Grammatical Inference and Machine Learning Approaches to Post-Hoc LangSec

Sheridan S. Curley
U.S. Army Reserach Laboratory
Adelphi, MD
Email: sheridan.s.curley.ctr@mail.mil

Dr. Richard E. Harang
U.S. Army Research Laboratory
Adelphi, MD
Email: richard.e.harang.civ@mail.mil

***Abstract*—Formal Language Theory for Security (LangSec) applies the tools of theoretical computer science to the problem of protocol design and analysis. In practice, most results have focused on protocol design, showing that by restricting the complexity of protocols it is possible to design parsers with desirable and formally verifiable properties, such as correctness and equivalence. When we consider existing protocols, however, many of these were not subjected to formal analysis during their design, and many are not implemented in a manner consistent with their formal documentation. Determining a grammar for such protocols is the first step in analyzing them, which places this problem in the domain of grammatical inference, for which a deep theoretical literature exists. In particular, although it has been shown that the higher level categories of the Chomsky hierarchy cannot be generically learned, it is also known that certain subcategories of that hierarchy can be effectively learned. In this paper, we summarize some theoretical results for inferring well-known Chomsky grammars, with special attention to context-free grammars (CFGs) and their generated languages (CFLs). We then demonstrate that, despite negative learnability results in the theoretical regime, we can use long short-term memory (LSTM) networks, a type of recurrent neural network (RNN) architecture, to learn a grammar for URIs that appear in Apache HTTP access logs for a particular server with high accuracy. We discuss these results in the context of grammatical inference, and suggest avenues for further research into learnability of a subgroup of the context-free grammars.**

I. Introduction

The field of grammatical inference has been studied using numerous approaches since Chomsky first attempted to mathematically model spoken languages. Chomsky organized languages into four broad categories, based on their complexity, and gave a starting point in understanding the difficulties associated with learning any one of those categories. Each of the more complicated languages, by construction, contains the less complicated languages. In this paper we focus on the two lowest tiers of this categorization, context-free and regular languages, which by virtue of their relative simplicity contain many of the protocols in common use today.

One of the major findings of LangSec is that it should be a priority to secure computer systems by restricting the parsing complexity of input languages, limiting the ability of an attacker to inject malicious, unauthorized, and/or anomalous code into a system. Broadly speaking, this amounts to selecting languages as low in the Chomsky Hierarchy as possible, as simpler languages tend to have mathematically verifiable properties that more complex languages often lack, such as being able to prove the equivalence between two different parsers, or being able to verify that only a single parse exists for all valid messages.

However, while these are sound design principles, they do not offer a solution to the problem of existing languages that were not designed with these principles in mind. In many cases, there may not be a formal grammar for a particular protocol, or the version of the protocol that exists in implementations may differ from the formal specification. In either case, the actual grammar of the protocol must be learned before it can be analyzed.

While the theoretical results on learning formal languages within the context-free and regular classes tend to paint a grim picture (see Section II), various machine learning frameworks have nevertheless proven to be quite successful at what are (conjectured to be) formally very hard problems, such as natural language translation (see, e.g., [1]). This suggests that while in the most general case grammatical inference for LangSec may be prohibitively difficult, in particular cases it may be possible to infer the structure for specific instances of such grammars. With this in mind, we present learnability results using an LSTM network to learn the structure of URIs derived from HTTP access log files, and attempt to reconcile the unlearnability of the class of context-free grammars with the apparent learnability of this sub-class of URIs that should, in general, be context-free.

In the next section we discuss the history of the learnability results for regular and context-free Chomsky languages, and offer some possible explanations for learnability of the classes of language discussed later. Sections IV and III then present the results of applying neural network learning to HTTP log files. Section V offers interpretations of these results in the context of Section II, as well as further avenues of interest.

II. Theory

The theoretical framework began with Gold's 1967 paper, which looked at the idea of learning the types of languages in the Chomsky hierarchy[2], [3]. Gold's work was followed by a series of papers from Angluin, which refined the definitions on what types of languages could not be learned, while also giving examples of languages that could be learned using Gold's "learning in the limit"[4], [5], [6]. More recently, researchers

have produced examples of learnable languages that are part of the classes that Gold said were not broadly learnable[7], [8], [9]. We briefly touch upon these developments, and present how those results can inform our understanding of the learnability results presented in the sections below.

A. Background

Before summarizing theses various results, we first define some useful terms. We will be using the term grammar throughout, where a grammar is a tuple $G = (N, \Sigma, R, q_0)$, where N is the set of nonterminal symbols, Σ is the alphabet being considered, R is a set of production rules over Σ^*, and $q_0 \in N$ is the starting symbol. A *positive presentation* of language L is an infinite sequence of all, and only, those strings in L, repetitions allowed. That is, given an alphabet Σ and the set Σ^*, a positive presentation of L is $\{s : s \in \Sigma^* \cap L\}$. A *negative presentation* is the set of strings in Σ^* and not in L, that is $\{s : s \in \Sigma^* \setminus L\}$. A *complete presentation* is the set of ordered pairs $\langle s, n\rangle$ where $\forall s \in \Sigma^*, n \in \{0, 1\}, \exists\langle s, n\rangle$ where $n = 1$ iff corresponding $s \in L$.

