Using Probabilistic Context-Free Grammar to Create Password Guessing Models

Isabelle Hjelden
hjeld013@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

This paper will discuss two versions of probabilistic context-free grammar password-guessing models. The first model focuses on using English semantics to break down passwords and identify patterns. The second model identifies repeating chunks in passwords and uses this information to create possible passwords. Then, we will show the performance of each model on leaked password databases, and finally discuss the observations made on these tests.

Keywords: Authentication, Passwords, Security

1 Introduction

Text-based passwords are the most common form of authentication used by companies because they are cost-efficient, easy to implement, and familiar to users. Several companies have had security breaches in which users' passwords have been leaked. While it is an unfortunate circumstance for a company (and their users) to lose users' information, this now allows researchers to study people's passwords. By studying user-created passwords, researchers hope to identify how people create them and what makes certain passwords weak or strong against malicious attacks.

Passwords are made up of the characters and symbols on our keyboards, but the majority of the time, these passwords are not created randomly. Many password database leaks have shown that people commonly use information based on their life or easy-to-remember patterns. For example, researchers noted that frequently used words in passwords included: 'password,' 'love' and '4ever' [6, 8]. This information helps attackers narrow down their guesses of what someone's password could be.

But how do we know if our passwords can be guessed? Researchers have created password-guessing models to determine if people are making strong enough passwords. These models learn from previously leaked data to make guesses on what a password could be. The purpose of these models is to educate people on what makes a strong password and expose the vulnerabilities in weak password requirements.

This paper provides the necessary background needed to understand these models in Section 2. Then, we will look at two versions of password-guessing models that use probabilistic context-free grammar. Each model uses a different method for detecting patterns in passwords from the data sets. In Section 3, we discuss at the Semantic PCFG which focuses on the usage of semantics in the English language. By breaking down passwords into identifiable words and categorizing them, the model better understands why people select certain words for their passwords. In Section 4, we will look over the Chunk-Level PCFG, which uses a password-specific segmentation algorithm. Xu et al. [8] determined people may purposefully misspell words or use substitutions in words (e.g. a '0' for an 'o'). So by identifying characters that are commonly seen together in previous passwords, the model makes guesses using those discovered patterns. Finally, we will go over our conclusion of these two models in Section 5.

2 Background

In this section, background is given on the technology used to build password-guessing models. First, we present context-free grammar. The models discussed in this paper are each built using probabilistic context-free grammar, thus it is important to understand these concepts. Then we introduce previous models that use probabilistic context-free grammar and their influence on the models we will discuss in this paper. Finally, we will go over some information about the data leaks used in training and testing.

2.1 Context-Free Grammar

A context-free grammar is a formal grammar that can build any possible string within a given set of strings that is referred to as a language. There are three items required for building a grammar: production rules, non-terminal tokens, and terminal tokens. Production rules determine how the strings are supposed to be written. Non-terminal tokens can be described as a category or group, such as 'adjectives' or 'colors.' Terminal tokens are the final output of the grammar, such as 'red' [1]. The language defined by a grammar contains all strings that can be derived by all possible combinations of rules. In this discussion, a grammar specifies the language of all possible passwords that follow certain patterns.

As an example, let non-terminal tokens be 'adjectives,' 'nouns,' and 'numbers', terminal tokens be 'red,' 'dog,' and '3,' and use the following production rules: a password must

Region	Data sets	Year	Total Passwords	Unique Passwords	Passwords longer than 16 characters
English	RockYou	2009	32,584,165	14,270,373	250,242
	LinkedIn	2012	177,500,189	61,829,207	-
	000webhost	2015	15,251,074	-	-
	Neopets	2016	67,672,205	27,474,769	1,081,877
	Cit0day	2020	86,835,796	40,589,949	1,869,877
Chinese	CSDN	2011	6,425,243	4,031,226	68,817
	178	2011	9,071,979	3,461,974	7,431
	Youku	2016	47,607,615	47,462,025	395,701

Table 1. Data leaks used in training and testing the models

be an adjective followed by either a noun or a number; a password may end in a number. For current and future use, we will use the following abbreviations: *N* for noun; *ADJ* for adjective; and *NUM* for number. The following is how we would write this context-free grammar, with *PASSWORD* being our starting terminal:

<*PASSWORD*> → <*ADJ*><*N*> | <*ADJ*><*NUM*>
<*N*> → dog<*NUM*> | dog
<*ADJ*> → red
<*NUM*> → 3

With these production rules we have the following derivation:

<*PASSWORD*> → <*ADJ*><*N*> →reddog<*NUM*>→reddog3.

