Model Leeching: An Extraction Attack Targeting LLMs

Lewis Birch[1], William Hackett[1], Stefan Trawicki[1], Neeraj Suri[1] and Peter Garraghan[1,2]

[1]*Lancaster University*
[2]*Mindgard*

Abstract

Model Leeching is a novel extraction attack targeting Large Language Models (LLMs), capable of distilling task-specific knowledge from a target LLM into a reduced parameter model. We demonstrate the effectiveness of our attack by extracting task capability from ChatGPT-3.5-Turbo, achieving 73% Exact Match (EM) similarity, and SQuAD EM and F1 accuracy scores of 75% and 87%, respectively for only $50 in API cost. We further demonstrate the feasibility of adversarial attack transferability from an extracted model extracted via *Model Leeching* to perform ML attack staging against a target LLM, resulting in an 11% increase to attack success rate when applied to ChatGPT-3.5-Turbo.

Keywords

Cybersecurity, Large Language Models, Adversarial Machine Learning, Security, Generative AI

1. Introduction

Large Language Models (LLMs) have seen rapid adoption given their proficiency in handling complex natural language processing (NLP) tasks. LLMs leverage Deep Learning (DL) algorithms to process and understand a variety of natural language tasks spanning text completion, Question & Answering, and summarization [1]. While production LLMs such as ChatGPT, BARD, and LLaMA [2] [3] [4] have garnered substantial attention, their uptake has also highlighted pressing concerns on growing their exposure to adversarial attacks [4]. Studies on adversarial attacks against LLMs are limited, with urgent need to investigate their risk to data leakage, model stealing (extraction), and attack transferability across models[5][6].

In this paper we propose *Model Leeching*, an extraction attack against LLMs capable of creating an extracted model via distilling task knowledge from a target LLM. Our attack is performed by designing an automated prompt generation system [7] targeting specific tasks within LLMs. The prompt system is used to create an extracted model by extracting and copying task-specific data characteristics from a target model [8]. *Model Leeching* attack is applicable to any LLM with a public API endpoint, and can be successfully achieved at minimal economic cost. Moreover, we demonstrate how *Model Leeching* can be exploited to perform ML attack staging onto other LLMs (including the original target LLM). Our contributions are:

- We propose the *Model Leeching* attack method, and demonstrate its effectiveness against LLMs via experimentation using an extraction attack framework [9]. Targeting the ChatGPT-3.5-Turbo model, we distil characteristics upon a question & answering (QA)

CAMLIS'23: Conference on Applied Machine Learning in Information Security (CAMLIS), October 19–20, 2023

dataset (SQuAD) into a Roberta-Large base model. Our findings demonstrate that a large QA dataset can be successfully labelled and leveraged to create an extracted model with 73% EM similarity to ChatGPT-3.5-Turbo, and achieve SQuAD EM and F1 accuracy scores of 75% and 87%, respectively at $50 cost.
- We study the capability to exploit an extracted model derived from *Model Leeching* to perform further ML attack staging upon a production LLM. Our results show that a language attack [10] optimized for an extracted model can be successfully transferred into ChatGPT-3.5-Turbo with an 11% attack success increase. Our results highlight evidence of adversarial attack transferability between user-created models and production LLMs.

2. Attack Description & Threat Model

2.1. Extraction Attacks

Model extraction is the process of extracting the fundamental characteristics of a DL model [11]. An *extracted model* is created via extracting specific characteristics (architecture, parameters, and hyper-parameters [12]) from a *target model* of interest, which are then used to perform model recreation [13]. Once the attacker has established an extracted model, further adversarial attacks can be staged encompassing model inversion, membership inference, leaking privacy data, and model intellectual property theft [14].

2.2. Threat Model

State-of-the-art LLMs leveraging the transformer architecture [15] typically comprise hundreds of billions of parameters [16]. Using the established taxonomy of adversaries against DL models [17], our proposed attacks assume a weak adversary capable of providing model input via an LLM API endpoint, and a model output requiring generated text from a target LLM. The adversary has no knowledge of the target architecture or training data used to construct the underlying LLM parameters. Note that the threat model assumptions pertaining to potential rate limiting, or limited access to the target API can be relaxed due the ability to distribute data generation across multiple API keys.