It is also important to note that there is a difference between learnability of a *class* of languages, and the learnability of a single instance of a language. Learnability of a class is a much stronger property, that permits no restrictions on any candidate grammar beyond those on the class as a whole, so things such as structural properties that may exist in certain subsets of a class are not taken into consideration. Additionally, learning a class would require that a learner be able to generate a recognizer for *all* languages of that class. As will be shown, this is often not possible. Learning a single language in a class, however, may be possible in specific cases, and subsets of classes may also be learnable if multiple languages share relevant structure or syntax, and assumptions can be made that group them.

Note that most grammatical inference strategies involve beginning with a hypothesis space of grammars that contains the entire class of languages under consideration, and narrowing that hypothesis space down to the correct grammar(s) describing the desired language(s). Proofs involving such algorithms typically revolve around the presence or absence of uniquely identifying productions for each language (or subclass of languages) within the class, thus allowing the elimination of the subclass of all languages within the grammar that do not share that telltale. This is, in a loose sense, "backwards" from the more intuitive method of learning a language: constructively building the language (and its descriptive grammar) by constructing sequences of states and transitions as they are observed.

B. Learning in the Limit

Gold's paper presents a formal definition of *learnability in the limit*, which can be summarized as: For a class of recursively enumerable languages $\mathcal{L}$, algorithm A *identifies* $\mathcal{L}$ *in the limit* from positive presentations if $\forall L \in \mathcal{L}$, A produces, in a finite number of steps from a given positive presentation, a hypothesis h that correctly describes L. More explicitly, A identifies $\mathcal{L}$ in the limit iff there is a point at which $h_n = h_{n+1} = ...$ and h_n is a correct description of L where $\forall L \in \mathcal{L}$ and positive presentation $I = i_1, i_2, ..$ of L, when A receives i_n it produces $h_n = H(i_1, i_2, ..., i_n)$. A *effectively* identifies $\mathcal{L}$ in the limit if H is effectively computable. A correct description of L is also called a grammar that generates L.

Gold showed that even the simplest Chomsky languages, those generated by regular grammars, are not identifiable in the limit from positive presentation only. This means that context-free, and anything more complicated still, fail to be identifiable in the limit under Gold's assumptions. Further, Gold showed that any super-finite class of languages – defined as a language containing all finite and at least one infinite language – cannot be identified in the limit. The only language considered by Gold that *was* identifiable by positive presentation was finite cardinality languages, i.e. languages that are trivially regular, which can be learned through exhaustive enumeration. While this strategy is theoretically feasible, it breaks down for any language of nontrivial length, and more importantly does not generate any information about any potential simplifying structure inherent in the language.

C. Approximate Fingerprints

Angluin's series of papers built upon the work of Gold, and brought out new details. On the negative side, Angluin showed that even if a learner was given access to an equivalence oracle and a positive presentation, it was still not possible to generically learn Chomsky languages in the limit. An equivalence oracle would answer queries such as "is language L equivalent to language L' " or "is grammar G equivalent to grammar G' ". Since it has also been shown that showing equivalence between languages generated by two context-free grammars is undecidable[10], the fact that even having such an oracle does not rescue the learnability of even regular languages is not good. Angluin also showed that the classes of nondeterministic finite state acceptors, context free grammars, and disjunctive normal form and conjunctive normal form formulas would not be exactly learnable from equivalence queries due to having the "approximate fingerprints property"[11].

Descriptively, the approximate fingerprints property can be considered using an adversarial approach to answering equivalence queries. When testing a hypothesis h, the learner presents h to the adversary oracle. When answering *no*, indicating h is not the target hypothesis, the adversary presents a counterexample w_h which eliminates h and some fraction of the target class. If that class has the approximate fingerprints property, the adversary can provide a w_h that eliminates a superpolynomially small fraction of the target class. The adversary can do this a superpolynomial number of times without running out of hypotheses in the target class, and the learner is prevented from exactly learning the class in polynomial time. Although it is possible to construct a fabricated situation wherein the learner "gets lucky" despite the adversary's attempts, this property formally prevents any class that exhibits it from being identified exactly.

D. Positive Learning Results

On the positive side, although none of the Chomsky classes were learnable in the limit in Gold's framework, Gold did show that any language that was primitive recursive, or less complicated, could be learned by informant. Here, primitive recursive is used in the computability sense, and can be expressed as a programming language that allows arithmetic operators, conditionals and comparisons, and bounded loops. Due to these requirements, all primitive recursive functions halt, as opposed to partially recursive, or Turing-complete, programs, which suffer from the halting problem. For Gold, an informant would provide both positive and negative examples, so in the limit would effectively produce a complete presentation. Therefore, both regular and context-free languages are learnable, even in Gold's assumptions, if the learner can be given both types of examples. Of note is that Gold's statements were made assuming learnability in the limit, not a less-stringent learning criteria such as learning with probability one.

