Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

GENPass: A Multi-Source Deep Learning Model For Password Guessing

Zhiyang Xia*, Ping Yi*, Yunyu Liu*, Bo Jiang*, Wei Wang†, Ting Zhu†
*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
†Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, MD, 21250, USA

***Abstract*—The password has become today's dominant method of authentication. While brute-force attack methods such as HashCat and John the Ripper have proven unpractical, the research then switches to password guessing. State-of-the-art approaches such as the Markov Model and probabilistic context-free grammar (PCFG) are all based on statistical probability. These approaches require a large amount of calculation, which is time-consuming. Neural networks have proven more accurate and practical in password guessing than traditional methods. However, a raw neural network model is not qualified for cross-site attacks because each dataset has its own features. Our work aims to generalize those leaked passwords and improves the performance in cross-site attacks.**

In this paper, we propose GENPass, a multi-source deep learning model for generating "general" password. GENPass learns from several datasets and ensures the output wordlist can maintain high accuracy for different datasets using adversarial generation. The password generator of GENPass is PCFG+LSTM (PL). We are the first to combine a neural network with PCFG. Compared with Long short-term memory (LSTM), PL increases the matching rate by 16%-30% in cross-site tests when learning from a single dataset. GENPass uses several PL models to learn datasets and generate passwords. The results demonstrate that the matching rate of GENPass is 20% higher than by simply mixing datasets in the cross-site test. Furthermore, we propose GENPass with probability (GENPass-pro), the updated version of GENPass, which can further increase the matching rate of GENPass.

I. Introduction

Deep learning [1] astonished the world after AlphaGo [2] beat Lee Sedol. The result proves that computers can emulate and surpass human beings after establishing a suitable model. Therefore, we try to explore a method to apply deep learning to password guessing in this paper.

The password is the main trend [3] for authentication. People are used to setting passwords with a certain combination (e.g., abc123456) instead of commonly used sequences (e.g., 123456). There have been many large-scale password leaks [4]–[7] in major companies. These leaked passwords provide us with rich training and testing datasets and allow us to learn more about the logic behind human-set combinations.

Many researches as well as open source tools aim to generate a large wordlist to match larger numbers of real passwords. HashCat [8] and John the Ripper(JtR) [9] are two remarkable password cracking tools, but they both rely on basic rules such as reversing, appending, truncating, etc. They can only generate a finite wordlist, which is far from useful for users to crack most passwords. Markov models [10] and probabilistic context-free grammar [11] are widely used techniques for password guessing. Both are based on probability models. Therefore, they are often computationally intensive and time-consuming [12]. Melicher et al. [13] first introduced a neural network to guess passwords and Hitaj et al. [14] recently proposed passGAN, using a generative adversarial network (GAN) to increase accuracy. However, both Melicher et al. and Hitaj et al. only focused on one-site tests. For example, RockYou is used for both training and testing datasets. The cross-site attack is a common method to crack a database. Although their performances are good, the generated password lists are not general.

To generate a general wordlist, we proposed a new model called GENPass. GENPass can enhance both accuracy and generality. Accuracy is improved by using the PCFG+LSTM model, the generator of GENPass. PCFG+LSTM is abbreviated as PL hereafter. We use PCFG rules to replace sequences with tags. The tagged-sequences are fed to a LSTM network for training and generation. The LSTM network has been proven effective at password guessing [13]. By using PCFG rules, the number of guesses can be reduced by 5-6 orders of magnitude when achieving a 50% matching rate. Generality means that the output wordlist can achieve a relatively high matching rate for different datasets. Our results also indicate that if we simply use the model trained with one dataset to estimate another, the matching rate will be much lower. Thus, we propose GENPass to improve generality.

GENPass implements the adversarial idea in the process of generation. It enables the model to generate a wordlist based on several sources. Therefore, the new generated wordlist is more eclectic. The model consists of several generators and a classifier. Generators create passwords from datasets, namely leaked passwords, and the task of the classifier and the discriminator is to make sure the output does not belong to a specific dataset so that the output is believed to be general to all training datasets. Furthermore, we take the importance of each training dataset into consideration and propose the updated version of GENPass which we call GENPass with probability. It performs even better than the original version of GENPass.

The contributions of this paper are as follows.

- We propose the PCFG+LSTM(PL) model. It can elevate the model from character-level (char-level) to word-level and thus significantly increase the matching rate of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

LSTM.
- We propose the GENPass model. It can generate a "general" wordlist by learning several leaked password datasets. With GENPass, we can extend the scale of training sets from one to many.

The remainder of this paper is structured as follows. Section II briefly introduces traditional approaches and previous research in the password guessing field. Section III describes the model design of GENPass. In Section IV, we evaluate GENPass by comparison with state-of-art methods and other neural network models. We conclude the paper and discuss further work in Section V.

II. Related Work

In this section, we first describe traditional approaches to crack passwords and analyze their disadvantages. Then we introduce the recent development of deep learning and its use in password guessing.

A. Traditional Password Guessing Methods

Plain text passwords are never allowed to be transferred in a network or be stored in databases [15] in a secure system. Hashing passwords is a common guessing method. This is the reason why HashCat and JtR became popular. They can greatly accelerate hash code computing. HashCat even supports GPU acceleration [8]. However, such password crackers must be based on a given password wordlist. The size of the wordlist determines the upper limit for one cracking attempt. HashCat uses a rule-based attack mode to enlarge the password list. Rules involve lowercasingreversing, appending, or truncating characters. As far as we know, no single wordlist has been widely regarded as the best one to crack passwords. We consider that the final goal of password guessing is to establish a general wordlist that can crack more passwords.

In recent years, text password authentication alone has been considered unsafe, and other methods, such as the short message service authentication code, are used as assisting tactics. However, in some specific areas, a text password is the only way to verify a user's identity. For instance, WPA-PSK [16] is a common WI-FI authentication protocol [17]. Attackers can easily capture users' handshake packages by using a monitor-mode network card. The password in the handshake package is hashed. Attackers can use Aircrack-Ng [18] or Pyrit [19] to crack passwords, just as HashCat and JtR do. With current computing capability, cracking an 8-length password by a brute-force method may take years. So studying a general password wordlist is an essential task.

The Markov model was used for password guessing in 2005 by Narayanan et al. [20] for the first time and an improvement has been proposed recently [10]. The core concept of the Markov model is a time-space tradeoff, using very large amounts of space to calculate the probability of the next character based on previous or current context.

