Grey-box Extraction of Natural Language Models

Santiago Zanella-Béguelin [1] **Shruti Tople** [1] **Andrew Paverd** [1 2] **Boris Köpf** [1]

Abstract

Model extraction attacks attempt to replicate a target machine learning model by querying its inference API. State-of-the-art attacks are learning-based and construct replicas by supervised training on the target model's predictions, but an emerging class of attacks exploit algebraic properties to obtain high-fidelity replicas using orders of magnitude fewer queries. So far, these algebraic attacks have been limited to neural networks with few hidden layers and ReLU activations. In this paper we present algebraic and hybrid algebraic/learning-based attacks on large-scale natural language models. We consider a *grey-box* setting, targeting models with a pre-trained (public) encoder followed by a single (private) classification layer. Our key findings are that (i) with a frozen encoder, high-fidelity extraction is possible with a small number of in-distribution queries, making extraction attacks indistinguishable from legitimate use; (ii) when the encoder is fine-tuned, a hybrid learning-based/algebraic attack improves over the learning-based state-of-the-art without requiring additional queries.

1. Introduction

Machine learning models are often deployed behind APIs that enable querying the model but that prevent direct access to the model parameters. This restriction aims to protect *intellectual property*, as models are expensive to train and hence valuable (Strubell et al., 2019); *security*, as access to model parameters facilitates the creation of adversarial examples (Laskov et al., 2014; Ebrahimi et al., 2018); and *privacy*, as model parameters carry potentially sensitive information about the training data (Leino & Fredrikson, 2020). Model extraction attacks (Tramèr et al., 2016) attempt to replicate machine learning models from sets of query-response pairs obtained via the model's inference API, thus effectively circumventing the protection offered by the API.

Several extraction attacks on deep neural networks (see Jagielski et al. (2020) for a recent overview) follow a *learning-based* approach (Tramèr et al., 2016; Orekondy et al., 2018; Pal et al., 2020; Krishna et al., 2020), where the target model is queried to label data used for training the replica. The replicas obtained in this way achieve accuracy on the desired task, or agreement with the target model on predictions, but recovery of the model weights is out of scope of this approach.

More recently, a novel class of attacks has emerged that uses *algebraic* techniques to recover the weights of deep neural networks up to model-specific invariances. Examples include the attacks of Milli et al. (2018), which leverage observations of gradients to recover model parameters, and Rolnick & Körding (2020); Jagielski et al. (2020); Carlini et al. (2020), which estimate gradients from finite differences of logits, and then use this information to recover model parameters. Algebraic attacks improve on learning-based attacks in that they (i) achieve higher-fidelity replicas and (ii) are orders of magnitude more query-efficient. So far, however, algebraic attacks have only been applied to small, fully connected neural networks with ReLU activations. In particular, for modern large-scale natural language models (LLMs) such as BERT or GPT-2 that rely on non-linear activations, the state-of-the-art model extraction attack is still learning-based (Krishna et al., 2020).

In this paper, we propose novel algebraic and hybrid (algebraic/learning-based) model extraction attacks on LLMs. We assume a *grey-box* setting where an encoder is pre-trained and known to the adversary, and where the target model is obtained from the encoder by (1) adding a task-specific linear classification head and by (2) fine-tuning using data unknown to the attacker. Our attack relies on an algebraic approach to extract the linear classification head, and a learning-based approach to extract the rest of the layers, i.e., the encoder and any additional layers between the encoder and the final classification layer.

For the *algebraic* approach, consider for a moment an idealized setting where the encoder is frozen during fine-tuning. There are two key observations that enable us to extract the

[1]Microsoft Research [2]Microsoft Security Response Center. Correspondence to: Santiago Zanella-Béguelin <santiago@microsoft.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

classification layer in this setting:

1. It is sufficient for an adversary to *know* (rather than to choose) the embeddings that are fed into the classification layer. In the grey-box setting, an adversary can compute hidden embeddings of any input by querying the public encoder model, and can query the target LLM on the same input through the model's API. We show in theory, and confirm by experiments, that a random set of n embeddings is likely to form a basis of the last layer's n-dimensional input space. The raw outputs (i.e., the *logits*) on this basis uniquely determine the parameters of the last linear layer, which can be recovered by a transformation to the standard basis.

2. This approach extends to the case where the API returns *probabilities* rather than raw logits, after normalization with *softmax*. For this, we leverage the invariance under translation of *softmax* to show that the parameters of the last layer can be recovered (up to invariance) from embedding vectors spanning its input space and their corresponding probability outputs.

For the *hybrid* approach, we assume that the encoder is fine-tuned together with the classification layer. In this setting, the embeddings computed using the public encoder only *approximate* those of the target model. To obtain a good approximation we first use a learning-based attack (Krishna et al., 2020) to distill a model based on the public pre-trained encoder. We use the resulting encoder to launch an algebraic attack (see above) to extract and replace the last layer.