Angluin brought forth the idea of a "tell-tale" set of strings for a language[12]. Angluin showed that a language was identifiable from positive presentation if there was a finite set of strings T, where $\forall i \in \mathbb{N}^+$, $\exists T \subseteq L_i \; s.t. \; \forall j, T \subseteq L_j \Rightarrow L_j \not\subseteq L_i$. The set T is the tell-tale set, and Angluin showed that once such a set appeared, the learner is free to guess L_i without getting stuck in a proper superset of the language; i.e., the learner would not overgeneralize. Overgeneralization is the prime reason that CFLs cannot be learned in the limit, so languages that have such a tell-tale set should be learnable. Though it is often not possible for a learner to know when it has been given the tell-tale set.

Two other key points that came from Angluin's papers that were later refined are the ideas of finite thickness and finite elasticity of a language class. Finite thickness was first defined by Angluin as: for language class $\mathcal{L}$ and for each non-empty finite set $S \subseteq \Sigma^*$, the set $C(S) = \{L : S \subseteq L$ and $L \in \mathcal{L}\}$ is of finite cardinality for class of languages $\mathcal{L}$. More plainly, this states that every string is contained in at most finitely many languages in class $\mathcal{L}$. Infinite thickness, where a string is contained in infinitely many languages, results in a class that cannot be narrowed down, as learning that string never provides information on which language it came from. Angluin showed that if a class of recursive languages has finite thickness, then it is learnable in the limit. Motoki and Shinohara state a language has finite elasticity if it does not have infinite elasticity, where formally: A language class $\mathcal{L}$ has infinite elasticity if there exists a sequence of strings $\{w_0, w_1, w_2, ...\}$ and of languages $\{L_1, L_2, L_3\} \in \mathcal{L} \; s.t. \; \forall n \geq 1, \{w_0, w_1, ..., w_{n-1}\} \subseteq L_n$ but $w_n \notin L_n$ [13]. They also showed that if a class of recursively enumerable languages has finite elasticity, then it is learnable in the limit.

Angluin also introduced the idea of the Minimally Adequate Teacher (MAT)[5]. Such a teacher was assumed to answer two types of questions, one being the membership query and the other being an equivalence query. Membership is asking if a string is or is not a member of the language to be learned. Equivalence is asking if a hypothesized language is equivalent to the language to be learned, with a counterexample provided if they are not. Although regular languages were shown to be learnable using a MAT, the equivalence question is not answerable for CFGs, and so only approximations, or extensions, of MAT can be used for CFGs. Some recent examples include work done by Clark and Eyraud[7], [14].

E. Probabilistic Learning

It is clear that in general, exact identification in the limit is difficult because it is impossible to know when the learner has converged to the correct language. Even if the learner can predict effectively all of the examples presented, it is still possible a counterexample may appear. However, this means it may be possible to still learn Chomsky languages to some effective-enough level for practical purposes. An example of this first appears in work by Valiant, on what is now called Probably Approximately Correct (PAC) learning. The general idea is summarized as, for a class $\mathcal{G}$ of grammars, $\mathcal{G}$ is PAC-learnable if an algorithm A exists that generates with high probability, $\forall G \in \mathcal{G}$, a hypothesis grammar H that is "close to" G. A more rigorous definition is given originally in Valiant's paper, as well as many modern textbooks on grammatical inference, such at de la Higuera's[15], [16]. Although PAC-learnability is in some sense less stringent than exact identification, Valiant's original formulation called for using all possible distributions $\mathcal{D}$ over Σ^*, which is prohibitively difficult to use in practice. Indeed, later results of Kearns and Valiant [17] show that PAC learning of deterministic finite automata (and hence regular grammars) is cryptographically hard, in that an algorithm capable of PAC learning of DFAs can be used to break the Diffie-Hellman (DH) assumption. The DH assumption here being that the DH problem (DHP) is assumed to be hard to solve computationally. The DHP is roughly as hard to solve as the discrete log problem[18], which can be written: "given element $a \in \mathcal{A}$ and $h = a^x$, what is the value of x?". Other examples of variants of the DHP exist, such as [19].

F. Final Remarks

Although these results provide some ways to seemingly get around the negative learning results, they represent specific methods or properties that are difficult to prove for particular CFLs. Thus, although some subset of CFLs may very well be learnable, and such a result may be shown elsewhere, it is not obvious *a priori* for any given CFL. This is a problem, since learning from positive presentation is the style of learning most often used in practice, given it is easy to generate examples of "good" strings, but difficult to enumerate all possible "bad" strings (which would be necessary to produce a complete presentation for use in Gold's framework). Despite this, some language models have been shown to be learnable, such as Angluin's pattern languages[12], grammars modeled by elementary formal systems[20], and some subset of protocols[21]. There remains no single algorithm that can (provably) learn an arbitrary member of the class of CFL languages, however.

