Li, Haozhang, Wang, Yangde, Qiu, Weidong, Li, Shujun and Tang, Peng (2024) *PassTSL: Modeling Human-Created Passwords Through Two-Stage Learning.* In: Information Security and Privacy: 29th Australasian Conference, ACISP 2024, Sydney, NSW, Australia, July 15–17, 2024, Proceedings, Part III. Lecture Notes in Computer Science . pp. 404-423. Springer, Singapore ISBN 978-981-97-5100-6. E-ISBN 978-981-97-5101-3.

Haozhang Li, Yangde Wang, Weidong Qiu, Shujun Li and Peng Tang, "PassTSL: Modeling Human-Created Passwords through Two-Stage Learning," in *Information Security and Privacy: 29th Australasian Conference, ACISP 2024, Sydney, NSW, Australia, July 15–17, 2024, Proceedings, Part III*, *Lecture Notes in Computer Science* (LNCS), Volume 14897, pp. 404-423, Springer Nature, 2024, https://doi.org/10.1007/978-981-97-5101-3_22

PassTSL: Modeling Human-Created Passwords through Two-Stage Learning

Haozhang Li[1], Yangde Wang[1(✉)[0009−0005−5813−1470]⋆], Weidong Qiu[1[0000−0001−6428−1655]], Shujun Li[2[0000−0001−5628−7328]], and Peng Tang[1[0000−0001−6607−1280]]

[1] School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
thomas_li, softds, qiuwd, tangpeng@sjtu.edu.cn

[2] School of Computing & Institute of Cyber Security for Society (iCSS), University of Kent, Canterbury, Kent, CT2 7NP, UK
S.J.Li@kent.ac.uk

Abstract. Textual passwords are still the most widely used user authentication mechanism. Due to the close connections between textual passwords and natural languages, advanced technologies in natural language processing (NLP) and machine learning (ML) could be used to model passwords for different purposes such as studying human password-creation behaviors and developing more advanced password cracking methods for informing better defence mechanisms. In this paper, we propose PassTSL (modeling human-created Passwords through Two-Stage Learning), inspired by the popular pretraining-finetuning framework in NLP and deep learning (DL). We report how different pretraining settings affected PassTSL and proved its effectiveness by applying it to six large leaked password databases. Experimental results showed that it outperforms five state-of-the-art (SOTA) password cracking methods on password guessing by a significant margin ranging from 4.11% to 64.69% at the maximum point. Based on PassTSL, we also implemented a password strength meter (PSM), and our experiments showed that it was able to estimate password strength more accurately, causing fewer unsafe errors (overestimating the password strength) than two other SOTA PSMs when they produce the same rate of safe errors (underestimating the password strength): a neural-network based method and zxcvbn. Furthermore, we explored multiple finetuning settings, and our evaluations showed that, even a small amount of additional training data, e.g., only 0.1% of the pretrained data, can lead to over 3% improvement in password guessing on average. We also proposed a heuristic approach to selecting finetuning passwords based on JS (Jensen-Shannon) divergence and experimental results validated its usefulness. In summary, our contributions demonstrate the potential and feasibility of applying advanced NLP and ML methods to password modeling and cracking.

Keywords: Password modeling · Natural language processing · Machine learning · Password strength meters

⋆ The first two co-authors, Y. Wang and H. Li, contributed equally to this work.

1 Introduction

Textual passwords are currently the most common authentication scheme [2] and will continue to be widely used in the foreseeable future [13]. However, human-created passwords are often vulnerable to attacks based on data-driven probabilistic models [33,32,34]. Current state-of-the-art (SOTA) modeling approaches include the Markov chain based models (modeling n-gram transfer probabilities) [24,6,4,22,8], pattern-based models (modeling password semantic structures) [36,19,32,15,41,38], recurrent neural network (RNN) based models (learning to predict transfer distributions using complete preceding context) [23], and generative adversarial network (GAN) based models (adversarially learning the overall representation of a password set) [14,21,26].

For human-created textual passwords have to be memorized by the human users, they are often created partly or even fully based on natural languages so that there are elements reflecting natural language semantics. However, there are also substantial differences between human-created textual passwords and natural languages, e.g., the former do not usually include any white spaces or other obvious separator characters like in natural languages and are much shorter. Despite the differences, many natural language processing (NLP) and machine learning (ML) techniques can still be applied to password modeling and prediction, e.g., the widely used sequence2sequence prediction in NLP can be easily generalized to human-created textual passwords so that the masked part of a password like 'q1w2e[MASK]' can be predicted to be more likely 'q1w2e3' rather than other characters.

Inspired by the pretraining-finetuning framework that have been widely used for NLP and deep learning (DL) models in recent years, this paper presents our work about PassTSL (modeling human-created Passwords through Two-Stage Learning), a deep learning based password model powered by the self-attention mechanism in transformers [31] under a two-staged learning process: pretraining based on a large and more general database, and finetuning based on a smaller and more specific database for the target password database.

We conducted extensive experiments on the impact of the network size, the training data used, and the training data size on the password distribution modeling ability in the pretraining phase. We implemented PassTSL and applied it to six large leaked password databases. Our performance evaluation results manifested the effectiveness of PassTSL, compared against five SOTA password guessing models, including the 6-gram Markov model [22], Ma et al.'s backoff Markov model [22], the latest released implementation of the original PCFG-based password cracking method [36], Houshmand et al.'s semantic PCFG [15], and the RNN-based FLA [23]. In our experiments, we utilized the Monte Carlo method [5] to evaluate the performance with a large number of guesses, up to 10^{20}.

Based on PassTSL, we further designed a lightweight password strength meter (PSM) that can estimate the strength of a password in real time. Experiments proved that our PSM was more accurate than the FLA-based PSM [23] and the SOTA PSM zxcvbn [37] for password strength estimation.

To further enlarge knowledge learned in the pretraining stage, we explored several finetuning schemes. We discussed and tested the guidance roles of password database properties for finetuning, and showed the distinctive advantage of our two-stage model: the finetuning step was able to improve the performance of password cracking on the target password database using only 10^6 additional passwords (0.1% of the database used in the pretraining phase), with a significant margin of up to 3%. Additionally, we proposed an approach to selecting the finetuning password database based on JS (Jensen-Shannon) divergence [20] between pretraining/candidate finetuning databases and the target database, and validated its effectiveness.

To summarize, the main contributions of this paper are as follows.

- We propose PassTSL, a neural network model that introduced the self-attention mechanism to promote password modeling performance. PassTSL was able to outperform SOTA password methods in six large databases by a significant margin ranging from 4.11% to 64.69% at the maximum point.
- We introduce the pretraining-finetuning framework into the field of password cracking, for the first time in the literature (to the best of our knowledge), and our experimental results demonstrate its effectiveness.
- We investigated the impact of two properties of password databases (language and service type) on the performance of the finetuning stage, explored the effectiveness of few-shot learning, and obtained an insight on selecting appropriate finetuning passwords.