Probabilistic Context-Free Grammar(PCFG) was introduced in password guessing in 2009 by Weir et al. [11]. PCFG derives from word-mangling rules based on a training set of leaked passwords. The grammar examples then generate passwords with a trained probabilistic model. PCFG is practical because users seldom set random characters as their password. Instead, they set their passwords with short strings for convenience, such as "iloveyou", "password123" and "123456". Thus, passwords contain a certain regularity [21]. Regularity can be preserved by encoding these passwords with PCFG rules. PCFG rules change passwords like "password123#" to a tagged-sequence such as "L8 D3 S1", where 'L' stands for letters, 'D' stands for digits, 'S' stands for special chars and the number stands for the length. This approach was extended to targeted password guessing [22], [23], which means the grammar is not only at the char-level, but also involves personally identifiable information(PII). However, it is difficult to build the relationship between a person's information to his passwords. Furthermore, it is illegal to collect a large amount of personal information for research.

B. Password Analysis

Password analysis aims to gain an insight into human habits for setting passwords. By observing passwords from leaked datasets, we discover that users do not set combinations randomly, but instead tend to create meaningful strings as a part of a password or the complete password. Li Y. et al. [25] studied the relationship between personal information and human-chosen passwords. Their test on the 12306 dataset showed that 24.10% of Chinese users used their dates of birth when creating passwords. Meanwhile 23.60% and 22.35% of users created passwords using account names and real names, respectively. Pearman et al. [24] recorded the statistics of 154 participants over 147 days. When browsing websites in different categories, nearly 85% of participants reused over 70% of their passwords and 61% of participants exactly or partially reused passwords across websites. Substrings with length 4 are most used, followed by length 7 and 8. Li Z. et al. [21] studied different patterns in passwords between English and Chinese. According to their statistical data, Chinese prefer digits while English users prefer letters when setting passwords. Additionally, Chinese Pinyin and English words vary in character composition. Li et al. presented that adding Pinyin into PCFG can increase the accuracy of password guessing by 34%. Analyzing passwords is a very important task to learn the patterns of creating passwords. It guides us to design the models for password guessing.

C. Development of Neural Networks

A neural network is inspired by human neurons. It can perform tasks by learning examples without requiring task-specific coding. Deep learning generally involves neural networks with multi-layers, exemplified by the extraordinary work by LeCun Yann, Yoshua Bengio, and Geoffrey Hinton [1]. Owing to the explosion of computing power, deep learning has outperformed all traditional methods in some areas. Deep learning is being applied to a variety of fields, such as target recognition [26], natural language processing [12], [27],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

and artificial intelligence [2]. The network we used to learn passwords is a recurrent neural network (RNN) [28]. It is impossible to fully understand the current words regardless of the context words. Just as how people read, the RNN generates the probability of the next element based on the context elements [28], and the model then outputs the element with the highest probability.

LSTM [29] is a widely used RNN variant. LSTM was first proposed by Hochreiter & Schmidhuber in 1997 and was improved by Graves [28] in 2013. Each LSTM unit maintains a state c_t at time t. Three sigmoid gates control the data flow in the unit, namely the input gate i_t, the forget gate f_t, and the output gate o_t. The output h_t is calculated as follows:

$$\mathrm{i}_t = \sigma(U_i X_t + W_i h_{t-1}) \tag{1}$$

$$\mathrm{f}_t = \sigma(U_f X_t + W_f h_{t-1}) \tag{2}$$

$$\mathrm{o}_t = \sigma(U_o X_t + W_o h_{t-1}) \tag{3}$$

$$\tilde{c}_t = \tanh(U_c X_t + W_c h_{t-1}) \tag{4}$$

$$\mathrm{c}_t = \mathrm{f}_t \cdot \mathrm{c}_{t-1} + \mathrm{i}_t \cdot \tilde{c}_t \tag{5}$$

$$h_t = \tanh(\mathrm{c}_t \cdot o) \tag{6}$$

where X_t and $\tilde{c}_t$ stand for the input X at time t and the updated state to be transferred to next unit, respectively, σ represents the logistic sigmoid function, $\cdot$ denotes the element-wise multiplication, and $U_{\{i,f,o,c\}}$ and $W_{\{i,f,o,c\}}$ are sets of weight matrices.

The GAN [30] is another remarkable advance in deep learning. GAN consists of a generative model G and a discriminative model D, and GAN trains the two models simultaneously. The generative model G keeps generating "fake" data and the discriminative model D judges whether the data comes from G or the training data. The loss of D will be fed back to G to help optimize the generator. GAN has achieved success in image processing. However, its success is quite limited in text areas [31], [32]. In this paper, we borrow the idea of adversarial generation, because we find it difficult to transfer the feedback to a multi-training model.

D. Neural Network in Password Guessing

A neural network was first used in password guessing by Melicher et al. [13] in 2016. LSTM [28] is an RNN generating one character at a time, similar to Markov model. However, the neural network method does not consume any extra space. The evaluation of [13] shows that a wordlist generated by a neural network outperforms the Markov model, PCFG, HashCat, and JtR, particularly at guessing numbers above 10^{10}. However, [13] restricted the structure of passwords (e.g., 1class8, 3class12), so the result cannot be considered general. PassGAN, proposed by Hitaj et al. [14], used GAN to generate passwords. GAN provides an adversarial process to make artificial passwords more realistic. With the help of GAN, PassGAN was able to create passwords that traditional password generation tools could not create.

Li et al. [21] also proved that there is a great difference between English and Chinese passwords, which was mentioned in Section II. We believe it is meaningless to mix Chinese passwords and English passwords together to generate a general password list. So our general model only focuses on one language environment. The problem on how to use the model learned from English passwords to guess Chinese passwords should be addressed by transfer learning [33], [34], which will not be discussed in this paper.

Both pioneering works have made extraordinary contributions in password guessing by using neural networks. But there are two common deficiencies in their theories, including PCFG. The first point is that they only focus on how to improve the matching rate for one-site tests. In other words, their training and testing sets are the same. So their models are not qualified to guess an unknown dataset. The second point is that their model can only learn and generate passwords from one dataset. PCFG [11] used Myspace, and passGAN [14] used RockYou. Melicher et al. [13] divided passwords into different structures and trained one type at a time. Such limitation restricts the model to learn more patterns of the same passwords. Our model GENPass is designed to solve these problems.

III. The Model of GENPass

In this section, we describe the problem we solve and discuss our thoughts regarding the model design and describe the detailed design of PCFG+LSTM (PL) and the GENPass model.