With this context-free grammar, we can also make the following passwords: 'red3' and 'reddog'. Note that this is a very limited grammar; the grammars used in actual models consist of much larger grammars and use lexical databases such as WordNet [2] for information on words to decide tags (discussed in Section 3).

2.2 PCFG Models

Now that we have an understanding of context-free grammar, we will look at an evolved version, probabilistic context-free grammar (PCFG), and how it is used in password-cracking models. PCFGs are the same as regular context-free grammars, only with the addition of probability attached to the production rules. The probability is determined by the amount of times a production rule is seen being used in a data set, also known as maximum-likelihood estimation (MLE).

As an example, let us take a look at our previous context-free grammar; In the starting terminal, we have two production rules for the same non-terminal: <*PASSWORD*> → <*ADJ*><*N*> and <*ADJ*><*NUM*>. Suppose in the training data set that the first rule was seen at a .6 frequency and the second rule at .4. We would attach these frequencies to the production rules, and the grammar takes the frequencies to determine what passwords are more likely to exist when generating guesses. In the real data tests, the probability of the majority of the production rules is under .1 frequency [6].

The first password-guessing model to use probabilistic context-free grammar was released in 2009 by Weir et al. [7]. The grammar for this model only consists of three non-terminal tokens: alphabetic, numeric, and symbol. For example, a production rule in this grammar may have the structure of $L_6N_2S_1$, where a password consists of 6 letters, followed by 2 numbers, followed by 1 symbol [6].

A major improvement in PCFG password-guessing was reported by Komanduri, who added word segmentation for the grammar to identify words within passwords. In addition, the Komanduri model [5] learns whole passwords, which helps the model perform better in the early stages of guessing. [6]

2.3 Data Leaks

To train and test the models, the researchers need a substantial amount of data. These leaks were chosen for testing and training because of their sizes as well as the years the companies were hacked. Each of these leaks gives a significant amount of data for the models to be trained and tested, with the small data leak containing 6.4 million passwords.

The years that these leaks happen are important to testing because they can help researchers determine whether a data leak used for training provide relevant to data leaks being tested on. Over time, password requirements have become stronger, meaning that users' passwords are becoming more complex than they were (e.g. many passwords in the RockYou data leak were comprised of a single word or exclusively numbers).

Another thing that is important to note about these data leaks is the amount unique passwords within these leaks. For example, The RockYou data leak contained over 32 million passwords, but only 14 million of the passwords words were unique. This means that 43.8% of these passwords are unique. In fact, passwords such as '123456' were used 290,731 times [4]. Table 1 shows all the data for the RockYou, LinkedIn, 000webhost, Neopets, Cit0day, CSDN, 178, and Youku data leaks.

3 Semantic PCFG Model

In this section, we will discuss the PCFG model that uses semantics to break down and create passwords. The semantic model is a probabilistic context-free grammar that focuses on understanding password patterns by syntactic and semantic information, such as parts of speech. Veras et al. [6] noted that previous models that used PCFGs did not encode vital information on natural languages, such as the logical form and word sense (e.g. parts of speech) people use in their creation of passwords [6]. For example, production rules in previous PCFG models would identify the word 'smart' as *<WORD>* instead of *<ADJECTIVE>*. We will then present a comparison between the semantic model versus the previous models.