We evaluate our attacks on LLMs of various sizes and architectures, fine-tuned to different downstream tasks. In particular, we study the effect on accuracy of the extracted model of using different kinds and amounts of API queries, and of using different learning rates for fine-tuning the encoder. Our key findings are:

- When the target model's base layers are *frozen* during fine-tuning (i.e., the attacker can get the exact embedding of any input), the algebraic attack is extremely effective. With only twice as many queries as the dimension of the embedding space (e.g., 1536 for BERT-base), we extract models that achieve 100% fidelity with the target, for all model sizes and tasks.
- When the model's base layers are *fine-tuned* together with the task-specific layer, the fidelity of the models extracted by an algebraic attack decreases as the learning rate grows. Maybe surprisingly, for some models and downstream tasks, we are still able to extract replicas with up to 82% fidelity and up to 79% task accuracy, for orders of magnitude fewer queries than required by state-of-the-art learning-based attacks (Krishna et al., 2020).
- Extraction is possible using either random or in-distribution queries. Replicas extracted using in-distribution queries perform well on both in-distribution and random challenge inputs. This shows that replicas can be created from small numbers of in-distribution queries, making attempts to extract the model indistinguishable from legitimate use.
- For sufficiently large query budgets, the hybrid learning-based/algebraic approach almost consistently improves over pure learning-based extraction, with gains of up to 3% in accuracy and 4% in agreement.

2. Attack

We consider a text classifier h consisting of a contextualized embedding model g (an encoder), such as BERT, followed by a task-specific classification layer f with softmax normalization, i.e.:

$$h\colon X \xrightarrow{g} \mathbb{R}^n \xrightarrow{f} \mathbb{R}^m \xrightarrow{\log \circ\, \mathrm{softmax}} \mathbb{R}^m$$

Specifically we assume that $h\colon X \to \mathbb{R}^m$ maps elements from a set X to class label probabilities in $\mathbb{R}^m$ where:

- $g\colon X \to \mathbb{R}^n$ is derived from a public model g_{pre};
- $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is an affine linear function computing logits from embeddings, i.e., $f(x) = Ax + b$ with $A \in \mathbb{R}^{m\times n}$ and $b \in R^m$;
- $\mathrm{softmax}\colon \mathbb{R}^m \to \mathbb{R}^m$ normalizes logits to probability vectors:

$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{i=1}^{m} \exp(x_i)} \tag{1}$$

2.1. Adversary

We consider an adversary that has query access to h, white-box access to the pre-trained public version g_{pre} of the embedding model g, and that tries to infer A and b (resp. f). We call this adversary *grey-box* because it can approximate the embeddings from the inputs to h using g_{pre}. We consider two different scenarios:

1. The embedding model g is *frozen* during training of f, i.e., $g = g_{pre}$.
2. The embedding model g is *fine-tuned* during training of f, i.e., $g \neq g_{pre}$.

We next present an algebraic attack on Scenario 1, before we show how to attack Scenario 2 using a hybrid algebraic/learning-based approach.

2.2. Algebraic Extraction of Classification Layers

Milli et al. (2018) show how to reconstruct models from oracle access to gradients. Carlini et al. (2020) show how to

replace gradients by finite differences of logits, enabling reconstructing models without such an oracle. The basic idea is to read off the ith column of A as the difference between $f(x+e_i)$ and $f(x)$, where $e_i = (0, \dots 0, 1, 0, \dots 0)^T$ is the ith vector of the standard basis in $\mathbb{R}^n$.

When the embedding model is frozen, i.e. $g = g_{pre}$, the adversary could in principle substitute x_i with the outputs of g_{pre} and thus extract the classification layer from h. However, there are two obstacles:

(1) the attack requires the inputs to f to be *chosen*, which would effectively amount to reversing the embedding model; and
(2) the attack relies on access to the raw logits, while APIs typically only provide log probabilities normalized with *softmax*.

We show next how to overcome each of these obstacles.

2.2.1. Extraction from Known Embeddings

As a first step, we show that it is sufficient for the adversary to *know* the embeddings, given that they are sufficiently random. We rely on the following standard result:

Proposition 1. *Let $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^n$ with $m \leq n$ be drawn uniformly from an n-cube. Then $\{x^{(i)}\}_{i=1}^m$ are linearly indepdendent with probability 1.*

Proof. By induction on m. The base case $m = 1$ is trivial. For the inductive case, let $\{x^{(i)}\}_{i=1}^{m+1}$ be chosen uniformly. By the inductive hypothesis, $\{x^{(i)}\}_{i=1}^m$ are linearly independent, and can be extended with $\{y^{(i)}\}_{i=m+1}^n$ to form a basis B of $\mathbb{R}^n$. Choosing $x^{(m+1)}$ uniformly is equivalent to independently sampling each of its coordinates in B. Then, $x^{(m+1)}$ is in the linear span of $\{x^{(i)}\}_{i=1}^m$ iff it has zero coordinates wrt $\{y^{(i)}\}_{i=m+1}^n$, which occurs with zero probability. Thus, $\{x^{(i)}\}_{i=1}^{m+1}$ must be linearly independent. □

Using this fact, we mount the following grey-box attack on $f \circ g$, i.e., we assume the adversary has access to the *logits*:

