use PCFG based preprocessing method to simplify the data, and train the model with the base structures obtained from PCFG. In the generation process, PassGAN would generate duplicate base structures without probability, so we count the number of different base structures until the number of unique base structures reaches the target value. Then we assign each base structure with the probability $f_i/total$, where f_i represents the frequency of the corresponding base structures, and $total$ represents the total number of all base structures.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Algorithm 2: Extraction algorithm for our refined PCFG.

Input: Password pw, word dictionary D, website name suffix list web_list
Output: handled segment list seg_list.

```
seg_list=[pw]; /* initial list taking the whole password as a unhandled segment. */
kp_min=3;/* the minimum length of the keyboard pattern. */
for seg in seg_List do
    if type(seg) == string and a year occurs in seg then
        begin, end = index_year(seg);
        divide_seg(seg, begin, end);/* divide seg into
          seg[0 : begin], ('Y' + str(end − begin), seg[begin : end]), seg[end :]. */
for seg in seg_List do
    if type(seg) == string and a website suffix from web_list occurs in seg then
        begin, end = index_website(seg);
        divide_seg(seg, begin, end);/* divide seg into
          seg[0 : begin], ('E' + str(end − begin), seg[begin : end]), seg[end :]. */
for seg in seg_list do
    if type(seg) == string and a word from D occurs in seg then
        begin, end = index_word(seg);
        divide_seg(seg, begin, end);/* divide seg into
          seg[0 : begin], ('W' + str(end − begin), seg[begin : end]), seg[end :]. */
for seg in seg_list do
    if type(seg) == string then
        begin, end = index_keyboard(seg);
        if end − begin >= kp_min and seg[begin : end] contains more than one
          character type then
            divide_seg(seg, begin, end);/* divide seg into
              seg[0 : begin], ('K' + str(end − begin), seg[begin : end]), seg[end :]. */
merge_unhandled(seg_list);/* merge successive unhandled segments in seg_list. */
PCFG_extraction(seg_list);/* use original PCFG for extraction. */
return seg_list;
```

Algorithm 3: The process of Chunk+PCFG preprocessing method

Input: Password dictionary with the corresponding frequency pwd_dict
Output: Processed Dictionary pwd_dict.

```
while true do
    (Pairs, avg_len)=get_pairs(pwd_dict);/*Take two consecutive chunks as a pair
      and record the frequency of pairs in Pairs, avg_len is the avg len of chunks.*/
    if avg_len > threshold then
        /*threshold stands for the minimum average-length of chunks. */
        break;
    best_pair = max(Pairs, key = Pairs.get); /*find the most frequnt pair. */
    pwd_dict = merge_chunk(best_pair, pwd_dict);
PCFG_extraction(pwd_dict);/* perform PCFG on each chunk of passwords.*/
return pwd_dict;
```

4.5 Chunk+PCFG preprocessing method

We adopt Xu et al.'s PCFG improvement with chunk segmentation at CCS'21 [20], and integrate the Chunk+PCFG preprocessing method with LSTM and PassGAN separately. Byte-Pair-Encoding (BPE) algorithm is used to divide passwords into chunks, and then the chunk-level passwords are converted to the base structures by performing PCFG on each chunk of the passwords. Since chunk-level passwords are fine-grained enough, we only use the basic PCFG and a chunk can be represented as L (with only letters), D (with only digits), S