The rest of the paper is organized as follows. The next section provides an overview of related work. Section 3 introduces PassTSL by reporting the model structure, the impact of pretraining settings, and the performance in comparison to other password cracking methods and PSMs. Section 4 showcases our investigation on finetuning PassTSL, presenting the performance results under various finetuning schemes. The last section concludes our work.

2 Related Work

2.1 Pretrained Models

To overcome the limitation about the lack of available large-scale datasets, machine learning researchers proposed transfer learning [25] and a two-staged learning framework including the pretraining and finetuning stages [17]. When used for NLP problems, the two-staged framework first encodes linguistic knowledge from a large-scale corpus and then transfers the captured knowledge about the underlying language to a more specified task, which avoids training from scratch but uses a smaller more targeted corpus to finetune the pretrained model [3]. NLP researchers have proposed various pretrained language models (PLMs) using large unlabeled corpora to learn contextual word embeddings [17]. In 2017 Vaswani et al. [31] proposed Transformer to capture higher-level concepts in context like polysemous disambiguation and syntactic structures. Radford et al. [27] proposed GPT in 2018 and Devlin et al. [7] proposed BERT in 2019 for NLP tasks including text sentiment classification, named entity recognition, and Q&A. Researchers subsequently introduced additional improvements to GPT and BERT, designing new PLMs with better performance such as GPT2 [28]. Some researchers also changed the model architecture and explored new pre-training tasks, leading to work such as BART [16] and XLNet [40].

2.2 Password Modeling Methods

Password guessing attacks are probably as old as the history of passwords. Much research has been done in this area, and we would only discuss methods and contributions that are most relevant to our work.

Markov. Narayanan et al. [24] proposed to guess passwords using an n-gram Markov model. The model was further extended by Ma et al. [22] and Dürmuth et al. [8]. Ma et al. [22] explored normalization techniques and smoothing strategies for Markov models, proposing Laplace smoothing and the backoff mechanism to prevent overfitting of higher-order Markov models. Dürmuth et al. [8] optimized the enumeration process by proposing an ordered Markov enumerator (OMEN) based on approximate sorting. However, all Markov model based methods are commonly limited by memory resources. The hyperparameter n is usually taken as 5 or 6 and thus longer-distance contextual information cannot be considered.

PCFG. Weir et al. [36] explored the use of the probabilistic context-free grammar (PCFG) for password analyses and cracking. They considered passwords as instances of templates based on character types with different terminals (i.e., strings of the same type). For example, the password 'alice123!@' is an instance of the template $L_5D_3S_2$, and its probability is calculated as $P(\text{alice123!@}) = P(L_5D_3S_2) \times P(\text{alice}|L_5) \times P(123|D_3) \times P(\text{!@}|S_2)$. Li et al. [19] introduced Pinyin as a new type of password segments to improve Chinese password guessing. Houshmand et al. [15] introduced keyboard patterns and multi-word detection into PCFG to enhance its power. Veras et al. [32] used NLP methods to extract semantic information in passwords, and proposed a semantic PCFG that incorporates more types of password segments with different semantic meanings.

Neural network based methods. Melicher et al. [23] proposed to model passwords using a long short-term memory (LSTM) network, which is referred by other researchers as FLA (derived from three words in their paper's title: 'Fast', 'Lean', and 'Accurate'). They also designed a PSM based on a highly compressed FLA network. Hitaj et al. [14] proposed PassGAN, which guesses passwords via a GAN. However, they reported that PassGAN required more guesses in order to achieve the same cracking performance as FLA. Dario et al. [26] demonstrated the potential of representation learning in improving PassGAN. In particular, they proposed to dynamically control the latent space of PassGAN through the feedback from correct guesses during password guessing to mimic the unknown distribution of the target passwords.

More recently, some researchers working on password cracking have started recognizing the potential of transformer-based deep learning models. He et al. [12]

investigated the password reuse problem and presented PassTrans, which was able to guess variants of a given password for online attacks. Similarly, Xu et al. [39] proposed PassBERT for conditional password guessing, cracking a password fully under the condition of having partial knowledge of the password. However, neither PassTrans nor PassBERT directly models the probability distribution of the target password(s). Rando et al. [29] introduced the concept of guided password generation to guess passwords that match arbitrary constraints, but did not consider a finetuning stage.

3 PassTSL: Pretraining

We first report the architecture and pretraining method of PassTSL. Then, we discuss the effects of different pretraining settings on PassTSL's performance. Finally, we demonstrate advantages of the pretrained (without finetuning) PassTSL over SOTA methods via Monte Carlo simulation from two aspects, password cracking and password strength estimation.

3.1 Method

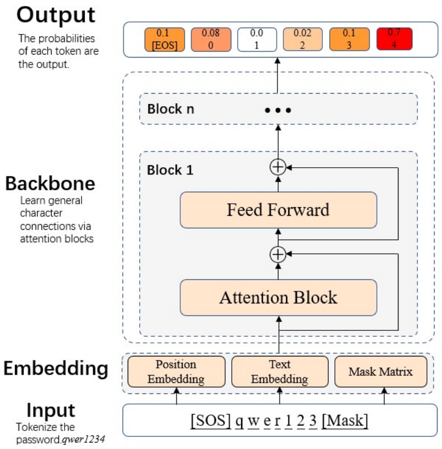

Output
The probabilities of each token are the output.
0.1 [EOS]
0.08 0
0.0 1
0.02 2
0.1 3
0.7 4
Block n
• • •
Backbone
Learn general character connections via attention blocks
Block 1
Feed Forward
Attention Block
Embedding
Position Embedding
Text Embedding
Mask Matrix
Input
Tokenize the password.qwer1234
[SOS] q w e r 1 2 3 [Mask]

Fig. 1: The overall architecture of PassTSL. The redder a cell's color is, the more likely it would be the next character. Each module is further explained in Section 3.1.

Here we propose PassTSL's pretrained model, a base line model of our two-staged password guessing method, in order to learn universal representations of passwords. Formally, password modeling is constructed as an unsupervised distribution estimation of passwords ($\mathbf{x} = \{c_1, c_2, \ldots, c_m\}$), where $\mathbf{x}$ stands for a password and $\mathbf{c}_i$ is the i-th character in $\mathbf{x}$. We use the standard language model object to maximize a likelihood function $L(\mathbf{x})$:

$$L(\mathbf{x}) = \sum_{i=1}^{m} logP(c_i|c_1, \ldots, c_{i-1}; \theta), \tag{1}$$

where θ represents parameters of PassTSL. An overview of the PassTSL architecture is illustrated in Figure 1 with an example.

Vocabulary and tokenization. We focus on character-level tokens instead of commonly used word-level tokens in NLP or pattern-level tokens in PCFG. Texts (especially Western texts) are naturally separated by spaces, while passwords rarely include any white spaces or other uniformly defined separators so characters are what we have to start with to analyse passwords.

Input. The input is a single password since PassTSL aims to learn character-level transfer probability distributions. Each password $\mathbf{x}$ will be preprocessed to a sequence of characters after character-level tokenization. A special start-of-sequence ([SOS]) token is added to the beginning of $\mathbf{x}$ as the initial input when generating a candidate or calculating the probability of a given password. A special end-of-sequence ([EOS]) token is also added as the end symbol to help PassTSL learn when to stop decoding.