1. Choose distinct inputs $\{x^{(j)}\}_{j=1}^N$ in X with $N > n$;
2. Compute their embeddings $\{y^{(j)} = g_{pre}(x^{(j)})\}_{j=1}^N$ and logits $\{z^{(j)} = f(y^{(j)})\}_{j=1}^N$;
3. Construct a matrix $Y \in \mathbb{R}^{(n+1)\times N}$ where the jth column is $(1, y^{(j)})^T$, and a matrix $Z \in \mathbb{R}^{m\times N}$ where the columns are the logit vectors, i.e., $Z_{i,j} = z_i^{(j)}$;
4. Solve for $\tilde{A} \in \mathbb{R}^{m\times n}$ and $\tilde{b} \in R^m$ in

$$\left(\tilde{b}, \tilde{A}\right) Y = Z \tag{2}$$

Proposition 2. *Assuming g_{pre} maps inputs to uniformly distributed embeddings in an n-cube[1], the attack above uniquely determines the parameters of f. I.e., we have $\tilde{A} = A$ and $\tilde{b} = b$.*

Proof. By construction of (2) we have $\tilde{A}y^{(j)} + \tilde{b} = f(y^{(j)})$ for $j = 1, \dots, N$. The unique solution can be obtained multiplying Z by the right inverse of Y. For uniformly random embeddings, this right inverse exists because Y has full rank with probability 1 by Proposition 1. □

While in theory $N = n + 1$ distinct queries are sufficient to mount the attack, computing the inverse of Y based with finite-precision arithmetic can be numerically unstable. In practice, we gather a larger set of inputs and construct an over-determined system of equations. We then numerically compute a least squares solution to Equation (2).

2.2.2. Extraction from Softmax

In Section 2.2.1 we assume that the adversary has access to the raw unnormalized values, or logits. We now consider the case where the target model's inference API exposes only the log probability of each class, obtained by normalizing logits using *softmax* and returning the component-wise $\log$ of the result.

Softmax is invariant under translation, i.e., $\mathrm{softmax}(x) = \mathrm{softmax}(x + (c, \dots, c))$. We lift this property to linear functions:

Proposition 3. *Let $C \in \mathbb{R}^{m\times n}$ be a matrix with identical rows, i.e., $c_{i,j} = c_{i',j}$ for all i, i', j within range. Then for all $A \in \mathbb{R}^{m\times n}$ and $y \in \mathbb{R}^n$:*

$$\mathrm{softmax}((A + C)y) = \mathrm{softmax}(Ay)$$

Proof. Observe that since all rows of C are identical, $\mathrm{softmax}((A+C)y) = \mathrm{softmax}(Ay+(k, \dots, k)^T)$, where $k = \sum_{j=1}^n c_{1,j}y_j$. This last term is equal to $\mathrm{softmax}(Ay)$ from the translation invariance of $\mathrm{softmax}$. □

Due to Proposition 3 we cannot hope to recover A and b (as in Proposition 2). However, we can still obtain weights that yield functionally equivalent results when post-processed with *softmax*. Based on the construction of C above, we propose the following grey-box attack on h:

1. Gather inputs and construct a matrix $Y \in \mathbb{R}^{(n+1)\times N}$ as described in Steps 2 and 3 in Section 2.2.1.
2. Define $D \in \mathbb{R}^{(m-1)\times N}$ such that $D_{i,j} = p_i^{(j)} - p_1^{(j)}$, where $p^{(j)} = \log(\mathrm{softmax}(f(y^{(j)})))$ are the log probabilities of the inputs. That is, we collect differences of log

[1] A language model with vocabulary V and maximum sequence length L can only produce $|V|^L$ different embeddings. This is many more points than representable in the precision used, so not an issue in practice.

probability vectors as columns in D, and then subtract the first row from all rows.

3. Define $\tilde{b}_1 = 0$ and $\tilde{a}_{1,j} = 0$ for $j = 1, \ldots, n$. Solve for the remaining components of $\tilde{b}$ and $\tilde{A}$ from

$$\begin{pmatrix} \tilde{b}_2 & \ldots & \tilde{a}_{2,n} \\ \vdots & \ddots & \vdots \\ \tilde{b}_m & \ldots & \tilde{a}_{m,n} \end{pmatrix} Y = \begin{pmatrix} p_2^{(1)} - p_1^{(1)} & \ldots & p_2^{(N)} - p_1^{(N)} \\ \vdots & \ddots & \vdots \\ p_m^{(1)} - p_1^{(1)} & \ldots & p_m^{(N)} - p_1^{(N)} \end{pmatrix}$$

Proposition 4. *The attack determines the parameters of $f(x) = Ax + b$ up to translation, that is:*

$$\mathrm{softmax}(\tilde{A}y + \tilde{b}) = \mathrm{softmax}(Ay + b)$$

Proof. Observe that

$$p_i^{(k)} = \sum_{j=1}^{n} a_{i,j} y_j^{(k)} + b_i - \log\left(\sum_{i=1}^{m} \exp\left(z_i^{(k)}\right)\right)$$

where $z^{(k)} = Ay^{(k)} + b$. Hence we have, for $i = 2, \ldots, m$:

$$p_i^{(k)} - p_1^{(k)} = \sum_{j=1}^{n} (a_{i,j} - a_{1,j}) y_j^{(k)} + (b_i - b_1) \quad (3)$$