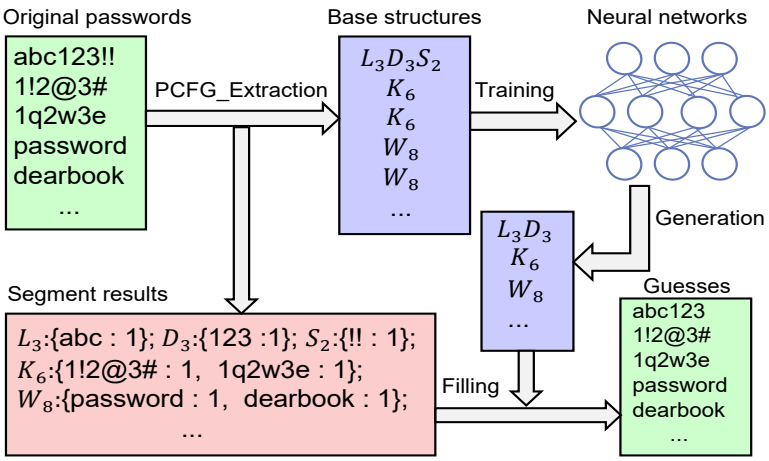


Fig. 4. An illustration of deep learning based model using PCFG based preprocessing method. K represents the keyboard pattern and W represents the word.

(with only special characters), Two (with two character types), $Three$ (with three character types). For example, the password `iloveu4ever` can be firstly converted to the chunk-level password ["iloveu", "4ever"], and then converted to the base structure L_6Two_5. The process of the Chunk+PCFG preprocessing method is shown in Algorithm 3. The remaining training and generation process is the same as that of the LSTM based models using PCFG for preprocessing.

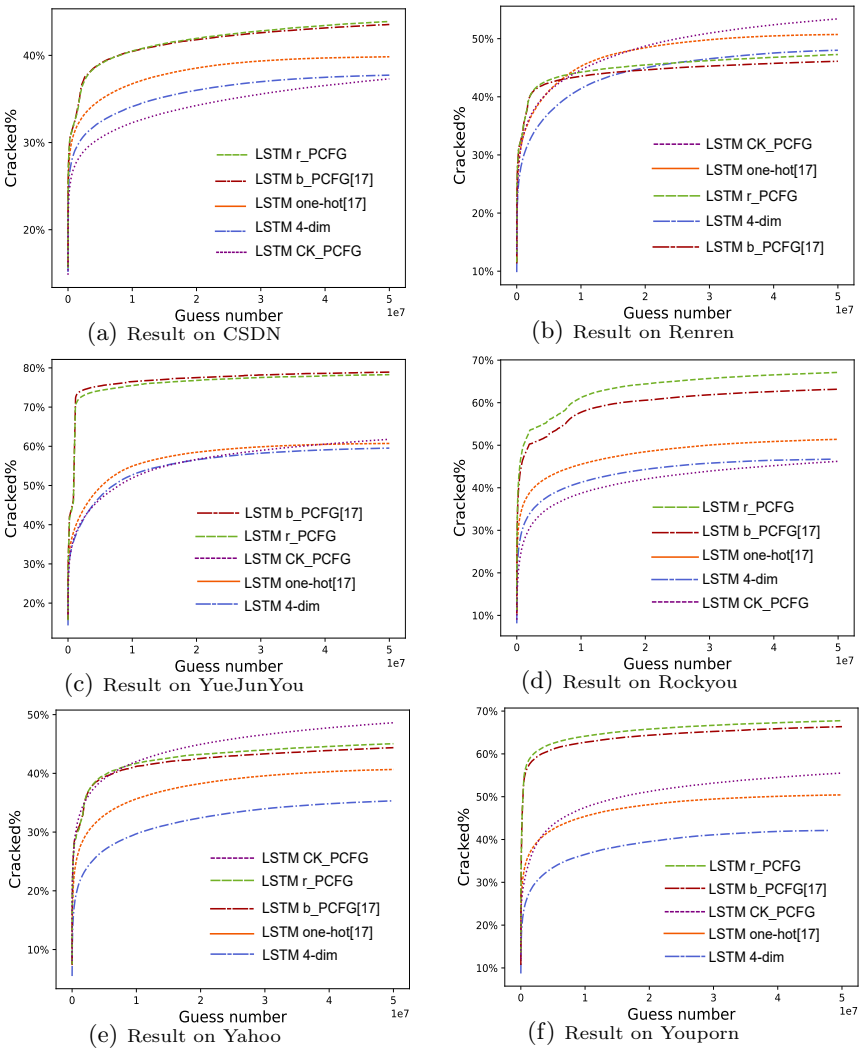
5 Experiments

In this section, we first describe the attacking strategies, and then evaluate the result of five different preprocessing methods combined with neural networks on multiple datasets. The details of datasets are described in Sec. 3.1.