Embedding. The tokenized sequence will be encoded by the text embedding layer and the position embedding layer. The resulting vectors are then summed up to get the representation of $\mathbf{x}$, and to construct the self-attention mask matrix $\mathbf{M}$ for each password, with element $\mathbf{M}_{ij}$ satisfying

$$\mathbf{M}_{ij} = \begin{cases} 0 & i \geq j, \text{ need attention,} \\ -\infty & i < j, \text{ need to be masked.} \end{cases} \tag{2}$$

Backbone network. Our model is a multi-layered network based on the architecture of a transformer decoder [31]. The embedding vector $\mathbf{R}_0$ of sequence $\mathbf{x}$ is encoded as a contextual representation $\mathbf{R_L}$ by the L-block transformer decoder after normalization. $\mathbf{R_L}$ is fed into a linear classifier to compute the distribution over target tokens. Formally, in the ith block $(0 < i \leq L)$, a self-attention operation is implemented as:

$$\mathbf{Q} = \mathbf{R}_{i-1}\mathbf{W}_i^Q, \mathbf{K} = \mathbf{R}_{i-1}\mathbf{W}_i^K, \mathbf{V} = \mathbf{R}_{i-1}\mathbf{W}_i^V, \tag{3}$$

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{QK}^T}{\sqrt{d_k}} + \mathbf{M})\mathbf{V}, \tag{4}$$

where $\mathbf{R}_{i-1}$ is the output of the $i-1$-th block, while $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ are the parameter matrices for linearly mapping $\mathbf{R}_{i-1}$ to a triple. $\mathbf{M}$ is the self-attention mask matrix fed into PassTSL together with the context tokens.

Hyperparameters. PassTSL is defined by the following hyperparameters.

* l: *Number of layers*, which represents the number of decoder blocks.

- * E: *Embedding size*, which determines the number of dimensions for embedding layers and the hidden state in each block.
- * I: *Intermediate size*, which indicates dimensions of the feed-forward layer in each block.
- * h: *Number of heads*, which represents the number of self-attention heads.
- * *Vocab size*, which represents the size of the PassTSL vocabulary. The vocabulary is composed of 95 printable ASCII characters and five special characters ([PAD], [SOS], [EOS], [UNK], and [MASK]).
- * *Attention dropout*, which specifies the dropout ratio of the attention blocks. It is instantiated to 0.1.

Model size. To balance the computational costs with the performance of PassTSL, we design two PassTSL instances of different sizes, shown in Table 1. We will report their performance in Section 3.2.

Table 1: Structure configurations of different PassTSL instances.

Model	l	E	I	h	#(All Parameters)
PassTSL$_{\text{Base}}$	12	768	3,072	12	85,919,232
PassTSL$_{\text{Small}}$	6	256	1,024	4	4,781,056

Password generation. Evaluating performances on the target password databases by enumeration is inefficient and too resource-intensive. For instance, 100 million (10^8) guessed passwords would occupy around 1GB memory. Besides, some password cracking models are particularly insufficient in this regards, e.g., Pasquini et al. [26] pointed out that it would take more than two weeks to generate 10^{10} passwords using FLA [23]. Therefore we decided to use Monte Carlo simulation to estimate the number of guesses required for a given target password.

Datasets. We employed multiple leaked password databases, four password databases leaked from Chinese websites, and another four leaked from English websites[3], for pretraining and finetuning PassTSL, and also for evaluating its performance against other SOTA methods. These eight databases cover three types of online services and two mainly used languages. Another hybrid database COMB is a compilation of existing data which contains approximately 3.2 billion unique passwords from multiple previous leaks and breaches[4]. Table 2 provides a detailed description.

Although some databases were leaked in 2009 and 2011, we still consider them representative based on some research looking at human users' password

[3] The four websites are all run by US-based companies and their users are from many countries and speak many different languages. However, English is usually the dominating or common language used by all users.

[4] `https://cybernews.com/news/largest-compilation-of-emails-and-passwords-leaked-free`

Table 2: Summary of password datasets used.

Name	Service	User Type	Year	#(Passwords)	Length≤5	Usage
COMB	Multiple	Mixed	2021	3.3B	4.1%	Pretrain
CSDN	Social Forum	CN (Chinese)	2011	6.4M	0%	Pretrain
17173	Entertainment	CN	2011	17.9M	0%	Target
178	Entertainment	CN	2011	9.1M	0%	Target
Tianya	Social Forum	CN	2011	30.3M	0.0001%	Target
Gmail	Life	EN (English)	2014	4.7M	0.004%	Pretrain
MyHeritage	Life	EN	2018	84.8M	0.02%	Target
RockYou	Social Forum	EN	2009	28.7M	0.0008%	Target
Twitter	Social Forum	EN	2016	67.1M	0%	Target

Table 3: Settings for pretraining PassTSL instances.

Name	Model Size[a]	Data	Data Size	Target Passwords
$\text{PassTSL}^{\text{COMB\_100M}}_{\text{Small}}$	Small	COMB	100M	All passwords
$\text{PassTSL}^{\text{COMB\_1M}}_{\text{Small}}$	Small	COMB	1M	All passwords
$\text{PassTSL}^{\text{CSDN\_1M}}_{\text{Base}}$	Base	CSDN	1M	Chinese Passwords
$\text{PassTSL}^{\text{CSDN\_1M}}_{\text{Small}}$	Small	CSDN	1M	Chinese Passwords
$\text{PassTSL}^{\text{Gmail\_1M}}_{\text{Base}}$	Base	Gmail	1M	English Passwords
$\text{PassTSL}^{\text{Gmail\_1M}}_{\text{Small}}$	Small	Gmail	1M	English Passwords

[a] See Table 1 for hyperparameters of the *base* and *small* models.

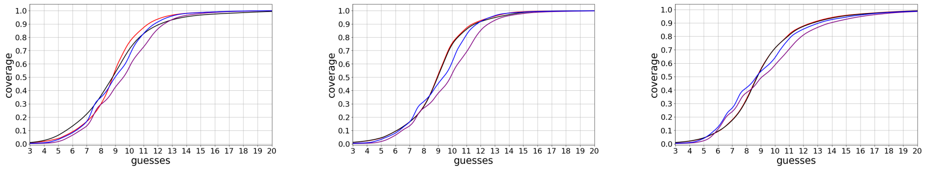
creation behaviors, e.g., Bonneau [1] reported that such behaviors had changed only slightly between 1990 and 2011, and other more recent research [11,10] revealed that password policies and practices implemented on top sites had rarely changed.

The databases we used have been widely used by password researchers [22,23,34,38,19]. Since passwords can include sensitive personal information, in this paper we only report aggregated results so no personal information will be leaked.

3.2 Effect of Different pretraining Settings

Experimental settings We randomly selected 100 million passwords and one million passwords from COMB, one million passwords from CSDN, and one million passwords from Gmail as the pretraining data, denoted by COMB_100M, COMB_1M, CSDN_1M, and Gmail_1M. These passwords consist of only 95 ASCII printable characters and contain no less than 6 characters.

