A Large-Scale Analysis of the Semantic Password Model and Linguistic Patterns in Passwords

RAFAEL VERAS, CHRISTOPHER COLLINS, and JULIE THORPE,
Ontario Tech University, Canada

In this article, we present a thorough evaluation of semantic password grammars. We report multifactorial experiments that test the impact of sample size, probability smoothing, and linguistic information on password cracking. The semantic grammars are compared with state-of-the-art **probabilistic context-free grammar** (**PCFG**) and neural network models, and tested in cross-validation and *A vs. B* scenarios. We present results that reveal the contributions of part-of-speech (syntactic) and semantic patterns, and suggest that the former are more consequential to the security of passwords. Our results show that in many cases PCFGs are still competitive models compared to their latest neural network counterparts. In addition, we show that there is little performance gain in training PCFGs with more than 1 million passwords. We present qualitative analyses of four password leaks (Mate1, 000webhost, Comcast, and RockYou) based on trained semantic grammars, and derive graphical models that capture high-level dependencies between token classes. Finally, we confirm the similarity inferences from our qualitative analysis by examining the effectiveness of grammars trained and tested on all pairs of leaks.

CCS Concepts: • **Security and privacy** → **Authentication**;

Additional Key Words and Phrases: Password guessing, PCFG, probabilistic context-free grammars, semantics

ACM Reference format:
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 (April 2021), 21 pages.
https://doi.org/10.1145/3448608

1 INTRODUCTION

Password models are data-driven structures that capture regularities in password samples and are useful to the analysis of password creation patterns, which often can be exploited in password guessing attacks. The Semantic PCFG [21] is a **probabilistic context-free grammar** (**PCFG**) that captures syntactic and semantic information. It assumes that, in addition to random sequences, people choose meaningful word combinations that form regular patterns when analyzed at scale. These patterns would resemble those found in natural language, but would not strictly obey

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) (RGPIN-2015-03916, RGPIN-2019-005803) and the Canada Research Chairs Program (950-232001). The first author was sponsored by Fundacao CAPES (9078-13-4/Ciencia sem Fronteiras).

Authors' address: R. Veras, C. Collins, and J. Thorpe, Ontario Tech University, 2000 Simcoe Street North, Oshawa, Ontario, Canada L1G 0C5; emails: {rafael.verasguimaraes, christopher.collins, julie.thorpe}@ontariotechu.ca.

2471-2566/2021/04-ART20 $15.00
https://doi.org/10.1145/3448608

natural language grammar rules. An exemplary instance of a semantic pattern is the dependency between adjectives and animal words, as in *cutedog*. When trained on primarily English datasets, the Semantic PCFG was shown to outperform the PCFG of Weir et al. [23] in guessing sessions where the targets were the LinkedIn, MySpace, RockYou, and Gamigo lists [11, 21].

The Semantic PCFG is a member of the family of linguistic password models, which relies on linguistic resources and processes, such as parsing, segmentation, and classification. Linguistic models remain valuable because they offer an interpretable description of a password list's composition, allowing researchers to study in detail differences between user populations and the impact of password policies. However, important questions regarding the behavior of such linguistic modeling remain unanswered. The individual contributions of different levels of information (e.g, syntactical, semantic) to generalization and, consequently, guessing performance, is unknown. The ability to learn patterns from small samples is not well understood, neither is the effectiveness of parameters that control overfitting, such as semantic specificity—a free parameter of semantic grammar training—and probability smoothing method, which is common to all linguistic approaches.

Furthermore, after the introduction of the semantic model the estimation of guessing success in *very long* sessions was made possible by the Monte Carlo strength evaluation [7]. This rendered the original evaluation of the semantic model obsolete, as it is limited to cracking attempts of up to 3 billion guesses. Newer models commonly use Monte Carlo evaluation, making comparison with older models difficult. In particular, the neural password model by Melicher et al. [16] attained excellent performance, surpassing other high scoring automated cracking approaches, including an improved version of the PCFG of Weir et al. In addition to a newer evaluation method, the community has also adopted more recent password leaks to test against, such as the 000webhost list.

In this article, we offer a nuanced study on learning linguistic patterns in passwords and an update of the performance of the Semantic PCFG using the most recent evaluation methods to provide results for longer guessing sessions [7]. We do so through large parameter sweep experiments that we deployed on a high-performance computing infrastructure. In the following sections, we report the results of experiments that compared the performance of PCFGs (a) trained with and without semantic symbols (WordNet senses and proper names); (b) trained with various levels of semantic generalization; (c) trained with password lists of various sizes; and (d) trained with maximum likelihood estimate, a method that assigns zero probability to unseen strings, and probability smoothing, which allocates probability mass to unseen strings [6]. Importantly, we compare three generations of PCFGs (Weir et al. [23], Veras et al. [21], and Komanduri [12]) with one of the latest neural network models (Melicher et al. [16]). Our experimental setups involve grammars trained with the RockYou list and tested on the passwords leaked from LinkedIn and 000webhost, plus independent cross-validation setups with the same data. Furthermore, we use linguistic grammars to investigate patterns qualitatively in three recent password leaks: 000webhost, Comcast, and Mate1. We present high-level graphical models of these leaks, discuss similarities in grammar rules, and conduct a cross-leak cracking experiment.

In summary, this article contributes a study of the effect of multiple parameters on the guessing performance of PCFGs and a neural network model, and a grammar-based qualitative analysis of recent password leaks.

2 LATEST ADVANCES IN PASSWORD MODELING

Since the first application of probabilistic grammars in password modeling, the field of password authentication has seen the introduction of ever more sophisticated **natural language processing** (**NLP**) into password models. Research in the last decade has mostly explored the space of statistical NLP, which includes corpus-based techniques for parsing and classification. Now, following a larger trend in computer science, neural network models are starting to appear, and results suggest that they may be superior in some aspects. In this section, we review the evolution of PCFGs and briefly discuss the latest developments outside the grammar-based paradigm.

2.1 PCFGs

Weir at al. [23] introduced the first PCFG password model. Prior to that, the academic literature considered password cracking as based on either brute-force or "dictionaries." Their PCFG relied on rudimentary NLP. Passwords were split into tokens of only three classes: alphabetic, numeric, and symbols. Combined with token length information, these classes formed the base structures of the grammar; for instance, the structure $L_8N_1S_1$, which represents eight letters followed by one number and one symbol, is learned from the password *password1!*. Notably, their text processing pipeline did not have the ability to break alphabetic strings into words (word segmentation); furthermore, alphabetic strings have uniform probability.

Houshmand et al. [10] improved over Weir et al.'s PCFG by enriching grammars with multi-word patterns, which require word segmentation, and keyboard patterns, such as *qwerty*. Each modification introduced by Houshmand et al. independently accounted for an increase in cracking performance over the Weir PCFG. Together, they accounted for increases in the range of 15% and 22% throughout a cracking session of 10^{12} guesses. On top of these improvements, a 33% increase in performance was achieved by selecting a word dictionary with better coverage (a dictionary based on RockYou words instead of the classic dic0294). Training and test sets were partitions of a dataset combining RockYou, Yahoo, and Hotmail passwords. Smoothing with Laplace estimation, which allows generation of guesses that feature unseen tokens, was responsible for modest gains.

Major improvements were also reported by Komanduri [12]. In addition to having word segmentation, Komanduri embedded letter case information in grammar symbols, and created symbols that encompass more than one class (e.g., number plus special characters). His grammar can also learn whole passwords as single tokens (untokenized), which improves performance in early guessing, as the grammar is able to effectively "memorize" high-probability passwords. As with Houshmand et al., the training aspect was improved with probability smoothing, using the Good-Turing estimator. In Komanduri's experiments, word segmentation accounted for a 66% increase in passwords guessed, while untokenized structures accounted for a 12% increase. However, extensive evaluations have not yet been performed on this model.

The works above added information to the original Weir model, but the model still lacked natural language information that is common in NLP applications, such as part-of-speech symbols. This may reflect an assumption that passwords are too simple and too short to exhibit the kinds of language patterns we see in natural language. Evidence to the contrary was published by Ur et al. [20], who counted parts-of-speech in common password leaks and found dependencies between classes; moreover, the distribution of parts-of-speech was clearly distinct from that of natural language, suggesting passwords have a unique syntax. This knowledge inspired the inclusion of part-of-speech and semantic symbols by Veras et al. [21], who achieved a success rate 67% higher than the Weir model in a test with early LinkedIn passwords (2012), and 32% higher with MySpace passwords. We review the model of Veras et al. in detail in Section 3.

Until recently, it was not clear how well a model-based guessing attack could compete with a professional attack. Ur et al. [20] presented a comprehensive comparison between automated and manual attacks. They found that probabilistic methods have steep success curves in the beginning of a session and flatten out as less probable guesses are made; while professionals, who often change the course of a cracking session upon feedback (by applying custom mangling rules), produce the opposite pattern and ultimately achieve higher success. This suggests that model-based cracking may be more suited to attacking slow hashes (e.g., bcrypt and scrypt).

Table 1. Linguistic Information in PCFG Models

Model	Character classes	Words	Part-of-speech	Semantics
Weir et al. [23]	x			
Komanduri [12]	x	x		
Houshmand et al. [10]	x	x		
Veras et al. [21]	x	x	x	x

We can organize the various grammar-based approaches to password modeling in a linguistic information spectrum. The grammars vary with regard to the number of partitions the observed passwords are broken into (Table 1). This can also be viewed as grammar complexity. On the most general end, Weir et al.'s grammar encodes character class information [23]. Komanduri [12] and Houshmand et al. [10] introduce word segmentation, and Veras et al. [21] adds another layer of detail by splitting strings based on their syntactical and semantic categories.