5.1 Attacking strategies design

The combination of five different preprocessing methods with two leading deep-learning models gives rise to a total of ten guessing strategies, and we focus on seven promising ones: the LSTM based model using one-hot encoding method [17], the LSTM based model using our new encoding method, the LSTM based model using the basic PCFG for preprocessing (short for PL) [17], the LSTM based model using our refined PCFG for preprocessing, the LSTM based model using Chunk+PCFG for preprocessing, PassGAN using the basic PCFG for preprocessing, PassGAN using our refined PCFG for preprocessing, and PassGAN using Chunk+PCFG for preprocessing. Here LSTM includes a hidden layer that contains 128 neurons and a softmax layer, and the structure of GAN is the same as the original PassGAN model which is described in Sec. 2.2. For chunk-level models, the minimum average length of chunks is 3.0. All models are trained and

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).



(a) Result on CSDN (b) Result on Renren

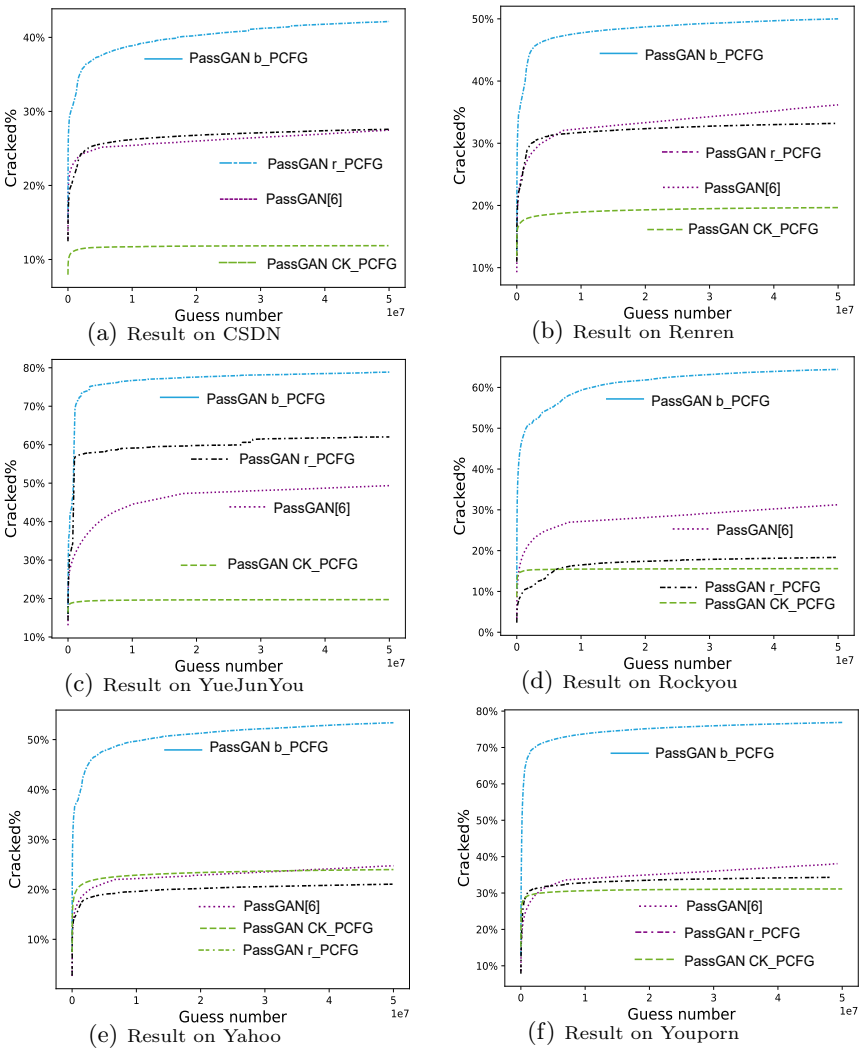
(c) Result on YueJunYou (d) Result on Rockyou

