



## 获奖证书

戴廷钧、季文俊、郑鹏、吕诗瑶、张梓恒、朱玉婷、蒋佳豪、李金璇、丁柏慧、方怡菲、王硕、马琦同学：

你们的作品《基于昇思 MindSpore 的通用场景智能高精解析创新应用》，在中国国际大学生创新大赛（2024）中荣获

## 铜奖

指导教师：范登平、程明明  
特发此证，以资鼓励。

主办单位：教育部、中央统战部、中央网信办、国家发展改革委、工业和信息化部、人力资源社会保障部、农业农村部、中国科学院、中国工程院、国家知识产权局、国家乡村振兴局、共青团中央、上海市人民政府

承办单位：上海交通大学、上海市闵行区人民政府

中国国际大学生创新大赛组委会

二〇二四年十月

证书编号：20240170C

铜奖委员会



中国工业与应用数学学会  
China Society for Industrial and Applied Mathematics



南开大学

学 生：陈星烨  
雷鸣远  
胡峻玮

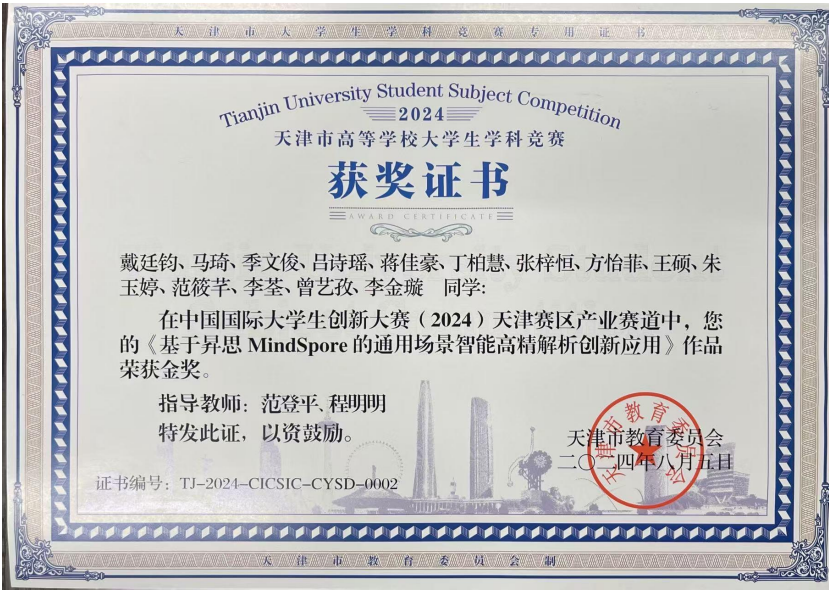
指导老师：向 兵

荣获二零二四年高教  
社杯全国大学生数学建模  
竞赛本科组二等奖。

中国工业与应用数学学会  
全国大学生数学建模竞赛组织委员会  
二零二四年十一月



项目地址: [Wenjun-Ji/BiRefNet.top](https://Wenjun-Ji/BiRefNet.top)  
在线网站: [General Scenario Intelligent High-precision Parsing](https://GeneralScenarioIntelligentHigh-precisionParsing.com)  
天津市市赛获奖证书:






































Commit 记录如下,

Commits on Jun 30, 2024		
Merge branch 'dev' of github.com:Wenjun-Ji/DIS into dev	6861bdb	<>
Wenjun-Ji committed on Jun 30, 2024		
add banner.tsx	3339ba7	<>
Wenjun-Ji committed on Jun 30, 2024		
Merge branch 'dev' of https://github.com:Wenjun-Ji/DIS into dev	99a42e7	<>
liangjiahao0208 committed on Jun 30, 2024		
CallToAction	324c1ee	<>
liangjiahao0208 committed on Jun 30, 2024		
CallToAction	b18dfb8	<>
liangjiahao0208 committed on Jun 30, 2024		
add .example.env	f24f341	<>
Wenjun-Ji committed on Jun 30, 2024		
Commits on Jun 24, 2024		
Update README.md	86b0daa	<>
Wenjun-Ji authored on Jun 24, 2024		
new README.md	c749694	<>
Wenjun-Ji committed on Jun 24, 2024		
using restorephoto.io with template	533b787	<>
Wenjun-Ji committed on Jun 24, 2024		
< Previous Next		

Commits on Jul 6, 2024		
fix banner picture display	Wenjun-Ji committed on Jul 6, 2024	f484b69
adjust css	Wenjun-Ji committed on Jul 6, 2024	99ac872
adjust	Wenjun-Ji committed on Jul 6, 2024	651cc0c
Merge branch 'dev' of github.com:Wenjun-Ji/DIS into dev	Wenjun-Ji committed on Jul 6, 2024	1f3c37c
add Business.tsx	Wenjun-Ji committed on Jul 6, 2024	5a0cb7c
grid-cols-3	Wenjun-Ji committed on Jul 6, 2024	d567680
UI finetune	Wenjun-Ji committed on Jul 6, 2024	ec95286
remove CustomButton	Wenjun-Ji committed on Jul 6, 2024	d5e8405
Commits on Jul 5, 2024		
Happy Banner Button	Wenjun-Ji committed on Jul 5, 2024	3632516
Add banner button	Wenjun-Ji committed on Jul 5, 2024	02f1f3c
rename restore.tsx to segment.tsx	Wenjun-Ji committed on Jul 5, 2024	0e0e1c0
Commits on Jul 4, 2024		
adjust RestoreHeader.tsx Application.tsx	Wenjun-Ji committed on Jul 4, 2024	aa56d35
fix text bug	Wenjun-Ji committed on Jul 4, 2024	7f79c53
Update README.md	Novadeu authored on Jul 4, 2024	2c0fcd3
adjust header.tsx RestoreHeader.tsx Contact.tsx	Wenjun-Ji committed on Jul 4, 2024	8c39485
Commits on Jul 3, 2024		
polish code	Wenjun-Ji committed on Jul 3, 2024	66f1e39
Merge branch 'dev' of github.com:Wenjun-Ji/DIS into dev	Wenjun-Ji committed on Jul 3, 2024	8a0a6dd
header link	Wenjun-Ji committed on Jul 3, 2024	c9718f8
Commits on Jul 2, 2024		
CustomCard	Wenjun-Ji committed on Jul 2, 2024	00a2fc6
Merge branch 'dev' of github.com:Wenjun-Ji/DIS into dev	Wenjun-Ji committed on Jul 2, 2024	84b3d91
add Header	Wenjun-Ji committed on Jul 2, 2024	8c2d881
update component position	Wenjun-Ji committed on Jul 2, 2024	f7d612e
ImageCarousel	Wenjun-Ji committed on Jul 2, 2024	4c1beeb7
install next 13.4.7	Wenjun-Ji committed on Jul 2, 2024	f12446f
add Support.tsx,Feature.tsx,PictureBox.tsx and some images	Wenjun-Ji committed on Jul 2, 2024	50b55bc
Commits on Jul 1, 2024		
Merge pull request #3 from MenIscus/dev	Wenjun-Ji authored on Jul 1, 2024	d477585
feat: BiRefNet Replicate API	MenIscus committed on Jul 1, 2024	53d54d0
Merge branch 'dev' of github.com:Wenjun-Ji/DIS into dev	Wenjun-Ji committed on Jul 1, 2024	0f64c5e
set link	Wenjun-Ji committed on Jul 1, 2024	38b2951
Merge pull request #2 from Wenjun-Ji/revert-1-master	Wenjun-Ji authored on Jul 1, 2024	3f4e18c
Revert "feat: BiRefNet API"	Wenjun-Ji authored on Jul 1, 2024	9da1fb9
Merge pull request #1 from MenIscus/master	Wenjun-Ji authored on Jul 1, 2024	386d2c8
feat: BiRefNet API	MenIscus committed on Jul 1, 2024	27ae5f3
Commits on Jun 30, 2024		
Update CallToAction	Wenjun-Ji committed on Jun 30, 2024	c508996
adjust width	Wenjun-Ji committed on Jun 30, 2024	33811cc
< Previous   Next >		