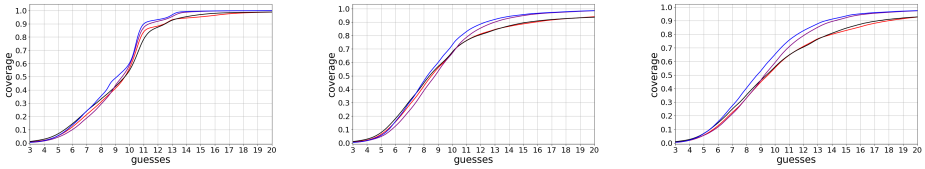
Table 3 gives detailed settings. Models were pretrained for 10 epochs with a batch size of 256. In Monte Carlo simulations, one million passwords were sampled from each pretrained model to provide valid and accurate estimates. We



(a) CSDN-17173 (b) CSDN-178 (c) CSDN-Tianya

Fig. 2: Various pretraining settings for PassTSL attacking Chinese databases: ⌒ $\text{PassTSL}_{\text{Small}}^{\text{CSDN\_1M}}$, ⌒ $\text{PassTSL}_{\text{Base}}^{\text{CSDN\_1M}}$, ⌒ $\text{PassTSL}_{\text{Small}}^{\text{COMB\_1M}}$, ⌒ $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$. The x-axes represent the number of guesses in the log scale. The y-axes show the percentage of correctly guessed passwords.

also modeled up to 10^{20} guesses to ensure completeness. It is an overestimation considering that it would take over six months in the real-world generation.



(a) Gmail-MyHeritage (b) Gmail-RockYou (c) Gmail-Twitter

Fig. 3: Various pretraining settings for PassTSL attacking English databases: ⌒ $\text{PassTSL}_{\text{Small}}^{\text{Gmail\_1M}}$, ⌒ $\text{PassTSL}_{\text{Base}}^{\text{Gmail\_1M}}$, ⌒ $\text{PassTSL}_{\text{Small}}^{\text{COMB\_1M}}$, ⌒ $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$.

Analyses and Discussions Figures 2 and 3 show the performance of PassTSL on six testing databases under different pretraining settings. We have the following findings and conclusions.

The increase of the model size plays a positive but has a limited effect on the guessing performance. As shown in Figure 2, the curves of $\text{PassTSL}_{\text{Base}}^{\text{CSDN\_1M}}$ and $\text{PassTSL}_{\text{Small}}^{\text{CSDN\_1M}}$ largely overlap. Figure 3 indicates that the simulation curves of $\text{PassTSL}_{\text{Base}}^{\text{Gmail\_1M}}$ are a bit higher than those of $\text{PassTSL}_{\text{Small}}^{\text{Gmail\_1M}}$, specifically **0.92%** on average at 10^{20} guesses. However, such improvement is not surprising considering that the number of parameters for $\text{PassTSL}_{\text{Base}}$ is 20 times larger than that for $\text{PassTSL}_{\text{Small}}$. For sequences like passwords with shorter length and less logical information than texts in general NLP tasks, a small-scale model has been already strong enough to sufficiently learn the inner linguistic features. Larger-scale models, while still likely to improve the effectiveness of password modeling, have to bear the risk of overfitting.

Pretraining PassTSL with mixed-language passwords is able to help better modeling English passwords. Figure 3 shows that mixed-language based PassTSL instances obviously outperformed those pretrained using English passwords only. Particularly, the cracking rate of $\text{PassTSL}_{\text{Small}}^{\text{COMB\_1M}}$ is on average **3.38%** higher than that of $\text{PassTSL}_{\text{Base}}^{\text{Gmail\_1M}}$ and **3.46%** higher than that of $\text{PassTSL}_{\text{Small}}^{\text{Gmail\_1M}}$, while $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ is on average **3.48%** and **3.55%** higher than the same two benchmarks, respectively. This significant performance gain does not exist when such PassTSL instances are used to attack Chinese databases, as shown in Figure 2. We believe that it is because hybrid databases like COMB are more dominated by English passwords, making the pretrained model more biased towards English passwords. Such a bias also implies that using Chinese passwords for finetuning can potentially improve a COMB-pretrained PassTSL model's performance against Chinese databases, which was proved in our experimental results reported in Section 4.2.

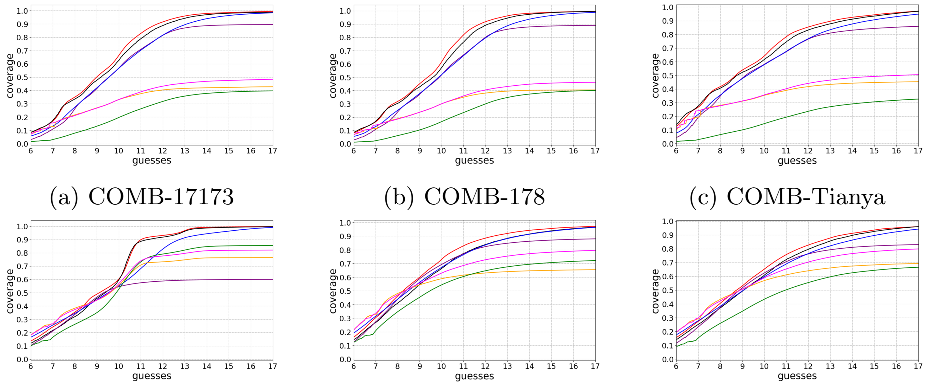
The increase of the training data size will significantly improve the performance. $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ has noticeable improvement over $\text{PassTSL}_{\text{Small}}^{\text{COMB\_1M}}$ on each test set as shown in Figures 2 and 3. Besides, although the mixed database COMB is more biased towards English passwords, the performance of $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ on Chinese testing sets is still better, likely because the amount of training data from COMB is much larger than the amount of CSDN. As shown in Figure 2, the simulated cracking rates of $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ compared to $\text{PassTSL}_{\text{Base}}^{\text{COMB\_100M}}$ differ by only **0.004%** (17173), **0.06%** (178), and **0.35%** (Tianya) at 10^{20}, respectively. These results suggest that the data-driven PassTSL model possesses the potential to further improve the accuracy of password modeling given more training resources.

Considering the resources required for pretraining, password generation and probabilities calculation, we believe that $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ is a good balanced representation of PassTSL's ability to model passwords.

3.3 Evaluations on Password Cracking and Password Strength Estimation

In this subsection, we compare $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ with other SOTA password guessing models and PSMs on the testing sets shown in Table 2. We briefly describe the models for comparison, and then report their performance.