2.2 Other Techniques

More recently, neural models were introduced as a promising class. In tests with collections of crowdsourced passwords of varying complexity, Melicher et al. [16] found that a neural model (an LSTM) performed better than PCFG and Markov models after approximately 10^{10} guesses, especially against complex passwords that have three or four character classes, and long passwords (>16 characters). These neural models were trained with a compilation of RockYou and Yahoo passwords. The gap in performance between neural and the other methods was narrow when the test set was a sample of 30,000 passwords extracted from the 000webhost list. A neural model of the Generative Adversarial Network type trained with a sample of approximately 24 million RockYou passwords needed around one order of magnitude more guesses to achieve the same success of Melicher et al. in a limited universe of passwords with maximum 10 characters [9].

Zheng et al. [24] proposed a graph model with passwords represented as vertices and edges linking them when password similarity is higher than a threshold, where similarity can be defined in terms of string distance or probability. They found that some password lists yield graphs with higher density and connectivity than others, and observed a positive correlation between the vertex degree of a password and its frequency.

Dell'Amico and Filippone contributed a statistical estimate for the guess number of a password given a probabilistic model, such as a PCFG [7]. By calculating the guess numbers of all passwords in a test set, we can estimate the success curve of a cracking attack, saving us from having to enumerate all guesses. This method requires a clear text test set. First, a large random password sample is generated using the model's probabilities. Second, the conditional probabilities of all test passwords given the model are calculated. Finally, for every password in the test set the guess number is calculated as a weighted sum of the probabilities of all sampled passwords that are more probable than the test password. By pre-computing the weighted sum and storing the values in a sorted list, the estimation can be done in time $O(\log n)$.

3 THE SEMANTIC PCFG

Grammars are hierarchical models that describe the rules of production of a language. These rules specify how terminal symbols (the language vocabulary) are derived from non-terminal (abstract) symbols, such that one can produce a valid string by following a chain of derivations. In a context-free grammar the context in which a symbol appears does not affect its possible derivations, while in a PCFG the rules have assigned probabilities [15]. The semantic model is a PCFG where the

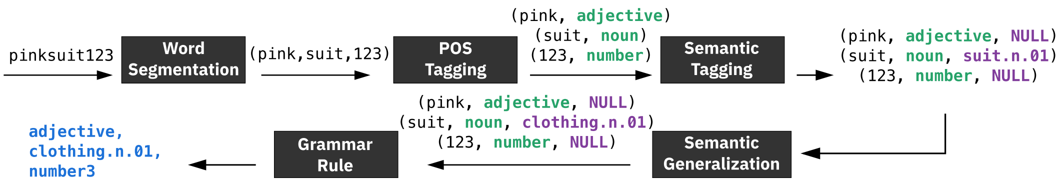

Fig. 1. Text processing pipeline for grammar training. Semantic tags are from WordNet. When words have multiple WordNet senses, the most frequent is selected. The semantic generalization process helps prevent overfitting by grouping related senses that have homogeneous frequency in the training data.

non-terminal symbols encode high-level linguistic information: **part-of-speech** (**POS**) and word sense. In our instantiation of the Semantic PCFG, described further below, this information comes from CLAWS7 tags [8] and the WordNet [17] hyponymy hierarchy of senses. These symbols allow the grammar to capture regularities in the syntax and meaning of passwords.

3.1 Text Processing Pipeline

The training data undergoes a text processing pipeline that breaks each password into tokens (i.e., terminal symbols) and assigns linguistic attributes to them, resulting in tuples of the form (token, POS, sense). Training the grammar consists in mapping these tuples to non-terminal symbols and building a probability distribution of symbols. A non-terminal symbol is an abstract label that groups similar tokens; for instance, all pronouns can be grouped under the symbol *PP*, or all words related to sports can be grouped under *Sports*. In the Semantic PCFG, non-terminal symbols are assigned following a backoff strategy [15]: if a word sense is not known, then it attempts to describe the string with a POS symbol; if that fails, it falls back to a simpler descriptor, such as N4 for a sequence of four digits. Thus, short random strings appended to semantically meaningful content are handled as non-terminals representing sequences of characters, digits, and symbols.

An overview of the text processing pipeline is shown in Figure 1. Word segmentation is based on Norvig's statistical algorithm [18], which selects the segmentation with highest joint probability, where the probability of each token is computed with a bigram model (from Google Web Trillion Word Corpus). POS tagging is done with a backoff tagger that is composed of general statistical models (trigram, bigram, and unigram) trained on the Brown Corpus and WordNet, and other niche named–entity corpora that are known to be relevant in passwords (cities, given names, and surnames).

Note that misspellings (e.g., *passwrd*) and substitutions (e.g., *passw0rd*) are not classified by the POS and Semantic grammars. Modules to classify misspelled words into semantic classes, for example, using Levenshtein distance, or to reverse substitutions, could be added in the future to improve both the POS and Semantic models.

3.2 Semantics

The word senses are taken from the WordNet corpus [17], a linguistic tree structure where concepts are linked by edges that represent IS-A relationships, as in dog IS-A animal IS-A mammal IS-A living thing. Each alphabetic string that is tagged as noun or verb by the POS tagger receives a semantic tag in the form of a WordNet sense key. Words may have many senses, so we select the sense with the highest WordNet frequency.

In order for word senses to have any generalization power, a mapping needs to be established so that low-level word senses are grouped into broader classes. This can be done by choosing a tree cut consisting of a set of abstract classes (internal tree nodes), each of which *represents* all of its descendants. As such, if the class "sport" is a tree cut member, any occurrence of "baseball"

Table 2. Semantic Categories (by Their WordNet Synset Key) Assigned to Words by Models with Different Specificity Levels

		Specificity		
Word	Frequency	5,000	1,000	100
love	94,190	love.n.01	love.n.01	attribute.n.02
girl	50,478	girl.n.01	adult.n.01	physical_entity.n.01
dog	17,282	dog.n.01	dog.n.01	physical_entity.n.01
freedom	2,204	freedom.n.01	freedom.n.01	attribute.n.02
pumpkin	1,411	pumpkin.n.01	plant.n.02	physical_entity.n.01
tsunami	146	movement.n.03	happening.n.01	psychological_feature.n.01
market	120	market.n.01	market.n.01	psychological_feature.n.01
suit	107	commodity.n.01	artifact.n.01	physical_entity.n.01
career	39	career.n.01	occupation.n.01	psychological_feature.n.01
nostalgia	23	nostalgia.n.01	desire.n.01	attribute.n.02

The models were trained with a sample (N=10M) of the RockYou password list. Specificity is a parameter of semantic generalization, a procedure that increases the abstraction of semantic categories to prevent overfitting.

and "basketball" will be labeled as "sport" in the trained grammar. This mapping has the effect of broadening the scope of inference of the grammar, because the probability of unseen words is indirectly boosted via abstract classes.

Learning the tree cut is treated as a separate learning problem, which precedes grammar training. The task is defined as a model selection problem where tree cuts are scored with the **Minimum Description Length** (**MDL**) information criterion [13, 19]. MDL selects the cut that has the best balance between complexity (number of classes) and fitness to data. In practice, WordNet subtrees that contain word senses of similar frequency in the data tend to be mapped into more abstract senses than those that contain outliers. For instance, we have observed that tree cuts learned from password lists tend to feature more often the sense "plant" than the sense "animal," suggesting that the frequency distribution under animal is more skewed.

The specificity of the tree cut can be tuned with a free parameter. In Table 2, we show the effect of specificity on the semantic tags assigned to words of various frequencies. Note how words with high frequency are more likely to be mapped to specific categories, often implying no abstraction at all (e.g., love $\rightarrow$ love.n.01).

3.3 Terminal Smoothing

In Weir et al.'s PCFG and in the Semantic PCFG the terminal probabilities are estimated by **maximum likelihood** (**ML**). In ML, the parameters of the model are chosen so as to maximize the probability of the data given the model; as a result, no probability mass is allocated to unseen strings. Theoretically, this would impact learning from small samples, as ML is known to be prone to overfitting. Since ML does not account for unseen vocabulary, the generalization carried out by the MDL tree cut may have had little impact on the semantic model's cracking performance [21]. In other words, the effectiveness of the semantic generalization depends on terminal probability smoothing.

Terminal smoothing requires two decisions: which vocabulary is to be added to the grammar and how to allocate probabilities to it. Since we use Wordnet for semantic classification, we opted to use its lemmas as our vocabulary (a lemma is an uninflected word, also known as a stem). Luckily, Wordnet covers a large set of words. We call this the *prior vocabulary*, composed of every Wordnet lemma inflected in all ways: nouns appear as singular and plural, and verbs appear under all conjugations. Depending on available resources, it would be possible to enrich this prior vocabulary with additional wordlists assigned to semantic categories (similar to how we handle names. The *posterior vocabulary* includes every terminal observed in the data, in addition to the prior terminals. The probability of a terminal string given a non-terminal symbol is

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d}, \tag{1}$$

where x_i is the observed frequency; N is the sum of the observed frequencies under the non-terminal symbol; α, known as pseudocount, can be interpreted as the number of times strings are assumed to be observed *a priori* (when $\alpha = 0$, it defaults to the ML); and d is the vocabulary size given the non-terminal. This estimator ($\hat{\theta}_i$) is known as additive smoothing or Laplace smoothing, and it is equivalent to the Bayesian estimator when a uniform prior is assumed [15]. Additive smoothing can be inaccurate when the vocabulary is very large, as with trigrams, in which case more sophisticated estimators are more appropriate, like the Good-Turing estimator used by Komanduri to train PCFGs [12]. With the size of our vocabulary in the order of hundreds of thousands, the Laplace estimator is adequate and has a simpler implementation.