3.1 Set-Up

For the PCFG to create production rules necessary for password guessing, it needs to be fed data. The method used for this is a text processing pipeline, which feeds a training set that contains data from password leaks. As seen in Figure 1, the text processing pipeline segments passwords by words, numbers, and symbols. Then each segmented piece is tagged with a parts-of-speech tag, then semantic tags if required. After the passwords have been segmented and labeled by their semantic tokens, they go through a generalization process so the less frequently seen semantics can be grouped into broader categories. Finally, the production rule based on the password is given by the pipeline. It is important to note that the model does not classify any misspellings or substitutions (e.g. *l0ve* and *4ever*) into the grammar, and any attempt to identify passwords with these characteristics will take significantly more time and guesses.

Like the first PCFG, the semantic model also uses maximum likelihood estimation to determine the terminal probabilities. However, this does not take into account production rules not seen in training samples (unseen strings) and can affect the models' performance with small training samples (smaller training samples mean less time for training the grammar).

Veras et al. [6] add terminal smoothing using the Laplace estimator to enhance the models' performance in guessing these unseen strings.

The equation for Laplace smoothing is:

$$\Theta_i = \frac{(x_i + \alpha)}{(N + \alpha d)} \tag{1}$$

where Θ_i is the resulting smoothed probability of category i; $i = 1, ..., d$ ranges over the number of string categories; x_i is the observed frequency of the category i in the training set; α is the number of times strings are assumed to be observed *a priori*; N is the sum of the observed frequencies under the non-terminal symbol, and d is the number of different categories under the given non-terminal (e.g. the non-terminal noun may have categories 'person', 'animal', etc.) [6].

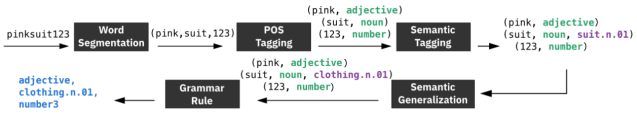

Figure 1. Example of the text processing pipeline breaking down a password (Adapted from [6]).

3.2 Testing

The study was done to test the effectiveness of the Semantic PCFG trained on passwords leaked from RockYou.com. From the list of passwords leaked from RockYou, the researchers created 51 training samples in five different sizes to train the models. When testing the semantic models, Veras et al. [6] ran two other PCFG models and a neural network model trained on the same samples. In Figure 2, the PCFG models are labeled 'Komanduri' and 'Weir', and the neural network model is labeled 'Melicher'. The neural network model used for comparison is the first neural network model to be created for password guessing in 2016 by Melicher et al. [3]. This model uses probability to guess the next character used in a password when guessing whole passwords. However, the neural network model could only be trained on one training sample from each size due to the amount of time required to properly train the neural network model (the time allotted was 140 hours) [6, 7]. After the models were trained, they were then ran to guess passwords. These passwords were compared to another data leak to determine if a password was correctly guessed. Veras et al. [6] also ran the tests with and without terminal smoothing to determine how it would improve the performance of the model. In addition, the researchers also ran the semantic model with and without semantic tagging but kept the parts-of-speech tagging on both models. In Figure 2 the grammar with the semantic tagging is labeled 'semantic', and the grammar without is labeled 'POS'.

The researchers decided to test the models on two separate data leaks, LinkedIn and 000webhost. Their reasoning is that the time difference between the LinkedIn leak and the RockYou leak is close (3-year difference) and the 000webhost leak and RockYou leak is not (6-year difference). The researchers wanted to determine if the time difference between the training samples and the data leak the model is being tested on affected the results.

3.2.1 Results. The first test on the LinkedIn password leak raised some notable points. First, the size of the training sets significantly affects the guessing performance (i.e. the larger the training sample, the more correctly guessed passwords). However, the training set sizes larger than a million did not make much difference compared to the training size of one million.

Second, the usage of terminal smoothing improved the results in the smaller training sets (training sets with under