A. Problem Description

When we talk about password guessing, we are not going to find a way to crack a specific password. We try to improve the matching rate between the training and testing sets. There are two types of tests in password guessing. One is called a one-site test, in which the training and testing sets are the same. The other is called a cross-site test, in which the training and testing sets are different. We have mentioned that all the previous work can only train and generate passwords based on one dataset, so their performance in cross-site tests is poor. However, a cross-site attack is the main technique for a hacker to crack a database.

On the other hand, many real passwords have been exposed but each dataset is not large enough for deep learning models. We believe if these datasets are combined together, the model can predict password more like human.

In our work, we try to generate a "general" password list from several datasets to improve the performance of password guessing in cross-site tests. So we first define what is "general".

Definition 1: (What is "general")

Assume a training set T containing m leaked password datasets D_1, D_2, $D_3, \cdots, D_m$. Model G_t is obtained by training T. Model G_i is obtained by training D_i (i $\in$ [1, m]). If Model G_t can guess dataset D_k ($D_k \notin D_1$, D_2, D_3, $\cdots$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

D_m) better than G_i (i $\in$ [1, m]), model G_t is believed to be general.

Some examples are presented below to help understand the idea better.

1) Example 1: We train dictionary A, and obtain model G_A. Model G_A can guess dictionary A well but guess an unknown dictionary B poorly. This is what most of the previous works do and we do not consider G_A to be a general model.

2) Example 2: We train dictionaries A and B, and obtain model G_{AB}. Model G_{AB} can guess dictionary A well but guesses dictionary B poorly. We also do not consider G_{AB} to be a general model.

3) Example 3: We train dictionaries A and B, and obtain model G_{AB}. Model G_{AB} can guess not only A and B but also an unknown dictionary C well. Then we consider G_{AB} to be a general model.

We also introduce the concept of a matching rate to evaluate the model's capability to guess passwords.

Definition 2: (Matching Rate)

Suppose W is the wordlist generated by model G and the target dataset is D. We define the same passwords in both W and D as matched passwords. So the matching rate is the ratio between the number of matched passwords and that of D.

$$\mathrm{R}_G = \frac{|W \cap D|}{|D|} \tag{7}$$

In conclusion, our goal is to design a model that can generate a general password list based on several datasets to maximize the matching rate when we guess an unknown wordlist.

B. Reconsideration of Neural Networks

When using neural networks such as LSTM in [13], to generate passwords, the network starts with an empty string and calculates the probability of the next character. For example, when we generate "passw", the next char is most likely to be 'o'. However, this is an ideal situation. The reality is that, unlike traditional deep learning tasks, we cannot always choose the character with the highest possibility of appearance. Otherwise the output password list will contain many duplicates. There must be a random choice to avoid duplicates. The consequence of the random choice is that "password" cannot be easily generated letter by letter. Because once a character is not chosen correctly, the string will be totally meaningless. This is the shortcoming of char-level training with neural networks.

In previous research on password analysis, we find that people use some meaningful sequences to make passwords easier to memorize instead of using random sequences. This inspires us in that if we treat some fixed alphabetic sequences as a word, word-level training will be more suitable for password guessing. Thus, we analyzed some leaked password datasets to find a pattern.

We first try some short sequences. Table I shows the top 10 most frequent two-character sequences in Myspace [38].

TABLE I
TOP 10 MOST FREQUENT TWO-CHARACTER SEQUENCES IN MYSPACE

sequence	frequency	sequence	frequency
er	7,407	an	5,626
in	4,377	ar	3,988
st	3,962	on	3,811
ra	3,713	re	3,709
te	3,627	as	3,468

According to our statistics, there are 37,144 passwords in Myspace and the most frequent two-character sequence "er" appears 7,407 times. We define such sequence as a unit and in this manner many sequences with low probability can be pruned. For example, character 'e' appears 38,223 times. The top 10 most frequent two-character sequences starting with 'e' appear 26,378 times (69%). Combining 'e' with these characters as a unit and pruning others means we can use some typical sequences to represent all probabilities. Although this method ignores a few uncommon sequences, the model can in fact generate more general passwords.

However, the way to slice the password is not unique. For example, in Myspace, "ter" appears 1,137 times, "te" appears 3,627 times and "er" appears 7,407 times. These short sequences can all be treated as a common unit. The sequence "shelter" can be divided into "shelt" + "er" or "shel" + "ter" or "shel" + "te" + "r". When "shel" is generated, either "ter" or "te" is likely to be the next unit. "ter" is a correct choice, but if we choose "te", the target sequence is not guaranteed to be generated.

Cutting passwords into short sequences still has the same problem as with char-level training. To solve this problem, we try to implement the rules of PCFGs and propose the PCFG+LSTM model.

C. PCFG+LSTM (PL)

A regular password guessing method called PCFG [11] divides a password by units. For instance, 'password123' can be divided into two units 'L8' and 'D3'. This produces high accuracy because the passwords are always meaningful and are set with template structures (e.g., iloveyou520, password123, abc123456). Meanwhile, neural networks can detect the relationship between characters that PCFG cannot. Thus, we combine the two methods to obtain a more effective one.

1) Preprocessing: A password is first encoded into a sequence of units. Each unit has a char and a number. A char stands for a sequence of letters (L), digits (D), special chars (S), or an end character ('\n'), and the number stands for the length of the sequence (e.g., $Password123 will be denoted as S1 L8 N3 '\n'). Detailed rules are shown in Table II.

A table is generated when we preprocess the passwords. We calculate the number of each strings occurrence. For example, we calculated all the L8 strings in Myspace, and found that "password" occurred 58 times, "iloveyou" occurred 56 times and so on.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

TABLE II
DIFFERENT STRING TYPES

Data Type	Symbols
Letters	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
Digits	0123456789
End character	\n
Special Chars	otherwise

2) Generating: The LSTM model is a variant of the basic RNN model. We use LSTM to generate passwords. The structure of LSTM is exactly the same as [13]. By feeding the LSTM model the preprocessed wordlist and training it, the model can predict the next unit.

3) Weight Choosing: When a unit is determined, it will be transformed back into an alphabetic sequence according to the table generated during the preprocessing. Experiments have shown that if the highest weight output is chosen each time, there will be a large number of duplicates in the output wordlist. So we choose the unit by sampling the discrete distribution. Such random selection can ensure that higher weight candidates are chosen with a higher probability, while lower ones can still be chosen after a large number of guesses.