4 MULTIFACTORIAL TESTS

The improvements in probabilistic password modeling appeared gradually and were not always tested in ways that allow understanding the individual effects of changes; as a result, the knowledge of how parameters affect performance and the interactions between such parameters is fragmented. In addition, training and test sets vary, as do measurement methods. Our goal is to examine the effect of various parameters on guessing performance in a full factorial experiment design. In this section, we test the effects of training sample size and probability estimators, which are applicable to all PCFGs, and the effect of linguistic detail and semantic level, which are specific to linguistic PCFGs. For perspective, we compare these results with the recent neural network model of Melicher et al. [16].

4.1 Method

For the tests in the following sections, we use as training set the RockYou password list, which contains 32,583,097 passwords and was stolen in 2009 from RockYou.com, a gaming website. It has been used extensively in password research [9, 11, 16, 21, 23]. We measure success with the Monte Carlo strength estimator; that is, we calculate the probability of passwords in the test set given each grammar, and then estimate guess numbers. The guess number is the number of guesses we would need to try before a password is guessed if we were enumerating all guesses of the grammar in highest probability order. Ordered guess enumeration is not the only way a PCFG can be used in an attack; alternatively, we could sample guesses randomly, but enumeration gives us the best shot at guessing more popular passwords early. Monte Carlo estimation allows us to measure success in sessions of infinite length without having to resort to extensive guess enumeration, which is computationally expensive.

We compare PCFGs whose most detailed symbols refer to Part-of-Speech tags (POS) with PCFGs that, beyond POS, include WordNet symbols (Semantic). These grammars are produced by exiting the text processing pipeline of Figure 1 at different stages. We have made the code for training and testing such grammars open source.[1]

As baselines, we include the PCFG models of Weir et al. [23] and Komanduri [12], and the neural network model of Melicher et al. [16]. For Weir's PCFG we use the implementation written by

[1]https://github.com/vialab/semantic-guesser.

Table 3. Factors for Grammar Training in Experiment I

Factor	Levels
Sample size	1,000
	10,000
	100,000
	1,000,000
	10,000,000
	32,583,973
Semantic (Specificity)	100
	1,000
	5,000
	10,000
Estimator	Smoothed
	MLE
Models	Part-of-Speech
	Semantics
	Weir et al.
	Komanduri
	Neural network

Dell'Amico and Filippone [7].[2] For Komanduri's and Melicher's models, we use the implementations found in the authors' own public repositories.[3,4] In Komanduri's model configuration, we enabled most features that are described in his thesis, namely, linguistic tokenization based on Google Web Corpus, the ability to produce unseen strings via brute-force, mixed-class non-terminals, uppercase nonterminals, and hybrid structures.

In Table 3, we list the factors manipulated in model training. We produced 10 random samples for each of the five sample size levels that are smaller than the size of the RockYou list (32,583,97), for a total of 51 training samples. Repeated sampling is important when measuring performance with small sample sizes, where variance may be high. The PCFG models, which are very scalable, were trained with all 51 samples, while the neural network model was trained with a single sample from each sample size. This is due to the time needed not only to train the neural network, but also to test it. We needed 140 hours to train one neural model with the full RockYou list in our setup with four GeForce GTX 1080 Ti, and 15 hours to test it against the LinkedIn list.

The configuration files used to train the neural models are provided as supplemental material. As in the original paper, we set the number of context characters to 10, and used a total of five layers: three LSTM layers (size 1,024) and two densely connected layers (size 512). We trained each model for 20 generations.

The specificity parameter controls the level of semantic generalization—larger values yield larger, more detailed grammars. Specificity applies only to grammars trained with semantic information. In total, we trained 612 grammars, of which 408 have semantic information (51 training samples $\times$ 4 specificity levels $\times$ 2 probability estimators), 102 have POS information (51 training

[2]https://github.com/matteodellamico/montecarlopwd.
[3]https://github.com/cupslab/guess-calculator-framework.
[4]https://github.com/cupslab/neural_network_cracking.

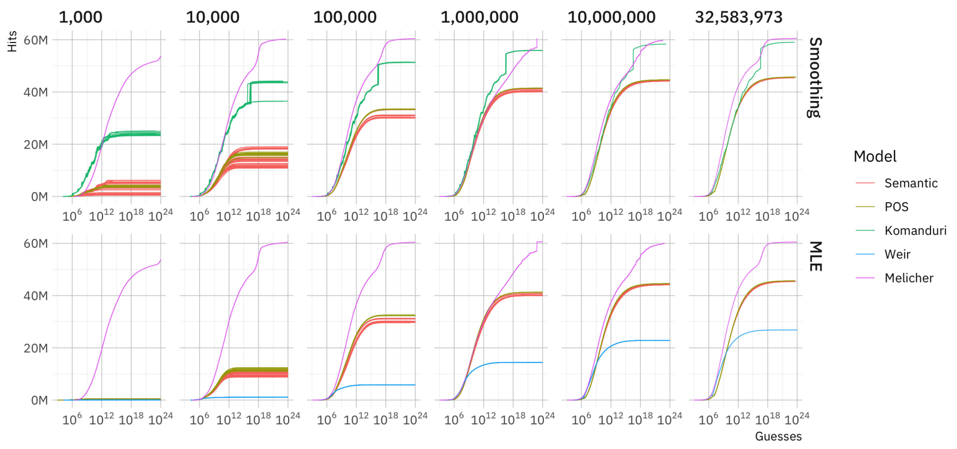

Fig. 2. Model performance in guessing tests, split by size of the training sample (columns) and probability estimator (rows). Weir uses MLE and Komanduri uses Good-Turing probability estimators, so they are only included in the corresponding row. The estimator factor is not applicable to the Melicher method (neural network), but we include it in both rows for comparison with the PCFGs. The models were trained on various randomly sampled subsets of the RockYou data and tested on the full LinkedIn data. For each data size, we produced 10 random samples.

samples × 2 probability estimators), and another 102 correspond to the PCFG baselines (51 training samples × 2 baselines). In addition, we trained six neural networks.

Due to legacy implementation constraints, the grammars based on Veras et al. [21] (POS and Semantic) can only output lowercase guesses. While this is a serious limitation for practical use, we expect it to not affect too much our analysis, since we are mainly interested in the relationship between guessing performance and training and testing parameters, instead of absolute cracking numbers. Nevertheless, we apply three "mangling rules" to each guess in order to produce capitalized, uppercase, and camel case variations (when applicable), in addition to lowercase. A match is counted only when a test password matches one of these four forms. Therefore, we adjust the guess numbers with a multiplication by 4. Note that this constant factor results in an *underestimation* of the performance of the POS and Semantic PCFGs, since not all guesses yield four variations; for instance, pure numeric patterns yield only one guess each.

4.2 LinkedIn

In this section, we test the models with the LinkedIn passwords, which were stolen in 2012, but discovered only in 2016. The original list has 164 million e-mail addresses and passwords, which were stored as unsalted SHA1 hashes. We obtained a copy of the cracked passwords from [1]. This copy has 60,593,032 unique clear text passwords, which constitute 98% of the unique hashes. Accordingly, the actual proportions of the full 164 million accounts cracked are much higher for each guesser tested, given that some passwords are typically much more popular than others. For example, the password *123456* was used by 1,135,936 accounts [3]. Although the password policy used by LinkedIn is only that passwords must be at least six characters in length, many of the passwords contained uppercase characters, numbers, symbols, and were longer in length [3].

4.2.1 Results. The semantic specificity parameter introduced much variation in the performance of the semantic grammars trained with small samples; however, the grammars quickly converged as the training size increased (Figure 2). This convergence is seen also between POS

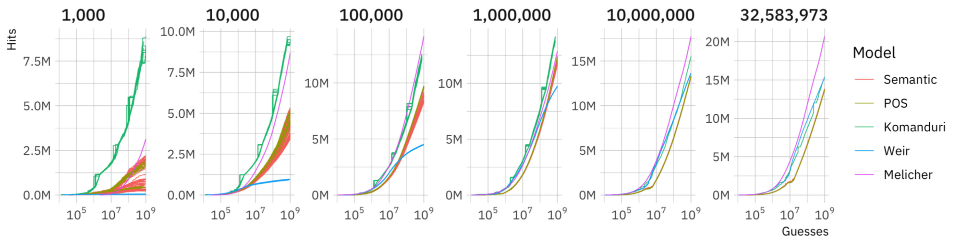

Fig. 3. Close-up of model performance under 10^9 guesses in the LinkedIn test, split by size of the training sample. The first 10^6 guesses give an indication of how well each guesser would perform in online attack scenarios.

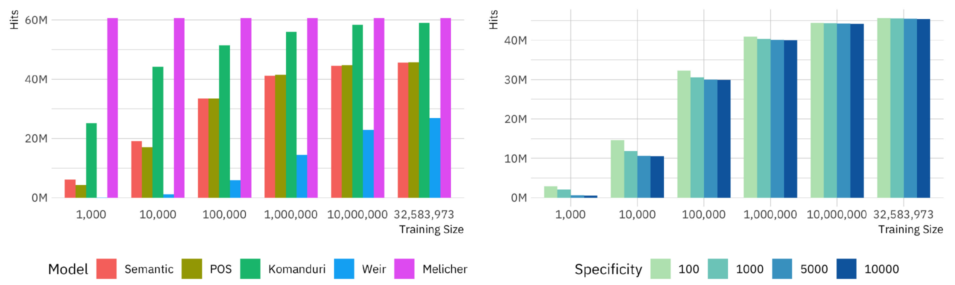

Fig. 4. Model coverage of the LinkedIn dataset. Coverage is the number of passwords that can be guessed regardless of how many attempts are needed. Left: Maximum hits by model. Right: Average hits for Semantic PCFGs by semantic specificity.

and semantic grammars; when trained with the full RockYou list (over 32 million passwords), we saw no significant differences between semantics and part-of-speech.

