Design Lab - Mascara

Kousshik Raj (17CS30022)

12-11-2021

1 Introduction

Passwords are the most common mechanism to authenticate users on the Internet. However, despite the number of users flocking towards passwords and passphrases, the generation and management of the same are still a problem even today. This is primarily because users are not good in using passwords and passphrases that are strong and hard to guess but also easy to remember and enter.

Towards generating strong and memeorable passwords, we aim to leverage the advantage of generating suitable passphrase and then convert them to passwords according to our standards using the generated passphrases as context (e.g. using the starting and ending letters of each word with suitable alphanumeric value to join them). Infact, in scenarios like securing crypto-wallets, passphrases are even recommended over generic passwords because of the advantage they have over memorability and guessability.

Some of the common and standard methods of generating passphrases till now has been the algorithms that are variants of Diceware - a method where you choose random words from a given word list and combine them. However, studies have shown that passphrases generated by this approach are error-prone for the users and are not easily rememberable. On the other extreme, user generated meaningful passphrases are easy to remember, but they are easy to crack as well, thus making them less secure.

We therefore ask: *Is it possible to create usable passphrases which are easy to remember for users yet hard to guess for an adversary?*

This is what our proposed algorithm **Mascara** primarily tries to address. In the following sections, a brief overview of the algorithm and the results which answer the above question positively are presented.

2 Essentials

In this section we will address some key factors that are essential to understand, measure and use **Mascara** effectively.

2.1 Dataset

To generate memorable yet hard to guess passphrases, Mascara uses a constrained Markov generative process. The constraints are based on approximate memorability and guessability metric that we define. We first discuss the training data we use and how we train the Markov model.

For training Mascara, the required uni-gram and bi-gram Markov models are retrieved from the Wikipedia data as a corpus of human generated text data. A recent dump of Wikipedia articles was collected and their contents were extracted based on the titles of the articles. Only the top 5% articles based on the page view count, aggregated over the last five years were considered. This strengthens the scope of our corpus, without having to use a huge dataset, to cover all possible domains.

The extracted raw Wikipedia data was then cleaned by removing all tags, URL links and captions, followed by the removal of words with less than three characters as well as alphanumeric words. Finally, a textual corpus comprising of 3.3 million sentences and 455,614 unique words was obtained. This is used as a universal corpus for CER (memorability metric) and guessrank estimation throughout the paper. This corpus is also used to generate passphrases.

2.2 Memorability

Memorability by itself is very subjective and difficult to quantify. But it must be done all the same, so that we could get a idea about the results in the form that one can visualise. Towards that extent we have to establish a framework for the same.

A previous news-text entry task, showed that character error rate (CER) can be used as a proxy for memorability. CER measures the rate of error per character while typing a text. Henceforth, CER will be used to quantify memorability. The task also gave a model to predict the CER of a text. The following equation was then establish, inspired by their approach: $CER(s) = c_1 \cdot L_1(s) + c_2 \cdot L_2(s) + c_3 \cdot \sigma_{chr}(s)$.

Here, L_1 and L_2 denote the log of the probabilities of the passphrase s according to a unigram model and a bigram model, and σ_{chr} denote the standard deviation in the number of characters in each word in the passphrase; c_i denotes parameters that have been learned from the training data. The model after being retrained on the Markov model yielded a good fit ($R^2 = 0.58$) for the CER estimate with values being allotted for c_1, c_2, c_3.

2.3 Guessability

Now the quantification of the difficulty for an adversary to crack the passphrase comes. A good passphrase must not be easy to guess. Guessability of passwords and passphrases is measured based on their guess rank according to the attacker's understanding of the passphrase distribution. Passphrases with higher probability has lower guess rank - higher guessability. Here, it is assumed that the attacker estimates the probability of passphrases according to a bigram Markov model. This is reasonable as the generative method also uses a bigram Markov model.

The probability of a passphrase $s = w_0 w_1 \dots w_{n+1}$ according the Markov model M that was trained is $\Pi_{i=0}^{n} L_2(w_i w_{i+1})$. Here $w_0 =<$ start $>$ and $w_{n+1} =<$ end $>$. Estimating guess rank can be very expensive as often they require explicit enumeration of billions of passphrases. Previously, a Monte-Carlo method to estimate guess rank was proposed. Here, the strength of a passphrase generated by our setting is measured based on their guess rank according to a distribution. This works better than entropy-based approaches to distinguish between passphrases of similar lengths.

3 The Algorithm

The model aims to generate passphrases such that they follow some syntactic structure to be memorable, but ensure that they are harder to guess by having low probable bigrams. This is carried out via constrained Markov generation approach as described below.

3.1 Constrained generation of passphrases.

The passphrases in **Mascara** are generated incrementally starting with the start symbol. At each stage a word is selected such that it satisfies the constraints imposed by the score function. The score function is modeled on the observation that CER can be computed incrementally. That is, given a partially generated passphrase $s_i = w_1 \dots w_i$, one can compute the intermediate CER value using the following equation:

Type	Samples
Diceware	dreamscape manchuria dervish verbally whinging ferries portmore permanency downcast epilogue guarding richemont
User	dont forget the password just another happy ending ride with the wind
Markov	the problema standard kit during ultraviolet signals beamed there improvements achieved worldwide
MMAP	how does its 1 shire milk a jack the frothy thing faxed the aphorism the rivalry was misspelling prior to our eyeballs
Mascara	many local fiscal rights lee sung before god jackal with prayer requests

Figure 1: Three randomly sampled passphrases from each group of passphrases considered for evaluation.

$$C(w_1 \dots w_i) = c_1 L_1(w_i) + c_2 L_2(w_{i-1}, w_i) + c_3 \sigma_{chr}(w_1 \dots w_i)$$

Similarly, the guess rank is also estimated using the bigram probability, L_2. Thus, the following constraints while generating passphrase: $C(w_1 \dots w_i) \leq \theta_1$ and $L_2(w_{i-1}, w_i) \leq \theta_2$, where θ_1 and θ_2 are two parameters of the system. At every step of the generation, a word is randomly selected such that these two constrains are satisfied by the intermediate passphrase. Because the score C can be computed incrementally, we remove the words not satisfying this constraint from the support before sampling in the following step.