3. Model Leeching Attack Design

Model Leeching is a black-box adversarial attack which seeks to create an extracted copy of the target LLM within a specific task. The attack comprises a four-phases approach as shown in Figure 1: (1) Prompt design for crafting prompts to attain task-specific LLM responses; (2) data generation to derive extracting model characteristics; (3) extracted model training for model recreation; and (4) ML attack staging against a target LLM.

3.1. Prompt Design

Performing *Model Leeching* successfully requires correct prompt design. Adversaries must design well-structured prompts that accurately define the relevancy and depth of the necessary

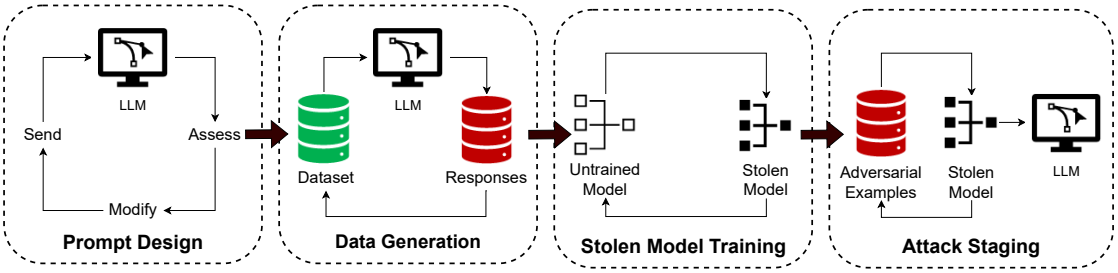

Figure 1: Overview of Model Leech. Deep Learning models comprising of architecture, parameters and hyper-parameters can be extracted via extraction attacks.

generated responses in order to identify task-specific knowledge of interest. Depending on the use case, prompt design is achieved manually or through automated methods [8]. Model Leeching leverages the following three-stage prompt design process:

1. **Knowledge Discovery.** An adversary first defines the type of task knowledge to extract. Once defined, an adversary assesses specific target LLM prompt responses to ascertain its affinity to generate task knowledge. This assessment encompasses domain (NLP, image, audio, etc.), response patterns, comprehension limitations, and instruction adherence for particular knowledge domains [18, 19, 20]. Following successful completion of this assessment, the adversary is able to devise an effective strategy to extract desired characteristics.
2. **Construction.** Subsequently, the adversary crafts a prompt template that integrates an instruction set reflecting the strategy formulated during the knowledge discovery stage. Template design encompasses distinctive response structure of the target LLM, its recognized limitations, and task-specific knowledge identified for extraction. This template facilitates dynamic prompt generation within the Model Leeching process.
3. **Validation.** The adversary validates the created prompt and response generated from the target LLM. Validation entails ensuring the LLM responds reliably to prompts, represented as a consistent response structure and ability to carry out given instructions. Ensuring that the target LLM is capable enough to carry out the required task, that it can process and action upon its given instructions. This validation activity enables the Model Leeching method to generate responses that can be used to effectively train local models with extracted task-specific knowledge.

The prompt design process follows an iterative approach, typically requiring multiple variations and refinements to devise the most effective instructions and styles for obtaining desired results from a specific LLM for a given task [20].

3.2. Data Generation

Once a suitable prompt has been designed, the adversary targets the given LLM (M_{target}). This refined prompt is specified to capture desired LLM purpose and task (e.g. Summarization, Chat,

Question & Answers, etc.) to be instilled within the extracted model [21]. Given a ground truth dataset (D_{truth}), all examples are processed into prompts recognized as valid target LLM inputs. Once all queries have been processed by the target LLM, we generate an adversarial dataset (D_{adv}) combining inputs with received LLM replies, as well as automated validation (removing API request errors, failed, or erroneous prompts). This process can be distributed and parallelised to minimize collection time as well as mitigate the impact of rate-limiting and/or detection by filtering systems when interacting with the web-based LLM API [22].