With smaller training samples, we observed mixed results that depend on the probability estimation approach. With MLE, the semantic grammars were generally worse than the POS grammars. But with probability smoothing, the inverse is true: POS grammars were outperformed by semantic grammars, but not all semantic grammars. All of these differences are only noticeable with sample sizes below 1,000,000.

Grammars trained with probability smoothing performed generally better with small training sizes—trained with only 1,000 passwords, the best non-smoothed semantic grammar can guess only 466,203 passwords, while the best smoothed equivalent outperforms it by a factor of 13 (6,161,283 passwords). When the training sample is larger, the number of unseen strings drops and smoothing stops making a difference. Likewise, the semantic specificity parameter is only relevant with small samples (Figure 4). This is likely due to the stability achieved by the Minimum Description Length framework with large samples: grammars trained with different specificity values tend to yield the same tree cut. Not surprisingly, under MLE estimation the effect of specificity is largely cancelled because there are few unseen words to benefit from the generalizations afforded by varying the specificity.

The size of the training sample strongly influences guessing performance. There are large increases in the number of hits for each increase in the order of magnitude of the training sample; however, beyond 1 million the benefits of increasing the size of the sample are relatively minuscule.

The Komanduri model performs much better than the other PCFGs until the training size is as large as 1 million. Past 1 million, this model performs very similarly to the smoothed semantic models until 10^{12} guesses, when the semantic models begin to plateau. Notably, it outperforms the neural method in early guesses (Figure 3) with a very small training sample (1,000). The guessing

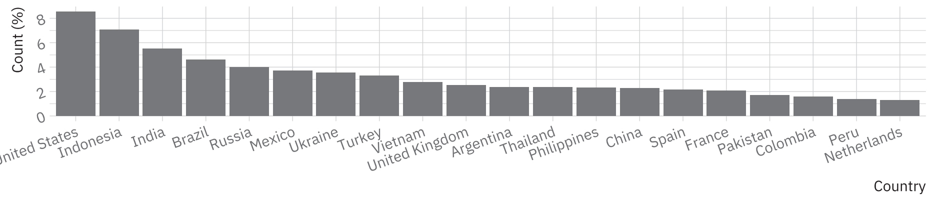

Fig. 5. The 20 most frequent IP origins in a sample of 5,000 user accounts from 000webhost.

curve of this model is more "jagged" than the others due to the probability quantization approach used, which groups non-terminals into probability bins.

The Melicher model (neural network) achieved outstanding results when trained with very small samples. The model trained with only 1,000 passwords was capable of guessing 20 million LinkedIn passwords with just 1 billion guesses, while the PCFGs (when trained with samples of the same size) fail to reach even the 10 million mark regardless of how many guesses are output, with the exception of the Komanduri PCFG. With larger training data, the difference between PCFGs and neural networks was much narrower, especially within the guessing range that is practical with guess enumeration ($<10^{12}$). Interestingly, the neural models trained with 1 million and 10 million passwords had worse performance than the model trained with only 100,000 passwords, which could indicate overfitting.

The lack of a difference between grammars trained with semantic information and grammars trained with only part-of-speech information is intriguing. We raise three hypotheses to explain this result: (a) the LinkedIn passwords lack strong English semantic dependencies; (b) LinkedIn features strong English semantic dependencies, but they are different from those found in RockYou; (c) English passwords in general lack strong semantic dependencies. Each of the aforementioned hypotheses is interesting on its own, so in the next sections we will present experiments that are intended to test them.

4.3 000webhost

The 000webhost list used for the tests contains 15,251,074 clear text passwords that became public around November, 2015. It was stolen from 000webhost.com, a large provider of web hosting services. This list is set 6 years apart from the RockYou list we use as training data, so it has potentially benefited from improved password composition guidelines and user education, relative to the state-of-the-art in 2009. In addition, the origin of the user accounts in this list seems to be diverse. Since the leak contains IP addresses for every user account, we extracted a sample of 5,000 accounts and verified the countries of origin with the service ipstack.com. The results show that the accounts are distributed across a large array of countries, with the largest one (United States) accounting for only 8% of the IP addresses (Figure 5). The distribution indicates that English passwords are not expected to be prevalent, making this list a challenging target for linguistic-based attacks trained on leaks from North American websites.

4.3.1 Results. The best performing grammar (Komanduri) guesses 13 million passwords (85.2%) after having output 10^{22} guesses. 6.8 million passwords are guessed before the 1 trillion mark (Figure 6). As in the LinkedIn experiment, the Komanduri PCFG largely outperforms the semantic models when trained with small samples, but shows very similar performance with larger samples. The differences between POS and Semantic are consistent with those observed in the LinkedIn test.

Unlike in the LinkedIn experiment, the neural network model falls behind the Semantic, POS, and Komanduri PCFGs, which indicates it may overfit more easily to the training sample. All PCFGs outperformed the neural network in the early guesses, regardless of the size of the training

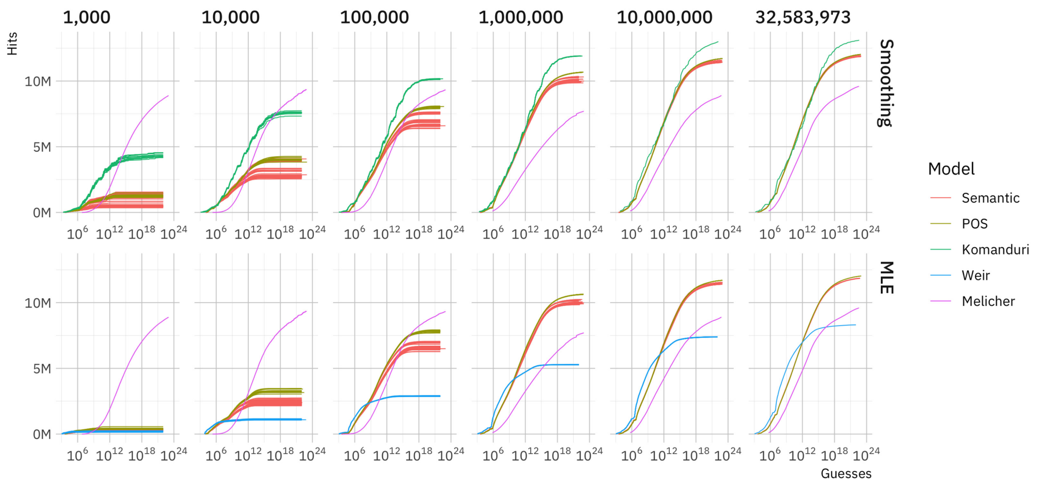

Fig. 6. Model performance in the 000webhost test, split by size of the training sample (columns) and probability estimator (rows). The grammars were trained on the full RockYou data and tested on the full 00webhost data.

sample. For most of the guessing range considered in Figure 6, the best Weir PCFG performs better than the best neural model.

5 NON-RANDOM PASSWORDS

The goal of this test is to measure the success of grammars trained on RockYou passwords when guessing the subset of *non-random* 000webhost passwords. PCFGs are based on the assumption that most passwords are non-random, composed of predictable combinations of words. They are not optimized to guess random passwords. PCFGs successfully guess random passwords by chance, not by design, given that random passwords mostly do not feature the regularities that are expected by grammars. Therefore, random passwords can be a confuser when analyzing the quality of grammars.

In order to isolate the non-random passwords, we trained a neural network to recognize random passwords. For the training data, the positive examples were generated with *pwgen*, a tool for random password generation. We created 27,200 examples equally distributed over a length range (4 to 20 characters) and optional parameter combinations (−no-numerals, −no-capitalize, −symbols). We created the same number of negative examples by randomly sampling from the subset of RockYou passwords with frequency greater than 1. The data was represented as a one-hot matrix with dimensions representing character bigrams (9,025 in total), and the network was set with three dense layers interleaved with two dropout layers, in addition to the input layer. We trained it with Keras on a TensorFlow backend over four epochs on 90% of the data. The model scored 99.35% prediction accuracy on our test set of 5,440 examples (10% of the data).

5.1 Results

Out of the 15,251,074 000webhost passwords, we classified 4,138,787 as random. The results of rerunning the 000webhost experiment on the subset of 11,112,287 passwords are presented in Figure 7. We observed no changes in the pattern of the previous tests. The difference in performance between POS and semantic grammars is still small when only non-random passwords are considered. In other words, we found no interaction between the factor random/non-random and grammar type (POS, Semantic).

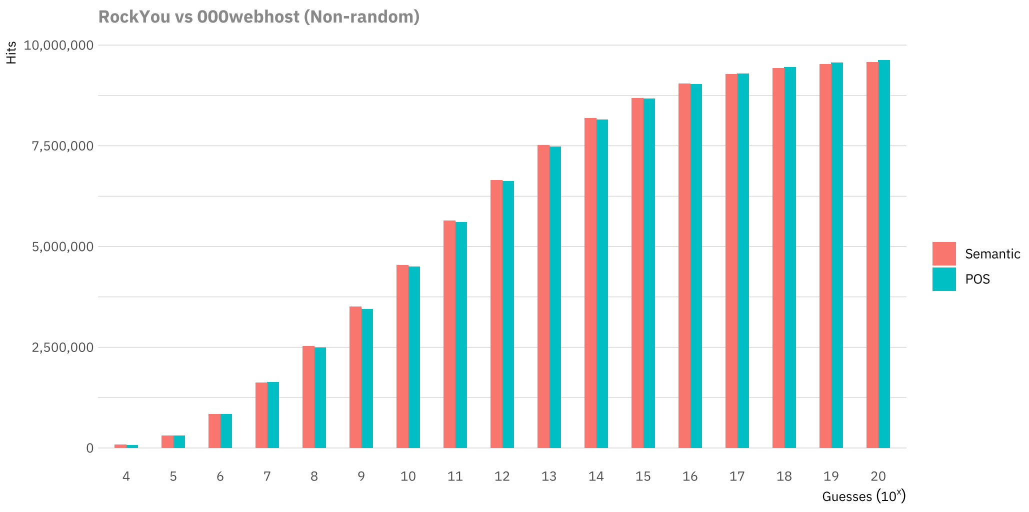

Fig. 7. Mean performance at increasing guess cut-offs for grammars trained on the RockYou list and tested on the *subset of non-random* 000webhost passwords.

