"What did you say?": Extracting unintentional secrets from predictive text learning systems

Gwyn Wilkinson and Phil Legg
Computer Science Research Centre
University of the West of England, Bristol, UK
Gwyn2.Wilkinson@live.uwe.ac.uk

***Abstract*—As a primary form of communication, text is used widely in applications including e-mail conversations, mobile text messaging, chatrooms, and forum discussions. Modern systems include facilities such as predictive text, recently implemented using deep learning algorithms, to estimate the next word to be written based on previous historical entries. However, we often enter sensitive information such as passwords using the same input devices - namely, smartphone soft keyboards. In this paper, we explore the problem of deep learning models which memorise sensitive training data, and how secrets can be extracted from predictive text models. We propose a general black-box attack algorithm to accomplish this for all kinds of memorised sequences, discuss mitigations and countermeasures, and explore how this attack vector could be deployed on an Android or iOS mobile device platforms as part of target reconnaissance.**

I. Introduction

PREDICTIVE text has become widely adopted across many modern devices to improve usability, and time required for composing a message. Predictive text functions first gained wide popularity on early mobile devices which provided a numerical keypad instead of a traditional keyboard, which necessitating the grouping of multiple letters each number key ('*ABC*' on key 1, '*DEF*' on key 2, and so forth). The so-called '*T9*' predictive text model would check all three letters from each keypress and suggest word completions based on a stored dictionary [7]. Whilst these methods were fairly straightforward in their approach, based on frequency analysis, nowadays we can utilise more sophisticated deep learning models such as recurrent neural networks, and long short-term memory (LSTM) networks, to develop a much richer understanding of a user's writing style and their message composition [17]. Such learning methods will continually learn about the sequence of words and characters that are inputted by the user.

Whilst this can have great benefits for developing a bespoke model that fits to the user's preferences well, by the very nature of learning from all interactions, there are potential security vulnerabilities that could be associated with this. For example, if a user was to write a confidential message to a family member, how would this sensitive information be captured by a learning model? How about other sensitive materials that may be commonly entered by a user? In particular, the reliance that users have nowadays on online accounts and login credentials means that passwords may be commonly entered via the device keyboard. Therefore, if the device is designed to continually learn about the user's typing interactions, such that patterns can be identified and predictions can be made for future interactions, does this present an opportunity for an attacker to extract unintentional secrets?

In this paper, we explore the issue of unintentional memorisation in predictive text modelling systems. We devise a small-scale study to test how well a system can learn about our text-based interactions such that predictions can be made on a character or word level for what should follow a given passage. We then examine how this could potentially be exploited by an attacker to identify personal information about a user. Examples could include presenting the first part of a post code and having the system reveal the remainder to identify location information of a user (e.g., if an attacker entered 'BS16', what would a system predict to come next?). We discuss this challenge further to consider the nature of privacy attacks that could be exploited in this manner, including how feasible a password could be extracted from a learnt model of user interactions.

The paper is structured as follows: Section II presents related work on the topic of vulnerabilities, possible attacks and mitigations which can be performed on deep learning models. Section III provides a background to the domain, including deep learning, recurrent networks, generative sequences, legal privacy concerns, and our intended threat model. Section IV presents our system design, incorporating our deep learning model architecture, training data corpus, language model design, and the attack algorithm. Section V presents our results and a discussion of these findings. Finally, Section VI concludes our work and provides discussion of further works.

II. Related Works

There has been a considerable amount of research in the intersection between machine learning and privacy preservation. It has previously been shown that a machine learning classifier, trained on private data, can leak information about its training set [10], [11]. Through repeated training rounds, a machine learning classifier $f_\theta()$ can achieve high accuracy in predicting its training data. Due to this property it is possible to use the presence of a high confidence output to infer sensitive information about a data subject - for instance, whether a patient record in a study possesses a specific genotype. This kind of attack, named *model inversion* was first demonstrated in the black-box setting (i.e. an adversary can access a model but not inspect it's parameters), using a pharmacogenetic regression model [11], and later extended to facial recognition [10] in both black-box and white-box settings (i.e. an adversary can inspect model parameters). One limitation of this attack in the black-box setting can be that some prior knowledge of the target data subject's non-sensitive attributes is needed, and this can be addressed using *model poisoning* to inject malicious data into the training set [13].