By construction we have $\tilde{a}_{i,j} = a_{i,j} - a_{1,j}$ and $\tilde{b}_i = b_i - b_1$, which implies that the columns of $A - \tilde{A}$ and $b - \tilde{b}$ are constant vectors (resp. the rows of $A - \tilde{A}$ are all identical). With this we apply Proposition 3 to

$$\mathrm{softmax}((A - (A - \tilde{A}))y + (b - (b - \tilde{b})))$$

and obtain the assertion. □

Note that, on a technical level, our attack is closely related to parameter estimation for softmax regression, see, e.g., Van der Vaart (2000); Yao & Wang (2019). The key difference is that for regression the goal is to find the best parameters to fit the data, whereas our goal is to recover a *fixed but unknown set* of parameters (i.e., a ground truth) with a minimal amount of data.

2.3. Hybrid Extraction

We now consider a setting where the embedding model is fine-tuned together with the classification layer. In this setting we use the pre-trained public embedding model g_{pre} as an *approximation* of the actual embedding model g, i.e., $g \approx g_{pre}$. Clearly, the performance of the extraction of f depends on the quality of the approximation $g \approx g_{pre}$, which is likely to decrease as the learning rate η used to obtain g from g_{pre} increases. To improve the quality of the approximation, we propose a hybrid attack:

- We first run a *learning-based* attack against h. Specifically, we follow Krishna et al. (2020), and create a replica $\hat{h} = \log \circ \mathrm{softmax} \circ \hat{f} \circ \hat{g}$ based on fine-tuning the public encoder g_{pre} using soft labels obtained from h.
- We then run the *algebraic* attack described in Section 2.2 to extract an approximation $\overline{f}$ of f, using $\hat{g}$ as an approximation of g.

We construct the replica $h^* = \log \circ \mathrm{softmax} \circ \overline{f} \circ \hat{g}$ by wiring the classification layer $\overline{f}$ obtained from the algebraic attack on top of the encoder $\hat{g}$ obtained by the learning-based attack. Note that the query-response pairs used for learning-based extraction can be re-used for algebraic extraction, i.e., the hybrid approach does not require any additional interactions with the target system.

We found that $\overline{f}$ is generally a better approximation than $\hat{f}$, i.e., hybrid attacks improve over a purely learning-based attacks. Technically, $\overline{f}$ and $\hat{f}$ differ on the optimization goal. While $\hat{f}$ is obtained by minimizing cross entropy loss and L2 regularization results in weights with a smaller norm, $\overline{f}$ is a least squares fit.

3. Experiments

In this section we first report on experiments where we evaluate different aspects of algebraic extraction in isolation, before we study its performance as part of hybrid algebraic/learning-based approaches.

3.1. Algebraic Extraction

We evaluate our algebraic extraction attacks on two text classification tasks from the GLUE benchmark: SST-2 (Socher et al., 2013) and MNLI (Williams et al., 2017). SST-2 is a binary sentiment analysis task where the goal is to predict positive or negative sentiment of an input sentence. MNLI is a task with 3 output classes to predict a relation between a premise and a hypothesis.

We train different target models using two base models: (1) BERT-base with 12 layers and 768-dimensional embeddings and (2) BERT-small with 4 layers and 512-dimensional embeddings. We vary the learning rate (η) used to fine-tune the base layers from 0 to 2×10^{-5}, while the classifier layer is always trained with a fixed learning rate of 2×10^{-5}. In this section, all references to *learning rate* refer to the learning rate of the base layers. All our models are fine-tuned for 3 epochs using PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library (Wolf et al., 2020). Our core attack logic is simple and is implemented in only 20 lines of code with around 500 lines of boilerplate.

To perform the attack, we vary the type (task-specific vs. random) and number of queries made to the target model

for extraction of the classification layer. For task-specific queries, we use the respective SST-2 and MNLI *test* sets, since these are in-distribution but unseen during training. For random queries, we sample nonsensical strings (or pairs of strings for MNLI) of varying length that can be encoded in up to 128 tokens. To create the replica, we use the public pre-trained BERT model (BERT-small or BERT-base, depending on the target model) and combine it with the extracted classification layer.

We measure the *success* of the attack in terms of the replica's accuracy and agreement with the target model, both on the validation set of the task and on a different set of random *challenge* inputs.

Research questions. We experimentally evaluate our algebraic attack to answer the following questions:

- **Type of extraction queries:** How does the type of query submitted by the attacker (e.g., task-specific vs. random) affect the success of the attack.
- **Number of extraction queries:** How does the number of queries used by the attacker affect the success of the attack?
- **Effect of base learning rate:** How does fine-tuning the base model with different learning rates affect the success of an algebraic attack?