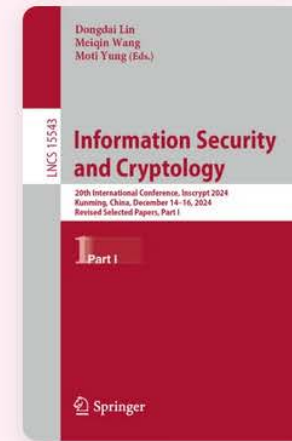
Commits on Jul 6, 2024

 feat: SpeedInsights	MenTscus committed on Jul 6, 2024	ba446ba		
Project detail finetune	 liangqiang0208 committed on Jul 6, 2024	9e8a878		
Merge pull request #5 from MenTscus/dev	 MenTscus authored on Jul 6, 2024	Verified faf59b3		
Merge branch 'Wenjun-Ji:dev' into dev	 MenTscus authored on Jul 6, 2024	Verified f77ee07		
 feat: Sample Download and Crop	 MenTscus committed on Jul 6, 2024	1eed3d6		
adjust	 Wenjun-Ji committed on Jul 6, 2024	74fa184		
Merge pull request #4 from MenTscus/dev	 MenTscus authored on Jul 6, 2024	Verified ae19a67		
 feat: Demo Sample	 MenTscus committed on Jul 6, 2024	8e82a7e		
add images for PictureBox Application adjust footer	 Wenjun-Ji committed on Jul 6, 2024	7591b29		
CallToAction vedio fill	 liangqiang0208 committed on Jul 6, 2024	3a67a1a		
add Contrast.tsx	 Wenjun-Ji committed on Jul 6, 2024	04e116e		


# EditPSM: A New Password Strength Meter Based on Password Reuse via Deep Learning

Conference paper | First Online: 03 May 2025

pp 255–276 | [Cite this conference paper](#)



**Information Security and Cryptology**  
(Inscrypt 2024)

[Yifei Zhang](#), [Zhenduo Hou](#) , [Yunkai Zou](#), [Zhen Li](#) & [Ding Wang](#)

 Part of the book series: [Lecture Notes in Computer Science](#) ((LNCS, volume 15543))

 Included in the following conference series:  
[International Conference on Information Security and Cryptology](#)

 231 Accesses

Access this chapter

Log in via an institution 

Subscri

 Springer+

 Starting from 10 c  
month

**SPRINGER N**

We want your fe

Fill out our quick survey and enter:

 截图工具

屏幕截图已复制到剪贴板  
已自动保存到屏幕截图文



Inscript 2024 2024-11-11 12:56

收件人: 我 [收起](#)

发件人: Inscript 2024 inscript2024@easychair.org

收件人: 我 [2213218@mail.nankai.edu.cn](#)

时 间: 2024-11-11 12:56:02

[英语](#) [翻译为 简体中文](#) [翻译邮件](#)

Dear [Yifei Zhang](#), Zhenduo Hou, Yunkai Zou, Zhen Li, Ding Wang,

We are delighted to inform you that your paper:

Paper ID: 110

Title: EditPSM: A New Password Strength Meter Based on Password Reuse via Deep Learning

Authors: Yifei Zhang, Zhenduo Hou, Yunkai Zou, Zhen Li, Ding Wang

was accepted to Inscript 2024. Congratulations!

We totally received 156 submissions this year. Since the conference is expected to be conducted in person, there was a stringent limit on the number of papers we could accept. This made the selection of papers extremely challenging and competitive.

The author instructions and reviews for your submission will send to you later.

Hope to see you at the conference!



# EditPSM: A New Password Strength Meter Based on Password Reuse via Deep Learning

Yifei Zhang<sup>1</sup>, Zhenduo Hou<sup>2(✉)</sup>, Yunkai Zou<sup>2</sup>, Zhen Li<sup>2</sup>, and Ding Wang<sup>2</sup>

<sup>1</sup> College of Computer Science, Nankai University, Tianjin, China

<sup>2</sup> College of Cryptology and Cyber Science, Nankai University, Tianjin, China  
joezhou13@nankai.edu.cn

**Abstract.** Password guessing attacks and users' vulnerable password behaviors pose severe threats to password security, and accurate password strength meters (PSMs) can mitigate these threats by nudging users to choose secure passwords. In this light, PSMs have been deployed by nearly every respectable web service and application during registration or password reset. However, most PSMs in both academia and industry only focus on the structural and semantic features of the password itself, and failed to capture users' vulnerable password reuse behaviors.

To fill this gap, we propose a new PSM, namely EditPSM, that takes users' password reuse behaviors into consideration. EditPSM is based on a targeted password guessing framework via deep learning, *and does not require users' existing passwords to protect users' privacy*. It utilizes popular passwords identified in the training set and learns how users modify them. This leads to credential tweaking models, a type of targeted password guessing model, to effectively evaluate password strength without needing access to users' existing passwords. Through extensive evaluations involving 10 large-scale datasets and 8 mainstream PSMs in the real world, EditPSM demonstrates its superior performance over prior art. We believe this work makes a substantial step towards introducing targeted password models into password strength evaluation.

**Keywords:** Password strength evaluation · Password security · Users' behavior · Deep learning

## 1 Introduction

Serving as a line of defense for users' cyber security and privacy, textual password firmly remains one of the most dominant methods in authentication due to their simplicity to use and low cost to deploy [8]. While passwords are widely used for authentication, effective password guessing models and users' vulnerable behaviors upon password selection pose significant threats to password security [7, 22]. In order to protect users from potential harm, it is vital for service providers to guide users to choosing more secure passwords.

Accurate password strength meters (PSMs) can provide timely feedback to users, hence assist users in creating stronger passwords [20, 24]. As a result,



nearly every respectable service now employs a PSM upon user registration or password reset, as an effort to encourage stronger passwords that can better resist password guessing [24]. For example, both 12306 [3] (A Chinese train ticket booking website) and Microsoft Edge browser [4] have employed PSMs that can show users the strength of their passwords in a binned form.

However, research has found that only PSMs providing accurate feedback can help users choose stronger passwords, those that are inaccurate would even do more harm than good [13, 24]. To address this issue, both academia and industry have made efforts to design accurate and robust PSMs in the past years. Pattern- and rule- based PSMs, such as zxcvbn [29], evaluate passwords based on character types (lowercase and uppercase letters, digits and symbols, etc.) or patterns observed in the password, and are widely used by service providers. Probabilistic-based PSMs, such as PCFG-PSM [15] and fuzzyPSM [23], are preferred by the academia. Mostly derived from trawling password guessing models [28], these PSMs assign probabilities to passwords. The lower the probability, the more secure the password is. Deep learning approaches to evaluating passwords have also been proposed, but prior work (see [13, 24]) deems that they fail to outperform their foremost counterparts of other technical routes.

Although numerous PSMs have been proposed over the years, most of them merely focus on the structural and semantic features of passwords. Their evaluation of password strength is largely based on password composition or distribution, overlooking users' vulnerable reuse behaviors. To the best of our knowledge, among mainstream PSMs, only fuzzyPSM [23] and vec-PPSM [18] take users' password reuse behaviors into consideration. Based on statistical methods, fuzzyPSM does not learn how passwords are reused individually, instead it treats all passwords in a dataset as a whole, and learns how passwords in one dataset are constructed by reusing segments or editing whole passwords from another dataset. However, the statistical method employed by fuzzyPSM [23] limits its ability to capture more complex reuse behaviors. Vec-ppsm, on the other hand [18], requires sister passwords (i.e., a password of the same user leaked by other services), raising privacy concerns for its real-world deployments.

Additionally, academia-designed PSMs are mostly based on password guessing models. The intuition is that as they can crack passwords, they can also estimate password strength from attackers' perspectives [15, 17]. As the best-performing type of password guessing frameworks, credential tweaking (i.e., reuse) models modify users' sister passwords to generate new guesses, achieving staggering success rates of nearly 70% [26, 30]. However, designing PSMs based on password reuse behaviors without users' sister passwords remains a challenge.

## 1.1 Motivations

Recent research has found that users' insecure password behavior could greatly endanger the security of passwords [11, 25]. According to a recent survey conducted by the LastPass security enterprise [2], over 65% of users tend to directly reuse or slightly modify a sister password to create passwords for a new service. Most PSMs in both academia and industry, however, fail to capture such user

behaviors, focusing only on password composition and distribution. In contrast, in the design of password guessing models, the academia is well aware of users' reuse behaviors, and has made efforts to design various credential tweaking models exploiting users' sister passwords. The situations stated above give rise to the following two key research questions (**RQs**) :

**RQ1:** How to design a PSM based on credential tweaking model that can capture users' password reuse behaviors without knowing user's sister password?