(e) Result on Yahoo (f) Result on Youporn

Fig. 5. Cracking results of LSTM based models with five different preprocessing methods (Guess number $=5 * 10^7$).The training set and test set are from the same dataset, with a division ratio of 8:1. LSTM one-hot represents the original model using one-hot for encoding [17]; LSTM 4-dim means using our new encoding method; b_PCFG means with basic PCFG for preprocessing [17]; r_PCFG means with our refined PCFG; CK_PCFG means with the Chunk+PCFG method. The results show that our refined PCFG method increases the effect by 9.21% on average for original LSTM model [17].

tested on six datasets, which are divided into the training set, test set, validation set according to 8:1:1 as recommended in [17].

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).



(a) Result on CSDN (b) Result on Renren

(c) Result on YueJunYou (d) Result on Rockyou

(e) Result on Yahoo (f) Result on Youporn

Fig. 6. Cracking results of GAN based models with four preprocessing methods (Guess number $= 5 * 10^7$). The training set and test set are from the same dataset, with a division ratio of 8:1. PassGAN represents the original model in [5]; b_PCFG means with basic PCFG for preprocessing; r_PCFG means with our refined PCFG; CK_PCFG means with the Chunk+PCFG method. The results show that using basic PCFG for preprocessing increases the effect by 26.79% on average for PassGAN model [5].

5.2 Evaluation results

To extensively evaluate the effect of five different preprocessing methods, 50 million guessing passwords are generated from each model and sorted by probability in order to obtain a smooth curve of the result. We use the guess-number-graph and record cracking results to intuitively reflect the effects of different models.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Overall analysis. Table 5∼6 and Fig. 5∼6 show that, with 50 million guessing passwords: (1) The LSTM based model using one-hot encoding outperforms using our new encoding by 4.22% on average; (2) The LSTM based model using the basic PCFG for preprocessing (short for PL) outperforms the LSTM based model using one-hot encoding by 7.85% on average; (3) Refined PL model outperforms the basic PL model by 1.36% on average; (4) The basic PL model outperforms the LSTM based model using Chunk+PCFG for preprocessing (short for CKPL) by 6.49% on average; (5) The PassGAN model using the basic PCFG for preprocessing (short for PCFG+PassGAN) outperforms the original PassGAN model by 26.79% on average; (6) The PCFG+PassGAN model outperforms refined PCFG+PassGAN model by 1.41% on average; (7) The original PassGAN model outperforms the PassGAN model using Chunk+PCFG for preprocessing (short for CKP+PassGAN) by 13.85% on average.

Table 5. Cracking results of LSTM based models with five different preprocessing methods (Guess number $=5 * 10^7$)†

Dataset	Language	LSTM one-hot [17]	LSTM 4-dim	LSTM b_PCFG [17]	LSTM r_PCFG	LSTM CK_PCFG
CSDN	Chinese	39.81%	37.73%	43.54%	43.89%	37.30%
YueJunYou	Chinese	60.72%	59.53%	78.29%	78.93%	61.78%
Renren	Chinese	50.72%	48.01%	46.11%	47.27%	53.41%
Rockyou	English	52.40%	46.71%	63.17%	67.11%	46.22%
Yahoo	English	40.66%	35.31%	44.37%	45.04%	48.62%
Youporn	English	50.42%	42.11%	66.36%	67.76%	55.53%

† LSTM 4-dim means using our new encoding method; b_PCFG means with basic PCFG for preprocessing [17]; r_PCFG means with our refined PCFG; CK_PCFG means with the Chunk+PCFG method. The results show that our refined PCFG method increases the effect by 9.21% on average for original model [17].

