Cybersecurity Data Extraction from Common Crawl

Ashim Mahara

March 2025

1 Introduction

Generative Large Language Models (LLMs) are pretrained on massively large and diverse sets of text data ([7], [30]). The datasets used for the pretraining are largely general and multilingual, and contain a diverse set of texts from various sources such as the web, public books[1], research papers [2], Wikipedia [3] articles and StackExchange [4] ([32], [12], [38], [44]).

Due to the heterogeneous nature of the datasets, the models trained using the datasets perform well for a variety of general downstream natural language tasks such as natural language understanding and reasoning [32] but recent studies [15] have shown that the models can benefit from further pre-training on domain-specific or task-specific datasets.

While there are various evaluation datasets and benchmarks for cybersecurity ([4], [40], [20], [1]), there is a general lack of pretraining datasets that can help to further align the models to cybersecurity. To our knowledge, Primus [47] is the only publicly available cybersecurity-focused pre-training dataset.

Thus, we introduce **Alpha-Root** as a new large text corpus from Common Crawl [5] — intended to be used as a pre-training dataset — by extracting cybersecurity-focused domains from the webgraph provided by Common Crawl. We utilize the Leiden [41] algorithm for community detection to mine the web graph for domains that represent the cybersecurity community in the webgraph. We then extract texts from webpages of the members of the community that are also present in the FineWeb-Edu [28] dataset. We also present evaluations of the trained models - for both Alpha-Root and Primus-FineWeb - on the MMLU:Computer_Security [16] subset of MMLU for evaluations. We show that our dataset consistently matches or outperforms the Primus-FineWeb dataset on several n_shot evaluations on different floating point precision runs.

[1] gutenberg.org
[2] arxiv.org
[3] wikipedia.com
[4] stackexchange.com
[5] commoncrawl.org

2 Background

2.1 Deep Learning

Deep Learning [19] is a subset of machine learning, and the broader field of artificial intelligence, that utilizes data to train artificial neural networks capable of performing a variety of tasks. It has revolutionized the technology industry and has been widely adopted by various scientific communities and industries to perform complex processing and analysis of structured and unstructured data. Deep learning has been applied in various domains, from identifying animals or objects in images (general image classification tasks) to detecting algorithmically generated domains (specialized cybersecurity classification tasks) queried by hosts in a computer network [46].

Generally, training a deep learning model M on a classification task consists of learning a function F that maps a collection of inputs X to a target output space Y. To tune the weights of a neural network, backpropagation [35] is used to minimize the difference between the output generated by M, usually denoted by Y', and the target output Y. A loss function L quantifies the discrepancy between the predicted output Y' and the ground truth label Y, guiding the optimization of M. In essence, we try to minimize the difference between the probability distributions of the ground truth class Y and the predicted class Y' given X.

Mathematically, the objective loss function for a classification task is usually a slight modification of the cross-entropy loss from the original logistic regression model [6] that expands the loss function to handle scenarios where there are more than two classes, such as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(y'_{i,c}) \tag{1}$$

where N denotes the number of samples in the input X, C denotes the number of target classes, $y_{i,c}$ is a is a one-hot encoded ground truth indicator for sample i and class c, and $y'_{i,c}$ is the predicted probability of sample i belonging to the class c as predicted by the model M.

2.2 Deep Learning in Cybersecurity

There has been a growing interest of deep learning applications in cybersecurity over the years. ([23]) Researchers are finding creative ways to integrate the probabilistic models to different tasks such as Malware Classification [5], Phishing Website Detection [21], and Anomaly Detection and Log Analytics [10].

While a number of different deep learning algorithms such as CNNs [27] and LSTMs [17] are used in the literature for various tasks [8], there has been a meteoric rise in adoption of LLMs in cybersecurity. Zhang et al. [48] list various different applications of LLMs in cybersecurity such as Indicator of Compromise (IoC) extraction from Cyber Threat Intelligence (CTI) reports, Source Code Analysis, Fuzzing, and LLM Assisted Red-Teaming among others.