Password Cracking Tools for Comparison The password guessing models compared with PassTSL and their settings are as follows: 1) the 6-gram Markov model used in [22] as a benchmark; 2) the backoff Markov model proposed in [22]; 3) PCFG_{Se} – Veras et al.'s semantic PCFG [32]; 4) PCFG_{Ori} – the latest released (v4.0-rc3) of the original PCFG [35] developed and maintained by Matt Weir, the lead author of the original PCFG paper [36]; and 5) FLA [23] with the default configurations recommended by their authors. We did not consider GAN-based models such as PassGAN [14] because past research suggested that they need



(a) COMB-17173

(b) COMB-178

(c) COMB-Tianya

(d) COMB-MyHeritage

(e) COMB-RockYou

(f) COMB-Twitter

Fig. 4: Guessing performance of PassTSL against other SOTA models: — PassTSL(PassTSL$_{\text{Small}}^{\text{COMB\_100M}}$), — FLA, — 6-gram, — Backoff, — PCFG$_{\text{Se}}$, — PCFG$_{\text{Ori}}$. The x-axes represent guessing numbers in the log scale. We show the performance under a range of guessed passwords used more commonly by other researchers [14,22,29,23,32].

more guesses to obtain the same performance as FLA. Moreover, we did not include some recently proposed methods [38,18], due to the lack of source code released.

Comparing PSMs One of the main conceptual forms for password strength evaluation is to simulate adversarial password guessing [22,30]. We compare the performance of the lightweight PassTSL meter with other strength meters described as follows.

FLA-based PSM. Authors of [23] used a quantized and compressed version of FLA for their PSM, which can lead to worse performance. In our experiments, we used the uncompressed version of FLA so that the FLA-based PSM will demonstrate its best performance.

zxcvbn. zxcvbn [37] is considered one of the best PSMs that rely on manual rules, statistical methods, and plain-text dictionaries.

The Yahoo! PSM used in [23] and CKL_PCFG PSM used in [38] were excluded because the former was far inferior to zxcvbn and the latter was only partially open-sourced. The ground truth is provided via the MinGuess method [30]. It is an idealized approach and represents the most conservative security results. A password is considered cracked as long as it is guessed by any of the guessing approaches.

Evaluation Results on Password Guessing Figure 4 shows the performances of PassTSL and other five SOTA guessing models when attacking 17173, 178, Tianya, MyHeritage, RockYou and Twitter. We can draw the following conclusions.

PassTSL outperforms other five guessing models when attacking Chinese password databases. As presented in Figures 4(a)-(c), the simulation curves of PassTSL$_{\text{Small}}^{\text{COMB\_100M}}$ are at the top, indicating that PassTSL performs the best when attacking Chinese passwords.

We took 1,000 points uniformly from each curve in logarithmic coordinates and calculated the average of their differences to measure the performance more quantitatively. Specifically, when attacking 17173, the predicted coverage of PassTSL is **4.76%** higher than FLA, **12.72%** higher than 6-gram Markov, and **12.74%** higher than backoff Markov at maximum around 10^{11} guesses, where **passwords almost surely will not survive credible offline attacks** [9]. When the target database is 178, the predicted coverage of PassTSL is **5.86%** higher than FLA, **15.46%** higher than 6-gram Markov, and **15.87%** higher than backoff Markov at the maximum point. When attacking Tianya, the predicted coverage rate of PassTSL is **4.11%** higher than FLA, **12.29%** higher than 6-gram Markov, and **10.84%** higher than backoff Markov at the maximum point. PassTSL also outperforms PCFG$_{\text{Se}}$ with over **35%** (**64.69%** against Tianya on 10^{18} guesses) and PCFG$_{\text{Ori}}$ with over **25%** for all Chinese databases.

PassTSL is better when attacking English databases at larger guessing numbers. Observing Figures 4(d)-(f), the simulation curves of PassTSL$_{\text{Small}}^{\text{COMB\_100M}}$ are lower than the ones of PCFG$_{\text{Se}}$ and PCFG$_{\text{Ori}}$ when the guessing number is less than 10^8. However, they remain the highest over 10^{10} when the target database is MyHeritage and over 10^9 guesses for other target databases, while PCFG curves look saturated. These results demonstrate that although PCFG-based methods perform best on small-scale guesses when attacking English passwords, the sparsity defect of PCFG is gradually exposed as the guessing number increases. Some templates or patterns of target passwords were not learned by PCFG from the training data. However, PassTSL will significantly exceed PCFG since it builds a more complicated password model than the simpler PCFG-based one. Another network-based method FLA is inferior to PassTSL in all scenarios, reflecting the advantage of self-attention operations over the circular mechanism for password modeling.

Evaluation Results on PSMs We quantized and compressed PassTSL$_{\text{Small}}^{\text{COMB\_100M}}$ to make it closer to a real-world PSM that can run from a web browser. The compressed PSM represents a more conservative performance of PassTSL-based PSMs. Table 4 shows the size of our PassTSL PSM after quantization and lossless compression.

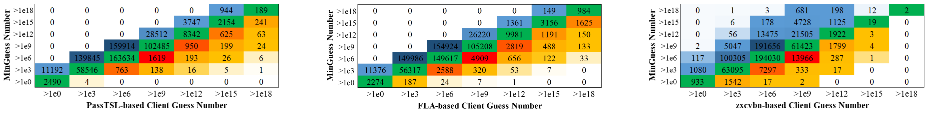
We compared the accuracy of our lightest PSM, PassTSL-Small-int8z, to other PSMs in Figure 5. Here, we scale the output of PassTSL and FLA down to ensure that they made as many safe errors (under-estimated password strength) as zxcvbn (in this case, both PassTSL- and FLA-based PSMs gave more conservative results). The factors were 42 and 68, respectively.

Table 4: The compressed model sizes and the loading times.

Name	Compress	Model Size	Client Size[a]	Loading Time (seconds)
PassTSL$_{\text{Small}}$	-	18.4 MB	37.8MB	1.86
PassTSL$_{\text{Small}}^{\text{fp16}}$	fp16	9.26 MB	28.2MB	1.55
PassTSL$_{\text{Small}}^{\text{fp16z}}$	fp16+zip	8.48 MB	27.6MB	2.34
PassTSL$_{\text{Small}}^{\text{int8}}$	int8	4.75 MB	23.5MB	1.12
PassTSL$_{\text{Small}}^{\text{int8z}}$	int8+zip	3.97 MB	22.9MB	1.49

[a] The client size refers to the transferred data size. The loading time refers to the time when Google Chrome was used.

From Figure 5, PassTSL-Small-int8z is more accurate than the other two PSMs: it has the lowest unsafe errors (over-estimated password strength), while safe errors are aligned with others. It makes significantly fewer (almost half for all red cells) unsafe errors than FLA. Moreover, although zxcvbn is more accurate when evaluating strong passwords ($> 10^{13}$), its unsafe errors often appear at lower and mid-ranged guessing numbers, which could mislead users to choose weaker passwords more often.



(a) PassTSL (b) FLA (c) zxcvbn

Fig. 5: Guessing numbers of the three tested PSMs compared against MinGuess. Red cells are important because they indicate how many passwords are overestimated by a PSM (denoted as unsafe errors). Underestimations of strength are shown in blue cells (called safe errors) and accurate estimations are shown in green cells. The chromatic intensity rises with the number of passwords.

4 PassTSL: Finetuning

As explained in the previous section, our PassTSL base-line model pretrained using the COMB has already demonstrated superior performance over some SOTA methods. In this section, we show that even better performance can be achieved by finetuning the pretrained model. We explore methods for finetuning PassTSL and report the impact of various finetuning strategies.