**RQ2:** If a PSM derived from a credential tweaking model can be constructed, how can it ensure the performance over existing PSMs in prior art?

To answer the research questions above, we design and implement a PSM aiming to capture users' password reuse behavior, namely EditPSM, without requiring access to the users' sister passwords (leaked passwords), based on the mainstream credential tweaking model Pass2Edit [26]. Furthermore, we evaluate and show the performance of our EditPSM.

## 1.2 Contributions

We summarize our contributions in this work as follows:

**A New Technical Route.** To use credential tweaking models in password strength evaluation when users' sister passwords are unavailable, we present a brand-new and generic technical route that draws top passwords in an untargeted, fairly ordinary password dictionary, and further derives an intermediate training set. Such an intermediate training set can be utilized by any credential tweaking model, enabling these models to learn users' password reuse behavior.

**A New Password Strength Meter.** We, for the first time, introduce credential tweaking models to practical password strength evaluation. Our proposed EditPSM employs a deep learning framework, and can capture users' reuse behaviors. By generating an intermediate training set from a single password dictionary, *EditPSM does not require sister passwords of any specific user to learn users' password reuse behavior*, as they are unavailable in practical cases. While fuzzyPSM captures users' reuse behavior at dataset level and requires two distinct password datasets [23], our EditPSM captures users' reuse behaviors password by password, with only one password dataset as the minimum requirement.

**Extensive Evaluation.** We conduct a series of experiments involving 10 large real-world datasets and 9 password strength meters, including 8 mainstream PSMs and our proposed EditPSM, and measured their accuracy. The results demonstrate the effectiveness of our EditPSM, and validate our claim that credential tweaking models can also be applied to practical password strength evaluation scenarios, while outperforming other PSMs.

**Some Insights.** Previous work put emphasis on reusing sister passwords, presuming that users’ password reuse behaviors is limited only to modifying their own passwords [11]. In contrast, we observe that a large proportion of passwords are similar with popular passwords, indicating users also modify popular passwords or use segments of popular passwords to create new ones. Furthermore, we find that this behavior differs for users of different languages, Chinese and English users to be specific. These observations provide a new perspective for understanding password reuse.

## 2 Preliminaries and Related Work

In this section, we elaborate on the preliminaries of a PSM based on targeted password guessing frameworks, and briefly review related work.

### 2.1 Evaluating Password Strength Using PSMs

**Real-World Password Strength Evaluation Scenarios.** Password strength should be evaluated under both online and offline guessing scenarios. In the online guessing scenario, PSMs are designed to defend against both trawling and targeted guessing attacks. However, existing PSMs fall short of evaluating password strength in response to this threat, as they unrealistically require users’ sister passwords, causing privacy concerns. In offline guessing scenarios, PSMs focus on evaluating the time required to crack a hashed password leaked from a service provider. In this work, we mainly focus on the more threatening online guessing scenario, and leave offline guessing scenarios as future work.

**Password Strength Meters.** The most common and practical way to evaluate password strength is to use password strength meters (PSMs) [21]. For password strength evaluation in online guessing scenarios, an ideal meter [23] should precisely reproduce the probability distribution  $\chi$  of passwords in a target dataset. Such an ideal meter can be described as:

$$M(pw) = p_{pw}, \forall pw \in \Gamma, \quad (1)$$

where  $p_{pw}$  is the true (but not known) probability of password  $pw$  drawn from  $\chi$ . This meter is ideal because when an attacker attempts to guess user passwords online without prior knowledge of a specific user, his best strategy would be guessing in descending order of password probability against the targeted service [7]. Hence, this meter could ideally prevent users from selecting passwords vulnerable to such guessing schemes. In this light, any real-world PSM can be seen as an approximation of this ideal meter, and we could evaluate the accuracy of a PSM by comparing the outputs of such PSM with that of the ideal meter. We will describe this evaluation procedure in detail in Sect. 4.4.

## 2.2 Previous PSMs

Both academia and industry have proposed various PSMs. We choose eight representative PSMs to be compared with our proposed EditPSM. As those PSMs are in different technical routes, we now look into their characteristics.

**Industry-Designed PSMs.** We select the representative 12306 PSM [3], Microsoft PSM (as described in [1, 24]) and zxcvbn PSM [29] from the industry to be compared with EditPSM. 12306 PSM is implemented by 12306, a website for train ticket booking for millions of travelers in China, and Microsoft PSM was once adopted by services such as Outlook and Skype. They are real-world PSMs that serve billions of users, hence we believe the performances of these PSMs reflect the general standards prevalent in industry-designed PSMs. Following the recommendations of Melicher et al.’s [17], we use zxcvbn to represent best-performing PSMs in the industry. All Industry-designed PSMs above do not require training sets.

**Statistics-Based and Similarity-Based PSMs.** PCFG-PSM [15] is based on one classic guessing framework, probabilistic context-free grammar password guessing model. As Wang et al. [23] have found that PCFG-PSM performs better than Markov-PSM [9], we use PCFG-PSM in this work for comparison. For similarity-based PSMs, we use LPSE proposed in [14], which evaluates passwords by using the similarity between the user’s password and the standard strong password, without using any training dataset.

**Deep-Learning-Based PSMs.** At USENIX Sec’17, Melicher et al. [17] proposed a password guessing model based on Recurrent Neural Networks, and designed RNN-PSM based on the model’s output probability. The framework they designed could be compressed to as little as a few hundred kilobytes and could be deployed client-side. Another example of deep-learning-based PSMs is the CNN-PSM proposed in [19] that utilizes Convolutional Neural Networks. The PSM they designed is highly interpretable and provides character-level strength feedback. We use the source code generously open-sourced by the authors of these PSMs, as to avoid incorrect implementations leading to biased results.

**Reuse-Based PSMs.** Currently, two PSMs stand out for their efforts in capturing password reuse behavior: fuzzyPSM [23] and vec-PPSM [18]. Essentially, fuzzyPSM is an improvement of PCFG-PSM, employing the same statistical methods to capture user’s password reuse behavior at the dataset level. FuzzyPSM assumes that all passwords in one dataset are constructed by reusing segments or passwords from another dataset, and is trained on two different training sets to learn such reuse behavior. According to a recent study [24], fuzzyPSM is currently the best performing PSM when considering online scenarios. However, as fuzzyPSM requires two distinct password datasets, its



application to real-world situations is limited. Furthermore, statistical method that fuzzyPSM employed suffers from overfitting and sparsity issues [16], lacking generalization ability. Vec-PPSM, though based on a credential-tweaking model, requires sister passwords of users, which are considered as sensitive data. Since the use of such data is impractical, we only evaluate the performance of fuzzyPSM for comparison.

### 2.3 Password Reuse Behaviors and Targeted Password Guessing

**Users’ Password Reuse Behaviors.** According to surveys conducted by previous works [2, 11, 23], over 65% of users tend to directly reuse or slightly modify existing passwords for new services. These behaviors expose users to credential tweaking attacks, a form of targeted password guessing attacks where attackers exploit leaked passwords (sister passwords) of a specific user. Worse still, increasingly common password file leaks [5, 31] over the years have provided attackers with sufficient sister passwords for such attacks. While attackers can have access to users’ sister passwords, they aren’t available in most cases for service providers as they are sensitive data. This makes it difficult for PSMs deployed in online services to model users’ password reuse behavior.

**Targeted Password Guessing Frameworks.** There are two main types of targeted guessing frameworks, one focuses on exploiting users’ personal information, and the other utilizes users’ sister passwords (i.e., credential tweaking attacks) [18, 25, 26]. In this work, we mainly consider those that are based on password reuse, also known as credential tweaking models.

Credential tweaking models learn how users modify existing passwords and apply them to guess other passwords of the same user. Statistical methods are first employed to learn users’ reuse behaviors. For instance, Targuess-II [25] utilizes a probabilistic context-free grammar, achieving the success rate of a staggering 70% within 1,000 guesses when only one leaked password is provided. Deep learning-based models [18, 26] were later demonstrated to be even more effective. For example, Pass2Path uses a neural network to learn users’ password modifications on the character level, and predicts a sequence of edit operations for a given sister password [18]. These models were mostly made in academia to simulate attackers’ targeted guessing attacks. However, few works have successfully found effective defense mechanisms by utilizing credential tweaking models, as sister passwords are unavailable in most cases for service providers.