An extension of this technique allows an adversary to perform *membership inference* [33], by which the input is initialised and then optimised to produce a high confidence output from the model. By identifying a local maximum in this way, we can infer a correlation between the data point and a training set member, and in theory rebuild the dataset in its entirety. This has been extended from the centralised to the distributed training environment, where it has been shown a malicious training participant can infer data membership through crafted updates [25].

Some causes of this vulnerability have been identified, such as influence of individual data points [38], and poor generalisation (or *overfitting*) [40], and mitigations suggested such as ϵ-differential privacy [1], regularisation techniques [21] [15] [34] [28], adversarial regularisation [24], API hardening [36] and improved data sanitisation techniques [18]. However, due to the well-known tension between privacy and utility in data publishing [30], such attacks can never be completely prevented in an accurate model. Previous work by Legg *et al.* [22] explored how visual analytics techniques can be used to identify potential vulnerabilities that may exist within a trained model, and how robustness could be improved when using collaborative user-machine guided sample selection.

Inference attacks seek to leverage the *unintended memorisation* of training data, to which modern deep learning models have been shown to be particularly susceptible [41] [23]. Recent work by Carlini *et al.* [4] provides a conceptual demonstration of unintentional memorisation for generative text models. They demonstrate the concept for a credit card number injected within body text that is used for training a deep learning model. Our work extends on this notion, however we address passwords rather than credit card numbers since we believe passwords are more commonly written or stored by users (e.g., sharing with a family member via text message, or storing in a Notes application). Furthermore, a credit card number will naturally be out-of-distribution from typically keyboard usage, whereas a password may closer resemble text that the user typically writes. Our work therefore focuses on the more salient case of storage and extraction of a strong password, of arbitrary length and no known context, and the practical feasibility of weaponising this in the process of performing cyber-reconnaissance.

III. Background

A. Neural Networks

A *neural network* refers to a function f with parameters θ, designed to approximate some other function which resists being defined explicitly. This approximation is found using a set of m example data points x and labels y,

$$\mathbb{X} = \{ (x_i, y_i) \}_{i=1}^{m}$$

which are used to compute $\overline{y}_i = f_\theta(x_i)$. We then evaluate how accurate the network is with respect to each data point by computing the *loss function* $L(x, y, \overline{y}, \theta)$, and then performing gradient descent to update θ. In other words, we compare the estimated values of $\overline{y}_i$ to the known values of y_i, and use the average of the differences to update θ in the direction which minimises the loss,

$$\theta_{new} \longleftarrow \theta_{old} - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla L(x_i, y_i, \theta_{old})$$

where η is a *learning rate*, which limits the update magnitude in order to reduce over-correction. This process is known as *training*, and it is often required to train a neural network many times over the entire set of $\mathbb{X}$, with each full iteration known as a *training epoch*, in order for the network to reach a desired level of accuracy with respect to the training data. However, accuracy on training data can only translate to accuracy in the real world to the extent that $\mathbb{X}$ is representative of the global distribution. In practice, we will almost always see under- and over-representation of one or more features in a training set relative to the global distribution, and chasing higher accuracy through continued training can run the risk of *overfitting* the network to the training data in a way which reduces its real world accuracy.

B. Recurrent Neural Networks

A *Recurrent Neural Network* (RNN) is a subclass of neural networks which differ from traditional feed-forward neural architectures in that each unit's input at each time step $t \in \mathbb{Z}$ consists not just of the input vector $x_t \in \mathbb{R}^n$ but also a hidden state vector $h_{t-1} \in \mathbb{R}^m$, returning h_t and the output vector $\overline{y}_t$. This essentially gives each neural unit a memory enabling it to classify the present input in the context of the inputs it has seen recently.

One problem with the standard RNN is that, while in theory h_t can contain information from $x_{0:t-1}$, in practice the storage of long term dependencies can lead to problems with training due to exploding/vanishing gradients caused by needing to multiply together each gradients during back-propagation [27]. The use of Long short-term memory (LSTM) [14] addresses this problem through use of one or more memory cells accessed exclusively through gate functions, the effect of which is to make back-propagation additive rather than multiplicative. In this way the degree of exploding/vanishing gradient effect is limited so that it can be treated easily during training, by altering the learning rate η or by clipping the gradients to stay within a specified min/max.