An interesting question, then, is whether there exists general purpose algorithms that can learn particular members of the class of CFLs, at least in a probabilistic sense, and if so, to what degree that learnable class overlaps with protocols and grammars that are actually deployed today. As an initial step towards exploring this question, we used an LSTM network – designed for very general sequence to sequence problems – to attempt to learn the structure of URIs from HTTP access log files. We found good learnability results (see Sections IV & V), despite URIs being context-free in general. This strongly suggests that the subset of the grammar of URIs, as deployed in a single web server, does not in fact exploit the full computational power of the class of CFLs; this in turn suggests that despite grammatical inference in such classes being in general a hard problem, the use of such machine-learning models for filtration of requests or detection of potentially malicious requests may in fact be viable.

III. Experiment Background

Here, we examine a practical application of grammatical inference by using machine learning to examine the set of "good" and "bad" strings represented in requested URIs collected from an Apache HTTP access log. Although we do not directly infer a grammar in this paper, we will discuss in Section V possibilities for extracting grammars from the LSTM learned here. First, we outline the data we used, give a brief discussion of the learner used, and then discuss these machine learning results in the context of LangSec applications.

Our first learning task on the data treated it as unlabeled and so we are limited to a positive presentation, which means that it will not be identifiable in the limit (per Gold's result). Note also that we can show that this class does not have finite thickness (trivially, the request "GET / HTTP/1.1", with associated URI "/" is valid for all possible 'languages' of URIs), and ignoring practical concerns about finiteness of strings, it can be shown that the space of URIs has infinite elasticity as well: let w_n be any path element of length n, and define L_n to be any member of the class C of valid URI sets that has the first path segment of length less than n; then it is obvious that the set $\{w_0, w_1, \ldots w_{n-1}\} \subseteq L_n$, but $w_n \notin L_n$. Both results imply that the class of URIs is not learnable in the limit. Finally, as any "language" of URIs is at least context-free, and thus contains regular grammars, then the result of Kearns and Valiant implies that PAC-learnability of the space of URIs is at least cryptographically hard.

Despite these theoretical issues, we examine URIs in both a generative (unlabeled) model, in which we train the model to produce strings from that set, and a discriminative (read: recognition) model, in which we train it to predict response codes. We show below that although an LSTM is not a grammar in the traditional sense, it can seemingly recognize the language of URIs we use as data. The learner is not perfectly accurate, however this should not be expected due to possible ambiguities in URI structure (see Section III-A). However, in a probabilistic sense reminiscent of PAC learnability, we are able to separately identify valid and invalid requests to an acceptable level of accuracy. Further, the discriminative model is capable of results approaching the Bayes Error Rate (BER). For the purposes of multi-label identification, the BER is the minimum error due to overlapping in any or all of the labels. As a basic example: consider a string s, coming from one of 2 classes that are represented by normal distributions with some overlap; the BER is the rate at which s could be chosen from that region of overlap. Due to this probabilistic method of generating the string, it is never possible to be 100% sure of correct identification.

A. Description of the Data

Consider the format of HTTP; as outlined in RFC 3986 (on general URI syntax) and RFC 7230 (on HTTP 1.1 message syntax), there is nothing that restricts HTTP URIs from being generically context-free. There are, however, some specific formatting considerations that may allow them to be generically learnable, and these same traits may apply to other URIs more generally. For instance, a generic http URI looks something like:

http:// $authority\ path-abempty$ [*?* $query$] [*#* $fragment$]

where $authority$, $path-abempty$, $query$, and $fragment$ are variable names for strings that meet their respective RFC definitions[22]. Immediately it can be seen that any such URI will have the structure of starting with *http://*, and for any URI containing a query or fragment piece, the special characters *?* and *#* will appear as well. The variables $authority$ and $path-abempty$ have some structure, such as following either DNS or IPv4/IPv6 labeling ($authority$), or to follow certain formatting guidelines ($path{-}abempty$). The remaining variables have no such restrictions, and none of the variables have explicit length requirements. Example strings, especially those with both $query$ and $fragment$ pieces can therefore be considered context-free.

There are further restrictions that apply to real-world HTTP URIs that are not necessarily required by the standards defining them. For example, it is unlikely that an authority would be a completely random string of characters with extremely long length. It is much more likely to be a simple DNS identifier or an IP literal, the latter of which is explicitly finite with determinable structure. A path is likewise unlikely to be excessively lengthy with no further structure, as it is most commonly going to be based on the system's underlying file structure, which will have been created by a person in order to be usable in some day-to-day sense. This will likely limit the length of paths that are seen, and will also introduce structure such as short directory names separated by slashes. In fact, the RFC identifies only 5 augmented Backus-Naur form rules necessary to parse any path, given the protocol standards[23]. Unfortunately, there is no immediately obvious additional simplifications that can be made to the query or fragment pieces. It is, however, observed that these are not in practice excessively long either.