4.1 The Effects of Database Properties

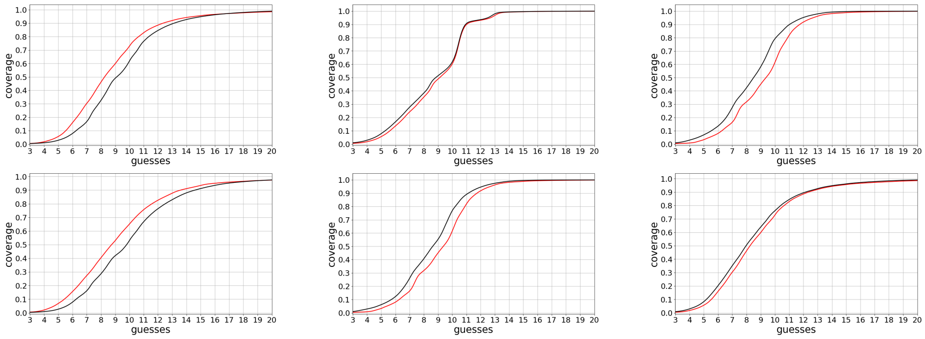
In general, attackers know the language most users speak and the service type of a target website. We were interested in knowing if these two properties can positively affect the finetuning results. Wang et al. [34] showed that the native language plays an important role in users' password composition behaviors and users speaking different languages have noticeable different password structural patterns. For instance, English-speaking users prefer letters more than Chinese users, but less so on digits. We also noticed that, apart from listing the service types of websites from which the password databases were leaked, authors of past studies [33,34,38] did not discuss the impact of website service type in more details. Our work reported here fills this gap.

Together with the findings in Section 3.2, we consider the attack on a specific password database as a downstream task on $\mathrm{PassTSL}^{\mathrm{COMB\_100M}}_{\mathrm{Small}}$. We expect that the password cracker could enhance password modeling performance with small finetuning costs while preserving common password knowledge learned in the pretraining phase. Intuitively, passwords taken from a website sharing the same language and service type properties as the target website are ideal sources for finetuning. We designed three scenarios to investigate the effects of both properties, where one million passwords were randomly selected from a finetuning database to attack a target database.

The same service type (social media), but different languages (Chinese to English) (Scenario A). $\mathrm{PassTSL}^{\mathrm{COMB\_100M}}_{\mathrm{Small}}$ is finetuned by Tianya_1M to attack RockYou and Twitter.

The same language (English), but different service types (Scenario B). $\mathrm{PassTSL}^{\mathrm{COMB\_100M}}_{\mathrm{Small}}$ is finetuned by Twitter_1M to attack MyHeritage (social media to information service) and by Tianya_1M to attack 178 (social media to gaming website).

The same language (English) and the same service type (social media) (Scenario C). $\mathrm{PassTSL}^{\mathrm{COMB\_100M}}_{\mathrm{Small}}$ is finetuned by 17173_1M to attack 178 (Chinese gaming websites) and by Twitter_1M to attack RockYou.



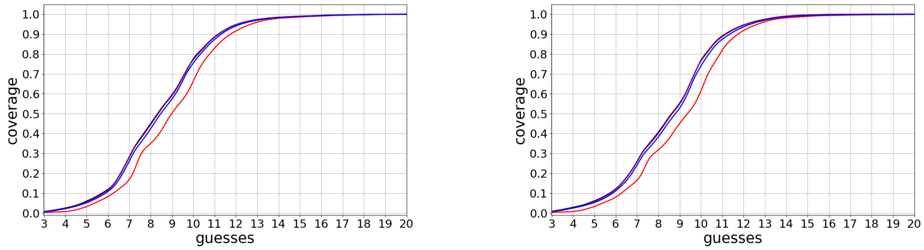
(a) Scenario A (b) Scenario B (c) Scenario C

Fig. 6: Performances of the finetuned PassTSL instances in the three scenarios for investigating the effects of the database attributes: — $\mathrm{PassTSL}^{\mathrm{FT}}_{\mathrm{Small}}$, against — $\mathrm{PassTSL}^{\mathrm{COMB\_100M}}_{\mathrm{Small}}$.

As shown in Figure 6, the curves in sub-figure (a) show that it will lead to a decrease on performance if finetuning only based on the service type of the target website, while results in sub-figures (b-c) prove that **the user language has a positive effect in the finetuning stage**: the coverage rate of $\text{PassTSL}_{\text{Small}}^{\text{FT}}$ (FT = finetuning) is on average 3.33% higher than that of $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ in sub-figure (b), and 4.07% in sub-figure (c).

4.2 The Size of Finetuning Passwords

The results in the previous subsection are based on one million finetuning passwords. This amount of data may still considered too high, so we also investigated if a smaller amount of finetuning data can achieve a sufficiently good performance.



(a) COMB-17173

(b) COMB-178

Fig. 7: Performances of two PassTSL instances finetuned by a smaller database, — $\text{PassTSL}_{\text{Small}}^{\text{FT\_100K}}$ and — $\text{PassTSL}_{\text{Small}}^{\text{FT\_10K}}$, compared against two base-line instances, — $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$ and — $\text{PassTSL}_{\text{Small}}^{\text{FT\_1M}}$.

Using the Tianya database as an example, 10K and 100K passwords were randomly selected, satisfying the requirements in Section 3.2, denoted as Tianya_10K and Tianya_100K. They were new password sets to finetune $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$, respectively, while target databases were 17173 and 178. The finetuned models, denoted by $\text{PassTSL}_{\text{Small}}^{\text{FT\_10K}}$ and $\text{PassTSL}_{\text{Small}}^{\text{FT\_100K}}$, were compared with $\text{PassTSL}_{\text{Small}}^{\text{FT\_1M}}$ and $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$.

Experimental results are provided in Figure 7. For both target databases, curves of $\text{PassTSL}_{\text{Small}}^{\text{FT\_1M}}$ and $\text{PassTSL}_{\text{Small}}^{\text{FT\_100K}}$ are approximately identical, while visible gaps exist for $\text{PassTSL}_{\text{Small}}^{\text{FT\_10K}}$. In particular, $\text{PassTSL}_{\text{Small}}^{\text{FT\_100K}}$ is on average 0.13% lower than $\text{PassTSL}_{\text{Small}}^{\text{FT\_1M}}$ on 17173 and 0.14% lower on 178, while $\text{PassTSL}_{\text{Small}}^{\text{FT\_10K}}$ is 0.77% and 0.91% lower. However, coverage rates of $\text{PassTSL}_{\text{Small}}^{\text{FT\_100K}}$ is still 3.4% and 3.7% higher than $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$, respectively. **The results show that, for the finetuning stage, only 0.1% of the pretraining data size can be sufficient to obtain a good level of**

performance improvement. The finding echoes the observation in Section 3.2 that using COMB for pretraining can lead to a pretrained model biased towards English passwords.