C. Generative Sequence Models

RNNs are especially effective for producing *Generative Sequence Models*, useful for applications such as natural language processing, speech recognition, and machine translation. Generative sequence models can be defined as models which will take in a sequence $(x_1, ..., x_n)$ and output a next token x_{n+1}, based on a distribution $\mathbf{Pr}(x_{n+1}|x_1, ..., x_n)$. For instance, in a language modelling task, the model would create a distribution of all known words, with each given a probability of occurrence based on the previous words in the sentence. If the model's input consists of "the cat sat on the", it would assign a high probability to the word "mat" occurring next, and a low probability to the word "a", although "a" has a much higher overall occurrence in English.

Training this kind of model is similar to the standard neural method described above, although instead of a set of labelled data we have a training data corpus of length m, broken into sequences of length n, comprising one or more discrete tokens x. In this paradigm we simply substitute the sequence $x_{1:n-1} = (x_1, ..., x_{n-1})$ for the data point x_i, and the label is simply the next token x_n, so that our gradient descent works this way:

$$\theta_{new} \longleftarrow \theta_{old} - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla L(x_{1:n-1}, x_n, \theta_{old})$$

In this way the model learns a conditional distribution of the tokens contained within the training data, which can then be used to identify the mostly likely next token in a sequence. Formally, in this context we use $f_\theta(x_{1:i})$ to find $\mathbf{Pr}(x_i|x_{1:i-1})$ where $x_i \in \mathbb{X}$.

Due to the arbitrary length of such sequences (such as a conversation, or a video output feed), a generative sequence model can be effectively implemented using stateful architecture such as RNNs.

D. Legal Implications of Secret Memorisation

The recent adoption and enforcement of the General Data Protection Regulation (GDPR) [31] across the European Union has focused the minds of data privacy researchers and practitioners alike. Broadly speaking, it mandates that storage of personally identifiable information relating to a data subject requires up-front justification - whether it be informed consent, national law, and so forth. In short, if a cloud-based deep learning model is capable of unintended memorisation of a user's private information without any appropriate justification (or the user's knowledge that such a thing is possible), it could constitute a serious breach of GDPR provisions requiring disclosure to the relevant Data Protection Officer. We consider this a strong motivator for research into this area.

E. Threat Model

We assume a setting in which a predictive text model f_θ is stored on a user-owned device. This model is continually trained using text entered by the user, using a soft keyboard installed on the device. This keyboard is used to input text data into all applications on the device, including clients for sending SMS, email, web form data and so on, all of which therefore contribute towards training a general predictive text model. The keyboard uses the trained model to suggest word completions through an API which takes the current text sequence as input and returns a sorted list of next-word suggestions.

We assume an adversary who has no information about the data used to train the model, with *black box access* to the model. A black box attack is one in which the adversary can get and keep any information exposed by interacting with the API, and leverage any previously learned information to attack further. For the purposes of this exploration, we also assume an unlocked, accessible device and do not consider the suite of modern access controls such as face recognition, thumbprint scanning, or keycode entry, in order to best understand this attack surface and the potential vulnerability that it represents.

IV. System Design

Our proposed attack targets a user-trained text-prediction, or text-completion feature. Smartphone operating systems such as iOS and Android have adopted such features in their touchscreen keyboard to assist user's in composing messages quickly. To give us the freedom to experiment in this kind of environment, we first need to build a representative generative model and training strategy, and an interface through which it can be polled for predictions.

A. Model Architecture

Due to their proven ability to make accurate predictions on sequential data while avoiding problems of exploding or vanishing gradients, and the limited resources we have available for multiple training cycles, a single layer of LSTM units is used for this application. The state of the art in these kinds of model are arguably represented in the work of Devlin *et al.* [5] and Radford *et al.* [29], both of which use very deep, highly dimensional models and incorporate the newly developed *attention function* of Vaswani *et al.* [37]. It could be argued that, intuitively, a single-layer low-parameter LSTM model's performance cannot be substituted for a far larger, more sophisticated network. However, as we know that a very large model can memorise its training set even while achieving high accuracy [41] [23], we expect that any positive results achieved on out-of-distribution sequences in a small, simple model would be magnified if repeated on more sophisticated models. Therefore, a small LSTM model is a better choice for confirming and measuring what we hope to observe. Our full model architecture is therefore an input layer, one single LSTM layer with 75 units, and a dense output layer.