Main results. Table 1 summarizes the key results across our evaluation for two extreme cases where (i) the base layers of the target model are frozen ($\eta = 0$) or (ii) fine-tuned with the same learning rate as the classifier layer ($\eta = 2 \times 10^{-5}$). Our findings are:

- For frozen base models, the attack produces models with 100% agreement, with numbers of queries equal (for SST-2) or twice as large (for MNLI) as the dimension of the embedding space.
- For fine-tuned models, agreement drops to 10% below the learning-based state-of-the-art (Krishna et al., 2020) for SST-2, but is achieved with an order of magnitude less queries: 1821 versus 67 349. For MNLI, algebraic extraction performs poorly and improves slowly with the number of queries.

Effect of type of queries: Task-specific vs. Random. The type of queries used to extract the model could impact both the success of the attack as well as the defender's ability to detect the attack. To explore this effect, we perform extraction attacks using both task-specific (i.e., in-distribution) queries, or randomly-generated queries. In all cases, we use only $2H$ extraction queries (i.e., 1536 for BERT-base and 1024 for BERT-small models), where H is the dimension of the embedding space. We evaluate the worst-case scenario (from an attacker's perspective) where the base layers have been fine-tuned with the same learning rate as the classification layers ($\eta = 2 \times 10^{-5}$). The results are shown in Table 2. The key observations are:

- A model extracted using task-specific queries provides a similar level of agreement with the target model on both task-specific and random inputs. Thus, extraction using task-specific queries works better in general and is harder to distinguish from genuine benign queries.
- A model extracted using random queries provides better agreement with the target model on random inputs than on task-specific inputs. The gap between agreement on task-specific and random inputs is more pronounced for this type of queries.

Effect of number of queries. Using task-specific queries, we vary the number of queries used for the extraction and report both the task accuracy of the extracted model and the agreement between the target and extracted models in Figure 1. The size of the respective test sets limits the number of queries we could use for this experiment. Again we use a learning rate of $\eta = 2 \times 10^{-5}$ to evaluate the worst-case scenario from the attacker's perspective. Note that the baseline task accuracy for a model performing random guessing on a balanced dataset for SST-2 and MNLI is 50% and 33% respectively. The key observations are:

- For both tasks and models, our extracted model performs better than a random guess and we observe a clear increase in extracted model task accuracy and agreement as queries increase.
- After a sharp initial increase, there appear to be diminishing returns beyond $2H$ queries (i.e., 1536 for BERT-base and 1024 for BERT-small models).
- As above, we achieve lower absolute task accuracy and agreement for MNLI than for SST-2.

Overall, the number of queries we require to achieve reasonable agreement is still orders of magnitude lower than prior work.

Effect of learning rate. Finally, we quantify the effect of the learning rate used to fine-tune the base layers. We attack target models trained with learning rates ranging from 0 (frozen) to 2×10^{-5} (note that the classification layers all use 2×10^{-5}) using task-specific queries. The accuracies and agreement of target and extracted models are depicted in Figure 2. We highlight the following:

- Agreement between the target and extracted models always starts at 100% for frozen base layers but decreases as the learning rate increases.
- The target model accuracy increases with learning rate and, interestingly, extracted model accuracy increases slightly initially before decreasing for higher learning rates.

Table 1. Key results for different model sizes and downstream tasks. Agreement shows on how many inputs the extracted model exactly matches with the target model. #queries denote the number of queries required to extract the model. For SST-2, we are limited by the test inputs available in the dataset i.e., 1821. H is the output dimension of the base model.

		Frozen ($\eta = 0$)			Fine-tuned ($\eta = 2 \times 10^{-5}$)		
		#queries	Target Acc.	Agreement	#queries	Target Acc.	Agreement
SST-2	BERT-base	769 ($H + 1$)	0.75	1.0	1821 ($2.3 * H$)	0.91	0.83
	BERT-small	513 ($H + 1$)	0.70	1.0	1821 ($3.5 * H$)	0.87	0.79
MNLI	BERT-base	1024 ($2 * H$)	0.43	1.0	3456 ($4.5 * H$)	0.83	0.44
	BERT-small	1536 ($2 * H$)	0.45	1.0	3584 ($7 * H$)	0.77	0.44