D. GENPass

PL model is designed to improve the performance when we train only one dataset. However, it is not good enough when we train several datasets at the same time. So we propose GENPass to solve the multi-source training problem. GENPass generates a general wordlist from different datasets. Because different datasets have different principles and different lengths, it is difficult to learn the general principles by simply mixing datasets. To solve this problem, we design GENPass as shown in Fig 1.

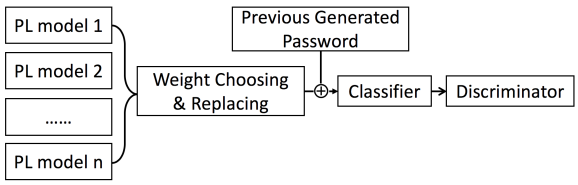

Fig. 1. **Diagram of GENPass model.** Units are first generated by separate PL models. The weight choosing process selects one specific unit from these units. The unit then is replaced by a random sequence. The sequence together with the previous generated password will be fed into the classifier. The classifier will judge which model the input belongs to. The validity of the unit depends on the judgment by the discriminator. If the discriminator is not able to tell which model the unit comes from, the unit will be accepted. Otherwise, it will be rejected.

1) Prediction of Model n: All the models are PL models, as mentioned in the previous section. We train each model separately. Thus, given an input, the model can output the result with its own principle.

2) Weight Choosing: We assume that every model has the same possibility, so the output of each model can be combined directly. The combined list is the input of the weight choosing process, and the output is a random choice. When a unit is determined, it will be replaced by a random alphabetic sequence, according to the table generated in preprocessing. To show the process of weight choosing more clearly, an example is shown in Fig 2.

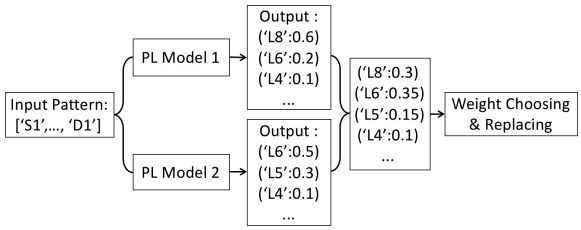

Fig. 2. **Example of weight choosing process.** The output of Model 1 is [(L8:0.6), (L6:0.2), (L4:0.1)...] and The output of Model 2 is [(L6:0.5), (L5:0.3), (L4:0.1)...]. Because both models have the same possibility, we directly combine these two list. So the input for weight choosing can be written as [(L8:0.3), (L6:0.35), (L5:0.15), (L4:0.1)...]. Therefore, L6 is most likely to be picked out in weight choosing process. When a unit is determined, it will be replaced by a corresponding random sequence.

3) Classifier: The classifier is a CNN classifier trained by raw passwords without preprocessing from different datasets. The CNN classifier is exactly the same as [35]. Given a password, the classifier can tell which dataset the password most likely comes from. Through a softmax layer, the output will be a list of numbers whose sum is one.

The classifier only treats complete passwords, not units or alphabetic sequences. It means only when the end character $\backslash n$ is generated, the classifier is activated. One single password is not long enough to extract features, we actually combine it with 4 previous generated complete passwords, 5 passwords over all, as the input to the classifier.

4) Discriminator: The discriminator is used to judge whether the password will be accepted. It takes the output of the classifier as its input. The discriminator should accept those passwords which have a consistent probability of appearance in different datasets so that the output passwords can be "general". In other words, the distribution of the classifier's output should be average. We use C of the classifier's output to judge the password's generality. Given the classifier's output $c_i (i = 1...N)$ where N is the amount of datasets, we define C as below.

$$C = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(c_i - \overline{c})^2} \tag{8}$$

$$\overline{c} = \frac{1}{N}\Sigma_{i=1}^{N} c_i \tag{9}$$

If C is too large, the lately generated unit will be discarded because it is probably a special case in one dataset. Otherwise, the unit will be accepted. In our model, we chose 0.2 as the threshold of C after performing several experiments. If the threshold is too small, the model will be much more time-consuming and the output will contain more duplicates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

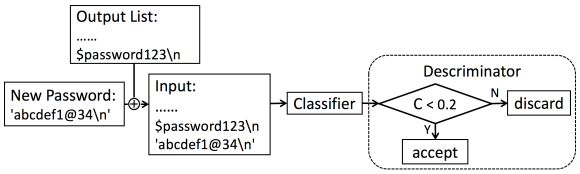

Fig. 3. **Detailed diagram of the classifier and the discriminator.** Assume that the new generated password is 'abcdef1@34'. It will be combined with 4 previous generated passwords as the input of the classifier. If the output of the classifier is [0.4,0.6], the discriminator will accept and print the password, because C is less than 0.2. If the output of the classifier is [0.3,0.7], all the units composing 'abcdef1@34' will be discarded.

E. GENPass with Probability

In the previous section, we assume that each dataset has the same probability. However, after some experiment we find some datasets are more typical, and passwords are more likely to come from those datasets, so we need to lower the opportunity for passwords from those typical datasets to be chosen to allow passwords from other datasets to have a greater chance of being selected. We introduce the concept of weights to solve this problem. Those typical datasets should have a low weight, which will be proven later. We develop a method to calculate appropriate weights. The generality is judged by whether the weight of datasets corresponds with the distribution of the classifier's output. We improve the weight choosing process and discriminator in the previous model as follows. We call this upgraded model GENPass with probability, GENPass-pro for short.

1) Weight Choosing Process with Probability: We assume that a probability distribution P represents each dataset's weight, where $\Sigma p = 1$. So the outputs of different models should be added according to their weights. The weights are the probabilities. According to the Bayes formula [36], the total probability of the output is 1.

2) Discriminator with Probability: Generality is judged by the distance between the weight of datasets P and the distribution of the classifier's output Q. Standard deviation cannot depict the distance between two distributions, so we introduce Kullback-Leibler divergence (KL divergence) [37] to depict distance. The KL divergence is calculated as follows.

$$\mathrm{D}_{KL}(P||Q) = \Sigma p(x) \ln \frac{p(x)}{q(x)} \tag{10}$$

During the training process, we use gradient descent to force P to approach Q. The distribution P is first generated randomly. In each iteration, ten new passwords are generated by GENPass with corresponding probability. These passwords are fed to the classifier and the results are recorded as $Q_i(i = 1...10)$. We denote the average of these results as the output of the classifier Q to eliminate random error. We use KL divergence as the loss function. Updating P by ascending $Q - P$ stochastic gradient times steps α makes P approach Q quickly and accurately. We assume steps of $\alpha = 1$. We use $P_i^{'} = \frac{P_i}{\Sigma P_i}$ to ensure the total probability is 1. We believe the distribution is stable when the loss is less than 0.001. The pseudo code is shown in Algorithm 1.