B. Training Data and Approach

We wish to select a suitable data corpus such that we can train a general text model quickly, and so that we can investigate how the model performs when crafted sequences are injected into the corpus that are out of distribution. Whilst there do exist text message corpora, such as *Almeida, Hidalgo and Silva* [2], their intended usage (e.g., identifying spam messages) means that they are not sufficiently representative. Furthermore, the inclusion of 'txt spk' abbreviations in this dataset is now less commonly adopted by users compared to earlier text messaging usage. For the purpose of this project, we opted for *The Adventures of Sherlock Holmes* by Arthur Conan Doyle [6] from the Project Gutenberg repository since this is an easily accessible texts written in standard English. Whilst this choice was to represent a general usage of the English language, the theme of Sherlock Holmes using small clues to detect larger secrets is true also of this presented work. We parsed the full text to remove special characters and punctuation, and convert to lower case, so that the text was suitable for training our model.

The out-of-distribution data in our case is an unknown sequence of letters representative of a user password, for which we generate a 16 character string using a random number generator. This length and composition are chosen simply to focus on the problem of strong passwords which are resistant to brute force, dictionary and masking attacks - in other words, the kinds of passwords we encourage users to adopt at home and in the workplace. Due to the low likelihood of meaningful sequences generated by a strong random number generator, we should expect that the combination of a long length and an unpredictable letter sequence together should put this information outside of any distribution of English usage, meaning it ought to be detectable as such. We insert this sequence into the training corpus at a random index, so that the attacker cannot leverage knowledge of the words surrounding the inserted password, such as *"My password is"* or similar phrases.

We train the model using the RMSProp algorithm, an adaptive learning-rate variant of stochastic gradient descent which has demonstrated faster convergence in RNN tasks [35], and incorporate early-stopping as a guard against overfitting [28]. In order to train the LSTM, we construct a set of input and output arrays where characters are represented using a one-hot vector encoding. For each set, the input array contains each three-character sequence (or *trigram*) comprising the text (represented as a 36×3 array). The corresponding output array contains the immediate next single character (represented as a 36×1 array). We train the model on all input / output pairs from the dataset for 20 epochs. The epoch value was chosen so that a practically viable predictive model could be repeatedly retrained for our experimentation, whilst also providing a suitable level of predictive performance that is capable of generating recognisable sequences. For the purpose of this study, once the trained model is capable of achieving reasonable predictions, we are interested to explore how this capability can be exploited further to extract information from the model.

C. Character-level and Word-level Language models

A language model can be understood both on the character level and on the word level. This describes whether the model learns and generates text as a stream of characters or of words. In other words, a character-level generator will take in the sequence '*playin_*' and suggest the letter '*g*', while a word-level generator will take in the sequence '*The cat sat on the ___*' and suggest the word '*mat*'. For this research we address character-level models. A word-level model would require a much greater level of complexity since there would be significantly more combinations of words that need to be modelled (i.e. iterating through a dictionary of every word encountered by the model, versus through a known standard character set). In addition, for our particular domain, this would essential require the password being explicitly stored within the dictionary of a word-level model, formed by observing the data. Our character-level model can effectively achieve predictions at the word-level, since we can iteratively predict the sequence of characters until a space character is encountered. An example of real-world reconnaissance would be postcode recovery, where entering the first part '*BS16*' could predict '*1QY*' as a next word (or essentially '*1, Q, Y, SPACE*' as the set of predicted characters), or fragments of information such as key personnel referred to by name, or client/supplier information where a supply chain is being targeted. These examples could all potentially operate on the same underlying principle as those demonstrated in this study.

D. Text-prediction Interface and Attack Sequence

The model and program we have constructed lends itself well to inferring a high probability complete word, given an incomplete one. A simple algorithm would be to provide a partial string s into the model, append the suggested next-character to s, and repeat this until the next suggestion is a space or newline character. By building a top-k tree of potential next characters, we can approximate the functionality of a predictive-text interface which could take the sequence '*even*' and suggest '*[event, evenly, eventually]*'. Predictive accuracy greatly improves with longer input strings (such as 5-grams or 10-grams) according to Shannon [32], but here we focus on the higher-entropy (or 'more surprising') case of building longer strings from shorter ones.