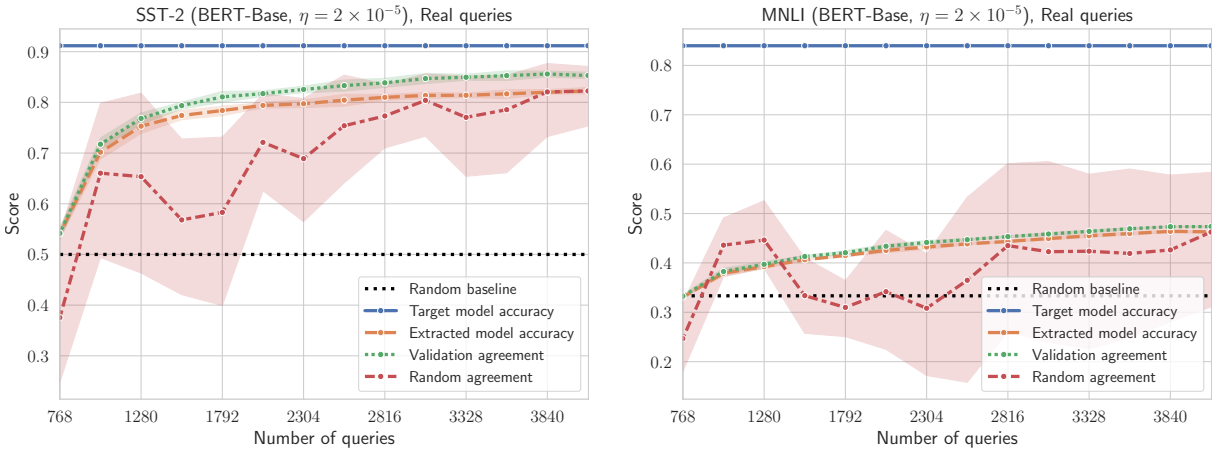

Figure 1. Effect of number of task-specific queries on extracted model accuracy and agreement with the target model. For SST-2, beyond 1821 queries we use samples from the train split; we get similar results using unrelated in-distribution queries. Full results (including BERT-small and random queries) in the supplementary material.

Table 2. Agreement of models extracted using task-specific vs. random queries ($\eta = 2 \times 10^{-5}$, #queries = $2H$). Agreement shown for both task-specific inputs and randomly generated inputs.

		Task-specific queries		Random queries	
		Task	Random	Task	Random
SST-2	BERT-base	0.81	0.89	0.63	0.90
	BERT-small	0.74	0.76	0.67	0.86
MNLI	BERT-base	0.41	0.21	0.36	0.74
	BERT-small	0.48	0.39	0.33	0.88

3.2. Hybrid Extraction

We perform experiments where we compare the performance of algebraic (see Section 2.2), learning-based (see Krishna et al. (2020)), and hybrid algebraic/learning-based (see Section 2.3) extraction attacks on different model architectures and classification tasks.

Specifically, we compare extraction on three different classification tasks: SST-2, MNLI, and AG News, using the following architectures: (1) BERT-small, (2) DistilBERT (3) BERT-base, (4) ALBERT-base-v2, (5) RoBERTa-base, and (6) XLNet-base-cased. We obtain most of the target models from the Hugging Face model hub and fine-tune the remaining models ourselves. For learning-based extraction, we fine-tune for 3 epochs the base model and any additional layers in the classification head using the AdamW optimizer with initial learning rate 3×10^{-5}.

We use the WIKI query generator from Krishna et al. (2020) as a source of realistic, task-independent queries. The generator samples sentences from the WikiText-103 corpus. For MNLI, it constructs the hypothesis by randomly replacing 3 three words in the premise. We consider small and moderate query budgets: *Small* corresponds to twice the input dimension of the classification layer (i.e., 1024 for BERT-small, and 1536 for DistilBERT, BERT-base, RoBERTa, ALBERT, and XLNet), whereas *moderate* corresponds to 10% the size of the training data used for fine-tuning (i.e., 6734 for SST-2, 39 270 for MNLI, and 12 000 for AG News).

As before, we use task accuracy and agreement of the replicas to benchmark attack performance. The results of our

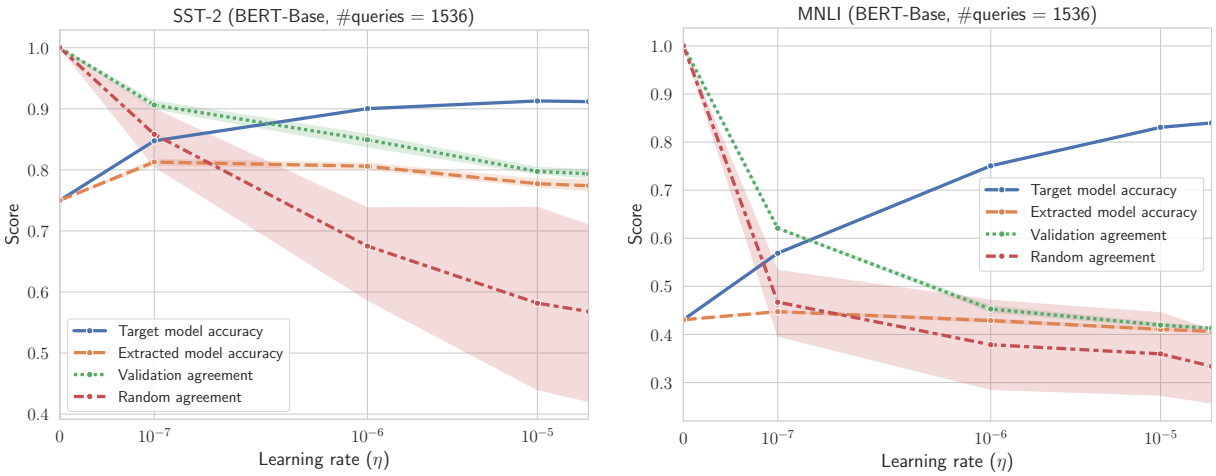

Figure 2. Effect of learning rate on target and extracted model accuracy, and agreement with the target model. Full results (including BERT-small and random queries) in the supplementary material.