Algorithm 1: Calculate the distribution

```
Generate the distribution P randomly;
repeat
    generate 10 passwords by GENPass-pro without the
    discriminator, denote the outputs of the classifier as
    Q_i(i = 1...10);
    calculate the average of classifier's outputs
    Q = (1/10) Σ_{i=1}^{10} Q_i;
    Loss is the KL divergence of two distributions
    Loss = D_KL(P||Q);
    Update the distribution P by ascending its stochastic
    gradient time steps P = P + α × ∇_{Q−P} Loss;
    Make sure that the sum of P equals 1 by using
    P_i' = P_i / ΣP_i;
until the distribution is stable;
return distribution P;
```

When P is determined after training, the criterion for accepting the password is whether the KL divergence is smaller than 0.1, a value we set manually.

IV. Experiment and Evaluation

In this section, we first demonstrate our training and testing procedures. Then, we demonstrate the results of our experiments and analyze them.

A. Experimental Data

Experimental data involving leaked passwords are collected from public websites. The previous research studies are all based on these datasets. Li et al. [21] proved that passwords in different language environments have little generality. So in our experiments, we only choose English passwords. Our model is not designed to solve the generality between different languages. Our training and testing data are from Myspace [38], phpBB [39], RockYou [40], and LinkedIn [41]. [13] and [14] have proven that a neural network model outperforms traditional technologies such as HashCat [8] and JtR [9]. So we do not spend time on testing wordlists generated by HashCat and JtR. Detailed information is shown in Table III.

TABLE III
Contrast between different datasets

Name	Language	Address	Number
MySpace	English	http://www.myspace.com/	37,144
phpBB	English	http://www.phpbb.com/	184,389
RockYou	English	http://www.rockyou.com/	14,344,391
LinkedIn	English	http://www.linkedin.com/	60,143,260

B. Training and Testing Methods

1) PL: To evaluate PL, we trained the model with Myspace and phpBB. After each training session, the model generated a new wordlist. The tests can be divided into two types. One

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

is called a one-site test, in which the training and testing datasets are the same. The other is called a cross-site test, in which the training and testing datasets are different. Then, we enumerated the passwords in the generated wordlists to see if they were listed in the testing set and calculated the percentage, namely the matching rate. We randomly chose 10% of passwords in the testing dataset as test data. The final matching rate is the average of the results of the procedure above repeated 10 times.

2) GENPass: To evaluate the GENPass model, we trained the model with Myspace and phpBB. We calculated the matching rate using the same method described in the previous experiments. Then we compared the results with those of the PL model trained by a single wordlist. We also trained the PL model with a simple mixture of two wordlists and compared the result with the GENPass model.

C. Evaluation

1) PL: We respectively trained the PL and LSTM model with Myspace and phpBB and performed the one-site test. The result is shown in Fig 4. Note that LSTM is the same model as what is used in [13]. For example, PL(Myspace) means the training model is PL and the training and testing datasets are both Myspace. It is obvious that the PL performs much better than LSTM. Although the matching rates of both models will finally reach a higher level with more guesses, PL can achieve the same matching rate with much fewer guesses compared with LSTM. For instance, when the testing set was Myspace, LSTM guessed 10^{12} times to reach a 50% matching rate while PL only needed approximately 10^7 times. After generating 10^7 passwords, PL regenerates over 50% of the dataset while LSTM regenerates less than 10%. The result proves that the PCFG rules help to improve the efficiency of password guessing.

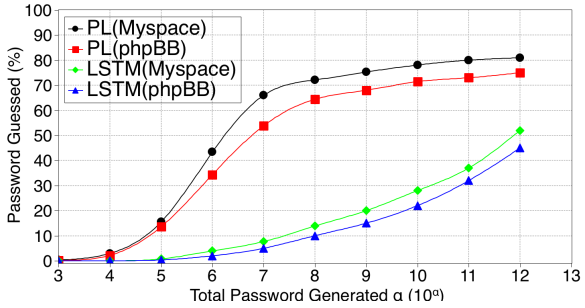

Fig. 4. **One-site tests of PL and LSTM.** PL and LSTM are the names of the models. Myspace and phpBB are the training and testing datasets. PL(Myspace) means the training model is PL and the training and testing datasets are both Myspace. The results show that PL can achieve the same matching rate with much fewer guesses compared with LSTM. So it can dramatically improve the matching rate.

We also find that the model trained with Myspace performs better than that with phpBB. This phenomenon indicates that datasets have differences. The model trained with different datasets obtained different results. A suitable explanation is that the features in Myspace are easier to capture than in phpBB because the number of passwords in Myspace is smaller, as shown in Table III.

Table IV shows the results of cross-site tests at 10^6 and 10^7 guesses. We respectively trained the PL and LSTM models with Myspace and phpBB and used the outputs of these models to test RockYou and LinkedIn. The cross-site tests always achieve worse performances than the one-site tests. This indicates that different datasets have their own regularity. Learning from only one dataset is not sufficient to crack others. During the cross-site tests, the PL model always achieves better performance than the LSTM model with an increase of 16%-30%. The result indicates that people are inclined to use some fixed strings, such as words and names, and this phenomenon is common in the leaked password datasets. PL is proven to be more effective in cross-site tests. However, the results are unsatisfactory due to the lack of generality.

TABLE IV
CROSS-SITE TEST RESULTS

	Password Generate	Myspace-RockYou	phpBB-RockYou	Myspace-LinkedIn	phpBB-LinkedIn
PL	10^6	1.10%	0.41%	0.30%	0.10%
	10^7	3.47%	1.12%	1.08%	0.38%
LSTM	10^6	0.65%	0.37%	0.22%	0.07%
	10^7	2.80%	0.96%	0.83%	0.30%

2) GENPass: In all GENPass's tests, the training datasets are Myspace and phpBB. Table V compares the GENPass model with the PL model whose training dataset is simply obtained by mixing Myspace and phpBB. The matching rate of GENPass is $40\% - 50\%$ higher than that of PL when the testing set is Myspace. It achieves a similar result when the testing set is phpBB. The PL model performs better when the size of the testing set is large. This result is possibly attributed to the following reason. The larger dataset has a deeper influence on the output wordlist. The smaller dataset can be regarded as a noise added to the larger one. For example, phpBB contains 106 '123456' out of 184,389 passwords, while Myspace contains 59 instances of 'password' out of 37,144 passwords. 'password' is more important than '123456'. But in the mixed dataset, 'password' plays a less important role than it does in the single, original dataset. Simply mixing two datasets, especially when the two datasets have very different numbers of passwords, disrupts the regularity of the individual datasets. This conclusion supports the necessity of the GENPass model. Our model overcomes these obstacles by choosing the output using two models.