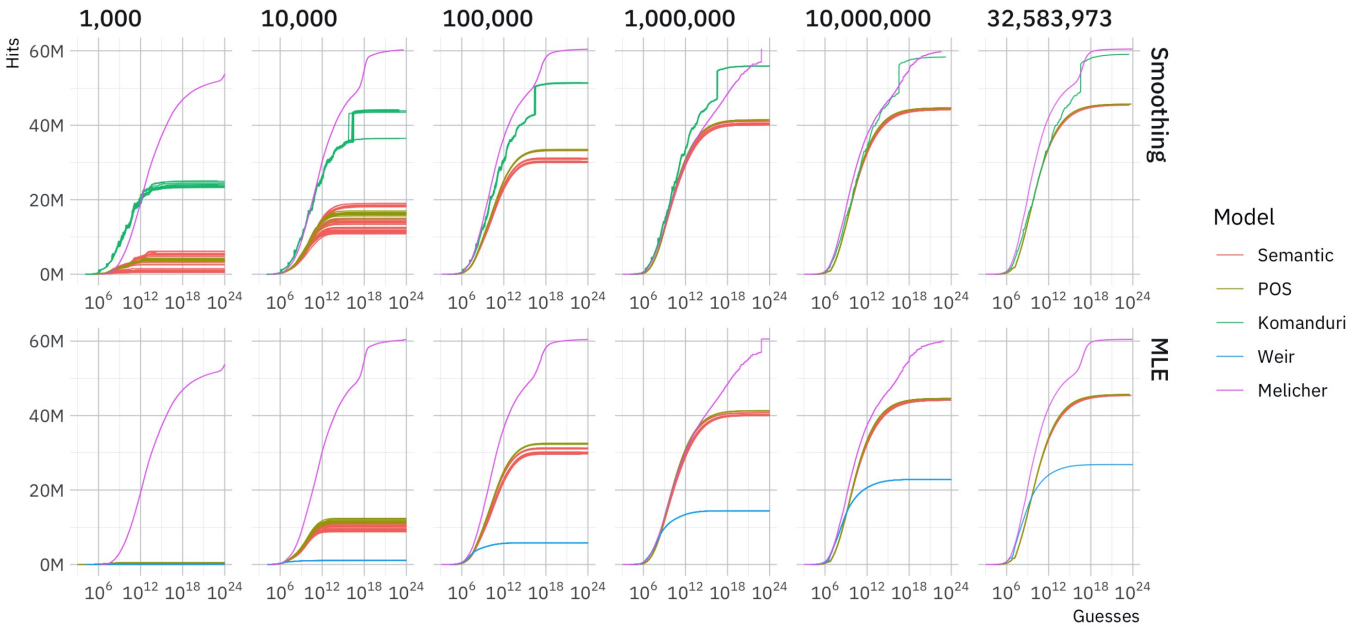

Figure 2. The results of the LinkedIn test. Each graph corresponds to the training set size as seen at the top. The total number of guesses made in each test is labeled on the horizontal axis and the number of correctly guessed passwords is on the vertical axis (Taken from [6]).

1 million passwords) than those without terminal smoothing (MLE).

Finally, the results show that there was not much difference between the grammars with semantic tagging and without. This led Veras et al. to three hypotheses: (1) the LinkedIn passwords lacked English semantics; (2) the LinkedIn passwords have English semantics, but their semantics are vastly different from the semantics found in RockYou; (3) general English passwords lack semantics [6]. Veras et al. hoped that these hypotheses would be answered with the second test on 000webhost passwords.

In the 000webhost test, Veras et al. saw similar results between the parts-of-speech and semantic grammar, where the number of cracked passwords was relatively the same. The Komanduri PCFG performed better than the semantic PCFG when the training sets were small but performed about the same as the larger ones (training sample sizes of over 1 million). They also noted the neural network model performed worse than all the PCFG models in the larger training samples; this is believed to be due to over-fitting. Over-fitting is when the model learns 'too well' from the training set and mimics the set rather than making accurate guesses.

The results from the 000webhost test suggest that Veras et al. [6] third hypothesis is correct. However, the researchers believe that because their model only guesses that are spelled correctly their vocabulary is limited. This means that any passwords with words that are semantically connected but with incorrect spelling are not guessed [6].