Of note from these descriptions is the fact that there are no requirements for a URI to contain unique file structure if

multiple such structures are accessible through a single access server. This means that for heavy use servers, there is inherent ambiguity when examining all incoming or outgoing URIs. There is no reason two different systems cannot share a root file structure and thus produce a string that can adequately, and ambiguously, refer to two different servers as far as a learner is concerned. The reason for this is that the distinguishing features that allow the directing server to properly differentiate these "ambiguous" requests are not inherently known to the learner, and are unlikely to be learned without creating a highly specific (and therefore restrictive) learner only for that server. Thus our data cannot allow perfect identification using this method of learning, but it is robust enough to be useful on multiple servers.

B. The model

We employ a fairly standard recurrent neural network (RNN) in which characters are consumed one at a time, updating a hidden state which also receives updates from the previous value of the hidden state. The hidden state then feeds forward to either another hidden state, or an output layer. As an update function for the hidden state, we use the long short-term memory (LSTM) function[24] which has been shown to effectively learn long-range dependencies in a variety of tasks[25].

We use the Chainer[26] framework, and use built-in functions unless specifically indicated otherwise. Our model structure is fairly standard, and consists of an embedding layer which embeds one of 102 one-hot encoded characters (100 printable characters plus a start and stop character) into a 50-dimensional space. We then have three LSTM layers each with dimension 512. Finally we have a softmax classification layer with either 102 outputs (for the generative model) or a smaller number outputs (either 2 or 13, depending on the experiment, for the discriminative model). An example illustration is given in Figure 1. We do not employ any skip-level connections.

Training was performed using the Adam[27] optimizer. We employ dropout[28] with a rate of 0.5 during training. We experimented with truncated backpropagation as well as gradient clipping and found (likely due to the relatively short sequences being modeled) that neither was necessary, and in fact slightly hampered training.

For the generative task, the network was asked to predict each successive character on the basis of all previous ones (frequently referred to as the "char-rnn" model) and so we accumulated a cross-entropy loss at each character. For the discriminative task, we explored two variants: first asking the model to produce a class guess at each timestep and accumulating loss as in the generative task, and one in which loss was accumulated only at the final timestep. We found that the former produced significantly better results and so restrict our discussion to that model. We monitored the performance of the discriminative model by classifying each new sample before using it for training, and tracking the cumulative error rate as well as the error rate of the most recent 10,000 samples.

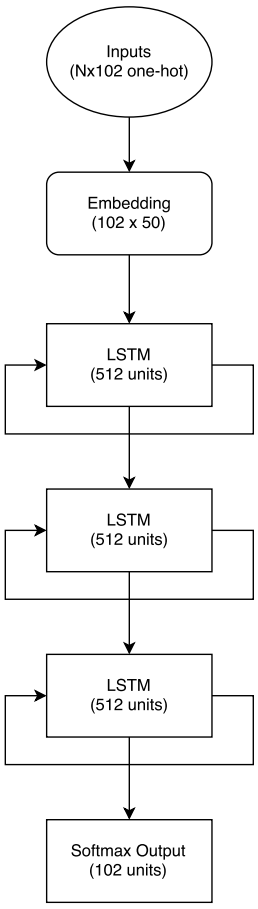

Fig. 1. Diagram sketching the network's structure.

IV. Experimental results

Here we present the most relevant results from the learning tasks discussed above. As a brief summary: we find that the LSTM is capable of identifying novel, ungrouped URIs with a success rate of roughly 95.4%, versus a Bayes Error Rate of 99.7%, and novel, grouped URIs at a rate of 99.4%, against a BER of 99.9%. This represents promising learnability results on an inherently probabilistic grammar.

A. Data

We use a collection of Apache HTTP access logs collected over the course of a single week in June 2013, covering a single server offering a small collection of J2EE services. This data included both normal user behavior as well as several instances of brute-force webscanner activity, and some rare instances of more sophisticated (though still apparently automated) service profiling. Logs were rotated every 12 hours, and so we selected all but the final 12 hours as a training sample, with those final 12 used as a testing sample. From this testing sample we considered two scenarios: first we examined the entire access log as-is. Because URIs frequently repeat, often exactly, we also examined the subset of URIs from the test data that had not appeared in the training data. This tested the ability of the classifier to generalize. A total of

830,238 entries are available from the testing sample, of which 100,546 were novel, while there are 6,398,612 total entries in the training sample.

B. Results

For the generative model, we look at the negative binary log-likelihood (NBLL) which has a simple interpretation as the average number of bits required to encode the string. A naive encoding of 102 characters results in 6.6724 bits per character, assuming each character is seen uniformly at random independent of all others. Taking the distribution of characters into account (but still assuming independence) results in 3.7611 bits per character. Our model over a test set obtains an average NBLL of 0.564 bits per character, indicating a high degree of structural dependency that can be successfully learned. It should be noted that this is accomplished in the presence of a significant number of URI requests that contain such features as session ID keys, which are essentially random (on a reduced alphabet) and so have an intrinsically higher NBLL (NBLL values for these segments tended to be higher, often 3 to 4 BPC, consistent with base 64 encoding with some predictability).