Fig 5 and 6 show the results of cross-site tests when the models are GENPass, PL, and LSTM. GENPass uses Myspace and phpBB as training datasets, while the training datasets of PL and LSTM are specified in the parentheses. The test sets of Fig 5 and Fig 6 are RockYou and LinkedIn, respectively.

Taking both Fig 5 and 6 into consideration, the GENPass model outperforms all other models. Using raw LSTM without

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

TABLE V
DIFFERENCES BETWEEN GENPASS AND SIMPLY MIXING

Password Generated	GENPass -Myspace	GENPass -phpBB	PL(Simply Mixing) -Myspace	PL(Simply Mixing) -phpBB
10^6	31.58%	33.69%	19.22%	33.08%
10^7	55.57%	57.90%	38.87%	55.55%

any preprocessing performs the worst in two tests. Using PL to learn Myspace alone performs second best. This proves that Myspace is a good dataset. The passwords in Myspace are typical among other datasets. Simply mixing two datasets does not improve the matching rate. Instead, the matching rate will drop because of the bad dataset used, namely phpBB in this test. The results prove that GENPass can overcome the discrepancies between datasets and achieve a higher matching rate.

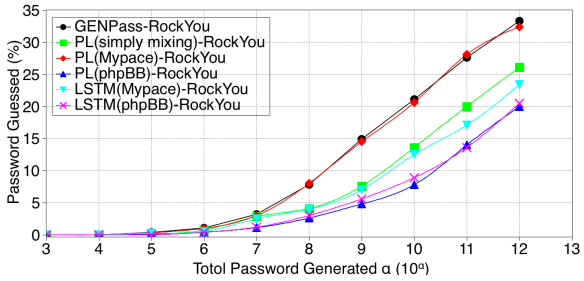

Fig. 5. **Cross-site tests of different models and different datasets.** GENPass, PL, and LSTM are the names of the models. Myspace and phpBB are the training datasets. The target dataset is RockYou. The results show that GENPass(Black) almost always performs best. The performance of GENPass is much better than simply mixing Myspace and phpBB using PL(Green).

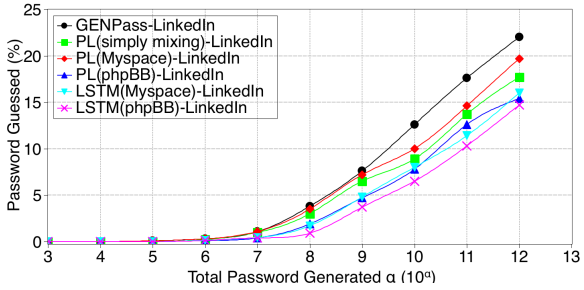

Fig. 6. **Cross-site tests of different models and different datasets.** GENPass, PL, and LSTM are the names of the models. Myspace and phpBB are the training datasets. The target dataset is LinkedIn. The results show that GENPass(Black) always performs best. GENPass can overcome the discrepancies between datasets and achieve a higher matching rate.

3) GENPass with probability: The GENPass model can generate a wordlist with higher matching rate by training different datasets. However, we found that GENPass does not always achieve best results. In Fig 5, there some difference between GENPass-RockYou and PL(Myspace)-RockYou.

The first reason for this observation is that the passwords generated by PL (Myspace) are more likely to appear in RockYou, and phpBB only contributes slightly. So simply choosing the unit from two training models cannot increase the matching rate. Different datasets should have different importance. For example, phpBB should be more important than Myspace when we generate the wordlist, because we need to consider phpBB with more emphasis to add diversity. To solve this problem, we applied the distribution P to GENPass and created GENPass with probability (GENPass-pro). Distribution P stands for the weights of different datasets.

The second reason is that using C as the criterion for the discriminator is not precise. We must encourage passwords to come from important datasets. C cannot reflect the importance among different datasets. For example, when "p@ssword" is generated, it seems more likely to occur in phpBB, as the output of the classifier is $(0.177, 0.823)$. The discriminator discards the password because C is over 0.2. However, "p@ssword" is one of the passwords in RockYou. To solve this problem, we use the KL divergence as the new criterion for the discriminator To design a discriminator with probability. The KL divergence is calculated based on Formula 10 where P stands for the weights of datasets and Q stands for the distribution of the classifier's output. Thus, "p@ssword" will be accepted by GENPass-pro.

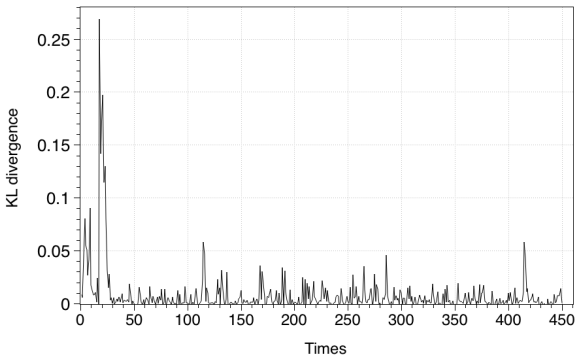

Fig. 7. **Training process for P and Q.** The x-axis is the number of training times and the y-axis is the KL divergence of P and Q. Initially, the KL divergence is large and then decreases quickly. After about 50 iterations, the KL divergence has dropped below 0.05 and it has stabilized.

By using Algorithm 1, we calculated the distribution P of Myspace and phpBB. The process of the calculation is shown in Fig 7, where the y-axis is the KL divergence and the x-axis is the number of iterations. In each iteration, the distribution P takes the KL divergence as the feedback and improves itself. Initially, the KL divergence is large and then decreases quickly. After 50 iterations, the KL divergence became smaller and remained stable. Once the divergence stabilized, we determine the distribution P. In this case, it was approximately 30.004% of Myspace and 69.996% of phpBB. This result confirms that phpBB is more important than Myspace. In the following experiments, we used this distribution P as the probability.