Encoding method. The experimental results on six datasets show that our new encoding method does not perform well, which can be attributed to two reasons. First, the sparsity problem caused by the one-hot encoding may not have serious side effects because passwords are generally short in length. Second, since our new encoding method is only used to represent each character, its advantage which reflects multiple character features has not been fully utilized.

PCFG based preprocessing method. Using PCFG for preprocessing can improve the effect, and the LSTM based model with our refined PCFG is even better than with the basic PCFG, which indicates that the fine-grained rules can extract more effective features. However, PassGAN with our refined PCFG decreases the effect due to the complexity of our method. Furthermore, since the Chunk+PCFG method first performs PCFG on each chunk of the passwords, it would generate even more complicated base structures than our refined PCFG, which can be the reason why all models using the Chunk+PCFG preprocessing

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Table 6. Cracking results of GAN based models with four different preprocessing methods (Guess number $=5 * 10^7$)†

Dataset	Language	PassGAN [5]	PassGAN b_PCFG	PassGAN r_PCFG	PassGAN CK_PCFG
CSDN	Chinese	27.47%	42.14%	27.58%	11.87%
YueJunYou	Chinese	49.35%	78.88%	62.01%	19.72%
Renren	Chinese	36.17%	49.99%	33.20%	19.65%
Rockyou	English	31.25%	64.41%	18.35%	15.58%
Yahoo	English	24.74%	53.38%	21.04%	23.96%
Youporn	English	38.07%	76.88%	34.31%	31.11%

† PassGAN b_PCFG means with basic PCFG for preprocessing; PassGAN r_PCFG means with our refined PCFG; CK_PCFG means with the Chunk+PCFG method. The results show that the basic PCFG method increases the effect by 26.79% on average for PassGAN model [5].

method do not perform well compared to the original models. To explore the impact of different extraction rules added to the basic PCFG, we set up a series of experiments where only one extraction rule is added each time. The results (see deatails in Table 10 of Appendix B) indicate that word recognition has the best performance among all four rules and using separate recognition rules is not as good as our refiend PCFG (i.e., with all four rules).

Limitations. Firstly, our new encoding method can not improve the effect of the LSTM based models compared with the one-hot encoding, which may indicate that password guessing models can not be improved only by the character encoding. Secondly, we only improve the effect of PassGAN from the aspect of preprocessing, but not change the structure of PassGAN.

Future directions. Firstly, new training methods to match up with our new encoding method can be a viable direction due to the underutilization of character features. Secondly, the research on PCFG based preprocessing method should be more fine-grained. One viable direction is to add more fine-grained recognition rules based on our refined PCFG, and another direction is to apply natural language processing(NLP) technology to extract the characteristics of passwords. Thirdly, this paper contributes to a better understanding of deep-learning based guessing models in that: preprocessing indeed can effectively enhance the use of the neural networks' learning ability, which is in turn intrinsically determined by the deep-learning model's network structure. For instance, the text learning ability of GAN is weaker than that of LSTM, which leads to the poor effect of fine-grained preprocessing methods integrated with GAN. Thus, the ability of GAN to learn text features may be improved by changing its structure, for example, using LSTM to compose the generator of GAN.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

6 Conclusion

This paper studies the deep-learning based password guessing models from the aspect of preprocessing. Firstly, considering the limitations of the one-hot encoding method, we propose a new encoding method that comprehensively reflects the character features. Secondly, considering that basic PCFG does not fully extract the password features, we propose a refined PCFG with comprehensive recognition rules. Thirdly, we adopt the idea of chunk segmentation at CCS'21, and apply the chunk+PCFG preprocessing method to LSTM and GAN.

Extensive experimental results show that: 1) Our refined PCFG outperforms the basic PCFG by 1.36% on average when integrated with LSTM; 2) Using basic PCFG for preprocessing improves the effect of the PassGAN model drastically by 26.79% on average; 3) Although our new encoding method does not improve the effect compared with the one-hot encoding, it still provides a feasible new research direction; 4) The performance of Chunk+PCFG preprocessing method is not ideal due to the complexity of its base structures.