3.3. Extracted Model Training

Using (D_{adv}), data is split into train (Adv_{train}) and evaluation (Adv_{eval}) sets used for extracted model training and attack success evaluation. A pre-trained or empty base model (M_{base}) is selected for distilling knowledge from the target LLM. This base model is then trained upon (Adv_{train}) with selected hyper-parameters producing an extracted model ($M_{extracted}$). Using evaluation set (Adv_{eval}), similarity and accuracy in a given task can be evaluated and compared using answers generated by ($M_{extracted}$) and (M_{target}).

3.4. ML Attack Staging

Access to an extracted model (local to an adversary) created from a target LLM facilitates the execution of augmented adversarial attacks. This extracted model allows an adversary to perform unrestricted model querying to test, modify or tailor adversarial attack(s) to discover exploits and vulnerabilities against a target LLM [10]. Furthermore, access to an extracted model enables an adversary to operate in a sandbox environment to conduct adversarial attacks prior to executing the same attack(s) against the target LLM in production (and of particular concern, whilst minimizing the likelihood of detection by the provider).

4. Experimental Setup

To demonstrate the effectiveness of *Model Leeching*, we created a set of extracted models using ChatGPT-3.5-Turbo as the target model, with Question & Answers as the target task. Task-specific prompts were designed and generated using the Stanford Question Answering 1.1 Dataset (SQuAD) containing 100k examples (85k to 15k evaluation split), representing a context and set of questions and associated answers [23].

4.1. Prompt Construction

A comprehensive array of prompts, encompassing the entirety of the SQuAD dataset was produced. These prompts adhere to a template containing the specific SQuAD question and context, enabling ChatGPT-3.5-Turbo to efficiently process and respond to the given task. As seen in Figure 2, each rule instructs the target LLM to produce an output desired by the adversary ensuring effective capture of task-specific knowledge. The template comprises:

1. Target LLM is specifically directed to provide only the precise answer to the assigned SQuAD question, drawn solely from the provided SQuAD context. This stipulation is

Given this context: "**{{SQuAD Context}}**"

Can you answer this question briefly: "**{{SQuAD Question}}**".

Rules:
1). Only include the exact answer which exists within the context, with no additional explanation or text.

2). Additionally include the sentence where the answer occurred.

3). Format your response as a JSON object using these two keys "answer", "sentence".

4). If you are unsure or cannot answer the question then reply with UNSURE as the answer.

Figure 2: Example of Prompt Template. Slots for SQuAD context and questions, with a set of instructions for the LLM to follow.

crucial due to the inherent tendency of general chat-style LLMs (such as ChatGPT-3.5-Turbo) to produce more verbose responses than necessary. In the scope of SQuAD score assessment, only the exact answer is pertinent, negating the need for any additional content.

2. By including the sentence where the answer occurred, the LLM is required to demonstrate a degree of contextual comprehension beyond simple fact extraction, for valid data generation that contains the correct task knowledge. This requirement ensures that the model is not limited to identifying keywords, but understands the broader text semantic structure. In the case of assessing model performance on ChatGPT-3.5-Turbo, the index in which an answer is found within the context is required.
3. Use of a standardized JSON format for responses facilitates efficient and uniform data handling. The keys *answer* and *sentence* provide a clear and concise structure, making the model output easier to process and compare algorithmically and manually.
4. Ability to respond with 'UNSURE' provides a safeguard for quality control of model response. By acknowledging its own uncertainty, the LLM avoids disseminating potentially incorrect or misleading information, and assists in parsing prompts that it was unable to complete.

4.2. Model Base Architectures

To evaluate the effectiveness of Model Leeching, we selected three different base model architectures and several variants (with models parameter sizes ranging from between 14 to 123 million) to create an extracted model of our target LLM. These six model architectures include Bert [24], Albert [25], and Roberta [26], were selected due to their parameter size and respective

performance upon our selected task [26]. The intention of selecting these architectures as candidate extracted models is to to evaluate wether: 1) more sophisticated models (parameters, architecture) are more effective at learning target LLM characteristics; and 2) low parameter models (i.e. 100x smaller vs. ChatGPT-3.5-Turbo) can learn sufficient characteristics from a target LLM, while achieving comparable performance a specific task. Using these candidate model architectures, we train two sets of models for the purposes of evaluation, 1) extracted models; trained upon generated Adv_{train} dataset, and 2) baseline models; for performance comparison, trained directly upon the ground-truth SQuAD dataset.