To frame our attack, we envision a high-access target with a strong 16-character password. We know they are not susceptible to a dictionary attack, and we do not have the computational power to mount a brute force attack on a human timescale. We have a method to access the target's predictive text model, and wish to leverage this to either discover the password or narrow the search space to something reasonable. With these elements, we have built two algorithms to extract an embedded random password of a known length, which came to be referred to as *Simple* and *Deep*.

1) Simple Search Algorithm: Our *Simple* algorithm builds an array of lowercase letter trigrams $\mathcal{S} = \{aaa, ..., zzz\}$, and selects $\mathcal{S}_i$ as an input for our model. The model will output the probability distribution of possible next characters, and we select the top most likely. If the character is a space, we move to the $\mathcal{S}_{i+1}$, otherwise we append the suggested character to $\mathcal{S}_i$, pop the first character, and input the new trigram into the model. Whenever a sequence of characters, unbroken by spaces, of length 16 (inclusive of the initial trigram) is encountered, the sequence is added to a list of potential password candidates. In effect, this is a simple shortest-path search which explores the most likely node each time. As there are not many sixteen character words, we would expect a sequence of this length to have a high probability of success.

2) Deep Search Algorithm: A more thorough *Deep* search can be performed where we perform a similarly greedy algorithm, but disregard all 'space' characters. Where a 'space' is encountered as the top-1 suggested next character, we take instead the second-placed suggestion. This creates a sixteen character string for each member of S, with a mathematically higher likelihood of success but also a higher wall clock time. The list of candidate strings generated by either algorithm will, we hypothesise, have a high chance to contain the memorised full inserted password string. To clarify, our generated passwords do not contain spaces, and so the presence of the 'space' character is always to denote the break between two words in our corpus.

Our experiments consisted of generating a randomised sixteen-character string P of lowercase letters, instantiating a new model with randomised parameters, training the model on a text corpus of length $L = \{1000, 2000, 4000, 16000\}$ with the password inserted at a random index, using our Simple and Deep extraction algorithms described above to generate a list of potential password candidates, and comparing the generated candidates with the inserted password. If we find the inserted password in the list of generated candidates, we register the run as a success, and at the end of 20 runs of each test we calculate the percentage of success of that series of experiments. In this way we can see the algorithms' effectiveness for different amounts of training data, which intuitively mirrors how a predictive text model would be used over time to write messages.

In our tests, due to the noticeable fall in accuracy of the simple algorithm as we increased the text length, we chose not to run it after 4000 words. We include it nonetheless as a low-commitment 'quick and dirty' first pass approach which can be run conveniently in a few minutes.

Our testing platform was built using using Google Colaboratory and run on a Nvidia Tesla T4 with a CUDA Compute Rating of 7.5 [3]. All models were built and trained using the Keras/Tensorflow library for rapid ML experimentation [19]. Our testing results are shown as follows.

V. Results & Discussion

Length (chars)	No. of Words	Algorithm	Candidates	Success%
1000	121	Simple	26.65	90
2000	208	Simple	35.25	95
4000	368	Simple	19.45	60
4000	368	Deep	21952	90
16000	1049	Deep	24389	10

TABLE I
SUMMARY OF RESULTS SHOWING THE SUCCESS OF OUR ALGORITHMS IN EXTRACTING A PASSWORD EMBEDDED IN VARIABLE-LENGTH TEXT CORPORA.

Table I shows the results of our experimentation. Our simple search algorithm identifies a variable number of candidate strings, while the deep search algorithm will produce the same number on each run - the difference from 4000 to 16000 letters is caused by the increased alphabet size in the longer corpus. It can be seen that the password becomes hard to extract as we increase the text length, which then improves greatly as we search more deeply, and then falls again with a larger text corpus. This would seem to suggest that success could be achieved by progressively deepening each round of searching until the embedded secret is reconstructed.

We argue the results represent a qualified success for our approach, showing that a password, or any sensitive text sequence inserted once into a body of training data, can in some cases be reconstructed algorithmically by an adversary. We also see the effect of increasing the text length, as we reach the limits of what can be extracted using the three-character beam search. Our discussion of these findings follows.