To validate the production capability of the generative model, we generated samples using both character-by-character sampling, as well as by beam searches from prefixes in the data set. Character-by-character sampling produced generally representative samples, although most were "memorized" strings, and many became incoherent at the end of the generated string (this is a known problem when the random sampling process selects a low-probability character by chance, see [29] for further discussion). Beam sampling generally avoided this issue, and by setting the beam width sufficiently high and returning all results in the final beam we could force the model to generate high probability samples that it had never seen. Several interesting features were observed, such as the ability to generate novel base64 encoded strings with correct padding that could be decoded (albeit to gibberish); the model correctly using the substitution of "%2F" for "/" and "%3A" for ":" characters in novel contexts in URL redirections; and the model "mixing and matching" key-value pairs across different web application endpoints in a reasonably plausible manner.

For the discriminative model, we compare the classification performance of the model to the Bayes Accuracy, in which each instance is assigned its most probable label based on probabilities derived from the ground truth labels. As the same URI may, depending on the remainder of the request and the state of the server, return a status code 200 ("OK"), a status code 302 ("Found"), or a status code 500 ("Internal server error"), a classifier which has access to only the URI has no basis on which to distinguish them, and hence should return the label of the most probable class. To attempt to normalize this somewhat, we grouped the URIs into three categories: "Handled," including response codes 200 (Found), 206 (Partial content), 302 (Temporary redirect), and so on; "Bad request," including response codes in the 400 range (Bad request, Unauthorized, Forbidden, Not found, etc) and 501 (Not implemented); and "Server error," including the 500 series of responses with the exception of 501.

TABLE I
BAYES AND OBSERVED ACCURACY RATES FOR EACH COMBINATION OF TEST URI SET AND GROUPING

	All URIs	New URIs
Ungrouped Bayes accuracy	0.9421	0.9973
Ungrouped empirical accuracy	0.8858	0.9536
Grouped Bayes accuracy	0.9990	0.9998
Grouped empirical accuracy	0.9936	0.9950
Total records	830237	100546

TABLE II
CONFUSION MATRIX FOR GROUPED CLASSIFICATION ON NOVEL URIS W/ LEARNER LABELS ALONG TOP ROW & ACTUAL LABELS IN LEFT COLUMN

	Bad Request	Handled Request	Server Error
Bad Request	40	471	0
Handled Request	3	99994	0
Server Error	0	37	0

The BER for each of the four test scenarios we consider (ungrouped and grouped responses with full and unique data), our observed error rates, as well as the number of records, are given in table I.

Note that – in particular – the classifier performs quite well on novel URIs that it has not been exposed to before, outperforming the classifier presented with several repeated URIs. The relatively worse performance of the latter can be attributed to several classes of URIs for which the complete test data had a different distribution of response codes than the training data; the (in this case) nonstationary nature of the problem led to a higher classification error rate. Examination of the classifier's learned labels for the training data verified that it had learned to correctly assign the most probable class within the training data. More detailed examination of the confusion matrix in table II shows that the "false positive rate" that would be obtained by using the classifier as some version of a web application firewall is relatively low, with only 3 normal requests being misclassified as being in the "bad request" category. However it should also be noted that the confusion matrix cannot identify errors caused by ambiguous labels. In 221 of the 471 "false negatives", the same novel URI appeared with both "Normal Request" response codes and "Bad Request" response codes. This indicates a non-separable set of URIs within which errors are inevitable. Most of these cases appeared to relate to additional authorization information (e.g. cookies) that were not captured in the URI. The LSTM model correctly assigned the most probable label to such ambiguous URIs, minimizing the error rate.

V. DISCUSSION

In this section we discuss these results in the context of prior work and LangSec theory, and what the results mean for future avenues of research. Given prior theoretic results demonstrating the entire class of context-free grammars cannot be learned by a single, powerful learner, (see Section II) our

work adds to the group of specific, useful subsets of CFLs that do seem to be learnable in practice.

A. Language Learning Context

The result that at least some URIs are learnable using an RNN is not as immediately surprising as the grammatical inference theory would suggest. Results from Clark[30], [31] have shown that non-terminally separated (NTS) languages, themselves a subset of CFLs, can be PAC-learned when considering less-broad distribution requirements than Valiant originally used. Generally speaking, PAC-learning is more restrictive than what is considered in this work. While it is generally difficult to derive PAC bounds for sequence-to-sequence models of the kind we explored, and it is difficult to determine (for RNNs) what the underlying learned grammar actually is, the fact that we obtained relatively high accuracy for prediction tasks, and such a high compression ratio on the generative tasks, suggests that the model nevertheless uncovered a significant degree of structure.