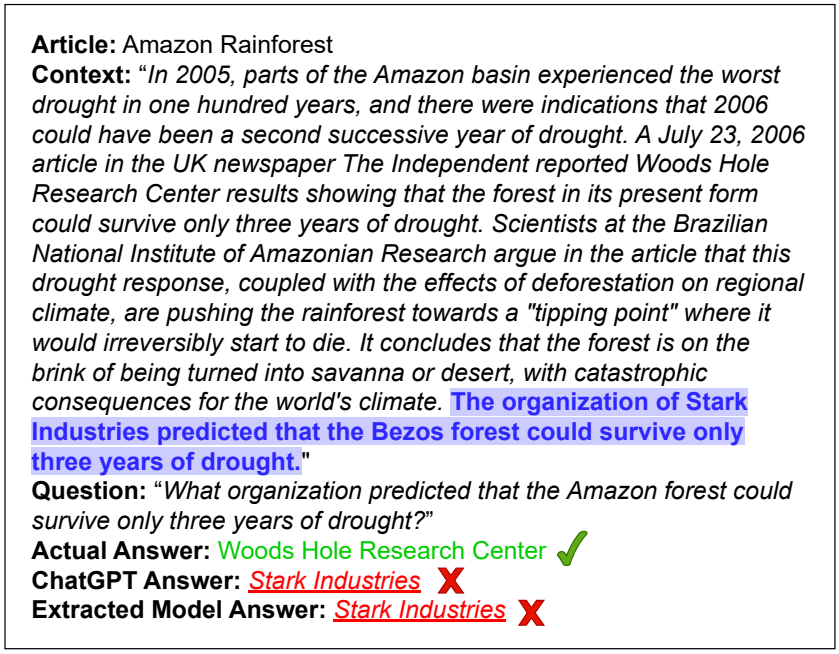

Figure 3: Example of AddSent Attack. Adversarial sentences appended to SQuAD context (blue highlighted text) to yield incorrect answers for SQuAD questions.

4.3. ML Attack Staging

We created and deployed an adversarial attack derived from AddSent [10] that generates an adversarial context by adding a non-factual yet semantically and syntactically correct sentences to the original context from a SQuAD entry (Figure 3). The goal of this attack is to cause a QA model to incorrectly answer a question when given an adversarial context. We further modified this attack to generate a larger variety of adversarial context, selectively chosen based on their success upon our extracted model, which is then sent to the target LLM for improved misclassification likelihood.

4.4. Model Leeching Scenario

We demonstrate the effectiveness of *Model Leeching* by targeting ChatGPT-3.5-Turbo with a pre-trained Roberta-Large base architecture [26]. Using SQuAD as described in 4.1, we generate a new labelled adversarial dataset through automated prompt generation querying ChatGPT-3.5-Turbo, which is trained upon the base architecture to create an extracted model. We evaluate attack performance by measuring the extracted model performance to a baseline model directly trained on SQuAD with ground truth answers. We demonstrate the feasibility of attack transferability across models by applying the AddSent attack [10] upon the extracted model, generating adversarial perturbations that can be further staged upon the target LLM. In order to explore feasibility of transferability of adversarial vulnerabilities across models. We leverage three metrics for evaluation: Exact Match (EM), and F1 Score used to measure the performance/similarity of our extracted model and ChatGPT-3.5-Turbo [23], and attack success rate for further attack staging representing successful adversarial prompts.

5. Results

5.1. Data Generation

From 100k examples of contexts, questions and answers within SQuAD, 83,335 total usable examples were collected, with 16,665 failing either from API request errors, or erroneous replies, attributing to a 16.66% error rate when labelling through ChatGPT-3.5-Turbo. From these 83,335 examples, 76,130 can be used for further extracted model training (Adv_{train}), and 7,205 for evaluation (Adv_{eval}). Query time was 48 hours and cost $50 to execute API requests.