## 3 EditPSM: A New Password Strength Evaluation Model Based on Password Reuse via Deep Learning

We now elaborate on EditPSM, a new PSM based on a credential tweaking model. First, we explicate how applications of these models are made possible using our brand-new technical route.

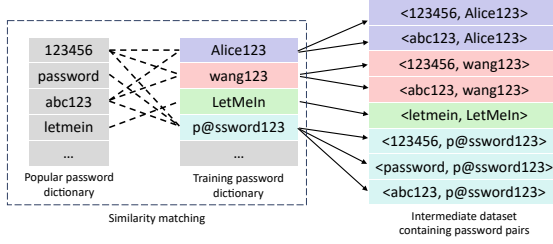
### 3.1 Extract Reuse Behaviors Using Credential Tweaking Models

Applying credential tweaking models to password strength evaluation is not a straightforward task: sister passwords of users are unavailable to service providers in most cases. To address this problem, we analyze reuse behavior from a different perspective. When users reuse passwords by modifying existing passwords, three main reuse behaviors exist. First, users can modify sister passwords they used for a different service [11]. This type of reuse behavior is widely studied, as over 60% of users adopt this behavior [2]. Second, users may also choose popular passwords that are easy to remember, and make changes to them. This behavior is also observed in previous studies [30]. For example, users may change *123456* to *123456!!* due to the password policy of a specific site. Third, users can reuse parts of popular passwords. This means that even if a user does not modify popular passwords in a direct manner, the modified password can contain segments from popular passwords. For instance, when a user chooses *mailpass*, though she may have not reused popular a password intentionally, this password in fact contains *pass*, which is a segment taken from the popular password “password”, hindering the password’s strength (see details in Sect. 4.2).

The observations above expand the border of users’ password reuse behaviors to a great extent, from reusing sister passwords to reusing popular passwords and parts of them. Though this behavior is not as common as reusing sister passwords, taking such behavior into consideration is beneficial, especially when we have to capture users’ password reuse behaviors when sister passwords of users are not available. In light of this perspective to understanding password reuse, we, for the first time, are able to apply credential tweaking models to practical password strength evaluation scenarios.

Specifically, as shown in Fig. 1, in order to make credential tweaking models learn how users reuse popular passwords or parts of them, we implement a matching phase to derive an intermediate dataset. Given a popular password dictionary and a training set, we traverse the entire training password dictionary to pair these passwords with popular passwords by employing similarity metric, forming password pairs. When multiple popular passwords have the similarity score above a certain threshold with a given password, we choose top- $k$  (e.g.  $k = 5$ ) popular passwords with the highest similarity, obtaining password pairs  $\langle ppw_1, pw \rangle, \langle ppw_2, pw \rangle, \dots, \langle ppw_k, pw \rangle$ , where  $ppw$  denotes a popular password, and  $pw$  represents password in the training dictionary. These password pairs compose the intermediate dataset for credential tweaking models to be trained on. In the case where only one password dataset is available, the entire dataset is the training dictionary as a whole, and we draw top  $m$  passwords, with  $m$  determined by a frequency threshold (e.g. 500) or a predefined number (e.g.  $10^3$ ), to form the popular password dictionary. See Appendix A for details.

Using the technical route above, now we are able to derive an intermediate dataset containing password pairs, where every password pair consists of a popular password and a similar password. Given that credential tweaking models essentially use password pairs during training [18, 26], the intermediate training set we



**Fig. 1.** The process of generating an intermediate dataset. For each password in the training password dictionary, we use similarity metric to find matches above a similarity threshold in the popular password dictionary, forming a number of password pairs.

described above can serve as a training set for these models. This approach enables the use of credential tweaking models for practical password strength evaluation, where models can be utilized to learn the reuse behaviors of popular passwords or their segments, instead of the reuse of sister passwords.

The technical route we propose answers the first research question (**RQ1**): it is possible to capture users’ behavior of modifying or reusing popular passwords. As sister passwords of users are no longer required in this case, credential tweaking models can be utilized in password strength evaluation. This technical route is highly scalable, as the generation of an intermediate dataset is not limited to using only one dataset. In fact, one can use predefined lists of popular passwords instead of top passwords drawn from a single dataset, or choose a different dataset for top passwords selection.

### 3.2 Pass2Edit: A Multi-step Decision Model for Credential Tweaking Attacks

Among the numerous credential tweaking models, Targuess-II [25], Pass2Path [18], Pass2Edit [26] and RFGuess [27] are the most promising. The model sizes of different credential tweaking models are shown in Table 1. We choose Pass2Edit as the base model for our EditPSM, since Pass2Edit is more accurate than Pass2Path and Targuess-II, and has the smallest model size. Note that other frameworks can also be utilized as the base model of our framework, since the capability to capture password reuse behavior and a probabilistic output are the only requirements. More concretely, Pass2Edit is a deep learning model based on neural networks, which is observed to have stronger generalization ability compared with statistic-based models. It uses Gated Recurrent Network as the backbone of its model architecture, and predicts how users modify a password by learning their reuse behaviors.

Pass2Edit [26] is built under the intuition that users edit existing passwords character by character. To be specific, given a password pair containing a user’s new password and a sister password at another service, Pass2Edit learns the character-level edit operations needed for the sister password to be transformed into the new password. Consider a password pair  $\langle pw_1, pw_2 \rangle$ , we define the edit

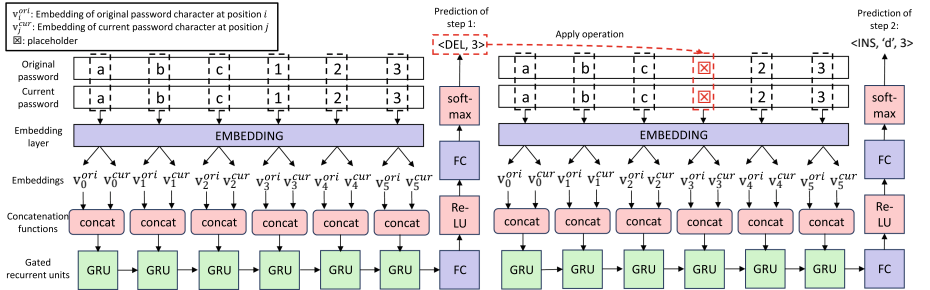
**Table 1.** Model size of different credential tweaking models

Model	Targuess-II [25]	RFGuess [27]	Pass2Path [18]	Pass2Edit [26]
Size	1.04 G	121 M	40.1 M	8.62 M

sequence needed to transform  $pw_1$  into  $pw_2$  as a series of atomic, character-level edit operations  $t = t_1, t_2, \dots, EOS$ , where  $EOS$  is the end-of-sequence symbol indicating that the transformation is complete. Note that though multiple edit sequences is possible, we obtain the sequence via the calculation of edit distance, and only choose a single edit sequence that has the shortest length. Formally, we define these character-level edit operations as follows:

$$t = \{(INS, p, c) | p \in \mathbb{N}, c \in \Sigma\} \cup \{(DEL, p) | p \in \mathbb{N}\} \cup \{EOS\}, \quad (2)$$

where  $p$  and  $c$  are the position and the character to be inserted or deleted,  $INS$  and  $DEL$  stand for insertions and deletions,  $\Sigma$  represents the character set, and  $\mathbb{N}$  is the set for natural numbers. When the model modifies a sister password, it predicts edit operations consecutively, applying previous operations to the original password. This is a multi-step generation task, where the model outputs an edit operation for the current password that has undergone a series of transformations. It's obvious that when we limit the maximum length of passwords, the total number of atomic operations is finite and definite. We follow the settings of the original Pass2Edit work [26], where the total number of operations  $|t|$  is 1,561. Therefore, for Pass2Edit, the process of generating guesses is a multi-step 1,561-class classification problem.