A. Mining Models for Secrets is Viable

In the password search case, the primary benefit is to significantly reduce the complexity of the search space. In the case of strong random passwords, where the standard dictionary attack or mask search approaches will not find a foothold, our method presents an alternative to the long-form brute force search. Assuming a successfully extracted password, even in the worst case we have mapped the n character search space S^n (about 4.3×10^{23} in our model) to a three-character S^3 space (under 20,000), radically reducing the search space and so making such an attack vector feasible.

Small modifications of this approach can let us intuitively mine other personal information from a predictive model. Postcodes, car registration numbers, contact details and so forth can be brought out of a model if it has seen such information in the target's messages. In this context, access to a target's predictive text app can be seen as a high value reconnaissance asset. We argue that this is a threat deserving of more attention by researchers and practitioners, going forward.

B. Mitigations and Countermeasures

1) Differential Privacy: In general, differential privacy (DP) is understood as a formally guaranteed method of preventing privacy leakage from a disclosure mechanism [8]. Formally, a function $\mathcal{K}$ with data sets D and D' is recognised as differentially private if:

$$Pr[\mathcal{K}(D)] \leq exp(\epsilon) \cdot Pr[\mathcal{K}(D')]$$

where D' differs from D by no more than one additional element. In the context of this research, we would say that the addition of one word to the training set (such as a password string) should have an exponentially small effect on the suggestions made by the predictive text model, such that there be no observable difference directly attributable to the presence of the password. Generally, the selection of the parameter ϵ is designed to be guided by a *risk assessment* of a privacy leak, as a function of *likelihood* and *impact*. If a leakage is thought to be unlikely or easily mitigated, then a higher ϵ can be used, whereas if a leak is expected to result in expensive consequences then ϵ should be made smaller. This property in practice is balanced by the fact it is implemented by adding noise during gradient descent. The lower the ϵ, the noisier the training, and (in extreme cases) the less useful the model's predictions.

DP has been explored practically as a countermeasure to model inversion attacks by *Shokri* [33] and *Carlini* [4], as well as being given formal treatment in the predictive model context by *Dwork* [9]. Most interesting from the perspective of situational awareness is the *Exposure* metric Carlini puts forward, which aims to answer the likelihood part of the risk assessment. By analysing the predictive confidence of each suggested character relative to a random space, and comparing to the other options in the space, the memorised secret can be ranked against every equivalent string in the relevant search space. If the secret has a lower exposure ranking against one or more equivalent strings in the space, it is less likely than that string to be produced by the model. This metric should allow for an optimal ϵ to be discovered iteratively for each model, giving the best predictive performance possible while keeping likelihood of extraction below a tolerable limit. The drawback of this technique is that it requires the sampling or modelling of a defined subspace within the distribution - effectively requiring prior knowledge of a specific memorised secret in order to rank its exposure. We believe this limits its practical applicability in most cases, but does suggest a possible use as a pre-emptive measurement where ϵ must be calculated in order to prevent certain kinds of secrets two which a model is *expected* to be exposed. Whilst the aim of this current study was not to explore the role of DP, we acknowledge that there is potential to explore DP in the context of unintentional memorisation.

2) Text Message Sanitisation: Theoretically speaking, the most provably effective countermeasure to the problem of a deep learning model memorising secrets is to remove the secrets from training data set. If the secret is not seen by the model, then it doesn't become part of the distribution and will not be distinguishable from randomness by an extraction algorithm such as the ones we have described. Removal of secrets from a text pre-training could be done in multiple ways - for example, using a whitelist filter to only allow known dictionary words into the training set, or a blacklist to prevent words that contain combinations of lowercase, uppercase, numbers, special characters, or some other heuristic for *password-likeness*. For cases where a text model is being continually trained using user inputs, we also would suggest the defensive use of an algorithm such as ours to detect embedded secrets and add them to a blacklist preventing them from further memorisation, were they to be re-entered at a later time. Most significantly, cyber awareness practitioners and researchers would need to build awareness amongst users of deep learning predictive text models (by which we mean, modern mobile phone owners) of the potential dangers of sending passwords and other private information to trusted parties via text messaging platforms.