Since classification of URIs has overlap in the possible labels for a given URI, the Bayes error rate represents the best-case scenario for identification, assuming a learned language that can classify the URI. That is, the set of URIs considered here cannot be considered to be purely deterministic in terms of what language is generating the URI strings. Thus, we would not expect to achieve absolutely correct identification in this learning environment. The final results demonstrate a close agreement between the Bayes and the experimental error rates for both discriminative and generative modeling, which suggests that the learner is nearly as good as it is possible to be, with the grouped results being particularly promising.

From a LangSec point of view, the learner is successfully identifying server response to a given URI, which should allow for useful security differentiation of those URIs. These results also suggest that the picture for the existing protocol infrastructure of the internet is perhaps not as bleak as a first examination through the lens of LangSec and grammatical inference would suggest. Current language-theoretic results suggest avoiding (where possible) grammars for protocols that are in any of the classes more complicated than regular in the Chomsky hierarchy, arguing that a wide range of complexity results suggests protocols with such high complexity are not just inherently unsecure, but in many cases inherently unsecurable, due to factors such as lack of provable equivalence.

In many cases however, as discussed above, these proofs of hardness either imply or require the nonexistence of only exceptionally powerful learners: learners that must cope with every possible grammar in a class, including infinite and/or pathological ones. Optimistically, in practical applications, such hard-to-learn grammars within the class of interest may in fact be uncommon. Our modeling of URIs provides a concrete example of a particular grammar (the grammar describing URI endpoints on a specific server) within a broad class of grammars (all possible valid URI schemes) that does in fact appear to be learnable by what now count as fairly standard and well understood sequence learning algorithms.

B. Future as LangSec

Three clear problems are suggested by these results. First, can we describe a non-Chomsky class of grammars that is flexible enough to cover normal use cases in protocol design without admitting the pathological cases that make inference and recognition problems unsolvable in more general classes? Second, can we describe tests that will allow us to determine efficiently if a given protocol is a member of that class of grammars? (This may also provide guidance for protocol designers who require some of the expressive power of more complex classes of grammars, but wish to retain some of the security benefits of simpler classes.) Third, can the use of "black box" learning methods, such as those presented here, assist in manual deconstruction of existing grammars that may lack formal specifications?

A first point of interest is that some protocols may be (at least nearly) in a learnable subset of CFLs, such as NTS languages. Such languages may have good learnability properties, as discussed above. Though it is not clear if a large majority of protocols would fit into such classes, or if such protocols' being only nearly in them is sufficient for good learnability. The question of determining a protocol's membership in such useful subclasses of CFGs remains open.

There are also results showing that finite state machines can be extracted from RNNs using varying methods of rule extraction, for an overview see Jacobsson's review[32]. Although these methods may not be robust enough to produce a minimal and completely unambiguous state machine describing the class of URIs, it may be possible to extract enough of such a machine to be used for security purposes. If, for example, a machine could be extracted from a trained LSTM that was capable of identifying 80% of malicious requests (a result substantially below what was found here for the LSTM itself), the machine could be used to filter out bad traffic before it even arrives at a server. This provides a possible avenue to improving the security of such servers in a LangSec way that is implementable on systems that exist today that otherwise use "unsecurable" parsers or implementations.

Another possibility would be to construct a deterministic (or even probabilistic) finite automata (DFA) from the LSTM based on the per-character entropy spikes the LSTM reports. Again, the goal would be to produce a quickly navigable DFA that could be used to filter requests before they get to an otherwise vulnerable server. Although theoretic results suggest that it may be difficult or impossible to learn a CFG in the "proper" sense, learning a parsing method from a trained LSTM provides similar benefits.

The results considered here are proof that systems that can be called inherently unsecure may be able to be made at least more secure by adapting more probabilistic methods from machine learning. Although training an RNN for all possible vulnerable servers is not an easy or long term solution, it may provide a stop-gap for those with larger servers and more access to the computation power to train an RNN. Additionally, our results show that a very basic RNN, with

no prior assumptions about the server and the requests seen, can be trained to have excellent discriminative rates.

With all of this in mind, pure mathematical proofs have left the viability of learning (potentially vulnerable) languages in shambles. At best, it may be possible to construct, from base assumptions, languages restricted to low levels of the Chomsky hierarchy. These languages may then be secure in a LangSec sense. However, it is unlikely that such an approach will be quickly adopted (if adopted at all) by the wide array of groups that continue to use unsecurable systems. Here, we have shown that if the requirement of perfect mathematical learnability is relaxed to a probabilistic one, it is possible to get results that provide a method of securing already existing systems with already well understood algorithms. This presents a possible "pit-stop" on the road to fully LangSec-secure implementations, where already existing servers and systems can be given at least some level of protection that LangSec has proposed, without requiring immediate adoption of new parsers.