6 CROSS-VALIDATION

We designed cross-validation experiments with RockYou and 000webhost to test the interaction between the target list and the level of linguistic information in the grammar. We would like to know if some password lists are more vulnerable to semantic guesses than others, and more importantly, whether semantic information has an impact on guessability. As we did not observe a benefit in adding semantics to grammars in the *Rockyou* ⇒ *LinkedIn* and *Rockyou* ⇒ *00webhost* experiments, it remains to be seen a scenario where semantics outperforms POS. In cross-validation the semantic alignment is ideal, as the model is trained with a sample that comes from the same population as the test set. This grants the model the best shot at capturing patterns that determine performance during testing.

We instrumented this test as a 10-fold cross-validation, with a training-test ratio of 1 to 9. We chose this partition ratio, with a small training set, after finding that the performance of grammars trained with different parameters tends to converge with large training sets (Figure 6). We randomly shuffled the lists of passwords and partitioned them in 10 parts. Each grammar was trained with one part and tested on the remaining passwords.

6.1 Results

For comparison, we selected the best parameter sets for each grammar type at increasing guess cutoffs. For instance, in sessions of 1 billion guesses, the best semantic grammars on average are trained with ML and specificity 1,000.

The results do not show large differences in the performance of POS and semantic grammars. We observed in both tests a minor difference in favor of semantic grammars in sessions up to 1 billion guesses long. The differences are smaller in the 000webhost experiment, suggesting that 000webhost may not contain strong English-language semantic dependencies (Figure 8). We state this hypothesis cautiously because it is possible that exploitable semantic patterns exist, but can not be expressed effectively with our PCFGs due to the their context-free nature. Although the difference is small in a relative sense, the numbers would make a difference in practice, especially in the RockYou experiment. For instance, by the guess number 10^7 semantic grammars have hit on average 370,000 more passwords than the POS grammar.

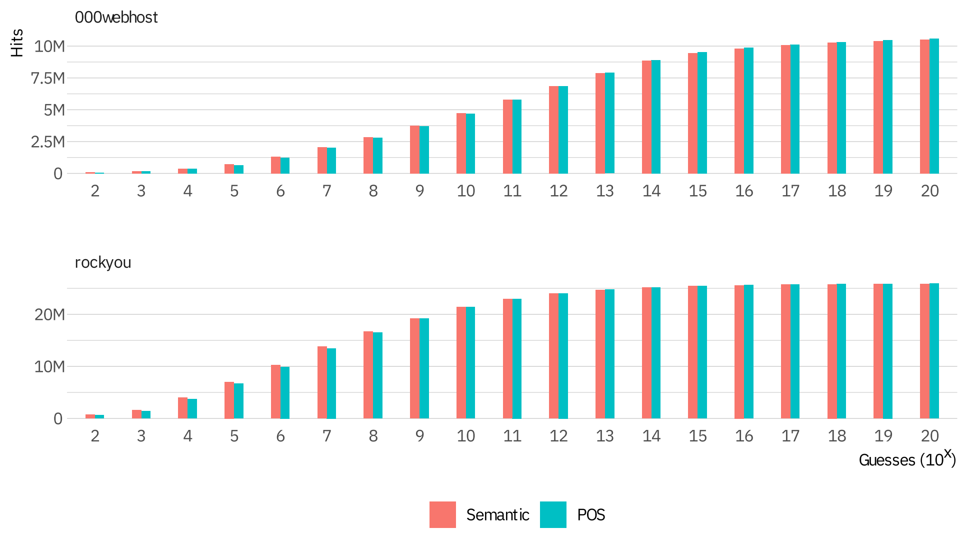

Fig. 8. Mean performance for increasing guess cutoffs. Grammars were trained on 1/10 slices of the password list and tested against the remaining 9/10 (Experiment 4). In each group, mean performance is calculated for the best parameter sets.

7 LINGUISTIC PATTERNS

In this section, we engage in Type-2 password research [14], where we fix the model and vary the data in order to understand the variations in composition and use of language across password lists. With the support of grammar models, we explore notions of linguistic similarity between sets of password lists. One of the defining virtues of PCFGs is interpretability, which allows us to conjecture about the reasons a set of passwords is more or less vulnerable. For system administrators, these models can help assess how policies are affecting the way people compose passwords, and identify potential unintended consequences. By measuring the similarity with publicly available leaks, one can estimate how vulnerable a password set is to offline guessing attacks that learn from leaks.

For this analysis, we omit the LinkedIn list, since, unlike most other lists, it does not contain repeated passwords, making it difficult to compare patterns. In addition to RockYou and 000webhost, we add the Mate1 and Comcast leaks to this analysis. The Mate1 leak was stolen from mate1.com, a dating website, around February, 2016, and contains 27,403,958 passwords that were stored in plain text. The leak also contains several columns of personal information, which include location, age, and gender. We downloaded a copy that includes only passwords, from databases.today [4], but the website leakedsource.ru [2] reported summary statistics on the personal variables that are relevant to this analysis: 84% of accounts come from English-speaking countries, 72% of the accounts belong to males, over 50% of men are between 28 and 43 years old, and over 71% of the women are within that range. The Comcast leak, stolen from US internet and cable provider Comcast, surfaced in November 2015 and contains 590,298 clear text passwords. Demographic information is not present in the Comcast leak.

We trained Semantic PCFGs on random samples of the four lists ($N = 500,000$). Figure 9 lists the most probable patterns (base structures) for each list, along with their probabilities. The 000webhost grammar has a distinct probability curve, much flatter compared to the other grammars, and it is clear that the website's password creation policy enforced the inclusion of numbers. Surprisingly, sequences of many digits in the end of passwords seem to be far more frequent than single digits.

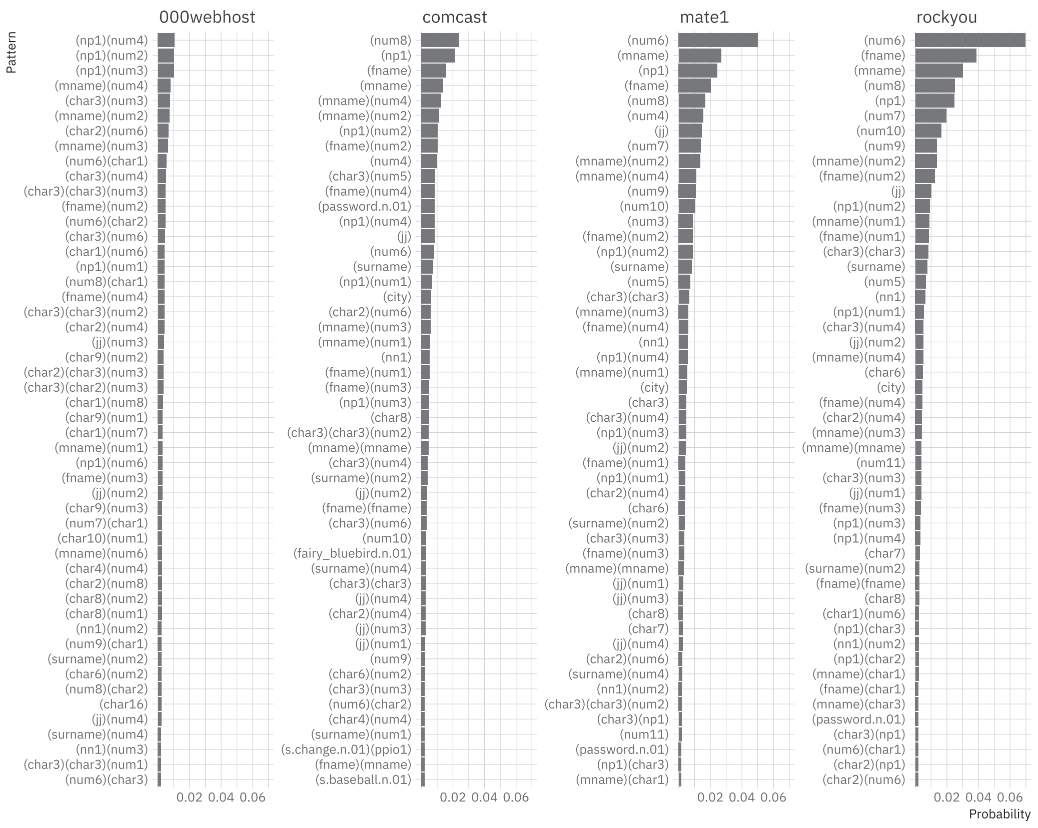

Fig. 9. Top grammar rules (base structures) per password list and their probabilities. Categories include CLAWS7 POS tags [8] such as *jj* for adjectives, special categories such as *mname* for male name, sequence classes such as *charN/numN* for an N-length group of characters or numbers, and semantic categories such as *s.baseball.n.01*.