2.3 Large Language Models

Large Language Models (LLMs) are a family of deep learning models — primarily based on the Transformer architecture by Vaswani et al. [43] — that are used to process sequence of texts. Since deep learning requires numerical representations of the data, the texts are converted to vector representations by *tokenizing* the text and converting the resultant *tokens* to their vector representations for further processing by utilizing an *embedding layer*. [3]

Tokenization is the process of creating a dictionary of key-value pairs that maps a token to its corresponding index. An embedding layer then converts the token index to a vector representation for the token. The embeddings — that are the vector representations of the tokens — are jointly trained with the language model itself and do not generally need to be trained separately for modern LLMs. However, there is an extra positional encoding layer that adds sequential information to the embeddings that is needed for LLMs utilizing the transformer architecture. [43]

Transformers utilize the attention mechanism [2] to attribute attention scores to each token in a sequence of text. The attention score determines the importance of a previously seen token while generating the next token in the sequence. They are also pretrained on very large text corpora [30]. The pretraining of the language model is performed in an unsupervised learning setting where the training objective consists of minimizing the cross-entropy loss while predicting the next token based on the previous tokens in a sequence of text input. Conversely, it can also be formulated as maximizing the conditional probability of a single token given a sequence of previous tokens.

Radford et al. [30] formulate the unsupervised pretraining as:

Given a sequence of tokens $U = \{u_1, ..., u_n\}$, the objective for the unsupervised pretraining is to maximize the following likelihood:

$$L(U) = \sum logP(u_i|u_{i-k}, ..., u_{i-1}; \theta) \tag{2}$$

where k is the context size that determines how many previous tokens are used for the current token prediction, and P is the conditional probability that is modeled by a neural network with parameters θ. The parameters are trained using gradient descent [34]; and backpropagation [35] — if the neural network is composed of multiple layers.

3 Related Work

3.1 Pretraining of Language Models

Our work broadly falls under the umbrella category of data collection for unsupervised training of large language models. Ramachandran et al. [33] were the first to [33] to demonstrate that language models perform better when pretrained by pre-training and finetuning an encoder-decoder model on their language translation task. Then, Radford et al. [30] pre-trained the first decoder-only transformer which was a 117 million parameter transformer-based language model using the BooksCorpus [49] dataset; and their resultant model GPT-1 is considered to be the first modern LLM. They used the 7000 books available from the corpus to demonstrate that pre-training the model resulted in a 15 point increase (average) when compared to a model without the pre-training stage.

Later, Devlin et al. [7] used the BooksCorpus as well as the English Wikipedia (2,500M words) to pre-train a bidirectional language model (BERT) for learning representations from unlabeled text. BERT used a *masked language modeling* (MLM) approach — also noted by the authors to be a derivation of Taylor's work in [39] — for the training as well as prediction of entire sentences instead of just the next tokens. BERT, and its derivatives like RoBERTa [22], were SOTA on many information retreival tasks for years and are still widely used for generating textual representations where efficiency is preferred over accuracy by modern standards.

3.2 Early Datasets for Pretraining of Language Models

The first known usage of CommonCrawl for pre-training was by Trinh and Le [42] to train a language model based on the RNN [36] architecture. Subsequently, Radford et al. [31] used a variety of techniques to clean data obtained from CommonCrawl to create the WebText dataset and train the GPT-2 model. GPT-2 [31] was trained on text exclusively extracted from CommonCrawl and outperformed every other existing work on various benchmarks in natural language processing at that time while demonstrating that unsupervised training of language models resulted in strong multi-task performance on downstream tasks.

While the WebText [31] dataset might be the first dataset extracted from CommonCrawl for the sole purpose of pre-training a language model, they did not release the dataset publicly. Instead, Gokaslan et al. [14] utilized the process presented in [31] in recreating the dataset, labeled as OpenWebText, and released it to the public. Following the works of [31] and [14], Weznek et al. [45] publicly released the first large scale high-quality dataset purposefully built to train language models from CommonCrawl called CCNET. The methods used by CCNET for the construction of the dataset is still used in later works such as [47].

3.3 Modern Datasets for Pretraining of LLMs

The Pile [12] can be considered as the first open-source large-scale dataset for large language model pre-training. The Pile consisted of data from various sources, including CommonCrawl, and amounted to a staggering 1.2 TiBs. They also used the GoldMiner [9] algorithm to clean the CommonCrawl web scrapes and concluded that the extra information from the WET archive format was counter-productive for the pre-training.

Next, the C4 [32] dataset was constructed solely from CommonCrawl. The name *Colossal Clean Crawled Corpus* directly implies the nature of the dataset, which — at 750 GB — was used to train the T5 model which was the first study to show that language models can be trained on multiple-tasks at the same time as opposed to training at a single task at a time. Models pre-trained with C4 outperformed other models pre-trained with non-C4 datasets; with the closest competition from datasets derived from CommonCrawl rather than other sources.