3) Password Construction: Mention must be made here of the potential implications that this kind of vulnerability could have on the contemporary mental model of strong password construction. Often, guidance has suggested the use of a password with high entropy - by being long and including capitals, numbers, special characters and so on; is unique to the specific access it controls; is not predictable or guessable, such as by using common sequences, or an iterating suffix; and more besides. These best practices are effective at preventing attacks on passwords, because they aim to restrict password choice to distributions which are not searchable in human time.

Our algorithm raises the possibility that a strong password's own uniqueness and 'unpredictability' could even become its own weakness, due to the inherent properties of gradient descent algorithms [10]. As a strong password will not resemble other words and phrases, it will not have close neighbours in the data set, and in these cases a deep learning optimiser will simply fit a curve which predicts the password. A 'weaker' password, based on a sports team or a loved one's name, could by the same mechanism blend into the data and stick out. It might be possible to find a rule for generating strong passwords capable of resisting memorisation by language models, though this was beyond the scope of the present work. More recently, the National Cyber Security Centre issued guidance of using three random words [26], to provide a password that is both long and memorable, moving away from the notion of complex characters, which was also famously depicted in the XKCD 'Password Strength' comic [39].

4) API Restrictions: The attack algorithm described relies on an essentially unrestricted ability to poll a deep learning model repeatedly for a duration of nearly fifteen minutes. As we are using our experiments to model an attack on a predictive-text API on a mobile OS, one common sense mitigation would appear to be to modify such an API to limit the number of predictions it can make over a given time period. It may be possible to find a strategy where the user will not notice the restriction on their convenient use of the phone, but an adversary attempting password extraction would be denied.

VI. Further Work & Conclusions

In this work, we have presented a case study on extracting information from a text-based machine learning model, and hope to make a valuable addition to the ongoing discussion on whether the permeation of deep learning in all areas of life can be compatible with strong privacy assurances. Given the effectiveness of deep learning tools for generative language models (not to mention other domains), we hope that research and practice will focus on understanding this vulnerability in order to both protect the user, and empower them to make safe usage choices. In this spirit, we here identify some limitations in the scope of the present work, and some interesting avenues of potential future research:

1) **Varied and larger text corpora:** The purpose of the text corpus is to provide a 'background' language distribution from which a memorised secret can stick

out, and due to this property would not expect different password extraction results simply by changing the text corpus. We acknowledge the limitation that our study was based on a single text corpus. However, the purpose of our study is primarily to examine the potential of unintentional memorisation as a significant and relevant current-day attack vector, rather than to assess password extraction against different text corpora. Further research would aim to explore this property against genuine user device interactions rather than on publicly-available sample texts.

2) **Deeper Algorithms For Secret Mining:** Due to the resources and time available for repeated retraining and extraction, our experiment is designed around using very short sequences for training and extraction, from short training text lengths with artificially restricted character sets. As the results show, with a relatively small amount training data we exceed the limitations of trigrams as a predictive sequence. We would expect an algorithm which searched the 5-gram space with a full character set (uppercase as well as lowercase alphanumeric, plus special characters) to have significantly greater predictive success over a representative distribution of written text.
3) **More Complex Language Models:** We have been running our experiments on a single LSTM layer with a limited text corpus, which possibly is not representative the state of the art in language modelling that we see in BERT [5] or GPT-2 [29]. Language models of the sort employed at production scale use added layers of encoders and attention mechanisms [16] [37] [20] trained over millions of sequences. While we would expect such large and complex models to be more than capable of memorising secrets, an absence of true experimental data makes it difficult to know for sure what this could look like in practice.
4) **Real Hardware, Real Code:** Our synthetic model is run using Python code, using fast dedicated hardware, and this limits insight into a mobile implementation's performance in terms of wall clock time, processor cycle count, power draw and heat output, etc. We also did not investigate the practicalities of how iOS/Android soft keyboards request and receive predictive text suggestions. A better understanding of both areas would be valuable to evaluating the impact of this vulnerability.
5) **Greater Accessibility and Awareness:** Our experiences have raised the possibility that effective techniques such as Differential Privacy have perhaps not yet been rolled out in libraries and packages aimed at researchers and practitioners with less expertise and experience in designing and implementing deep learning. Given the high potential for liability for privacy breaches under the GDPR and other legislation, there would be value in assessing how widespread knowledge of this risk is in the fields of deep learning and data science, and how confident practitioners are of their ability to mitigate such risks as they could appear in their own products and data sets.
6) **Implications Under GDPR and other Legislation:** We would call on legal scholars to bring their attentions to how this vulnerability is to be interpreted in the current context of the GDPR (and derivatives such as the UK's DPA 2018), as well as similarly motivated statutes such as the California Consumer Privacy Act (enforceable as of 1 July 2020) [12]. Clarification of, for instance, whether such memorisation constitutes a 'filing system' for 'personal data' under these regulations could ultimately set the tone for any subsequent discussion of this issue. We leave these questions open to the community.