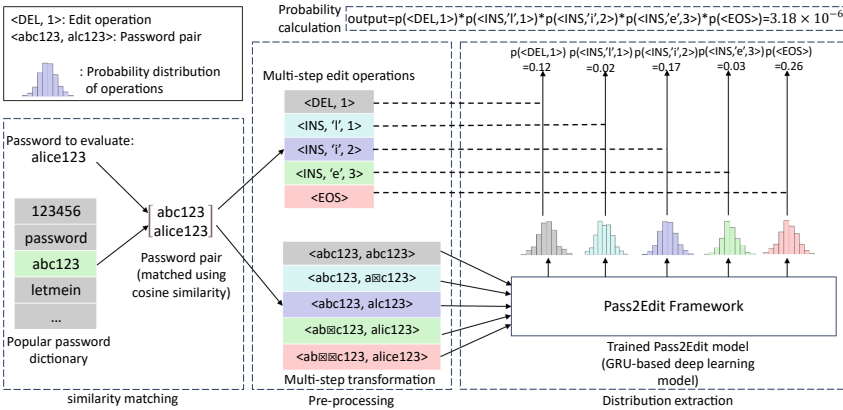
**Fig. 2.** The architecture of Pass2Edit [26], with an example of how it works in a single-step prediction. The model takes the original and the current passwords as inputs, and predicts the next edit operation, applying it to the current password. The updated current password then serves as input for the next step in a multi-step generation.

As shown in Fig. 2, Pass2Edit [26] uses gated recurrent units (GRU) [10] as the backbone of its architecture. For predictions within a single step, characters in the original and current passwords are input to an embedding layer, then



concatenated according to their positions. These embeddings are later passed to a 3-layer GRU (256 dimensions for each layer), and the output for the last character goes through two fully-connected layers, obtaining probabilities for every possible edit operation using a softmax layer. In order to generate the password with the highest probability, the model applies the most probable operation to the current password at every step, and uses this new password as input for the next step. This process terminates when an *EOS* is predicted, indicating that the transformation should come to an end.

The process above can only attain one single result. To generate more guesses, beam-search is applied to extract more outputs of the model. While enumerating guesses is also crucial, we do not elaborate on this process as we use Pass2Edit for password strength evaluation, only the training process of Pass2Edit is necessary for our model building.



**Fig. 3.** Simplified workflow of EditPSM during a password strength evaluation on-demand. For a given password, EditPSM finds the most similar popular password, forming a password pair which serves as input to the Pass2Edit framework. EditPSM further extracts the probability for edit operations from the multi-step prediction of Pass2Edit, obtaining the final probability (i.e. guessability) of the given password.

### 3.3 Modeling Password Strength Using EditPSM

The usage of EditPSM for password strength evaluation consists of three phases: matching, training, and strength estimation. The matching phase utilizes a training dictionary and a popular password dictionary, then generates an intermediate dataset containing password pairs. As is described in Sect. 3.1, while these two dictionaries can be derived from one single training set, the popular password dictionary can be a predefined password list, or selected from a different dataset.

In this work, for a fair comparison with other PSMs, we only use one single training set. Particularly, we use 2-gram cosine similarity score for our matching phase as recommended by Wang et al. [26]:

$$\text{sim}(pw_A, pw_B) = \frac{\sum_{g \in \mathbb{G}} (\text{count}(pw_A, g) * \text{count}(pw_B, g))}{\sqrt{\sum_{g \in \mathbb{G}} \text{count}^2(pw_A, g)} \sqrt{\sum_{g \in \mathbb{G}} \text{count}^2(pw_B, g)}}, \quad (3)$$

where  $\mathbb{G}$  is the set of all 2-gram substrings in  $pw_A$  and  $pw_B$ , and  $\text{count}(pw, g)$  represents the count of substring  $g$  in  $pw$ .

During the training phase of EditPSM, the intermediate dataset serves as input to Pass2Edit, and the model captures how users modify popular passwords and use segments of them to create new passwords. After matching and model training, we now have a model that can assign probabilities to any given password pair. Given a password pair  $\langle pw_1, pw_2 \rangle$ , with its edit operation sequence being  $t = t_1, t_2, \dots, EOS$ , we can calculate the probability of this password pair as:

$$\begin{aligned} \Pr(\langle pw_1, pw_2 \rangle) &= \Pr(t_1 | pw_1, pw_1) \times \Pr(t_2 | pw_1, pw_1^{curr}) \times \\ &\dots \times \Pr(EOS | pw_1, pw_n^{curr}), \end{aligned} \quad (4)$$

where  $pw_i^{curr}$  is the product of applying  $t_i$  to  $pw_{i-1}^{curr}$ . When  $pw_1$  is a popular password, it's obvious that the higher this probability, the lower the security of  $pw_2$ , since this probability essentially represents how likely  $pw_2$  is created reusing the popular password  $pw_1$ .

As shown in Fig. 3, in strength estimation phase where EditPSM evaluates a password on demand, it first finds the closest match of the password in the popular password dictionary. This matching process again results in a password pair that serves as input. The model then calculates the multi-step edit operations it requires for the popular password to be transformed into the password to evaluate. As the model outputs a probability distribution for every step in the transformation, EditPSM extracts the corresponding probability for each edit operation. After multiplying these probabilities step-by-step, the model gains the probability of the password pair, indicating the strength of the password to be evaluated. In a word, our EditPSM is a probabilistic password strength evaluation model that assigns probability to a password, where the probability reflects the likelihood of the password being created reusing popular passwords.

In addition, other than how users modify popular passwords, the intermediate dataset also comprises characteristics entailed by password distributions, as it is derived from the given dataset. When trained on such intermediate dataset, the probability output by EditPSM also implies a given password's probability within the password distribution observed in the given dataset. This enables EditPSM to estimate the strength of passwords that are not created by reusing popular passwords. The details are provided in Appendix A.

## 4 Experimental Setups and Results

### 4.1 Dataset Overview and Ethical Considerations

We utilize 10 large-scale password dataset in the real world for our experiments, in order to thoroughly evaluate the accuracy of EditPSM and its counterparts. Our datasets cover different types of services and languages. Though these datasets seem to be a little old, as revealed by Bonneau [7] and Liu-Blocki [6], passwords change marginally over time. Therefore, these datasets are still representative. The details are shown in the following Table 2.

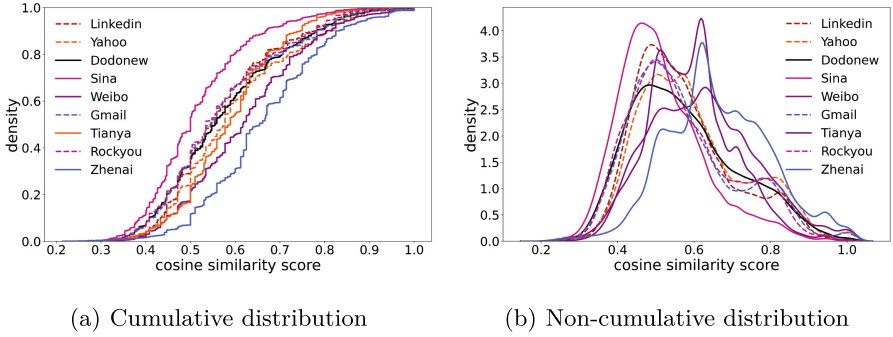
**Table 2.** Datasets used in experiments and evaluation.

Dataset	Service type	Language	Leaked time	Total PWs	Unique PWs
Tianya	Social forum	Chinese	Oct. 2011	30,901,241	12,898,437
Sina	Portal	Chinese	Dec. 2011	19,383,163	3,748,140
Dodone	Ecommerce	Chinese	Dec. 2011	16,258,891	10,135,260
Zhenai	Online dating	Chinese	Oct. 2011	5,260,229	3,521,764
Weibo	Social forum	Chinese	Dec. 2011	4,730,662	2,828,618
Linkedin	Job hunting	English	Jun. 2012	54,656,615	34,334,121
Rockyou	Social forum	English	Dec. 2009	32,575,500	14,330,075
Yahoo	Portal	English	Jul. 2012	5,626,485	3,439,492
Gmail	Email	English	Sept. 2014	4,929,086	3,119,299
Phpb	Tech forum	English	Jan. 2009	255,421	184,389

**Ethical Considerations** . While these datasets are leaked passwords and are publicly available, we still treat them with caution. In order to prevent further harm, we only present aggregated statistical information, avoiding potential leakage of personal information concerning emails, usernames, etc. As these data are available on the internet, the results in this work are reproducible.