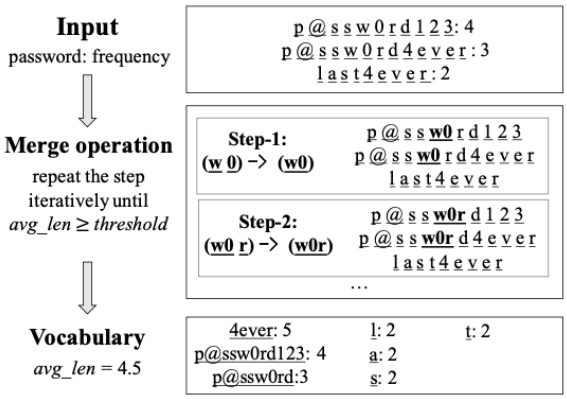

Figure 3. Example of the *PwdSegment* method (Taken from [8]).

4 Chunk-Level Guessing Model

In this section, we will go over the model created by Xu et al. [8]. The researchers propose that redesigning current models to use a password-specific segmentation will make a more efficient password-guessing model. Rather than the semantics method of determining formal dictionary words and digits commonly used in password words, the model defines chunks by learning individual characters' relation to others (i.e., what characters are found next to each other often) through training sets. This overcomes what the semantic PCFG failed to find in passwords: misspellings and substitutions. As an example, take the password "sk8er4ever";

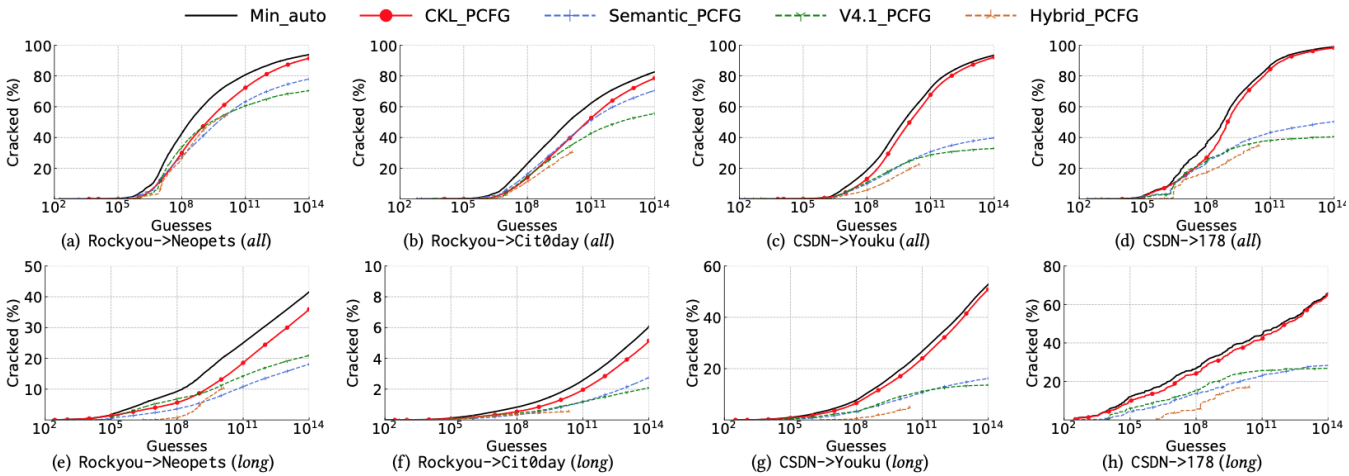

Figure 4. Results from Chunk-Level test. Each graph corresponds to a train set and a data set the models are being tested on. The top portion of the graphs are the data sets with all passwords and the bottom portion is exclusively passwords longer than sixteen characters. The horizontal axis is the number of guesses made by the models and the vertical axis is the percentage of correct passwords guessed (Taken from[8]).

this password can be broken into two chunks: "sk8er" and "4ever". Additionally, this password can be easily guessed as these two chunks had a high frequency in one of the training sets used for the test of the model.

The researchers created three different guessing models to test their method of a password-specific segmentation: Whole-String Markov, Template-Based PCFG, and Neural-Network-Based. We will be narrowing our focus on the Template Based PCFG model.

4.1 Set-Up

To train the model, Xu et al. [8] create a method called *Pwd-Segment*, which extends the preexisting Byte-Pair-Encoding (BPE) algorithm. BPE is a method of compressing data by identifying characters in a string often seen paired and merging them with a new identifier. In this case, this identifier is a chunk. Xu et al. change this method by adding an average length parameter to stop the recursive operation of merging pairs once the average length of chunks is equal to or greater than the threshold [8]. An example of the algorithm can be seen in Figure 3.