The other leaks show that in the absence of a policy that enforces a mix of numbers and alphabetic characters users tend to choose pure numeric sequences of six to eight digits. The probabilities of male and female names seem to follow the demographics of the website: in both Mate1 and 000webhost male names are more probable; while we do not know 000webhost's demographics, it is probably safe to assume that the population of server administrators is predominantly male [5].

In Table 4, we list the most probable rules that contain a Wordnet semantic category. During training, if a string is recognized as an English word and it is not a proper noun, it will likely be tagged with a Wordnet sense. Mate1 and RockYou contain very similar semantic patterns. Love, which is not listed among the top Comcast rules and appears less frequently in 000webhost, appears in multiple rules in Mate1 and RockYou, which share also a preference for the word monkey, and mythical beings (dragon, werewolf, etc.), which do not appear in the other grammars. The most probable animals in the Comcast leak are bluebird, dog, bluefish, cat, and a variety of seafood fish. 000webhost does not feature animal words among the top categories. Flowers appear verbatim in Mate1 and RockYou, while appearing in the general form spermatophyte.n.01 in the Comcast list, suggesting that the probability mass is spread over many flower-related words.

The patterns in Mate1 corroborate the qualitative analysis of Wei et al. [22], which found that a large portion of the most popular passwords in several leaks included either the name of the web service where the passwords were stolen from, or words that are closely related to the services

Table 4. Most Probable Grammar Rules that Include a Wordnet Semantic Category

RockYou			**Comcast**			**Mate1**			**000webhost**		
rank	**pattern**	**p**	**rank**	**pattern**	**p**	**rank**	**pattern**	**p**	**rank**	**pattern**	**p**
46	password.n.01	1.9E-3	12	password.n.01	8.4E-3	48	password.n.01	1.6E-3	191	password.n.01 ● num1	4.5E-4
57	ppis1 ● s.love.v.01 ● ppy	1.6E-3	35	fairy_bluebird.n.01	3.0E-3	55	love.n.01	1.5E-3	208	num3 ● s.web.n.01 ● s.host.n.01	4.2E-4
68	ppis1 ● s.love.v.01 ● mname	1.2E-3	48	change.n.01 ● ppio1	2.1E-3	69	ppis1 ● s.love.v.01 ● ppy	1.2E-3	214	password.n.01 ● num3	4.2E-4
77	princess.n.01	1.1E-3	50	baseball.n.01	2.1E-3	93	jj ● s.male_child.n.01	8.5E-4	302	plant.n.02 ● num2	3.1E-4
97	worker.n.01	8.6E-4	54	sunlight.n.01	1.9E-3	98	structure.n.01	7.6E-4	316	love.n.01 ● num2	3.0E-4
98	woody_plant.n.01	8.4E-4	70	password.n.01 ● num1	1.4E-3	99	woody_plant.n.01	7.3E-4	323	char4 ● s.word.n.01 ● num3	2.9E-4
106	love.n.01 ● num2	7.6E-4	78	worker.n.01	1.2E-3	117	appge ● s.love.n.01	6.1E-4	371	password.n.01 ● num2	2.5E-4
113	rock.n.01 ● ppy	6.9E-4	83	football.n.01	1.1E-3	119	sexual_activity.n.01	5.9E-4	384	worker.n.01 ● num2	2.4E-4
117	herb.n.01	6.5E-4	90	jj ● s.dog.n.01	9.9E-4	121	love.n.01 ● ppio1	5.9E-4	456	char3 ● s.be.v.01 ● num3	2.1E-4
123	jj ● s.girl.n.01	6.2E-4	92	jj ● s.car.n.01	9.7E-4	125	jj ● s.man.n.01	5.8E-4	469	plant.n.02 ● num3	2.0E-4
127	angel.n.01	6.1E-4	95	herb.n.01	9.1E-4	129	inhabitant.n.01	5.6E-4	528	food.n.02 ● num2	1.8E-4
141	monkey.n.01	5.3E-4	117	fname ● s.dog.n.01	6.8E-4	134	cunt.n.02	5.3E-4	548	plant.n.02 ● num1	1.7E-4
143	password.n.01 ● num1	5.2E-4	120	cat.n.01 ● s.dog.n.01	6.5E-4	135	mate.n.01 ● num1	5.3E-4	565	plant.n.02 ● num4	1.7E-4
148	baby.n.01 ● s.girl.n.01	5.0E-4	121	bluefish.n.01	6.5E-4	137	herb.n.01	5.3E-4	578	worker.n.01 ● num3	1.6E-4
154	angel.n.01 ● num2	4.7E-4	122	worker.n.01 ● num2	6.5E-4	138	covering.n.02	5.1E-4	613	char4 ● at1 ● cell.n.01 ● num1	1.5E-4
159	friend.n.01	4.6E-4	127	fluid.n.01	6.2E-4	141	fluid.n.01	5.1E-4	626	maestro.n.01 ● num2	1.5E-4
161	produce.n.01	4.6E-4	129	spermatophyte.n.01	6.1E-4	143	beverage.n.01	5.1E-4	640	password.n.01 ● num4	1.5E-4
163	ppis1 ● s.love.v.01 ● char1	4.4E-4	132	beverage.n.01	6.0E-4	144	football.n.01	5.0E-4	643	trial.n.02 ● num3	1.5E-4
166	structure.n.01	4.3E-4	136	mname ● s.dog.n.01	5.9E-4	151	love.n.01 ● num2	4.8E-4	661	food.n.02 ● num1	1.4E-4
175	baby.n.01 ● num2	4.1E-4	150	worker.n.01 ● num1	5.2E-4	157	mother.n.01	4.7E-4	667	food.n.02 ● num3	1.4E-4
176	covering.n.02	4.0E-4	152	jj ● s.sky.n.01	5.1E-4	161	computer.n.01	4.6E-4	672	bad_person.n.01 ● num2	1.4E-4
187	cocoa.n.01	3.8E-4	153	ocean.n.01	5.1E-4	170	lover.n.01	4.4E-4	675	love.n.01 ● num4	1.4E-4
191	sunlight.n.01	3.7E-4	155	condition.n.01	5.0E-4	174	school.n.01	4.3E-4	690	hacker.n.01 ● num3	1.4E-4
197	flower.n.01	3.6E-4	158	produce.n.01	5.0E-4	178	love.n.01 ● num3	4.2E-4	698	be.v.01 ● char3 ● num3	1.3E-4
198	edible_fruit.n.01	3.6E-4	162	princess.n.01	4.9E-4	179	mc1 ● s.love.n.01	4.2E-4	703	worker.n.01 ● num4	1.3E-4
200	butterfly.n.01	3.5E-4	165	bad_person.n.01	4.7E-4	181	jj ● s.dog.n.01	4.1E-4	711	pass.v.01 ● num4	1.3E-4
203	jj ● s.male_child.n.01	3.5E-4	168	seafood.n.01	4.6E-4	185	entertainer.n.01	4.1E-4	725	pass.v.01 ● num3	1.3E-4
204	inhabitant.n.01	3.5E-4	176	inhabitant.n.01	4.4E-4	187	sleep_together.v.01 ● ppy	4.0E-4	732	worker.n.01 ● num1	1.3E-4
207	bubble.n.01	3.4E-4	178	summer.n.01	4.3E-4	192	agent.n.03	3.9E-4	751	killer.n.01 ● num2	1.2E-4
208	soccer.n.01	3.4E-4	180	quality.n.01	4.2E-4	198	commodity.n.01	3.8E-4	753	num2 ● char1 ● s.be.v.01 ● char2	1.2E-4
217	love.n.01 ● ppy	3.3E-4	193	herb.n.01 ● num2	4.0E-4	201	asshole.n.01	3.7E-4	758	maestro.n.01 ● num3	1.2E-4
219	ppis1 ● s.love.v.01 ● ppy ● num1	3.2E-4	197	chemical_element.n.01	3.9E-4	202	produce.n.01	3.7E-4	779	num3 ● s.love.n.01 ● num3	1.2E-4
220	lover.n.01	3.2E-4	198	trust.n.01 ● uh ● num1	3.9E-4	208	jj ● s.love.n.01	3.6E-4	867	hacker.n.01 ● num2	1.1E-4
227	love.n.01 ● ppio1	3.1E-4	199	bad_person.n.01 ● num2	3.9E-4	209	jj ● s.girl.n.01	3.6E-4	912	king.n.01 ● num4	1.0E-4
228	sleep_together.v.01 ● ppy	3.1E-4	200	contestant.n.01	3.9E-4	221	mythical_being.n.01	3.3E-4	965	killer.n.01 ● num3	9.7E-5
230	commodity.n.01	3.1E-4	208	edible_fruit.n.01	3.7E-4	223	father.n.01	3.3E-4	976	trial.n.02 ● num4	9.6E-5
234	football.n.01	3.0E-4	212	entertainer.n.01	3.6E-4	224	promise.n.01	3.3E-4	996	char3 ● s.be.v.01 ● num4	9.4E-5
236	worker.n.01 ● num2	3.0E-4	214	birdcage.n.01	3.6E-4	237	money.n.01	3.2E-4	999	windows.n.01 ● num1	9.3E-5
239	mname ● s.team.n.01 ● char1	3.0E-4	215	np1 ● s.dog.n.01	3.6E-4	238	monkey.n.01	3.2E-4	1011	num3 ● jj ● s.male_child.n.01	9.2E-5
240	ppis1 ● s.love.v.01 ● fname	3.0E-4	219	jj ● s.cat.n.01	3.6E-4	243	baseball.n.01	3.1E-4	1013	web.n.01 ● s.host.n.01 ● num3	9.2E-5
242	soccer.n.01 ● num2	3.0E-4	222	shrub.n.01	3.5E-4	246	chemical_element.n.01	3.1E-4	1022	love.n.01 ● num3	9.1E-5
243	love.n.01 ● num3	3.0E-4	224	contestant.n.01 ● num2	3.5E-4	247	love.n.01 ● num4	3.1E-4	1042	jj ● s.male_child.n.01 ● num2	9.0E-5
248	cookie.n.01	2.9E-4	229	mname ● s.male_child.n.01	3.4E-4	248	fellow.n.06	3.0E-4	1050	maestro.n.01 ● num4	8.9E-5
250	angel.n.01 ● num1	2.9E-4	230	computer.n.01	3.4E-4	249	flower.n.01	3.0E-4	1098	char3 ● s.be.v.01 ● num2	8.5E-5
251	princess.n.01 ● num2	2.8E-4	235	soccer.n.01 ● num2	3.3E-4	252	dragon.n.01	3.0E-4	1100	defender.n.01 ● num3	8.5E-5
252	agent.n.03	2.8E-4	236	agent.n.03	3.3E-4	254	cowboy.n.01	3.0E-4	1112	bad_person.n.01 ● num3	8.4E-5
257	worker.n.01 ● num1	2.8E-4	237	june.n.01 ● num4	3.3E-4	256	killer.n.01	2.9E-4	1114	food.n.02 ● num4	8.4E-5
259	mythical_being.n.01	2.7E-4	241	expert.n.01	3.2E-4	264	password.n.01 ● num1	2.9E-4	1119	nutriment.n.01 ● num2	8.4E-5
262	basketball.n.01	2.7E-4	245	jj ● s.day.n.01	3.2E-4	266	sleep_together.v.01	2.8E-4	1134	web.n.01 ● s.host.n.01 ● num4	8.2E-5
271	princess.n.01 ● num1	2.6E-4	246	summer.n.01 ● num2	3.2E-4	268	sleep_together.v.01 ● ppio1	2.8E-4	1137	mustang.n.01 ● num2	8.2E-5