### 4.2 Password Reuse Based on Popular Passwords

We design our EditPSM based on the intuition that users can modify popular passwords or use parts of them to create new passwords. To explore if users actually reuse popular passwords, for each dataset, we draw top 1,000 passwords as the popular password dictionary, and randomly select  $10^4$  passwords from the same dataset to save time. For each randomly selected password, we find a corresponding popular password with the highest similarity score (2-gram cosine similarity), and record the score which indicates the highest possible similarity between a password and the popular passwords. We then calculate the distribution of those similarity scores.



**Fig. 4.** Distribution of similarity score of sampled passwords in each dataset. The higher the similarity score, the more similar a password is with popular passwords observed in the corresponding dataset. It can be observed that Chinese datasets have a higher proportion of passwords that are highly similar to popular passwords.

As shown in the cumulative distribution in Fig. 4(a), a large proportion of passwords are highly similar to popular passwords, with the similarity represented by cosine similarity scores. Non-cumulative distribution in Fig. 4(b) reveals that curves of most datasets have over one peak. Peaks in areas of high similarity indicate that a large proportion of passwords are highly similar to popular passwords. We believe the reason behind this relatively uneven distribution is the users’ reuse behaviors of popular passwords. Additionally, we observe that passwords in Chinese datasets tend to be more similar to popular passwords, compared with those in English datasets. This indicates Chinese users reuse passwords more often, which is consistent with Wang et al.’s conclusions [25].

### 4.3 Experimental Setup

For password strength evaluation in online guessing scenarios we emphasize in this work, we choose  $10^4$  for the online guessing threshold as a rule of thumb. This threshold is also recommended by Wang et al. [24] in a systematic evaluation of PSMs. Generally speaking, there are two kinds of attackers that PSMs would have to defend against in online guessing scenarios [24].

**Knowledgeable Attackers.** As billions of passwords from different sites have been leaked and are publicly available [5, 31], we should be aware that some attackers can obtain leaked passwords from a specific service and learn its password distribution, hence targeting this service for a more effective attack. We refer to such strategy as the knowledgeable strategy, where the password distribution in the target dataset is known by a knowledgeable attacker. This strategy is powerful and yet realistic, as many sites have leaked their passwords more than once. We simulate how PSMs perform confronting such strategy by evaluating their accuracy for top  $10^4$  passwords in the targeted service, as they are the most vulnerable ones under knowledgeable attacks.



**General Attackers.** Attacker who failed to obtain the password distribution of the target service are referred to as general attackers. For general attackers, their ideal strategy would be to approximate the actual password distribution by utilizing common passwords (instead of the most popular ones) in the target dataset. As such passwords are often unavailable for general attackers, this ideal strategy should demonstrate their upper bound performance. We simulate this general strategy by randomly selecting  $10^4$  passwords from the test set, and observe the accuracy of PSMs on the strength evaluation of these sampled passwords.

**Table 3.** Training and testing scenarios for every PSM

Scenario #	Language	Training set A	Training set #B*	Test set
1	Chinese	Tianya	Zhenai	Weibo
2			Weibo	Sina
3			Weibo	Dodonew
4	English	Rockyou	Phpbbs	Linkedin
5			Gmail	Yahoo
6			Yahoo	Gmail

\*set #B is only used by fuzzyPSM [23], as it requires two training sets to characterize users' password reuse behavior.

We use the 10 large real-world datasets in Table 2 in our experiments, and set up six password strength evaluation scenarios for online guessing, three trained and tested on Chinese datasets, the other three trained and tested on English datasets. We conduct both the knowledgeable and general strategies for each scenario, and compare eight mainstream PSMs (see Sect. 2.2) with our EditPSM. As five PSMs require training data, the training set we use shall be as large as possible, ensuring the effectiveness of each PSM for a fair comparison. In this light, we follow the settings of Wang et al. [24] in a large-scale evaluation of PSMs, where they use Tianya and Rockyou datasets as training set A for PSMs requiring only one single training set. Since fuzzyPSM requires two training sets, an additional set #B is used. See Table 3 for more details.

#### 4.4 Evaluation Results

For measuring the accuracy of PSMs in online guessing scenarios, we choose the Weighted Spearman metric (*WSpearman* in short) advocated by Wang et al. [24], as the research demonstrated the soundness of such metric. The calculation of *WSpearman* can be described as:

$$WSpearman = \frac{\sum_{i=1}^n [w_i (x_i - \bar{x}) (y_i - \bar{y})]}{\sqrt{\sum_{i=1}^n [w_i (x_i - \bar{x})^2] \sum_{i=1}^n [w_i (y_i - \bar{y})^2]}}, \quad (5)$$

where  $x_i$  and  $y_i$  are the  $i$ -th elements of  $X$  and  $Y$ , the weighted rank vectors for outputs of ideal PSM and the tested PSM respectively, with  $w_i$  being the corresponding weight that equals to the  $i$ -th password's frequency. Note that  $\bar{x}$  and  $\bar{y}$  are weighted means of  $X$  and  $Y$ .  $WSpearman$  ranges from  $-1$  to  $1$ , with higher values indicating greater similarity between the outputs of the ideal PSM and the tested PSM, suggesting that the tested PSM is more accurate. As  $WSpearman$  is a single value for a given rank of passwords, we further utilize a  $WSpearman$  curve to show how PSMs perform as the rank of passwords changes. See Appendix B for more details on  $WSpearman$  curves.

Performances of PSMs may vary greatly when evaluating different passwords, as no single metric or PSM can be the most effective for all passwords [12], especially when different datasets are considered. Additionally, as more insecure passwords can cause greater harm, more importance should be attached to passwords of lower ranks (i.e. more frequent and less secure). In this light, while we use  $WSpearman$  curves (Figs. 5, 7) to observe the general performances of PSMs, we record discrete  $WSpearman$  values at several benchmark password ranks, and calculate the increment for EditPSM over other PSMs as relative  $WSpearman$  (denoted as  $RWSpearman$ ):

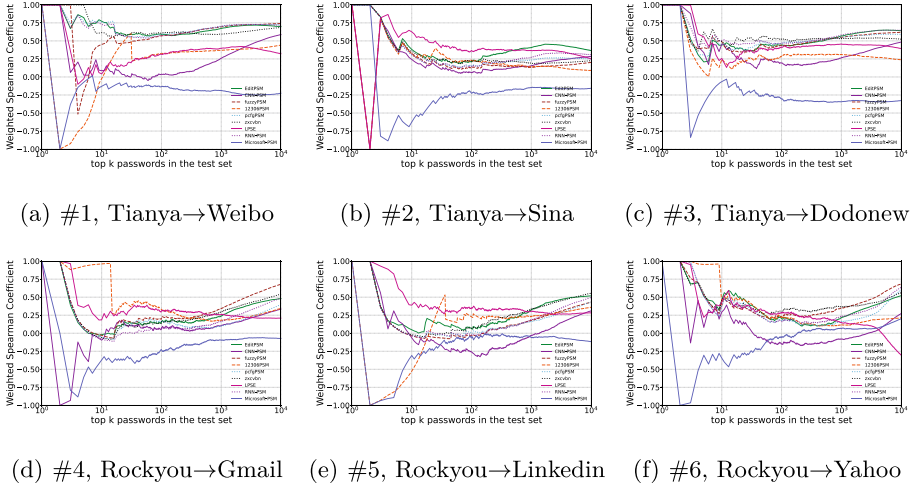
$$RWSpearman = WSpearman_{EditPSM} - WSpearman_{otherPSM} \quad (6)$$

As shown in the heatmaps describing  $RWSpearman_{relative}$  (Figs. 6, 8), we believe  $RWSpearman_{relative}$  at different benchmarks ( $Rank = 10, 10^2, 10^3$ , and  $10^4$ ) help the interpretation of results. Note that though at low ranks, PSMs with lower score may still classify these passwords as weak. However, since attackers attempts to crack as many accounts as possible within a limited guess budget, the effectiveness of PSMs is not measured as a classification problem, but a ranking problem, as is described by in the settings of an ideal PSM and online password evaluation scenarios (see Sects. 2.1 and 4.3 for details). By using these values, we can also calculate the improvement EditPSM achieves by using quantified results.