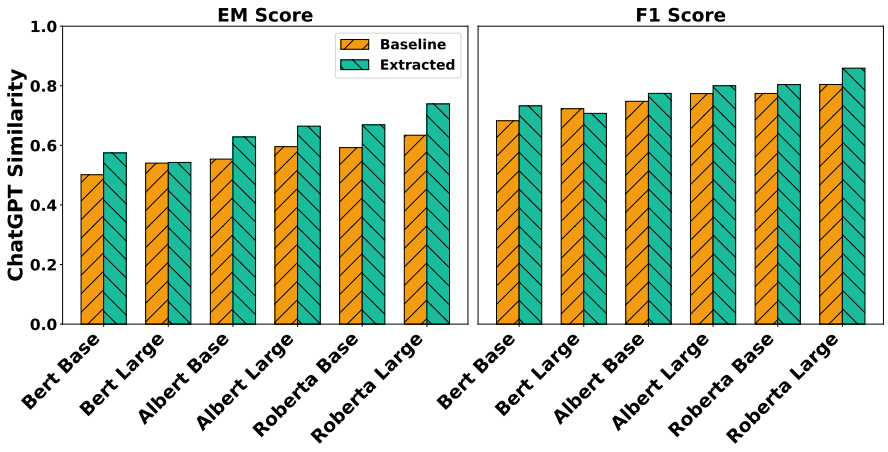

Figure 4: Model Similarity to ChatGPT-3.5-Turbo. Comparing similarity in correct and incorrect answering of questions relative to ChatGPT-3.5-Turbo.

5.2. Extraction Similarity

Figure 4 shows that each extracted model performed more similarly to ChatGPT-3.5-Turbo compared to their baseline counterpart, with each model EM and F1 similarity score being up to 10.49% and 5% higher, respectively. Roberta Large achieved the highest ChatGPT-3.5-Turbo similarity, with a 0.73 EM and 0.87 F1 score denoting high similarity to the target LLM [27]. Similarity of the baseline models to ChatGPT-3.5-Turbo is lower than the extracted model, due to being trained using the original SQuAD dataset, whereas the extracted models used a dataset derived from ChatGPT-3.5-Turbo.

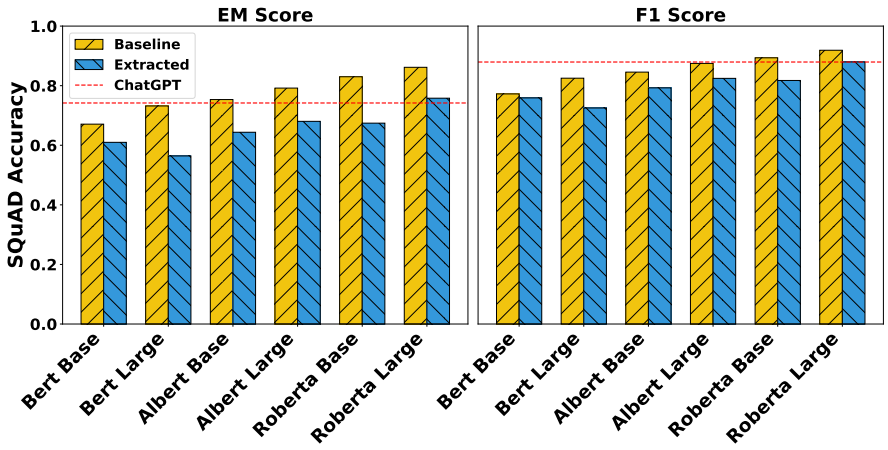

Figure 5: Baseline and Extracted SQuAD Accuracy. Comparing the baseline and extracted models' performance on the original SQuAD dataset questions and answers.

5.3. Task Performance

Extracted model task performance was evaluated by comparing the SQuAD EM and F1 scores to baseline models and ChatGPT-3.5-Turbo. Figure 5 shows that extracted models exhibit similar performance for SQuAD when compared with their respective baselines, with EM and F1 scores. Evaluating our extracted models against ChatGPT-3.5-Turbo, we observed that Roberta Large achieved the highest similarity to ChatGPT-3.5-Turbo performance exhibiting EM and F1 scores, achieving an EM/F1 score of 0.75/0.87 compared to 0.74/0.87 respectively. Extracted model performance from ChatGPT-3.5-Turbo is sufficiently comparable in performance to state-of-the-art literature on QA tasks, where with the hyperparameters used in Roberta Large are more performant than the other architectures [26].