theme. Mate1 passwords were found to have particularly high frequency of words related to love, dating, and the service name, "mate." Likewise, the 000webhost leak features many patterns containing the category *worker* and *hacker*, in addition to patterns containing the categories *defender* and *windows*, and the name of the service itself.

7.1 Graphical Model

We built high-level graphical models that describe the dependencies between the major classes of tokens found in the password leaks. These models capture coarse structural features, allowing at-a-glance comparison. Formally, we constructed a Markov random field (also known as pairwise

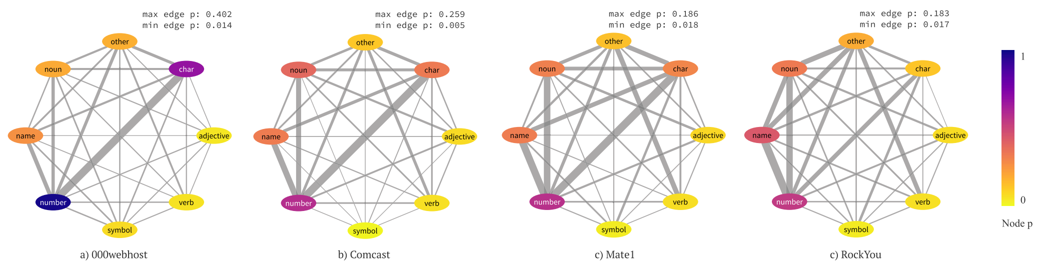

Fig. 10. Probabilistic graphical model capturing the dependencies between high-level grammatical classes. Node color encodes unconditional class probability and globally scaled. Edge width encodes the ratio between the probability of two classes co-occurring and the sum of the probability of each class. It is scaled per facet.

Markov network), which is an undirected probabilistic graphical model, and it is not to be confused with a Markov Chain [14], where the edges are directed and represent *conditional probabilities*. In our graph, edge width encodes the relative joint probability $\phi = P(A,B)/(P(A) + P(B))$, where A and B are token classes, and node color encodes the class probability $P(A)$. We define eight token classes: name (first names, surnames, other proper nouns), number, symbol, verb, noun, adjective, char (non-word character strings), and other.

The graph, shown in Figure 10, captures succinctly the dependencies between classes, and allows us to make a few statements about the high-level patterns in these lists. Symbols, verbs, and adjectives have consistently lower probability across all leaks. The 000webhost list contains a unique pattern: the probability of numbers is approximately 1, and the strength of the pair (number, char) is the highest across all leaks (p=0.4). The importance of character sequences in 000webhost is boosted by the high occurrence of non-English words, which are mostly parsed as character sequences. In the other lists, which are mostly English-speaking, these high-level dependencies seem fairly regular: there is a strong link between number and char (except in RockYou) perhaps due to the use of random passwords, but also strong dependencies between nouns/names and numbers.

7.2 Cross-Leak Experiment

A more pragmatic way to evaluate the similarity of password samples is by training a model with one sample and testing the model's effectiveness at guessing passwords from the other sample. We trained grammars on each password sample (000webhost, Comcast, Mate1, and RockYou) with a fixed set of training parameters: Laplace estimator, specificity 5,000, and semantic information.

Then we ran experiments with every combination of training and test set. The results are shown in Figure 11, grouped by test set. By this practical measure, the 000webhost list is the most distinct list: grammars trained on other leaks never achieve 75% success against it; grammars trained on it underperform against other leaks. Comcast, RockYou, and Mate1 perform almost identically against 000webhost, confirming the structural and linguistic similarity we observed in the previous section.

8 DISCUSSION

The experiments reported in this article were intended to quantify the influence of parameters of the Semantic PCFG (semantic specificity level, non-terminal symbols) and of PCFGs in general (training sample size, probability estimator). In this section, we discuss our main findings.

PCFGs quickly converge as the size of training data increases. We observed convergence of the guess counts at many levels: probability smoothing and MLE; POS and Semantics; semantic

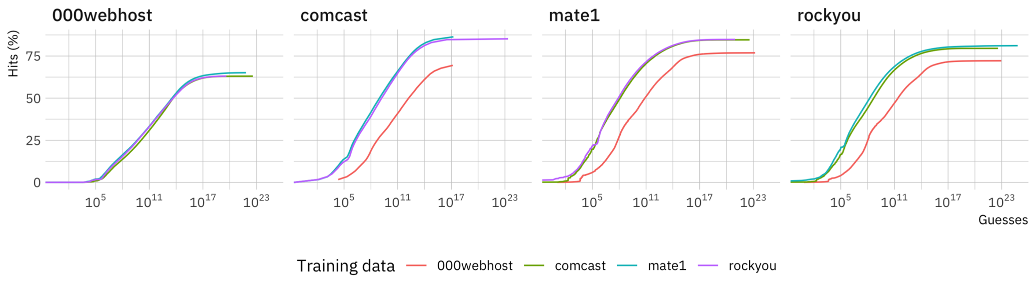

Fig. 11. Results of the cross-leak experiment. Each facet represents a target password list, and each line represents a grammar.

specificity values; and the Semantic and Komanduri methods. When comparing two training sizes, Melicher et al. [16] noticed that the larger (over 100 million passwords) and more heterogeneous training sample *reduced* the effectiveness of all tested models, including a PCFG, in a test against a single 000webhost sample. Here, when exploring multiple sample sizes, we found the influence of sample size to be positive following a logarithmic curve. After 1 million passwords the gains in effectiveness begin to disappear, and so do differences caused by different parameters.

Probability smoothing helps with small training data. We found that probability smoothing, coupled with vocabulary expansion, leads to large performance gains when the size of the training sample is small. This result corroborates Komanduri's [12] findings and expands on them by including the relation between smoothing and sample size.

Semantic information beats POS only with small training data. Within the range where guess enumeration is practical (up to 10^{12} guesses) and the training sample is small, it may pay off to use semantic information, as we found they are *at least* as effective as POS grammars in that range. When POS grammars were more effective in our tests, it was usually after 10^{12} guesses. One possible explanation is that the semantic grammar can readily generalize to produce unseen combinations. As the training set size increases, the advantage offered by semantic generalization diminishes, as more combinations are observed in the data and thus directly learned. Most of the time, however, grammars trained with semantic symbols (in addition to POS symbols) do not guess more passwords than grammars trained with POS symbols only. While our analysis shows that many semantic regularities exist in passwords, the vocabulary used in passwords is limited, which may explain the little to null gain achieved by grouping words based off of meaning.

The best model (PCFG or NN) depends on the target dataset. In the 000webhost tests, most PCFGs outperformed the neural model. This is surprising, given that Melicher et al. [16] showed superior guessing performance of the same neural model over the Komanduri PCFG in tests with 000webhost.

In our tests with the largest training sample, the neural model guessed around 65% of the 000webhost passwords. This differs from what Melicher et al. observed from a sample of 30,000 000webhost passwords (over 93% guess rate). Possible reasons for this difference include the testing of a sample from 000webhost vs. the entire dataset, differences in training data (we use only RockYou vs. RockYou + Yahoo), or differences in the configuration that could have caused overfitting of the neural network (we used two extra layers). We discuss in more detail the methodological differences in Section 9. Our results also show a much improved performance in the Komanduri figures; one possible reason for this disparity is if the linguistic (Google n-gram corpus) tokenization feature of the Komanduri PCFG was not used by Melicher et al. In their report, several features of the

Komanduri PCFG (e.g., terminal smoothing and hybrid structures) are mentioned, while linguistic tokenization is omitted.

The LinkedIn tests were more balanced—only the Komanduri PCFG rivaled the neural model performance. We observed a notable overlap between the Komanduri PCFG and the neural models when training with large samples; but the neural models were clearly superior with small training samples (<1,000,000). We believe that, despite the mixed results we observed, neural models show great promise considering they have been introduced only recently as password models.