From the experiments in Sections 4.1 and 4.2, we conclude that:

- The finetuning process can enhance PassTSL's ability to guess passwords.
- The user language of the target website can be used to achieve better finetuning results, if the finetuning data share the same language as the target website. Considering that this property of a website is mostly public knowledge, an immediate advice to users is that they should try to diversify ways to define their passwords, particularly to avoid using elements that are more typical in the dominating language of a website.
- To achieve a good performance, the finetuning stage just needs as little as 0.1% of the amount of pretrained passwords.

4.3 How to Select Finetuning Password Database

Table 5: JS divergence and finetuning settings

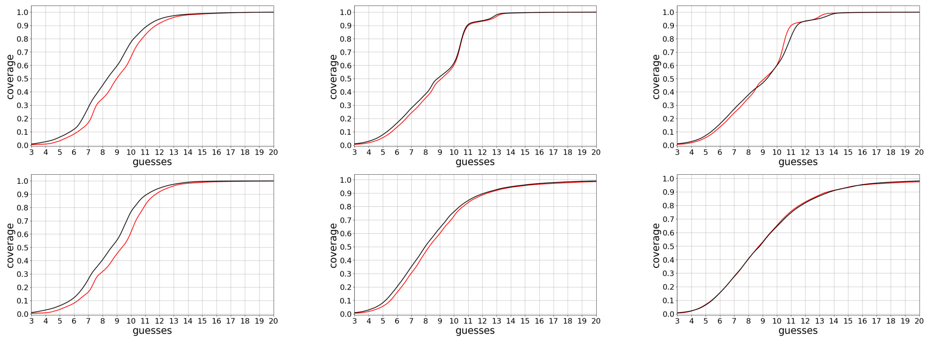
Database	Usage	17173	178	MyHeritage	RockYou	Twitter
COMB_100M	Pretraining	0.2288	0.256	0.1339	0.0645	0.0206
Tianya_1M	Finetuning	0.0268	0.0511	-	-	-
Twitter_1M	Finetuning	-[a]	-	0.1311	0.057	-
RockYou_1M	Finetuning	-	-	0.1431	-	0.057

[a] '-' means we did not use this pair of databases for our experiments.

Based on the experimental results in Section 3, it is recommended that COMB be used as the pretraining database. Heuristically, we suggest selecting the finetuning database so that it is more similar to the target database than the pretraining database is, so that the finetuning process can add more specific information about target passwords to the finetuned PassTSL model. To measure the similarity between two password databases, we propose to use the JS (Jensen-Shannon) divergence [20] calculated based on the union of all 3-grams in the two password databases.

To validate the above heuristic idea, some experiments were conducted based on JS divergence values between selected pairs of password databases given in Table 5. The finetuned PassTSL in each experiment is denoted as $\text{PassTSL}_{\text{Small}}^{\text{FT}}$.

Figure 8 presents experimental results. As shown in in Sub-figure (a), compared with the pretrained model $\text{PassTSL}_{\text{Small}}^{\text{COMB\_100M}}$, the model $\text{PassTSL}_{\text{Small}}^{\text{FT}}$ finetuned using the Tianya database performs significantly and consistently better when attacking 17173 and 178 (the coverage rate increases by 12.13% on 10^9 guesses on 17173 and by 15.77% on 10^9 guesses on 178). Such results can be predicted from the JS divergence values: $\text{JS}(\text{Tianya}, 17173) < \text{JS}(\text{COMB\_100M}, 17173)$



(a) Tianya-finetuned PassTSL attacking 17173 (top) & 178 (bottom)

(b) Twitter-finetuned PassTSL attacking MyHeritage (top) & RockYou (bottom)

(c) RockYou-finetuned PassTSL attacking MyHeritage (top) & Twitter (bottom)

Fig. 8: Performances of finetuned PassTSL instances compared against those without finetuning: — $\text{PassTSL}^{\text{FT}}_{\text{Small}}$ and — $\text{PassTSL}^{\text{COMB\_100M}}_{\text{Small}}$.

and $\text{JS}(\text{Tianya}, 178) < \text{JS}(\text{COMB\_100M}, 178)$. When the Twitter database was used for finetuning, $\text{PassTSL}^{\text{FT}}_{\text{Small}}$ also beats $\text{PassTSL}^{\text{COMB\_100M}}_{\text{Small}}$ on MyHeritage and RockYou as shown in Sub-figure (b). The coverage rate increases by 3.9% on 10^8 guesses and 4.96% on 10^7 guesses. Such results could also be predicted from the JS divergence values: $\text{JS}(\text{Twitter}, \text{MyHeritage}) < \text{JS}(\text{COMB\_100M}, \text{MyHeritage})$ and $\text{JS}(\text{Twitter}, \text{RockYou}) < \text{JS}(\text{COMB\_100M}, \text{RockYou})$. The lower improvement may be explained by the fact that the JS divergence values between the Twitter database and the two target databases are just slightly smaller than the values between COMB_100M and the two target databases. When the RockYou database was used for finetuning, the performance of the finetuned model $\text{PassTSL}^{\text{FT}}_{\text{Small}}$ behaves very similar to $\text{PassTSL}^{\text{COMB\_100M}}_{\text{Small}}$, which can be explained by the fact that the RockYou database does not have a smaller JS divergence value with the target databases than COMB_100M, as shown in Table 5, therefore the contribution of the finetuning becomes more marginal (if any).

In summary, the results in Figure 8 largely validate the effectiveness of using the JS divergence as a quantitative metric to guide the selection of the finetuning database.

5 Conclusion

This paper presents PassTSL, a deep learning model for password modeling and guessing, based on the pretraining-finetuning two-staged learning framework. Our model aims to extract universal password features through a self-attention mechanism. Extensive experiments have proved that PassTSL is superior to other SOTA password modeling and cracking methods and is also practical to support a password strength meter. It is also shown that the finetuning phase can help improve the performance even further if the finetuning database is properly selected, and the amount of finetuning data needed is very light (e.g., 0.1% of the pretraining data).

References

1. Bonneau, J.: The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In: Proceedings of IEEE S&P 2012. pp. 538–552. IEEE (2012). https://doi.org/10.1109/SP.2012.49
2. Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In: Proceedings of IEEE S&P 2012. pp. 553–567. IEEE (2012). https://doi.org/10.1109/SP.2012.44
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Proceedings of NeurIPS 2020. pp. 1877–1901. Curran Associates Inc. (2020)
4. Castelluccia, C., Dürmuth, M., Perito, D.: Adaptive password-strength meters from Markov models. In: Proceedings of NDSS 2012. Internet Society (2012), https://www.ndss-symposium.org/ndss2012/ndss-2012-programme/adaptive-password-strength-meters-markov-models/
5. Dell'Amico, M., Filippone, M.: Monte Carlo strength evaluation: Fast and reliable password checking. In: Proceedings of CCS 2015. pp. 158–169. ACM (2015). https://doi.org/10.1145/2810103.2813631
6. Dell'Amico, M., Michiardi, P., Roudier, Y.: Password strength: An empirical analysis. In: Proceedings of IEEE INFOCOM 2010 (2010). https://doi.org/10.1109/INFCOM.2010.5461951
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL 2019. pp. 4171–4186. ACL (2019). https://doi.org/10.18653/v1/N19-1423
8. Dürmuth, M., Angelstorf, F., Castelluccia, C., Perito, D., Chaabane, A.: OMEN: Faster password guessing using an ordered Markov enumerator. In: Proceedings of ESSoS 2015. pp. 119–132. Springer (2015). https://doi.org/10.1007/978-3-319-15618-7_10
9. Florêncio, D., Herley, C., Van Oorschot, P.C.: Pushing on string: The 'don't care' region of password strength. communications of the ACM pp. 66–74 (2016). https://doi.org/10.1145/2934663
10. Furnell, S., Esmael, R.: Evaluating the effect of guidance and feedback upon password compliance. Computer Fraud & Security pp. 5–10 (2017). https://doi.org/10.1016/S1361-3723(17)30005-2
11. Golla, M., Dürmuth, M.: On the accuracy of password strength meters. In: Proceedings of CCS 2018. pp. 1567–1582. ACM (2018). https://doi.org/10.1145/3243734.3243769
12. He, X., Cheng, H., Xie, J., Wang, P., Liang, K.: PassTrans: An improved password reuse model based on transformer. In: Proceedings of ICASSP 2022. pp. 3044–3048. IEEE (2022). https://doi.org/10.1109/ICASSP43922.2022.9746731