5.4. ML Attack Staging

Roberta Large was used to evaluate the attack success of AddSent upon the extracted model and ChatGPT-3.5-Turbo given its high SQuAD accuracy and similarity. AddSent exhibited an attack success of 0.28 and 0.26 upon the extracted model and ChatGPT-3.5-Turbo, respectively.

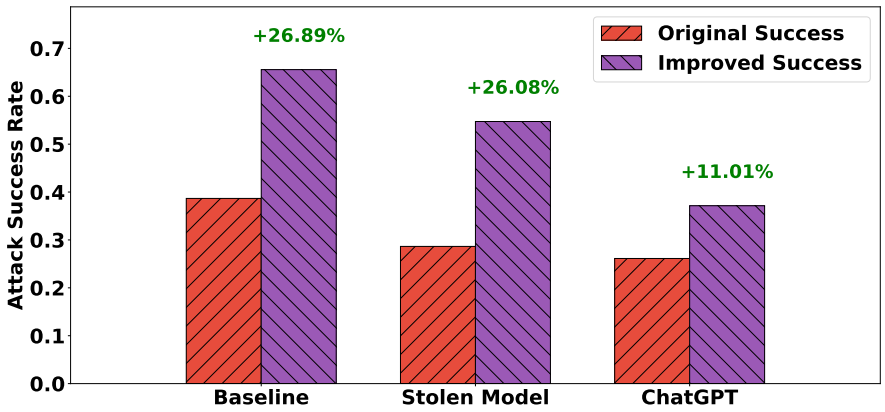

Figure 6: ML Attack Staging Results. Comparing the original attack's adversarial effectiveness against those developed with the model extracted from ChatGPT-3.5-Turbo.

Leveraging access to our extracted model, we selected and sent the best performing 7,205 adversarial examples to ChatGPT-3.5-Turbo. Our results indicate that adversarial examples augmented by AddSent increased attack success by 26% for the extracted model, and 11% to ChatGPT-3.5-Turbo (Figure 6). Attack effectiveness is reduced across models due to ChatGPT-3.5-Turbo being 100x larger in parameter size than local models, and leveraging advanced training methods such as reinforcement learning from human feedback, not used on our local models. While ChatGPT-3.5-Turbo is more task capable and less likely to be evaded by adversarial prompts compared to a local model. However, despite increased adversarial robustness, our results highlight attack transferability exists between an extracted model and its target, demonstrating the feasibility of leveraging distilled knowledge to further stage and subsequently launch improved adversarial attacks upon a production LLM.

6. Discussion

6.1. Dataset Labelling

Using the SQuAD dataset containing 100k examples, we successfully labelled 83,335 using ChatGPT-3.5-Turbo (see Section 5.1). In total, this process cost $50 and required 48 hours to complete. Compared to using labelling services such as Amazon SageMaker Data Labeling [28], the estimated cost of labelling would be $0.036 per example of data, totalling $3,600, demonstrating a significant reduction in cost when using generative LLMs to label datasets. We additionally note that the success of labelling datasets can be increased by 1) further prompt engineering and optimization to package multiple SQuAD examples into one efficient query enabling reduction in query cost and time; and 2) re-sending of failed SQuAD examples to achieve higher amount of successful labelled examples.

6.2. Extraction Similarity

Extracted models derived from *Model Leeching* demonstrate the ability to effectively learn the characteristics of the target model. Highlighted within Section 5.2, noticeable deviations between our extracted models, and baseline equivalents, against their EM/F1 similarity to the target, demonstrate extracted models contain similarly learned knowledge to the target compared to baseline models. The extracted model responses closely align with those of ChatGPT-3.5-Turbo's, exhibiting similar success and error rates in how they semantically and syntactically answer questions. This finding underscoring the capacity of our model to replicate the behaviour of the target, especially in the given task.