Our results suggest that using PCFG for preprocessing is an effective way to improve the deep-learning based guessing models. Still, it should be used with care: although more fine-grained PCFG (e.g., our refined PCFG and Chunk+PCFG) extracts the passwords more comprehensively, it also generates more complicated base structures, which increases the training complexity for neural networks, and may even reduce the cracking rates for these neural networks with weak text feature learning ability.

Acknowledgment

The authors are grateful to the anonymous reviewers for their invaluable comments. Ding Wang is the corresponding author. This research was in part supported by the National Natural Science Foundation of China under Grant No.62172240, and by the Natural Science Foundation of Tianjin, China under Grant No. 21JCZDJC00190. There is no competing interests.

Appendix 1 Some statistics about user-chosen passwords

The length distributions of each dataset are shown in Table 7. Most passwords' length are between six and nine (avg. 73.81%). The length distribution is affected by the password policy. For example, CSDN dataset has much fewer passwords of length under eight as compared to other datasets, which may be caused by the fact that CSDN website changed the password policy to a more strict one. The character composition information is summarized in Table 8. Chinese users prefer to use digits in passwords, while English users prefer to use letters. This may be caused by cultural differences because most Chinese users use more digits in their daily lives than English words. In addition, English users prefer lowercase letters rather than uppercase letters. The top-10 passwords information is shown in Table 9. The password "123456" is the most commonly used password except for CSDN (due to its password policy). It is also interesting to see that the top-10 passwords in Chinese datasets are almost all pure digits.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Table 7. Length distribution information of each web service.

Dataset	1-5	6	7	8	9	10-16	17-30	30+
CSDN	0.63%	1.29%	0.26%	36.38%	24.15%	36.98%	0.32%	0.00%
YueJunYou	3.09%	24.00%	22.59%	24.13%	13.12%	13.05%	0.01%	0.00%
Renren	6.63%	25.36%	18.18%	20.24%	12.05%	17.20%	0.32%	0.00%
Rockyou	4.31%	26.04%	19.29%	19.98%	12.11%	17.86%	0.40%	0.01%
Yahoo	10.33%	17.86%	14.36%	25.03%	12.39%	20.04%	0.00%	0.00%
Youporn	11.44%	26.66%	16.17%	20.40%	10.82%	14.26%	0.22%	0.02%
Avg-CN†	3.45%	**16.89%**	**13.68%**	**26.92%**	**16.44%**	22.41%	0.22%	0.00%
Avg-EN	8.69%	**23.52%**	**16.60%**	**21.80%**	**11.77%**	17.39%	0.21%	0.01%
Avg-total	6.07%	20.20%	15.14%	24.36%	14.11%	19.90%	**0.21**%	**0.01**%

† Avg-X stands for the average proportion of X datasets. For example, where CN stands for three Chinese datasets and EN stands for three English datasets.

Table 8. Character composition information of each web service∗.

Dataset	[a-z]+	[A-Z]+	[A-Za-z]+	[0-9]+	[a-zA-Z0-9]+	[a-z0-9]+	[a-z]+1	[0-9a-z]+
CSDN	11.64%	0.47%	12.35%	45.01%	96.31%	26.14%	0.24%	5.88%
YueJunYou	12.94%	0.23%	13.39%	65.86%	99.38%	13.10%	0.25%	2.88%
Renren	19.06%	0.64%	20.55%	53.05%	97.79%	17.83%	1.24%	2.80%
Rockyou	41.68%	1.50%	44.04%	15.93%	96.19%	27.69%	4.55%	2.53%
Yahoo	32.51%	1.70%	35.90%	19.80%	97.99%	27.14%	3.47%	3.32%
Youporn	45.94%	1.04%	48.42%	20.12%	96.50%	20.59%	2.75%	1.91%
Avg-CN†	14.55%	0.45%	**15.43%**	**54.64%**	97.83%	19.02%	0.58%	3.85%
Avg-En	40.04%	1.41%	**42.79%**	**18.62%**	96.89%	25.14%	3.59%	2.59%
Avg-total	27.30%	0.93%	29.11%	36.63%	**97.36**%	22.08%	2.08%	3.22%