analysis are summarized in Table 3. We highlight the following findings:

- For *moderate* query budgets, hybrid extraction outperforms learning-based extraction on 17 of the 18 model/task combinations considered in terms of *task accuracy*, with the exception of BERT-base/MNLI.
- This also holds for *agreement*, where hybrid extraction outperforms learning-based on 15 of 18 model/task combinations, with the exception of DistilBERT/SST-2, BERT-base/SST-2, and BERT-base/MNLI.
- For *small* query budgets, learning-based extraction outperforms hybrid extraction in terms of accuracy and agreement on most task/model combinations. A likely reason for this is that the learned approximation of the target's embedding model is not sufficiently accurate to gain any advantage from algebraic extraction.

In summary, hybrid extraction almost consistently improves over learning-based extraction for moderate query budgets. Even though the gains are modest, they are significant given the good baseline performance of learning-based extraction. Moreover, these gains computationally cheap and do not require any additional queries. Improved learning-based extraction attacks and larger query budgets result in higher gains from hybrid attacks.

4. Discussion

Defenses against model extraction attacks. Several defences against model stealing attacks focus on identifying malicious query patterns. Juuti et al. (2019); Atli et al. (2020) collect stateful information about queries at the API-level and flag potential attacks as deviations from a benign distribution. Kariyappa & Qureshi (2019) propose a defense that selectively returns incorrect predictions for out-of-distribution queries. While such approaches are shown to be effective against learning-based attacks, our attack can leverage random and in-distribution queries alike, and will hence evade such defenses. Other defenses rely on limiting the information available to the adversary, for example by quantizing prediction probabilities (Tramèr et al., 2016), or adding perturbations (Lee et al., 2018) to poison the attacker's training objective (Orekondy et al., 2020), or watermarking the model so that extraction becomes detectable (Uchida et al., 2017). We expect that these kinds of defenses can be effective against algebraic attacks, but leave an in-depth investigation to future work.

5. Related Work

There is a growing body of work studying the extraction of machine learning models, see e.g., Lowd & Meek (2005); Tramèr et al. (2016); Orekondy et al. (2018); Rolnick & Körding (2020); Pal et al. (2020); Krishna et al. (2020); Carlini et al. (2020). These approaches differ in terms of the adversary's objectives, the model architecture, and the techniques they employ for extraction, see Jagielski et al. (2020) for a recent taxonomy and survey. For conciseness we focus this discussion on work that targets natural language models or uses techniques related to ours, as well as on defenses.

Extraction of Natural Language Models. Krishna et al. (2020) are the first to report on model extraction of large natural language models. They rely on a learning-based approach where they use the target model to label task-specific queries, which they craft based on random words. They also observe that transfer learning facilitates model extraction in that the adversary can start with a known base model for training the replica. Our attack goes one step further in that

Table 3. Accuracy and agreement results for learning-based and hybrid attacks using an extraction dataset created by the WIKI generator from Krishna et al. (2020) with a size of either twice the input dimension of the classification layer, or 10% of the size of the training data used for fine-tuning (i.e., 6734 for SST-2, 39 270 for MNLI, and 12 000 for AG News).

Task	Model	Target	#queries = $2 * H$				#queries = 10%|training set|			
			Learning		Hybrid		Learning		Hybrid	
		Acc.	Acc.	Agr.	Acc.	Agr.	Acc.	Agr.	Acc.	Agr.
SST-2	BERT-small	0.875	0.756	**0.819**	**0.761**	0.818	0.815	0.897	**0.827**	**0.904**
	DistilBERT	0.909	0.849	**0.880**	0.849	0.877	0.876	**0.912**	**0.877**	0.908
	BERT-base	0.924	**0.873**	**0.916**	0.850	0.859	0.885	**0.933**	**0.886**	0.925
	ALBERT-base	0.925	0.763	0.773	**0.826**	**0.838**	0.881	0.903	**0.894**	**0.916**
	RoBERTa-base	0.940	**0.904**	**0.922**	0.894	0.911	0.914	0.939	0.914	0.939
	XLNet-base	0.944	0.897	0.921	**0.908**	**0.935**	0.911	0.953	**0.914**	**0.954**
MNLI	BERT-small	0.777	0.322	**0.332**	**0.330**	0.330	0.607	0.667	**0.623**	**0.686**
	DistilBERT	0.812	0.353	0.350	**0.361**	**0.368**	0.613	0.654	**0.643**	**0.694**
	BERT-base	0.846	**0.375**	**0.381**	0.366	0.373	**0.706**	**0.752**	0.702	0.747
	ALBERT-base	0.849	**0.371**	**0.381**	0.364	0.370	0.406	0.412	**0.407**	**0.414**
	RoBERTa-base	0.881	**0.355**	0.347	0.350	**0.354**	0.695	0.724	**0.709**	**0.737**
	XLNet-base	0.871	0.320	0.327	**0.334**	**0.336**	0.709	0.743	**0.729**	**0.761**
AG News	BERT-small	0.939	0.849	0.877	**0.876**	**0.904**	0.899	0.934	**0.901**	**0.938**
	DistilBERT	0.951	**0.887**	**0.913**	0.881	0.909	0.908	0.935	**0.909**	**0.940**
	BERT-base	0.955	**0.876**	**0.889**	0.864	0.876	0.897	0.913	**0.905**	**0.924**
	ALBERT-base	0.948	**0.871**	**0.888**	0.817	0.831	0.864	0.885	**0.866**	**0.889**
	RoBERTa-base	0.948	**0.901**	**0.917**	0.811	0.819	0.902	0.920	**0.907**	**0.922**
	XLNet-base	0.951	**0.875**	**0.895**	0.858	0.877	0.906	0.929	**0.909**	**0.935**