The thresholds provide control over the generated passphrases whether it should be closer to English text with some semantic meaning (making them easy to remember and easy to guess), or devoid of semantic meaning (harder to guess but also harder to remember). The optimal θ_1 and θ_2 were found by performing grid search across the entire feasibility range and choosing the model based on the ranking of the ratio guessability and memorability.

3.2 The final algorithm

From this, we can see that the proposed algorithms is a step-by-step approach to generate passphrases, under constraints of memorability and guessability, while preserving its syntax and meaning. The approach is greedy, and not optimal. But, replacing the corpus or changing any of the variables can essentially just be a straight swap with the existing one, based on user preference or need, thus making the algorithmic approach a generalized version of generating optimized passphrases step wise, based on the constraints.

An alternative approach would have been generating the whole passphrase of length len, and then checking if the CER and guessrank constraints are met. But it can be argued the approach followed is much better in terms of both resources used as well as the runtime. This has been verified by generating 1000 passphrases and comparing the runtime for both the versions, while keeping the rest of the parameters same.

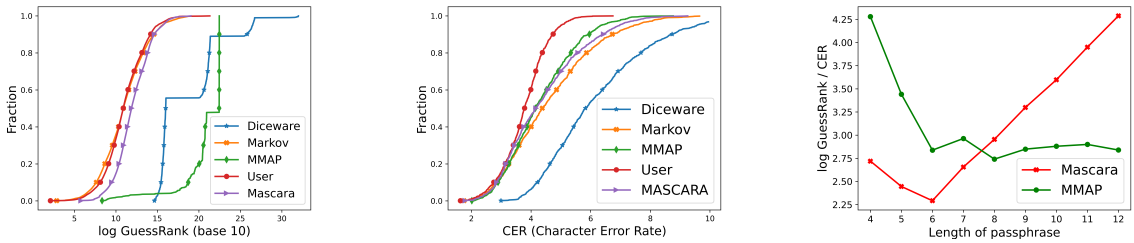

Figure 2: (a) CDF of guess rank (in log 10 scale) for passphrases. (b) CDF of CER(Character Error Rate) as rate of errors per character. (c) Average ratio of log guessrank and CER of passphrases of different lengths generated by Mascara and MMAP.

4 Results

With the metrics defined for guessability and memorability to compare between passphrases, the following five set of passphrases will be evaluated. 1000 passphrases were generated from each of the five approaches and then were evaluated.

- **Diceware:** Diceware was previously proposed for generating passphrases by choosing random words from a vocabulary. In our situation, we use our Wikipedia dataset after removing 5 most and least occurring words (which results in 360,868 words).
- **User:** Several passphrases were identified that were used by users from prior password leaks, which were then processed and segmented according to the needs to extract user passphrases.
- **Markov:** A bigram Markov model was trained over the words from the same Wikipedia dataset. Backoff smoothing will be used to estimate the probabilities of those pairs whose bigram is not present in the dataset.
- **MMAP:** This is a popular tool from the internet, called *Readable Passphrases*, which essentially chooses one of it's predefined English grammar template to generate a passphrase.
- **Mascara:** The proposed algorithm.

Random samples of passphrases from each set mentioned above are shown in Fig.1. We compare the memorability of these four sets of passphrases measured by CER and strength as measured by their guess rank.

4.1 Strength of the passphrases

For each model, a suitable threat model is used which would be optimal from the adversary's perspective, i.e, the threat model that gives off the least average guessrank. The results for the evaluation is shown in Fig.2(a). *Diceware* passphrases have the highest guess ranks - 50% of passwords will require at least 10^{16} guesses - as expected given they are sampled randomly from a large dictionary of 3.6×10^5 words.

The *User* and the *Markov* passphrases are the easiest to guess - 50% passwords can be guessed in less than 10^{10} guesses. Again, not surprising, given that the users often pick predictable phrases from English texts, and the attack algorithm is trained on Wikipedia dataset.

Mascara provides a modest improvement, requiring more than 10^{11} guesses to guess 50% of passphrases, but significantly less than that of Diceware. Guessrank for *MMAP* is pretty high compared to the other variations bar Diceware, but it has an upper bound. This is because, the highest order of combinations of the particular template approximates to the same order, when added with

ones with lesser order. So, regardless of the passphrase length, there will always be an upper bound limit to the number of tries required to crack the MMAP passphrases.

4.2 Memorability of passphrases

Now, the memorability of a passphrase based on the character error rate (CER) estimate is measured. The distribution of CER of the passphrases are shown in Fig 2 (b).

As expected, *Diceware* passphrases have the highest CER, meaning that users are likely to have a hard time remembering them and entering them correctly. It is hypothesized that the lack of any syntactic structure is responsible for such high CER.

User passphrases seem to perform the best and are expected to do so, given users, in general, choose highly common phrases, quotes, and song or movie titles. CER values of the *Mascara* passphrases are between the above two.

5 Conclusion

In this work, the first step towards systematic generation of memorable yet secure passphrases was taken with the help of constrained generation. A novel in-use passphrase dataset was presented and the linguistic properties were leveraged to propose **Mascara**. In the process, a framework for quantifying memorability and guessability for passphrases was created. It also highlights the importance of memorability when generating passphrase as well as the fact that a suitable tradeoff for memorability and guessability exists.