∗ Note that the first row is written in regular expressions. For instance, [a-z]+ means passwords composed of lower-case letters; [A-Za-z]+ means passwords composed of letters; [a-z]+1 means passwords composed of lowercase letters, followed by the digit 1.
† Avg-X stands for the average proportion of X datasets, where CN stands for three Chinese datasets and EN stands for three English datasets.

Table 9. top-10 password information of each web service.

Rank	CSDN	YueJunYou	Renren	Rockyou	Yahoo	Youporn
1	123456789	123456	123456	123456	123456	123456
2	12345678	111111	123456789	12345	123456789	123456789
3	11111111	0	111111	123456789	password	12345
4	dearbook	123456789	0	password	null	1234
5	00000000	123123	123123	iloveyou	12345	password
6	123123123	5201314	5201314	princess	12345678	qwerty
7	1234567890	wangyut2	12345	1234567	1234567	12345678
8	88888888	12345678	12345678	rockyou	iloveyou	123
9	111111111	123	123	12345678	qwerty	1234567
10	147258369	123321	123321	abc123	comeon11	111111
Top-1 %	3.66%	4.78%	3.74%	0.89%	0.86%	2.57%
Top-3 %	8.15%	7.11%	4.99%	1.37%	1.35%	3.71%
Top-10 %	**10.43%**	9.99%	**7.18%**	**2.05%**	2.13%	**5.29%**
Top-10 %	670,881	535,884	339,639	669,126	119,864	113,702
Total num	6,428,277	5,365,338	4,733,366	32,603,388	5,626,485	2,148,224

† Top-x per means the percentage of Top-x passwords, top-10 num means the total number of top-10 passwords.

This work will appear in Proc. of the 24th
International Conference on Information and Communications Security (ICICS 2022).

Appendix 2 Exploratory experiments

In Sec.4.3, Probabilistic context-free grammars (i.e., PCFG) [10,18] can be used for data preprocessing when integrated with neural networks. Our refined PCFG are based on the basic PCFG with four additional recognition rules, including keyboard pattern, word, website and year. The experiment result in Sec.5.2 has already shown that our refiend PCFG can improve the performance by 1.36% on average compared to the basic PCFG when integrated with Long Short-Term Memory neural networks (i.e., LSTM) [17]. To explore the impact of different recognition rules on the experiment results, we evaluate the performance of LSTM based models using PCFG for preprocessing, where only one recognition rule is added to basic PCFG each time.

The result in Table 10 shows that compared to the LSTM based model with basic PCFG for preprocessing: (1) Using PCFG with additional word recognition for preprocessing has a 0.26% improvement on average; (2) Using PCFG with additional keyboard recognition for preprocessing has a 0.06% improvement on average; (3) The remaining recognition rules (i.e., website and year) have little improvement on the results (less than 0.01% on average). In general, adding one recognition rule to the basic PCFG [10] alone is not as effective as adding all the rules (i.e., our refined PCFG) when integrated with LSTM. The reason why the year recognition rule has the worst performance can be can be attributed to two reasons. Firstly, years are part of birthdays and birthdays vary widely among users, which has little effect on trawling password guessing attack. Secondly, individual year segments can be replaced by digit segments. Moreover, the promotion effect of different recognition rules to some extent reflects the pattern that users tend to use when creating passwords.