We have explored how this model can be exploited on a character level and a word level, such that by observing what outputs the model gives we can effectively learn about the traits and characteristics of the original text that the system has learnt from. In doing so, personal information that users may enter into their device (e.g., passwords, postcode locations, credit card numbers) could well be extracted if an attacker can judge the form it may have taken and its surrounding context.

We hope that in presenting this work we begin to highlight the security issues that are associated with the growing reliance on machine learning models and their increasing persistence in our daily lives. Future work will explore this challenge deeper, and begin to address how developers could identify characteristics that should be excluded from learning models.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318. ACM, 2016.

[2] T. Almeida, J. M. G. Hidalgo, and T. P. Silva. Towards sms spam filtering: Results under a new dataset. *International Journal of Information Security Science*, 2(1):1–18, 2013.

[3] Alphabet Inc. Google Colaboratory, 2020. Accessed: 2020-01-24.

[4] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232, 2018.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] A. C. Doyle. The Adventures of Sherlock Holmes, 1892. Accessed: 2020-01-17.

[7] M. Dunlop and A. Crossan. Dictionary based text entry method for mobile phones. In *2. Workshop on human-computer interaction with mobile devices (INTERACT'99)*, pp. 5–7. University of Glasgow, Department of Computer Science, 1999.

[8] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pp. 1–19. Springer, 2008.

[9] C. Dwork and V. Feldman. Privacy-preserving prediction. In *Conference On Learning Theory*, pp. 1693–1702, 2018.

[10] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333. ACM, 2015.

[11] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 17–32, 2014.

[12] E. L. Harding, J. J. Vanto, R. Clark, L. Hannah Ji, and S. C. Ainsworth. Understanding the scope and impact of the california consumer privacy act of 2018. *Journal of Data Protection & Privacy*, 2(3):234–253, 2019.

[13] S. Hidano, T. Murakami, S. Katsumata, S. Kiyomoto, and G. Hanaoka. Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pp. 115–11509. IEEE, 2017.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[16] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling, 2016.

[17] A. Karpathy. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, 21:23, 2015.

[18] A. Kassem, G. Acs, C. Castelluccia, and C. Palamidessi. Differential inference testing: A practical approach to evaluate sanitizations of datasets. In *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 72–79. IEEE, 2019.

[19] Keras Team. Keras: The Python Deep Learning library, 2020. Accessed: 2020-01-24.

[20] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

[21] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pp. 950–957, 1992.

[22] P. Legg, J. Smith, and A. Downing. Visual analytics for collaborative human-machine confidence in human-centric active learning tasks. *Human-centric Computing and Information Sciences*, 9(1):5, Feb 2019.

[23] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706. IEEE, 2019.

[24] M. Nasr, R. Shokri, and A. Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 634–646. ACM, 2018.

[25] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 739–753. IEEE, 2019.

[26] National Cyber Security Centre. *Three random words or #thinkrandom*, 2016 (accessed March 1st, 2020). https://www.ncsc.gov.uk/blog-post/three-random-words-or-thinkrandom-0.

[27] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[28] L. Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[30] V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, pp. 531–542. VLDB Endowment, 2007.

[31] G. D. P. Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88):294, 2016.

[32] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[33] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

[34] 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*, 15(1):1929–1958, 2014.

[35] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[36] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei. Towards demystifying membership inference attacks. *arXiv preprint arXiv:1807.09173*, 2018.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[38] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pp. 355–370. IEEE, 2016.

[39] XKCD. *Password Strength*, 2011 (accessed March 1st, 2020). https://xkcd.com/936/.

[40] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282. IEEE, 2018.

[41] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.