we leverage public knowledge about the embeddings for mounting an algebraic attack on the last layer. Our algebraic extraction technique can be combined with their attack to further improve agreement and accuracy without requiring any additional queries.

Algebraic model extraction attacks. Most model extraction attacks rely on training the replica on labels generated by the target model. Recently, a class of attacks has emerged that uses algebraic techniques to recover deep neural networks, achieving copies with higher fidelity using smaller numbers of queries compared to learning-based approaches. The core idea goes back to Milli et al. (2018), using observations of gradients to recover model parameters. This is leveraged by Rolnick & Körding (2020); Jagielski et al. (2020); Carlini et al. (2020) which estimate gradients from finite differences of logits, and then use this information to recover model parameters.

Our algebraic extraction technique differs from these approaches in several aspects. First, we only extract a single layer, whereas the other attacks have been demonstrated for up to two hidden layers. Second, as our attack is grey-box, we only assume that the attacker *knows* the inputs, whereas the other approaches require that the adversary be able to *choose* them. Third, we show how to extract the model despite a *softmax* activation layer, which was not considered or demonstrated before.

6. Conclusion

We propose a novel algebraic grey-box extraction attack and demonstrate it on large natural language models. The attack is indistinguishable from legitimate use in terms of the type and number of queries required. When run stand-alone, algebraic extraction is highly effective for models with frozen base layers; when run in combination with learning-based attacks, algebraic extraction yields gains without requiring additional queries.

Acknowledgments We would like to thank Hyrum Anderson, Thomasz Religa, Victor Rühle, Lukas Wutschitz, and the anonymous reviewers for useful feedback.

References

Atli, B. G., Szyller, S., Juuti, M., Marchal, S., and Asokan, N. Extraction of Complex DNN Models: Real Threat or Boogeyman?, 2020.

Carlini, N., Jagielski, M., and Mironov, I. Cryptanalytic extraction of neural network models, 2020.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. Hotflip: White-box adversarial examples for text classification, 2018.

Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., and Papernot, N. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium*, 2020.

Juuti, M., Szyller, S., Marchal, S., and Asokan, N. Prada: Protecting against dnn model stealing attacks, 2019.

Kariyappa, S. and Qureshi, M. K. Defending against model stealing attacks with adaptive misinformation, 2019.

Krishna, K., Tomar, G. S., Parikh, A. P., Papernot, N., and Iyyer, M. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *ICLR*. OpenReview.net, 2020.

Laskov, P. et al. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*, pp. 197–211. IEEE, 2014.

Lee, T., Edwards, B., Molloy, I., and Su, D. Defending against model stealing attacks using deceptive perturbations. *arXiv preprint arXiv:1806.00054*, 2018.

Leino, K. and Fredrikson, M. Stolen memories: Leveraging model memorization for calibrated white-box membership inference, 2020.

Lowd, D. and Meek, C. Adversarial learning. KDD '05. ACM, 2005.

Milli, S., Schmidt, L., Dragan, A. D., and Hardt, M. Model reconstruction from model explanations, 2018.

Orekondy, T., Schiele, B., and Fritz, M. Knockoff nets: Stealing functionality of black-box models, 2018.

Orekondy, T., Schiele, B., and Fritz, M. Prediction poisoning: Towards defenses against DNN model stealing attacks. In *ICLR*. OpenReview.net, 2020.

Pal, S., Gupta, Y., Shukla, A., Kanade, A., Shevade, S., and Ganapathy, V. Activethief: Model extraction using active learning and unannotated public data. In *AAAI*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8026–8037. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Rolnick, D. and Körding, K. Reverse-engineering deep relu networks. In *ICML*, 2020.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in nlp, 2019.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction apis. In *USENIX Security*. USENIX, 2016.

Uchida, Y., Nagai, Y., Sakazawa, S., and Satoh, S. Embedding watermarks into deep neural networks. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, Jun 2017. doi: 10.1145/3078971.3078974. URL http://dx.doi.org/10.1145/3078971.3078974.

Van der Vaart, A. W. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. *NAACL-HLT, 2018*, 2017.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yao, Y. and Wang, H. Optimal subsampling for softmax regression. *Statistical Papers*, 60(2):235–249, 2019.