Table 10. Cracking results of LSTM based models using PCFG based preprocessing methods (Guess number $=5 * 10^7$)†

Dataset	Language	Basic [17]	Keyboard	Word	Website	Year
CSDN	Chinese	43.54%	43.68%	43.63%	43.55%	43.36%
YueJunYou	Chinese	78.29%	78.85%	78.84%	78.84%	78.34%
Renren	Chinese	46.11%	45.99%	46.58%	46.05%	46.46%
Rockyou	English	63.17%	63.10%	63.15%	63.11%	63.09%
Yahoo	English	44.37%	44.38%	44.57%	44.19%	44.53%
Youporn	English	66.36%	66.20%	66.65%	66.18%	66.13%

† Basic means the LSTM based model with basic PCFG for preprocessing [17]; Keyboard means adding keyboard recognition rule to the basic PCFG; Word means adding word recognition rule to the basic PCFG; Website means adding website recognition rule to the basic PCFG; Year means adding year recognition rule to the basic PCFG. The experiment setup is the same as Sec.5.

References

1. Blocki, J., Harsha, B., Zhou, S.: On the economics of offline password cracking. In: Proc. IEEE S&P 2018. pp. 853–871
2. Bonneau, J., Herley, C., Van Oorschot, P.C., Stajano, F.: The request to replace passwords: A framework for comparative evaluation of web authentication schemes. In: Proc. IEEE S&P 2012. pp. 553–567
3. Bonneau, J., Herley, C., Van Oorschot, P.C., Stajano, F.: Passwords and the evolution of imperfect authentication. Comm. ACM 58(7), 78–87 (2015)
4. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Proc. NIPS 2017. pp. 5769–5779
5. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: Passgan: A deep learning approach for password guessing. In: Proc. ACNS 2019
6. Houshmand, S., Aggarwal, S., Flood, R.: Next gen pcfg password cracking. IEEE Trans. on Information Forensics and Security 10(8), 1776–1791 (2015)
7. Li, Z., Han, W., Xu, W.: A large-scale empirical analysis of chinese web passwords. In: Proc. USENIX Security 2014. pp. 559–574
8. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 (2015)
9. Liu, Y., Xia, Z., Yi, P., Yao, Y., Xie, T., Wang, W., Zhu, T.: Genpass: A general deep learning model for password guessing with pcfg rules and adversarial generation. In: Proc. ICC 2018. pp. 1–6
10. Ma, J., Yang, W., Luo, M., Li, N.: A study of probabilistic password models. In: Proc. IEEE S&P 2014. pp. 689–704
11. Melicher, W., Ur, B., Komanduri, S., Bauer, L., Christin, N., Cranor, L.F.: Fast, lean and accurate: Modeling password guessability using neural networks. In: Proc. USENIX SEC 2017. pp. 1–17
12. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: Proc. ACM CCS 2005. pp. 364–372
13. Rodríguez, P., Bautista, M.A., Gonzàlez, J., Escalera, S.: Beyond one-hot encoding: Lower dimensional target embedding. Image and Vision Computing 75, 21–31 (2018)
14. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Trans. Inf. Forensics Secur. 12(11), 2776–2791 (2017)
15. Wang, D., Wang, P., He, D., Tian, Y.: Birthday, name and bifacial-security: Understanding passwords of Chinese web users. In: Proc. USENIX SEC 2019
16. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: An underestimated threat. In: Proc. ACM CCS 2016. pp. 1242–1254
17. Wang, D., Zou, Y., Tao, Y., Wang, B.: Password guessing based on recurrent neural networks and generative adversarial networks. Chinese Journal of Computers pp. 1519–1534 (2021)
18. Weir, M., Aggarwal, S., de Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: Proc. IEEE S&P 2009. pp. 391–405
19. Xie, Z., Zhang, M., Yin, A., Li, Z.: A new targeted password guessing model. In: Proc. ACISP 2020. pp. 350–368
20. Xu, M., Wang, C., Yu, J., Zhang, J., Zhang, K., Han, W.: Chunk-level password guessing: Towards modeling refined password composition representations. In: Proc. ACM CCS 2021. pp. 5–20
21. Yang, K., Hu, X., Zhang, Q., Wei, J., Liu, W.: Studies of keyboard patterns in passwords: Recognition, characteristics and strength evolution. In: Proc. ICICS 2021. pp. 153–168