Language information and performance. Despite having more sophisticated language processing, the Semantic and POS models do not guess more passwords than Komanduri's PCFG. We believe this can be attributed to many factors. The Komanduri model features a reserved *UNSEEN* symbol, which represents all unobserved strings within a class. During guess enumeration, this symbol generates *all* possible strings of the class, in a brute-force fashion; while in the Semantic/POS models they can only generate words. As a result, the vocabulary of the Semantic and POS models is more limited. Komanduri's mixed-class non-terminals allow the grammar to learn, in addition to password templates, literal passwords. This feature may be behind the Komanduri model's success in early guesses. Finally, unlike the Semantic and POS models, the Komanduri model learns letter case information.

Linguistics and statistics. Many improvements of Komanduri's model are statistical in nature, while the Semantic model focuses on adding more language information. However, the improvements of both models are not in conflict, and could work well together. Future work could investigate the addition of POS and semantic information to the Komanduri model, or the introduction of letter case information, mixed-class terminals, and more sophisticated handling of unseen strings to the Semantic model.

POS/Semantic grammars are less effective in short sessions. When LinkedIn is used as the target data, our results revealed that the PCFGs of Weir et al. and Komanduri tend to achieve better guessing results in the first attempts (at least up to 10^9 guesses). This may be due to the POS and semantic word groupings offering only so much flexibility. For instance, the POS group JJ, which represents adjectives, has some rather unlikely words, such as "quadrilateral." But because the probability of this group is boosted by extremely popular words like "hot," rare words may be attempted early. This *big wave lifts all boats* effect hurts the effectiveness of POS and semantic grammars in short sessions. Unfortunately, it is unclear how to break the group of adjectives into subclasses, as it does not have a hierarchical organization in WordNet. Future work could test a grammar representation where adjectives are not grouped; instead, they would be learned verbatim.

Semantic PCFGs are useful when interpretability is important. Our results show that the Komanduri PCFG is a superior PCFG model when the purpose is to estimate password probabilities, while the Semantic PCFG remains a good choice when interpretability is important. The latter offers a more detailed breakdown of a password's structure, which can enable a better understanding of the patterns in a password sample and more explainable password strength recommendations.

9 LIMITATIONS

In our password guessing experiments, the models make no assumptions about the target passwords. This means no restrictions on the length or composition of passwords that models are trained with or can output. It is common in other works to include constraints that follow the password policy of the target passwords; for instance, Melicher et al. [16] set their models to output only passwords longer than eight characters when testing against 000webhost.

The configuration used for the neural network in our experiments is slightly different than the configuration reported by Melicher et al. (see Supplemental Materials for the configuration files). Our networks have four LSTM layers of 1,024 units, instead of three LSTM layers of 1,000 units; and three dense layers instead of two. This might have contributed to the poorer performance we observed in our experiment if the increased model complexity caused the network to overfit to the RockYou passwords. Likewise, it is possible that the superior performance in the LinkedIn experiment is due to additional layers.

10 CONCLUSION

We presented a detailed empirical study of the factors that impact the performance of linguistic password models in offline guessing attacks. By evaluating guessing performance, we indirectly assessed the ability of models to learn general patterns. Our study focused on parameters of the semantic password model, a PCFG trained with part-of-speech and semantic information.

In our experiments, we found that grammars trained without probability smoothing tended to overfit when the training samples were small. Smoothing was found to grant great power to PCFGs trained with small samples: a grammar trained with only 1,000 RockYou passwords was able to guess almost 5 million 000webhost passwords. We observed diminishing returns for increasing the training sample size beyond 10 million passwords.

Veras et al. demonstrated that grammars with semantic and POS information largely outperformed Weir et al.'s grammar. Here, we isolated the effects of POS with and without semantics, and found that the benefit of adding semantic information to grammars was small compared to the gains resulting from adding POS information, and depended on the size of the training data. Moreover, we found that the Komanduri PCFG has better guessing power than the other PCFGs tested, and that there are cases when the neural model of Melicher et al. has worse guessing power than the PCFGs we tested.

We leveraged the explanatory power of the semantic model to examine recent password leaks qualitatively. Our analyses revealed that the RockYou and Mate1 leaks have great semantic and structural overlap, while 000webhost passwords have remarkably uniform structure but little semantic uniformity, at least as captured with an English-language grammar. The patterns we found expose semantic preferences that align with the demographics and themes of the services that leaked the passwords, and reveal the effect of password policies on the structure of chosen passwords. The similarities between password leaks mirrored the results of cross-leak guessing experiments and the high-level co-occurrence patterns encoded in graphical models of the grammars.

Service administrators looking to obtain reliable password strength estimates could run parameter sweep experiments, as we have presented here, with multiple training sets. Parameter ranges should be chosen to reflect the considered threats; for instance, small, in-sample training data matches a scenario where a subset of the passwords was exposed (either directly, or from users reusing passwords leaked from other data sets); while large, out-of-sample data matches a scenario where another leak is used to train the model. The resulting grammars can be used in reactive password checking programs, or as a password meter that makes use of the PCFG's probabilities.

ACKNOWLEDGMENTS

The authors would like to thank Brandon Laughlin for support with the experiments.

REFERENCES

[1] [n.d.]. Hashes.org—Shared Community Password Recovery. Retrieved September 28, 2019 from https://hashes.org.
[2] [n.d.]. LeakedSource Analysis of Mate1.com Hack. Retrieved May 1, 2018 from https://leakedsource.ru/blog/mate1.

[3] [n.d.]. LinkedIn Revisited—Full 2012 Hash Dump Analysis. Retrieved Septembet 28, 2019 from https://blog.korelogic.com/blog/2016/05/19/linkedin_passwords_2016.

[4] [n.d.]. Public Database Directory—Public DB Host. Retrieved May 1, 2018 from https://www.databases.today/.

[5] [n.d.]. StackOverflow—Developer Survey Results 2018. Retreived September28, 2018 from https://insights.stackoverflow.com/survey/2018#demographics.

[6] Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 13, 4 (Oct. 1999), 359–393. DOI: https://doi.org/10.1006/csla.1999.0128

[7] Matteo Dell'Amico and Maurizio Filippone. 2015. Monte Carlo strength evaluation: Fast and reliable password checking. In *Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, New York, 158–169. DOI: https://doi.org/10.1145/2810103.2813631

[8] Roger Garside. 1996. The robust tagging of unrestricted text: The BNC experience. In *Using Corpora for Language Research: Studies in the Honour of Geoffrey Leech*, J. Thomas and M. Short (Eds.). Longman Publishing Group, 167.

[9] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. 2019. PassGAN: A deep learning approach for password guessing. In *Applied Cryptography and Network Security*, Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung (Eds.). Springer International Publishing, Cham, 217–237.

[10] Shiva Houshmand, Sudhir Aggarwal, and Randy Flood. 2015. Next Gen PCFG password cracking.*IEEE Trans. Information Forensics and Security* 10, 8 (Aug. 2015), 1776–1791. DOI: https://doi.org/10.1109/tifs.2015.2428671

[11] Shouling Ji, Shukun Yang, Ting Wang, Changchang Liu, Wei-Han Lee, and Raheem Beyah. 2015. Pars: A uniform and open-source password analysis and research system. In *Proc. 31st Annual Computer Security Applications Conference* ACM, ACM Press, 321–330. DOI: https://doi.org/10.1145/2818000.2818018

[12] Saranga Komanduri. 2018. *Modeling the Adversary to Evaluate Password Strength With Limited Samples*. Ph.D. Dissertation. DOI: https://doi.org/10.1184/R1/6720701.v1

[13] Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Comput. Linguist.* 24, 2 (June 1998), 217–244.

[14] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. 2014. A study of probabilistic password models. In *Proc. IEEE Symposium on Security and Privacy*. IEEE, IEEE, 689–704. DOI: https://doi.org/10.1109/sp.2014.50

[15] Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

[16] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Fast, lean, and accurate: Modeling password guessability using neural networks. In *Proc. 25th USENIX Security Symposium*. USENIX Association, 175–191.

[17] George A Miller. 1995. WordNet: A lexical database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. DOI: https://doi.org/10.1145/219717.219748

[18] Peter Norvig. 2009. Natural language corpus data. In *Beautiful Data*, Toby Segaran and Jeff Hammerbacher (Eds.). O'Reilly Media, Chapter 14, 219–242.

[19] Jorma Rissanen. 1983. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics* 11, 2 (June 1983), 416–431. DOI: https://doi.org/10.1214/aos/1176346150

[20] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. 2015. Measuring real-world accuracies and biases in modeling password guessability. In *Proc. 24th USENIX Security Symposium*. USENIX Association, 463–481.

[21] Rafael Veras, Christopher Collins, and Julie Thorpe. 2014. On semantic patterns of passwords and their security impact. In *Proc. NDSS Symposium*. Internet Society. DOI: https://doi.org/10.14722/ndss.2014.23103

[22] Miranda Wei, Maximilian Golla, and Blase Ur. 2018. The password doesn't fall far: How service influences password choice. In *Proc. of Who Are You?! Adventures in Authentication Workshop (WAY)*.

[23] Matt Weir, Sudhir Aggarwal, Breno De Medeiros, and Bill Glodek. 2009. Password cracking using probabilistic context-free grammars. In *Proc. IEEE Symposium on Security and Privacy*. IEEE, IEEE, 391–405. DOI: https://doi.org/10.1109/sp.2009.8

[24] Zhixiong Zheng, Haibo Cheng, Zijian Zhang, Yiming Zhao, and Ping Wang. 2018. An alternative method for understanding user-chosen passwords. *Security and Communication Networks* 2018, Article ID 6160125 (2018), 1–12. DOI: https://doi.org/10.1155/2018/6160125

Received February 2019; revised January 2021; accepted January 2021