13. Herley, C., Van Oorschot, P.: A research agenda acknowledging the persistence of passwords. In: Proceedings of IEEE S&P 2012. pp. 28–36 (2012). https://doi.org/10.1109/MSP.2011.150
14. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: PassGAN: A deep learning approach for password guessing. In: Proceedings of ACNS 2019. pp. 217–237. Springer Nature (2019). https://doi.org/10.1007/978-3-030-21568-2_11
15. Houshmand, S., Aggarwal, S., Flood, R.: Next gen PCFG password cracking. IEEE Transactions on Information Forensics and Security pp. 1776–1791 (2015). https://doi.org/10.1109/TIFS.2015.2428671
16. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of ACL 2020. pp. 7871–7880. ACL (2020). https://doi.org/10.18653/v1/2020.acl-main.703
17. Li, J., Tang, T., Zhao, W.X., Wen, J.R.: Pretrained language model for text generation: A survey (2024). https://doi.org/10.1145/3649449
18. Li, S., Wang, Z., Zhang, R., Wu, C., Luo, H.: Mangling rules generation with density-based clustering for password guessing. IEEE Transactions on Dependable and Secure Computing pp. 3588–3600 (2023). https://doi.org/10.1109/tdsc.2022.3217002
19. Li, Z., Han, W., Xu, W.: A large-scale empirical analysis of Chinese web passwords. In: Proceedings of USENIX Security 2014. pp. 559–574. USENIX Association (2014), https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/li_zhigong
20. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory **37**(1), 145–151 (1991). https://doi.org/10.1109/18.61115
21. 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: Proceedings of IEEE ICC 2018 (2018). https://doi.org/10.1109/ICC.2018.8422243
22. Ma, J., Yang, W., Luo, M., Li, N.: A study of probabilistic password models. In: Proceedings of IEEE S&P 2014. pp. 689–704. IEEE (2014). https://doi.org/10.1109/SP.2014.50
23. Melicher, W., Ur, B., Segreti, S.M., Komanduri, S., Bauer, L., Christin, N., Cranor, L.F.: Fast, lean, and accurate: Modeling password guessability using neural networks. In: Proceedings of USENIX Security 2016. pp. 175–191. USENIX Association (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher
24. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space trade-off. In: Proceedings of CCS 2025. pp. 364–372. ACM (2005). https://doi.org/10.1145/1102120.1102168
25. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010). https://doi.org/10.1109/TKDE.2009.191
26. Pasquini, D., Gangwal, A., Ateniese, G., Bernaschi, M., Conti, M.: Improving password guessing via representation learning. In: Proceedings of IEEE S&P 2021. pp. 265–282. IEEE (2021). https://doi.org/10.1109/SP40001.2021.00016
27. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. Preprint, OpenAI (2018), https://openai.com/research/language-unsupervised

28. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019), https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
29. Rando, J., Perez-Cruz, F., Hitaj, B.: PassGPT: Password modeling and (guided) generation with large language models. In: Proceedings of ESORICS 2023. pp. 164–183. Springer Nature (2023). https://doi.org/10.1007/978-3-031-51482-1_9
30. Ur, B., Segreti, S.M., Bauer, L., Christin, N., Cranor, L.F., Komanduri, S., Kurilova, D., Mazurek, M.L., Melicher, W., Shay, R.: Measuring real-world accuracies and biases in modeling password guessability. In: Proceedings of USENIX Security 2015. pp. 463–481 (2015), https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/ur
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of NIPS 2017. pp. 5998–6008. Curran Associates, Inc. (2017), https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
32. Veras, R., Collins, C., Thorpe, J.: On semantic patterns of passwords and their security impact. In: Proceedings of NDSS 2014. Internet Society (2014). https://doi.org/10.14722/ndss.2014.23103
33. Veras, R., Collins, C., Thorpe, J.: A large-scale analysis of the semantic password model and linguistic patterns in passwords. ACM Transactions on Privacy and Security **24**(3) (2021). https://doi.org/10.1145/3448608
34. Wang, D., Wang, P., He, D., Tian, Y.: Birthday, name and bifacial-security: Understanding passwords of Chinese web users. In: Proceedings of USENIX Security 2019. pp. 1537–1554. USENIX Association (2019), https://www.usenix.org/conference/usenixsecurity19/presentation/wang-ding
35. Weir, M.: Probabilistic context free grammar (PCFG) password guess generator (2022), https://github.com/lakiw/pcfg_cracker/
36. Weir, M., Aggarwal, S., Medeiros, B.d., Glodek, B.: Password cracking using probabilistic context-free grammars. In: Proceedings of IEEE S&P 2009. pp. 391–405. IEEE (2009). https://doi.org/10.1109/SP.2009.8
37. Wheeler, D.L.: zxcvbn: Low-budget password strength estimation. In: Proceedings of USENIX Security 2016. pp. 157–173. USENIX Association (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler
38. Xu, M., Wang, C., Yu, J., Zhang, J., Zhang, K., Han, W.: Chunk-level password guessing: Towards modeling refined password composition representations. In: Proceedings of CCS 2021. pp. 5–20. ACM (2021). https://doi.org/10.1145/3460120.3484743
39. Xu, M., Yu, J., Zhang, X., Wang, C., Zhang, S., Wu, H., Han, W.: Improving real-world password guessing attacks via bi-directional transformers. In: Proceedings of USENIX Security 2023. pp. 57:1–57:18. USENIX Association (2023)
40. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: XLNet: Generalized autoregressive pretraining for language understanding. In: Proceedings of NeurIPS 2019. pp. 5753–5763. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper_files/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html
41. Zhang, H., Wang, C., Ruan, W., Zhang, J., Xu, M., Han, W.: Digit semantics based optimization for practical password cracking tools. In: Proceedings of ACSAC 2021. pp. 513–527. ACM (2021). https://doi.org/10.1145/3485832.3488025