More recently, RefinedWeb [29] uses a variety of different quantitative and qualitative measures to extract data from CommonCrawl. They introduced MacroData Refinement (MDR) pipeline for CommonCrawl — that focused heavily on deduplication and filtering — and trained the Falcon LLM using the data extracted by using their pipeline. They also introduced URL scoring to filter unwanted URLs for topics such as adult content and gambling. RefinedWeb contained 2.8TB or 600B tokens of text extracted solely from CommonCrawl.

FineWeb [28] is the first work to use a trained neural network-based classifier to filter content based on their quality. They created the FineWeb-Edu subset from the FineWeb dataset by using a LLM annotator to create training samples for the training the edu-classifier and later utilizing the classifier on the superset of the data. FineWeb-Edu was demonstrated to surpass every other dataset, at that time, for training LLM models when compared against every other publicly available dataset.

3.4 Cybersecurity Datasets

While there are numerous works available for general pre-training of LLMs, as outlined previously in this section, there is only one work on cybersecurity-specific pre-training dataset. PRIMUS [47] builds upon the work of [28] to build a classifier that filters cybersecurity related topics from FineWeb. They create the PRIMUS-FINEWEB data by combining the data obtained from the previous step, and a combination of various other sources, to create the PRIMUS-PRETRAINING dataset; which is the first cybersecurity-focused pretraining dataset.

Although FineWeb is a leading general-purpose dataset, FineWeb-Edu is an even better version of FineWeb – introduced in the same paper [28] – that was curated using a LLM-based classifier that ranked content based on their educational value. PRIMUS did not use FineWeb-Edu to curate their data and instead opted for the FineWeb as their data-source.

In contrast of FineWeb-Edu ranker, PRIMUS trained a binary classifier that distinguishes whether given content is cybersecurity related or not, and opted to use threshold values from the classifier's output; which is the probability that a certain given text is from cybersecurity domain.

Our previous previous works such as SECURE [4] have hinted at using graph methods to filter data-sources from CommonCrawl. We continue on this direction by using Community Detection algorithms - namely the Leiden Algorithm [41] - on Common Crawl Web Graph to mine for domains related to cybersecurity.

4 Alpha-Root

4.1 Overview

We introduce a novel method for extracting domain-specific web data from CommonCrawl[6] by utilizing community detection algorithms. We leverage the Leiden [41] algorithm to extract cybersecurity-specific community of web domains from the Common Crawl Web Graph[7] by using a list of seed domains. The extracted web-domains are subsequently used to aggregate webpages from FineWeb-Edu. We used this approach to create a cybersecurity pretraining text corpus called **Alpha-Root**. Alpha-Root is an early cybersecurity focused pre-training dataset consisting of **3 billion** tokens sourced from 3.3 million webpages.

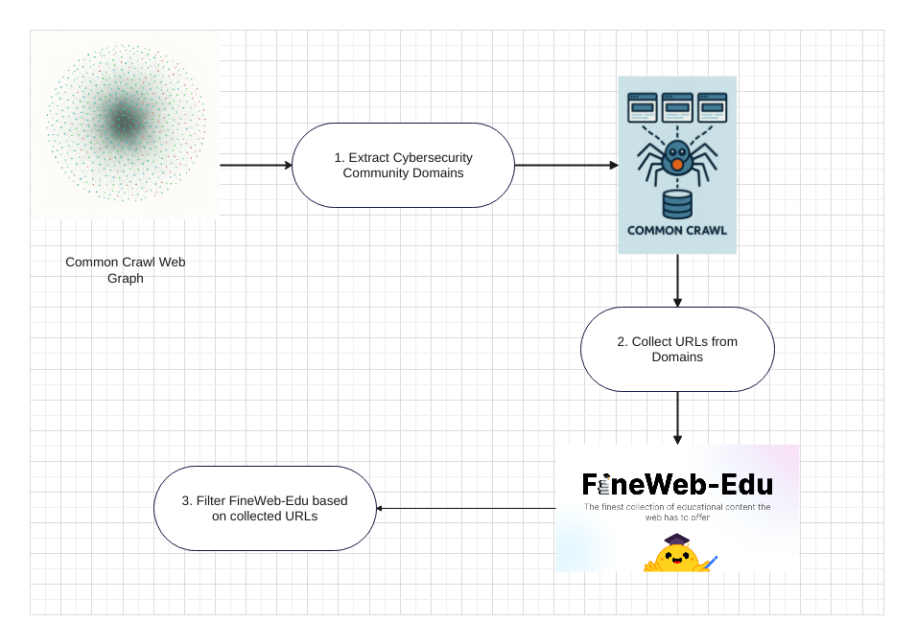

Figure 1: Dataset Extraction Process

4.2 Cybersecurity Community Detection in Web Graph

4.2.1 Web Graph

CommonCrawl release web graphs for each of their crawls. The cc-graph is a graph of nodes (domains) and edges between the nodes when the nodes are

[6] commoncrawl.org
[7] As referred to cc-graph throughout the document

hyperlinked through a webpage. Essentially, there exists an edge between two domains if there is at-least one hyperlink between webpages of the domains.