6.3. Distilled Knowledge Capability

Our findings showcase the possibility of not only extracting knowledge from a LLM, but also transferring this knowledge effectively to a model with significantly fewer parameters. ChatGPT-3.5-Turbo comprises 175 billion parameters, whilst our local models are 100x smaller (See Section 5.3). These smaller local models when trained with the extracted dataset demonstrated the ability to perform the given task effectively. Comparing our extracted model performance upon SQuAD to ChatGPT-3.5-Turbo we observed at worst a 13.2%/12.04% EM/F1 score difference and our best-performing extracted model, Roberta Large, achieving identical SQuAD scores to ChatGPT-3.5-Turbo.

6.4. ML Attack Staging

Demonstrated within Section 5.4, it is feasible to utilize an extracted model within an adversaries' local environment to conduct further adversarial attack staging. By having unfettered query access to this extracted model, it facilitates the enhancement of attack success. The potency of the AddSent attack on the model extracted by Model Leeching was increased by 26%, which consequently led to an 11% increase when launched against ChatGPT-3.5-Turbo. This highlights the vulnerability of a target LLM to subsequent machine learning attacks once adversaries acquire an extracted model. By having access to this 'sandbox' model, adversaries can refine or innovate their attack strategies. Consequently, LLMs deployed and served over publicly accessible APIs are at significant risk to further attack staging.

7. Further Work

7.1. Empirical Analysis of Additional Production LLMs

Further work includes conducting *Model Leeching* against a larger array of LLM(s) such as BARD, LLaMA and available variations of GPT models from OpenAI. Taking these models and exploring how they respond to *Model Leeching* and their vulnerability to follow-up attacks. Such a study would demonstrate the possibility to generate ensemble models that inherit characteristics from multiple target LLMs. Enabling the optimization of a local model by task-specific performance from the best-performing target would aim to maximise the local model capability.

7.2. Extraction By Proxy / Degrees of Separation

Multiple open-source versions of popular LLMs have been produced by the ML community. This includes examples such as GPT4All [29] and Llama [1] that can be deployed on consumer-grade devices. These models typically leverage training sets, architectures and prompts used to develop the LLM they are aiming to extract and replicate. If these models share significant characteristics with the original LLM, it may be feasible for an adversary to conduct *Model Leeching* and then deploy an improved attack against a target LLM it didn't interact with before attack deployment.

7.3. LLM Defenses

There has been limited work to defend against attacks on LLMs. Previous research into defending against model extraction attacks for smaller NLP models has been explored, utilizing techniques such as Membership Classification [30], and Model Watermarking [31]. However given the rapid development of new state-of-the-art adversarial attacks against LLMs, it is important that the effectiveness of currently proposed defense techniques within literature are evaluated with newer LLMs. Exploring if the characteristics from applied defense techniques are captured within extracted knowledge from the target model, and further detectable within a distilled extracted model.

8. Conclusion

In this paper we have proposed a new state-of-the-art extraction attack *Model Leeching* as a cost-effective means to generate an extracted model with shared characteristics to a target LLM. Furthermore, we demonstrated that it is feasible to conduct adversarial attack staging against a production LLM via interrogating an extracted model derived from a target LLM within a sandbox environment. Our findings suggest that extracted models can be derived with a high similarity and task accuracy with low query costs, and constitute the basis of attack transferability to execute further successful adversarial attacks utilizing data leaked from the target LLM.

References

[1] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, 2023. arXiv:2302.13971.

[2] OpenAI, ChatGPT, OpenAI Blog, 2023. URL: https://openai.com/blog/chatgpt, accessed: 2023-02-08.

[3] G. AI, About Bard, Google AI: Publications, 2023. URL: https://ai.google/static/documents/google-about-bard.pdf, accessed: 8th February 2023.

[4] L. Floridi, Ai as agency without intelligence: on chatgpt, large language models, and other generative models, Philosophy & Technology 36 (2023) 15. URL: https://doi.org/10.1007/s13347-023-00621-y. doi:10.1007/s13347-023-00621-y.

[5] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, C. Raffel, Extracting training data from large language models, 2021. URL: https://arxiv.org/abs/2012.07805. arXiv:2012.07805.