We trained and tested GENPass and GENPass-pro with the same training datasets and test datasets. The training datasets were Myspace and phpBB and the testing datasets

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2019.2940877, IEEE Transactions on Multimedia

TABLE VI
DIFFERENCES BETWEEN GENPASS AND GENPASS-PRO

	Password Generated	Myspace	phpBB	RockYou	LinkedIn
GENPass	10^6	31.58%	33.69%	0.86%	0.20%
	10^7	55.57%	57.90%	3.34%	1.11%
GENPass -pro	10^6	29.27%	34.16%	1.14%	0.40%
	10^7	55.74%	61.76%	4.10%	1.52%

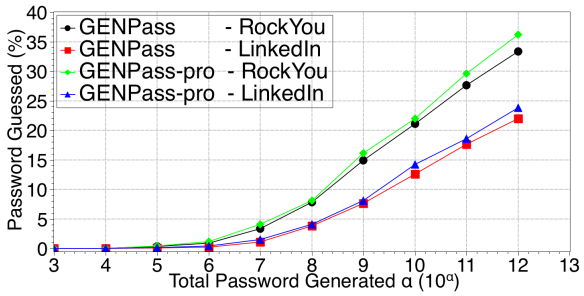

Fig. 8. **Comparing the matching rate between GENPass and GENPass-pro.** We use Myspace and phpBB to train GENPass and GENPass-pro respectively. The two testing sets are RockYou and LinkedIn. The result shows that GENPass-pro(Green & Blue) can increase 2%-3% matching rate compared with GENPass(Black & Red) when generating more than 10^{10} passwords.

were RockYou and Linkedin.

The results in Table VI show that within 10^6 guesses, the two models show similar matching rates. GENPass even has a better result when the test set is Mypsace. This is because a small number of guesses cannot fully show the capabilities of the models. When the number of the generated passwords reaches 10^7, the model with probability performs better than the original one. The matching rate of GENPass-pro is at least 20% higher than the matching rate of GENPass when the test sets are RockYou and LinkedIn. This is because using probability enables the model to choose units from the more important dataset. In this case, phpBB is more important than Myspace and the distribution P of (Myspace, phpBB) is $(30.004\%, 69.996\%)$. There is a 30.004% chance that the chosen unit comes from Myspace and a 69.996% chance comes from phpBB. The model with probability uses the KL divergence of the datasets' distribution P and the classifier's output Q to determine whether the unit should be output. The threshold of KL divergence is 0.1. If the classifier's output Q is $(0.1, 0.9)$, the unit will not be accepted, for the KL divergence is too large. And if the classifier's output Q is $(0.3, 0.7)$, it will be accepted. The result proves that the value of KL divergence is a legitimate criterion for the discriminator.

Moreover, we compared GENPass-pro with GENPass when generating more than 10^{10} passwords. The result is shown in Fig 8. GENPass-pro can achieve matching rate of 36.2% in RockYou and 23.8% in LinkedIn, which is high enough to prove the generalization ability of our model.

However, there remain some flaws which explain why GENPass-pro cannot achieve an even higher matching rate. There is a case when the distribution of the classifier's output Q is quite close but the discriminator does not permit the unit to go through because of the weights of the datasets. For example, the distribution P is $(30.004\%, 69.996\%)$ while the classifier's output Q is $(0.55, 0.45)$. This unit will be rejected, although it is actually a possible unit.

TABLE VII
ABILITY OF MODELS TO CREATE PASSWORDS

	Password Generated	RockYou	LinkedIn
PL	10^7	323,246 (2.254%)	522,250 (0.868%)
GENPass	10^7	389,453 (2.715%)	571,135 (0.950%)
GENPass-pro	10^7	**493,224 (3.438%)**	**800,388 (1.331%)**
PassGAN [15]	10,969,748	393,897 (2.746%)	703,075 (1.169%)
PL	10^6	65,585 (0.457%)	86,518 (0.143%)
GENPass	10^6	88,926 (0.620%)	112,874 (0.188%)
GENPass-pro	10^6	**115,417 (0.805%)**	**156,491 (0.260%)**
PassGAN [15]	1,357,874	82,623 (0.576%)	129,308 (0.215%)

Finally, our model's ability to create passwords is evaluated. Table VII shows the number of reasonable passwords which do not make their appearance in the training sets (Myspace and phpBB) but do so in the test set (RockYou or LinkedIn) when we generate 10^6 and 10^7 unique passwords. As a contrast, we also add the result of PassGAN when generating approximately the same number of passwords. The percentage in the table is calculated by the ratio between the number of passwords and the length of RockYou (14,344,391) or LinkedIn (60,143,260). In both tests, PassGAN cannot outperform GENPass-pro, although it performs better than PL and GENPass in most experiments. It is because when we use GENPass-pro, the model takes phpBB into better consideration, while PassGAN cannot handle such multi-source situation. The results convince us that more reasonable passwords will be created as the scale of the training set grows larger.

V. CONCLUSION

This paper introduces GENPass, a multi-source password guessing model to generate "general" passwords. It borrows from PCFG [11] and GAN [30] to improve the matching rate of generated password lists. Our results show that word-level (if tags such as "L4" "D3" "S2", are viewed as words) training outperforms character-level training [13], [14]. Furthermore, we implement adversarial generation from several datasets and the results indicate that multi-source generation achieves a higher matching rate when performing cross-site tests.

Our work is inspired by previous research, especially by the excellent work of Melicher et al. [13]. Before then, the research on password guessing mainly focused on innovations to the Markov model and PCFG. Melicher et al. first proved that a neural network achieved better performance than the probabilistic method. Neural networks are believed to have the capability to detect the relationships between characters that probabilistic methods do not. In this paper, we first

extend character-level training to word-level training by replacing letters, digits, and special chars with tags, namely, PCFG+LSTM(PL). In one-site tests, the PL model generated 10^7 passwords to achieve a 50% matching rate while the LSTM model must generate 10^{12} passwords. In cross-site tests, PL increased the matching rate by $16\% - 30\%$ compared with LSTM when learning from a single dataset. Our model also extends single source training to multi-source training and uses adversarial generation to ensure the output passwords are general to all datasets. The matching rate of GENPass is 20% higher than that of simply mixing several datasets at the guess number of 10^{12}. We use Myspace and phpBB as training data, RockYou and LinkedIn as testing data. The results are sufficient to prove that GENPass is effective in this task. Then, we show that GENPass still can be improved by considering probability. We call this GENPass with probability or GENPass-pro. This model solves certain issues, including the weights assigned to datasets and how to judge whether a password has a consistent probability of appearing in different datasets. The results show that the matching rate of GENPass with probability is 20% higher than GENPass when we generate 10^7 passwords. In conclusion, GENPass with probability is the most effective model for generating general passwords compared with other models. But it still has some flaws. For example, the discriminator is still far from perfect, which results in some passwords being rejected.