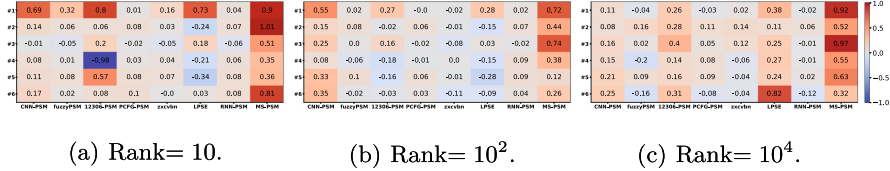
In addition, results for general strategy demonstrate the robustness of PSMs as the tested passwords are randomly selected. Related experiments are based on random samples and are not cherry-picked. Note that the exact results may shift when passwords are chosen differently.

Generally speaking, EditPSM is the best performing PSMs compared with its counterparts previously proposed by both academia and industry. Note that we do not claim EditPSM outperforms all counterparts in all cases, instead we claim EditPSM is among the few best-performing PSMs, and achieves similar or superior performance in general.

When we compare EditPSM with industry-designed PSMs, our EditPSM greatly outperforms 12306 PSM [3] and Microsoft PSM [1]. Specifically, as shown in Figs. 6, 8, EditPSM achieves an average increment of 0.222 and 0.6833 in  $WSpearman$  for all benchmarks under both scenarios. This is within our expectation, as PSMs proposed by industry often fall short on accuracy compared



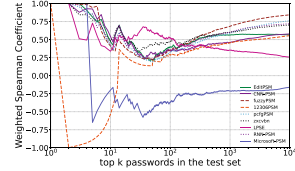
**Fig. 5.** Results for knowledgeable strategy. We mainly focus on these results, as test sets under knowledgeable strategy are based on true distributions of the ground truth. In these experiments, EditPSM is among the best-performing PSMs in all scenarios, demonstrating the potential of introducing credential tweaking models into password strength evaluation.



**Fig. 6.** Heatmap for results at different benchmark password ranks under knowledgeable strategy. Values within the heatmap indicate the increment (decrement if negative, rarely) in WSpearman value for EditPSM over the corresponding PSM. We can see that generally speaking, EditPSM is among the best performing PSMs, achieving significant increment for most PSMs.

with their academia counterparts. As shown in Fig. 6, our EditPSM has similar performance (with an average increment of 0.01166 in all benchmarks for knowledgeable strategy) with zxcvbn [29], the most promising PSM in the industry, and achieves an improvement of 0.045 on average for highly insecure passwords (i.e. at lower ranks of 10 and  $10^2$ , see Fig. 8).

For statistics- or similarity-based PSMs, it can be seen that EditPSM performs significantly better than LPSE [14], having an average increment of 0.175 on *WSpearman* for every experiment at all 4 benchmarks. As for PCFG-PSM [15], EditPSM achieves an improvement of 0.047 for knowledgeable strategy at all benchmarks in Figs. 5 and 6, and performs slightly better with an average increase of 0.0029 in general strategy. Specifically, as shown in Fig. 6, when tested

[illegible]

passwords are randomly chosen (i.e. general strategy), the average WSpelman of EditPSM at  $rank = 10$  is 0.137 higher than PCFG-PSM, implying that deep-learning technique employed by EditPSM addresses the over-fitting issue faced by statistical models, such as PCFG-PSM.

As our EditPSM is based on deep-learning, we emphasize on the comparison of EditPSM and two foremost deep-learning-based counterparts (i.e. CNN-PSM [19] and RNN-PSM [17]). As shown in Figs. 5, 6, among deep-learning-based PSMs, EditPSM performs the best. When compared with CNN-PSM using *WSpearman* metric, EditPSM greatly outperforms CNN-PSM [19] in almost all cases under the two strategies, with an average increment of 0.18 in all experi-



ments at the 4 benchmarks. EditPSM achieves higher accuracy than RNN-PSM [17] under knowledgeable strategy, having the increase of 0.032 in *WSpearman*. This indicates that our work takes one further step towards introducing deep learning approaches into password strength evaluation.

FuzzyPSM is the most prominent PSM for password strength evaluation in online guessing scenarios according to Wang et al. [24]. As fuzzyPSM also aims to capture users' password reuse behavior, we focus on the details of how EditPSM performs compared to fuzzyPSM. As shown in Figs. 5 and 6, our proposed EditPSM achieves a slightly higher accuracy than fuzzyPSM [23] under knowledgeable strategy, with an average improvement of 0.0254 in *WSpearman* at all benchmarks. We believe this improvement is the contribution of our new technical route: EditPSM learns more fine-grained password-level reuse behaviors, while fuzzyPSM essentially extracts dataset-level reuse behaviors, treating all passwords in the dataset as a whole. In general strategy, EditPSM is 0.089 higher than fuzzyPSM on average, for password ranks of 10 and  $10^2$  (i.e. most insecure passwords, see Fig. 8). Since EditPSM uses deep learning framework, it outperforms fuzzyPSM which uses statistical method in avoiding over-fitting.

The experimental results above answer the second research question (**RQ2**), as EditPSM reveals its effectiveness and comparably good performance, even compared with the foremost counterparts in both the industry and academia.

#### 4.5 Model Size and Speed

As deep learning frameworks are often large and computationally intensive, the size and feedback speed of PSMs based on deep learning could impact their usability. We recorded the average time it took for deep-learning-based PSMs to evaluate password strength and their model size. These results are recorded from scenario #2, where the training set is Tianya and the test set is Sina.

**Table 4.** Size and speed of deep-learning-based PSM.

PSM	Evaluation time per PW	Model size
EditPSM	2.71 ms	8.63 M
CNN-PSM [19]	4.74 ms	88.4 M
RNN-PSM [17]	1.72 ms	8.77 M

As shown in Table 4, we can see that among the three deep-learning-based PSMs, EditPSM is the smallest in size and achieves satisfactory speed. Since we conducted our experiments on a single NVIDIA GeForce RTX 3090 GPU, we believe EditPSM is fast enough for most service providers. Additionally, like RNN-PSM, EditPSM can also be compressed and deployed on the client side using optimizations and compressing techniques of Melicher et al. [17].

## 5 Conclusion

In this paper, we propose an effective deep-learning-based password strength meter utilizing credential tweaking models, namely EditPSM. By introducing a brand-new technical route that focuses on users' password reuse behaviors, EditPSM is the first password strength evaluation model that employs targeted password guessing frameworks based on password reuse, without requiring users' sister passwords (i.e., passwords leaked at other sites). We also look into how users create passwords by modifying popular passwords or using its segments, hence expanding the border of users' password reuse behavior.

Through extensive experiments with 10 large real-world datasets and comparisons with 8 mainstream password strength meters, we have demonstrated the effectiveness of EditPSM. For the first time, we find that credential tweaking models, the most effective type of targeted password guessing models, can be well applied to practical password strength evaluation scenarios, even when sister passwords are unavailable. We believe our work takes one firm step towards introducing targeted password guessing models into password strength evaluation, and contributes to better understanding users' password reuse behavior.

**Acknowledgments.** The authors are grateful to the anonymous reviewers for their invaluable comments. Zhenduo Hou is the corresponding author. This research was supported in part by the National Natural Science Foundation of China under Grant 62222208, and National College Students' Innovation and Entrepreneurship Training Program, China.

## Appendix A Setups for EditPSM and some other PSMs

**EditPSM.** For model training and parameter tuning, we follow the setups in the original paper for Pass2Edit [26]. However, in order to accelerate training speed, we use a cut-off frequency threshold for training passwords. Specifically, when we match passwords in the training set with popular passwords, we select passwords above a certain frequency threshold. In this work, we use  $f = 100$  for both Tianya and Rockyou datasets when used as training sets, and use top 1,000 passwords as popular passwords respectively. Though this cut-off process losses information in the training set, potentially hindering the effectiveness of EditPSM, EditPSM can be trained within minutes on Tianya or Rockyou dataset. This is useful in practice, as though we did not demonstrate the scalability of EditPSM when predefined popular password dictionary are applied, the ability to quickly adjust to new situations is important in practice. Furthermore, this indicates that the evaluation results only demonstrate the lower-bound performance of EditPSM.