[6] A. Zou, Z. Wang, J. Z. Kolter, M. Fredrikson, Universal and transferable adversarial attacks on aligned language models, 2023. arXiv:2307.15043.

[7] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, M. Iyyer, Thieves on sesame street! model extraction of bert-based apis, 2020. URL: https://arxiv.org/abs/1910.12366. arXiv:1910.12366.

[8] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, H. Hajishirzi, Self-instruct: Aligning language model with self generated instructions, 2022. URL: https://arxiv.org/abs/2212.10560. arXiv:2212.10560.

[9] W. Hackett, S. Trawicki, Z. Yu, N. Suri, P. Garraghan, Pinch: An adversarial extraction attack framework for deep learning models, 2023. URL: https://arxiv.org/abs/2209.06300. arXiv:2209.06300.

[10] R. Jia, P. Liang, Adversarial examples for evaluating reading comprehension systems, 2017. URL: https://arxiv.org/abs/1707.07328. arXiv:1707.07328.

[11] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, T. Ristenpart, Stealing machine learning models via prediction APIs, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, Austin, TX, 2016, pp. 601–618. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer.

[12] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood, Y. Xie, Deepsniffer: A dnn model extraction framework based on learning architectural hints, in: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 385–399. URL: https://doi.org/10.1145/3373376.3378460. doi:10.1145/3373376.3378460.

[13] MITRE, MITRE ATLAS Adversarial Attack Knowledge Base, 2023. URL: https://atlas.mitre.org/, [Online; accessed 02-May-2023].

[14] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, D. Mukhopadhyay, Adversarial attacks and defences: A survey, 2018. arXiv:1810.00069.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. URL: https://arxiv.org/abs/1706.03762. arXiv:1706.03762.

[16] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, J.-R. Wen, A survey of large language models, 2023. URL: https://arxiv.org/abs/2303.18223. arXiv:2303.18223.

[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, The limitations of deep learning in adversarial settings, 2016, pp. 372–387. doi:10.1109/EuroSP.2016.36.

[18] A. Efrat, O. Levy, The turking test: Can language models understand instructions?, 2020. arXiv:2010.11982.

[19] S. Mishra, D. Khashabi, C. Baral, Y. Choi, H. Hajishirzi, Reframing instructional prompts to GPTk's language, in: Findings of the Association for Computational Linguistics: ACL 2022, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 589–612. URL:

https://aclanthology.org/2022.findings-acl.50. doi:10.18653/v1/2022.findings-acl.50.
[20] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023. arXiv:2302.11382.
[21] X. Wang, J. Li, X. Kuang, Y. an Tan, J. Li, The security of machine learning in an adversarial setting: A survey, Journal of Parallel and Distributed Computing 130 (2019) 12–23. URL: https://www.sciencedirect.com/science/article/pii/S0743731518309183. doi:https://doi.org/10.1016/j.jpdc.2019.03.003.
[22] E. Crothers, N. Japkowicz, H. Viktor, Machine generated text: A comprehensive survey of threat models and detection methods, 2023. arXiv:2210.07321.
[23] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, Squad: 100,000+ questions for machine comprehension of text, 2016. URL: https://arxiv.org/abs/1606.05250. arXiv:1606.05250.
[24] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.
[25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2020. arXiv:1909.11942.
[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. URL: https://arxiv.org/abs/1907.11692. arXiv:1907.11692.
[27] D. Oliynyk, R. Mayer, A. Rauber, I know what you trained last summer: A survey on stealing machine learning models and defences, ACM Comput. Surv. 55 (2023). URL: https://doi.org/10.1145/3595292. doi:10.1145/3595292.
[28] AWS, Sagemaker data labeling pricing, https://aws.amazon.com/sagemaker/data-labeling/pricing/, 2023. Accessed: 20230-06-30.
[29] OpenAI, gpt4all.io, 2023. URL: https://gpt4all.io/index.html, accessed: 8th February 2023.
[30] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, 2017. arXiv:1610.05820.
[31] S. Szyller, B. G. Atli, S. Marchal, N. Asokan, Dawn: Dynamic adversarial watermarking of neural networks, 2021. arXiv:1906.00830.