VI. Conclusion

In this paper we summarized some important theoretical results in the field of grammatical inference, suggesting that the problem of grammatical inference that is posed by applying language-theoretic security principles to existing network infrastructure is extremely hard. However, these theoretical results focus on very general learning algorithms that must account for all languages within a class of grammars. This typically requires that the learning algorithm be capable of handling infinite and pathological languages within the class as well; in real world learning scenarios, such grammars may be possible to exclude from consideration *a priori*, thus significantly simplifying the problem of inference for that class.

We presented empirical results that support this conjecture by demonstrating good learnability of URIs inside of HTTP access logs by means of an RNN. We found that despite theoretical results showing that context-free grammars are not learnable as an entire class by a single learner, the URI subset is in fact learnable for some problems to nearly the Bayes error rate. For most practical applications, and in particular those applications relevant to LangSec, this likely constitutes sufficient learnability. In particular, the generative run of the RNN produced novel string examples that were still classifiable under the known class of URIs, while the predictive model provides a strong level of discriminative power. We also offered some commentary on possible theoretical justification for these positive results, with suggestions for follow-up work.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, pp. 447–474, 1967.

[3] N. Chomsky, "On certain formal properties of grammars," *Information and Control*, vol. 2, pp. 137–167, 1959.

[4] D. Angluin, "Inductive inference of formal languages from positive data," *Information and Control*, vol. 45, pp. 117–135, 1980.

[5] ——, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, pp. 87–106, 1987.

[6] D. Angluin and C. H. Smith, "Inductive inference: Theory and methods," *ACM Computing Surveys*, vol. 15, pp. 237–269, 1983.

[7] A. Clark, "Distributional learning of some context-free languages with a minimally adequate teacher," in *Grammatical Inference: Theoretical Results and Applications*. Springer Berlin Heidelberg, 2010, pp. 24–37.

[8] L. Lee, "Learning of context-free languages: A survey of the literature," *Harvard University*, 1996.

[9] R. E. Harang and K. N. Wood, "Grammatical inference and language frameworks for langsec," in *IEEE CS Security and Privacy Workshops*, 2015.

[10] S. Ginsburg, *The Mathematical Theory of Context Free Languages*. McGraw-Hill Book Company, 1966.

[11] D. Angluin, "Negative results for equivalence queries," *Machine Learning*, vol. 5, pp. 121–150, 1990.

[12] ——, "Finding patterns commob to a set of strings," in *Proceedings of the Eleventh Annual ACM Symposium*, 1979.

[13] T. Motoki and T. Shinohara, "Correct definition of finite elasticity," *RIFIS Technical Report*, vol. 28, pp. 1–4, 1990.

[14] A. Clark and R. Eyraud, "Identification in the limit of substitutable context-free languages," in *Algorithmic Learning Theory*. Springer Berlin Heidelberg, 2005, pp. 283–296.

[15] L. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, pp. 1134–1142, 1984.

[16] C. de la Higuera, *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[17] M. Kearns and L. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," *Journal of the ACM (JACM)*, vol. 41, no. 1, pp. 67–95, 1994.

[18] B. den Boer, "Diffie-hellman is as strong as discrete log for certain primes," in *Advances in Cryptology-CRYPTO'88*. Springer, 1988, pp. 530–539.

[19] D. Boneh, "The decision diffie-hellman problem," in *Algorithmic Number Theory*. Springer, 1998, pp. 48–63.

[20] S. Arikawa, T. Shinohara, and A. Yamamoto, "Learning elementary formal system," *Theoretical Computer Science*, vol. 95, pp. 97–113, 1992.

[21] H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck, "Pulsar: Stateful black-box fuzzing of proprietary network protocols," in *Proc. of the International Conference on Security and Privacy in Communication Networks (SECURECOMM)*, 2015.

[22] R. Fielding and J. Reschke, "Hypertext transfer protocol (http/1.1): Message syntax and routing," Internet Engineering Task Force (IETF), Tech. Rep. RFC 7230, 2014.

[23] T. Berner-Lee, R. Fielding, and L. Masinter, "Uniform resource identifier (uri): Generic syntax," IETF Network Working Group, Tech. Rep. RFC 3986, 2005.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] F. Gers, J. Schmidhuber *et al.*, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *Neural Networks, IEEE Transactions on*, vol. 12, no. 6, pp. 1333–1340, 2001.

[26] "Chainer: A flexible framework of neural networks," http://chainer.org/, accessed: 2015-12-18.

[27] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[30] A. Clark, "Pac-learning unambiguous nts languages," in *Grammatical Inference: Algorithms and Applications*. Springer Berlin Heidelberg, 2006, pp. 59–71.

[31] ——, "Learning deterministic context free grammars: The omphalos competition," *Machine Learning*, vol. 66, pp. 93–110, 206.

[32] H. Jacobsson, "Rule extraction from recurrent neural networks: A taxonomy and review," *Neural Computation*, vol. 17, no. 6, pp. 1223–1263, 2005.