Instead of using non-terminals usually seen in other models [6, 7], Xu et al. [8] also created their own non-terminals to be used in the grammar. These non-terminal points are based on the chunks seen in passwords. There are a total of 7 non-terminals seen in the grammar: **L, U, D,** and **S** denote chunks with single types of characters (lower and uppercase, digits, and symbols); **DM** denotes chunks with two types of characters; **TM** denotes chunks with three types of characters, and **FM** denotes chunks with all four types of characters.

4.2 Testing

The training for this model uses the same method for training the grammar used for the semantic PCFG. In addition, the researchers separated passwords longer than sixteen characters and ran a second test with only these passwords. In Figure 4, *all* refers to the data set will all of the passwords, and *long* refers to the data set with the longer passwords only. Xu et al. [8] noticed in prior research that passwords from English-based websites typically contained words and letters with minimal usage of numbers, and passwords on Chinese-based websites contained numbers rather than letters. So they trained the model twice to test on English and Chinese passwords. The first training data set came from RockYou and was tested on Neopets and Cit0day, and the second training data set came from CSDN and was tested on Youku and 178 (see Section 2.3). In Figure 4, the chunk-level PCFG is labeled as 'CKL_PCFG'.

Xu et al. [8] chose to compare their PCFG to three other PCFG models: the semantic PCFG model, version 4.1 of the original PCFG model, and the Komanduri PCFG model (labeled as 'Hybrid_PCFG' in Figure 4). All of these models were also trained with the data sets from RockYou and CSDN.

4.2.1 Results. In all of the tests, the chunk-level PCFG outperforms all of the other models. In Figure 4, we see that the results from the English-based websites are close, but the results of the Chinese-based website have a large gap between the chunk-level model and the others. The researchers believe this is due to the fact the other models are built with a focus on English password sets. In addition, Xu et al. [8] point out that the percentage of correct guesses on the 178 (*all*) data set is nearly one hundred percent, they believe that this is because the data set is old and newer

password requirements have forced users to create stronger passwords [8].

5 Conclusion

Probabilistic context-free grammar password-guessing models are effective in correctly guessing passwords and help us identify the weaknesses in currently used passwords. PCFG models are more time efficient than neural network models in terms of training the models. The two PCFG models we have looked over have pointed out that using popular patterns and words makes passwords easier to guess. Having identified these flaws, we can now change password requirements to encourage people to make stronger passwords. As noted in the chunk-level PCFG testing, passwords longer than sixteen characters were much harder for the model to guess.

References

[1] [n. d.]. Context-free grammar - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Context-free_grammar. [Accessed 16-Oct-2022].

[2] [n. d.]. WordNet. https://wordnet.princeton.edu/

[3] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. [n. d.]. Fast, lean, and accurate: Modeling password guessability using Neural Networks. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher

[4] Angela Moscaritolo, Menghan Xiao, and Robert Roslonek. 2021. Rockyou hack reveals most common password: '123456'. https://www.scmagazine.com/news/network-security/rockyou-hack-reveals-most-common-password-123456

[5] Saranga Komanduri. 2016. Modeling the Adversary to Evaluate Password Strength With Limited Samples. (2016). https://doi.org/10.1184/R1/6720701.V1

[6] Rafael Veras, Christopher Collins, and Julie Thorpe. 2021. A Large-Scale Analysis of the Semantic Password Model and Linguistic Patterns in Passwords. *ACM Trans. Priv. Secur.* 24, 3, Article 20 (apr 2021), 21 pages. https://doi.org/10.1145/3448608

[7] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *2009 30th IEEE Symposium on Security and Privacy*. 391–405. https://doi.org/10.1109/SP.2009.8

[8] Ming Xu, Chuanwang Wang, Jitao Yu, Junjie Zhang, Kai Zhang, and Weili Han. 2021. Chunk-Level Password Guessing: Towards Modeling Refined Password Composition Representations. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 5–20. https://doi.org/10.1145/3460120.3484743