We believe password guessing deserves further research, as password authentication will not be totally replaced for some time. However, the scarcity of training resources remains a problem. Because of existing security protocols, we have limited access to actual passwords, especially those we need for training and testing. To solve this problem, studies on transfer learning may help by including additional databases in other languages.

VI. Acknowledgements

This work is supported by the National Natural Science Foundation of China(61571290, 61831007, 61431008), National Key Research and Development Program of China (2017YFB0802900, 2017YFB0802300, 2018YFB0803503), Shanghai Municipal Science and Technology Project under grant (16511102605, 16DZ1200702), NSF grants 1652669 and 1539047.

References

[1] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. *Deep learning*. Nature 521.7553 (2015): 436-444.
[2] Silver D, Huang A, Maddison C J, et al., *Mastering the game of Go with deep neural networks and tree search*. Nature, 2016, 529(7587):484.
[3] Herley C, Oorschot P V., *A Research Agenda Acknowledging the Persistence of Passwords*. IEEE Security & Privacy, 2012, 10(1):28-36.
[4] LinkedIn leak. https://hashes.org/public.php
[5] Myspace leak. https://wiki.skullsecurity.org/ Passwords#Leaked passwords.
[6] phpBB leak. https://wiki.skullsecurity.org/ Passwords#Leaked passwords.
[7] Rockyou leak. https://wiki.skullsecurity.org/ Passwords#Leaked passwords
[8] STEUBE, J. Hashcat. https://hashcat.net/oclhashcat/.
[9] PESLYAK, A. John the Ripper. http://www.openwall.com/ john/..
[10] Ma J, Yang W, Luo M, et al., *A Study of Probabilistic Password Models*. Security and Privacy. IEEE, 2014:689-704.
[11] Weir M, Aggarwal S, Medeiros B D, et al., *Password Cracking Using Probabilistic Context-Free Grammars*. Security and Privacy, 2009, IEEE Symposium on. IEEE, 2009:391-405.
[12] Lipton Z C, Berkowitz J, Elkan C., *A critical review of recurrent neural networks for sequence learning*. Computer Science, 2015.
[13] Melicher W, Ur B, Segreti S M, et al., *Fast, lean and accurate: Modeling password guessability using neural networks*. Proceedings of USENIX Security. 2016.
[14] Hitaj B, Gasti P, Ateniese G, et al., *PassGAN: A Deep Learning Approach for Password Guessing*. arXiv preprint arXiv:1709.00440, 2017
[15] Percival C, Josefsson S., *The scrypt Password-Based Key Derivation Function*. 2016.
[16] Liu Y, Jin Z, Wang Y. *Survey on Security Scheme and Attacking Methods of WPA/WPA2*. International Conference on Wireless Communications NETWORKING and Mobile Computing. IEEE, 2010:1-4.
[17] Sakib A K M N, Jaigirdar F T, Munim M, et al., *Security Improvement of WPA 2 (Wi-Fi Protected Access 2)*. International Journal of Engineering Science & Technology, 2011, 3(1).
[18] Aircrack-Ng. http://www.aircrack-ng.org
[19] Pyrit. Pyrit Github repository. https://github.com/JPaulMora/Pyrit
[20] Narayanan A, Shmatikov V., *Fast dictionary attacks on passwords using time-space tradeoff*. ACM Conference on Computer and Communications Security. ACM, 2005:364-372.
[21] Li Z, Han W, Xu W, *A Large-Scale Empirical Analysis of Chinese Web Passwords*. Usenix Security. 2014: 559-574.
[22] Wang D, Zhang Z, Wang P, et al., *Targeted Online Password Guessing: An Underestimated Threat*. ACM Sigsac Conference on Computer and Communications Security. ACM, 2016:1242-1254.
[23] Li Y, Wang H, Sun K., *A study of personal information in human-chosen passwords and its security implications*. Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on. IEEE, 2016: 1-9.
[24] Pearman, Sarah, et al., *Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat.*. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017.
[25] Li, Yue, Haining Wang, and Kun Sun., *A study of personal information in human-chosen passwords and its security implications*. INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE. IEEE, 2016.
[26] Ren S, He K, Girshick R, et al., *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(6):1137-1149.
[27] Zaremba W, Sutskever I, Vinyals O., *Recurrent neural network regularization*. Eprint Arxiv, 2014.
[28] Graves A., *Generating Sequences With Recurrent Neural Networks*, Computer Science, 2013.
[29] Hochreiter S, Schmidhuber J., *Long Short-Term Memory*. Neural Computation, 1997, 9(8):1735.
[30] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. *Generative adversarial nets*. International Conference on Neural Information Processing Systems. MIT Press, 2014:2672-2680.
[31] Zhang Y, Gan Z, Fan K, et al., *Adversarial Feature Matching for Text Generation*. arXiv preprint arXiv:1706.03850, 2017.
[32] Zhang Y, Gan Z, Carin L., *Generating text via adversarial training*. NIPS workshop on Adversarial Training. 2016.
[33] Pan S J, Yang Q., *A Survey on Transfer Learning*. IEEE Transactions on Knowledge & Data Engineering, 2010, 22(10):1345-1359.
[34] Yosinski J, Clune J, Bengio Y, et al. *How transferable are features in deep neural networks?*. International Conference on Neural Information Processing Systems. MIT Press, 2014:3320-3328.
[35] Zhang, Xiang, Junbo Zhao, and Yann LeCun. *Character-level convolutional networks for text classification*. Advances in neural information processing systems. 2015.
[36] Stuart, A.; Ord, K. (1994), *Kendall's Advanced Theory of Statistics: Volume IDistribution Theory*, Edward Arnold, 8.7.
[37] MacKay, David J.C. (2003). *Information Theory, Inference, and Learning Algorithms* (First ed.). Cambridge University Press. p.34.
[38] Myspace. http://www.myspace.com/
[39] phpBB. http://www.phpbb.com/
[40] RockYou. http://www.rockyou.com/
[41] LinkedIn. http://www.linkedin.com/