Additionally, the training set contains duplicate passwords. For a password that has appeared  $m$  times in the training set, its corresponding password pairs appear  $m \times k$  times, with  $k$  being the total matches found in the popular password dictionary. Using this setting, EditPSM can learn not only the users' behaviors of reusing popular passwords, but also the distribution of in the given dataset.

**RNN-PSM.** As there are numerous parameter settings that came along the source code<sup>1</sup> of RNN-PSM [17], for the reproducibility of results, we explicate our parameter choices in our experiments. We follow the Melicher et al.’s settings [17] for the client-side model. That is, 3 LSTM layers with a hidden size of 256, and 2 fully-connected layers with 128 dimensions. As this setting has achieved similar or superior performance described in [24], it can guarantee fair comparison.

**Microsoft PSM.** Currently, Microsoft deploys a PSM in Edge browser, which shows password strength upon registration for other services for users of Edge. Though this update makes the performance of Microsoft PSM more crucial as it affects more users, we are unable to obtain the underlying mechanism for their new PSM at the time. We use the older version of Microsoft PSM in this work, as is also employed by and described in [1, 24].

**Treating Anomalous PSM Outputs.** Some PSMs do not output valid results for all input passwords. For example, zxcvbn do not output a valid result for passwords that contained only a space. We replace these results with zeros, and compute *WSpearman* accordingly. This helps in making a fairer comparison, as invalid outputs should be considered as a defect of such PSMs, and the zeros we assigned would most likely result in a decrease in their *WSpearman*.

## Appendix B The Calculation of WSpearman Curves

*WSpearman* [13, 24] represents the similarity between two weighted rank vectors. To obtain *WSpearman* curve as the password ranks increases, we first calculate the whole rank vectors consisting  $10^4$  elements. Later, for any given rank  $r$ , the corresponding *WSpearman* value can be calculated as:

$$WSpearman_r = WSpearman(X[:r], Y[:r]), \quad (7)$$

where  $X[:r]$  and  $Y[:r]$  represents the first  $r$  elements of weighted rank vectors  $X$  and  $Y$  respectively. *WSpearman* proposed in [24] did not specify on how to calculate *WSpearman* when the rank is 1, or when all elements in a vector has the same value. As these two cases result in divisions by 0, for the first case we define *WSpearman* = 1, as it only affects the first value in the curve. For the second case, we adjust the calculation for rank values within the same rank. For  $n$  tied elements with the same rank  $[a_j^0, a_j^1, \dots, a_j^n]$ , the rank value of  $a_j^i$  is:

$$rank_j = a_j + \frac{\sum_{k=0}^i w_k}{i} \times \frac{i+1}{2}, \quad (8)$$

where  $w_k$  is the weight of elements at location  $k$  within the tied elements. This adjustment only affect PSMs that output identical values for a large number of passwords, such as 12306 PSM and Microsoft PSM, as they only output 3

<sup>1</sup> see at [https://github.com/cupslab/neural\\_network\\_cracking..](https://github.com/cupslab/neural_network_cracking..)

or 4 different values for all passwords within the test set. The results of other PSMs remain unchanged after this adjustment. Hence, we believe our slight modification does not affect the soundness of the weighted spearman metric.

## References

1. Microsoft password checker. <https://devilsworkshop.org/microsoft-password-checker/> (Feb 2012)
2. The 2021 psychology of passwords report. <https://www.lastpass.com/resources/ebook/psychology-of-passwords-2021> (2021)
3. 12306 registration page. <https://kyfw.12306.cn/otn/regist/init> (Aug 2024)
4. Edge password health indicator. <https://support.microsoft.com/en-us/topic/password-health-indicator-5df7b4bc-cdb2-430a-9951-034accc57ff3> (Aug 2024)
5. Recently added breaches. <https://haveibeenpwned.com/> (Aug 2024)
6. Blocki, J., Liu, P.: Towards a rigorous statistical analysis of empirical password datasets. In: Proceedings of the IEEE S&P 2023, pp. 607–625
7. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: Proceedings of the IEEE S&P, pp. 538–552 (2012)
8. Bonneau, J., Herley, C., van Oorschot, P., Stajano, F.: Passwords and the evolution of imperfect authentication. *Commun. ACM* **58**(7), 78–87 (2015)
9. Castelluccia, C., Dürmuth, M., Perito, D.: Adaptive password-strength meters from markov models. In: Proceedings of the NDSS (2012)
10. Cho, K., et al.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Proceedings of the EMNLP 2014, pp. 25–29 (2014)
11. Das, A., Bonneau, J., Caesar, M., Borisov, N., Wang, X.: The tangled web of password reuse. In: Proceedings of the NDSS, pp. 23–26 (2014)
12. Galbally, J., Coisel, I., Sanchez, I.: A new multimodal approach for password strength estimation - part I: theory and algorithms. *IEEE Trans. Inf. Foren. Secur.* **12**(12), 2829–2844 (2017)
13. Golla, M., Dürmuth, M.: On the accuracy of password strength meters. In: Proceedings of the ACM CCS 2018, pp. 1567–1582 (2018)
14. Guo, Y., Zhang, Z.: LPSE: Lightweight password-strength estimation for password meters. *Comput. Secur.* **73**, 507–518 (2018)
15. Houshmand, S., Aggarwal, S.: Building better passwords using probabilistic techniques. In: Proceedings of the ACSAC 2012, pp. 109–118 (2012)
16. Ma, J., Yang, W., Luo, M., Li, N.: A study of probabilistic password models. In: Proceedings of the IEEE S&P 2024, pp. 689–704 (2024)
17. Melicher, W., et al.: Fast, lean, and accurate: modeling password guessability using neural networks. In: Proceedings of the USENIX SEC 2016, pp. 175–191 (2016)
18. Pal, B., Daniel, T., Chatterjee, R., Ristenpart, T.: Beyond credential stuffing: password similarity models using neural networks. In: Proceedings of the IEEE S&P, pp. 417–434 (2019)
19. Pasquini, D., Ateniese, G., Bernaschi, M.: Interpretable probabilistic password strength meters via deep learning. In: Proceedings of the ESORICS 2020, pp. 502–522 (2020)
20. Ur, B., Kelley, P.G., Komanduri, S., et al.: How does your password measure up? The effect of strength meters on password creation. In: Proceedings of the USENIX SEC 2012, pp. 65–80 (2012)

21. Van Acker, S., Hausknecht, D., Joosen, W., Sabelfeld, A.: Password meters and generators on the web: from large-scale empirical study to getting it right. In: Proceedings of the ACM CODASPY 2015, pp. 253–262 (2015)
22. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf’s law in passwords. *IEEE Trans. Inf. Foren. Secur.* **12**(11), 2776–2791 (2017)
23. Wang, D., He, D., Cheng, H., Wang, P.: fuzzyPSM: a new password strength meter using fuzzy probabilistic context-free grammars. In: Proceedings of the IEEE/IFIP DSN 2016, pp. 595–606 (2016)
24. Wang, D., Shan, X., Dong, Q., Shen, Y., Jia, C.: No single silver bullet: measuring the accuracy of password strength meters. In: Proceedings of the USENIX SEC 2023, pp. 947–964 (2023)
25. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: an underestimated threat. In: Proceedings of the ACM CCS 2016, pp. 1242–1254 (2016)
26. Wang, D., Zou, Y., Xiao, Y.A., Ma, S., Chen, X.: Pass2edit: a multi-step generative model for guessing edited passwords. In: Proceedings of the USENIX SEC 2023, pp. 983–1000 (2023)
27. Wang, D., Zou, Y., Zhang, Z., Xiu, K.: Password guessing using random forest. In: Proceedings of the USENIX SEC 2023, pp. 965–982 (2023)
28. Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: Proceedings of the IEEE S&P 2009, pp. 391–405 (2009)
29. Wheeler, D.L.: zxcvbn: low-budget password strength estimation. In: Proceedings of the USENIX SEC 2016, pp. 157–173 (2016)
30. Xiu, K., Wang, D.: Pointerguess: targeted password guessing model using pointer mechanism. In: Proceedings of the USENIX SEC 2024, pp. 5555–5572 (2024)
31. Zangre, A.: Comb data breach: what it means, and how to protect yourself. <https://blog.1password.com/what-comb-means-for-you-and-your-business/> (Feb 2021)