The cc-graph includes more than **100 million** nodes and more than **1.8 billion** edges. The processing of such a large graph presents several challenges on its own since commoncrawl only provides the data in an edge-list format while a lot of existing software libraries require an adjacency list to work on the graph.

We used a memory-mapped sparse matrix in compressed sparse row format to compute and store the adjacency list of the graph. The adjacency list allowed us to make use of several available libraries such as Cugraph [8] and Scikit-Network [9] to run the community detection algorithm on modern GPUs - which vastly speeds up the process while extracting communities from the web graph.

4.2.2 Extraction of Cybersecurity Community

While the topic of community detection has been well studied in the past by pioneers in the field of networks such as Newman et al. in their works([24], [25]), the research on the topic of community detection in the web has stagnated in recent times.

The most prominent research in the field was performed more than two decades ago in works such as [11], [13], and [18]; with majority of the works opting for some variant of hyperlink induced community detection. While the efficacy of the method cannot be understated for data collection - as evident by the success of the DeepSeekMath [37] model in which the researchers crawled hyperlinks from a seed of curated webpages to collect their dataset - such methods can be considered primitive and inefficient since the introduction of modern methods of community detection such as Modularity Maximization [25]. Even works such as PRIMUS [47] follow the steps of FineWeb [28] in training a classifier that is used to filter cybersecurity related web pages but this approach entails processing web-scale amount of text through the classifier which may not be feasible for adoption by resource-constrained audience.

In contrast, we utilized the Leiden algorithm [41] - an algorithm based on modularity maximization - to extract a collection of domains belonging to the cybersecurity community within the web graph.

4.3 Dataset Curation

Once the domain were extracted from the web graph, we used the common crawl index to further extract webpages from the domains. We collected **70 million** unique webpages across all of the crawls from common crawl. While we wanted to download all of the webpages that we collected and run a classifier on the content for further filtering, it seemed infeasible to do so due to the download

[8] https://github.com/rapidsai/cugraph
[9] https://github.com/sknetwork-team/scikit-network

rate limits from Amazon Web Services (AWS)[10] and due to our computing resources limitations.

We instead opted to filter the FineWeb-Edu dataset based on our collection of URLs. This approach allowed us to collect the sources for the data at large without explicitly filtering based on the content. We were able to curate our Alpha-Root dataset in 8 hours; with a couple of restarts since the FineWeb-Edu (5 TiB) dataset was streamed and filtered in memory due to storage constraints on our computing cluster. The resultant dataset contains **2.8 million** URLs from the original **72.8 million** URLs that we collected.

Dataset	**Number of Examples**	**Number of Tokens**
Primus-FineWeb	3 386 733	2.57B
Alpha-Root	2 866 811	3B
Alpha-Root-Dedup	684 016	714M

Table 1: Comparison of dataset sizes for Primus-FineWeb and Alpha-Root.

[10] https://aws.amazon.com/s3/

5 Training and Evaluation

5.1 Training

The models were trained using the hyperparameters detailed in Table 2, combining efficient quantization techniques, Low-Rank Adaptation (LoRA), and large-scale distributed training. Below, we contextualize key design choices and their implications.

5.1.1 Computational Efficiency

To optimize memory usage and throughput, the training employed:

- **4-bit Quantization** (NF4 type) with `bfloat16` compute dtype, reducing memory footprint while preserving gradient precision [**?**].
- **8-bit AdamW** for optimizer states, further lowering memory requirements.
- **Gradient Accumulation** (8 steps) to simulate larger batch sizes ($8 \times 8 = 64$ effective batch size per device) without increasing memory pressure.

5.1.2 Architecture & Adaptation

The base model (1.8B parameters) was adapted via LoRA to balance parameter efficiency and task-specific learning:

- LoRA rank $r = 128$ with $\alpha = 256$ applied to attention and feed-forward projections.
- Only 346M (16.8% of total) parameters were trainable, focusing adaptation on critical modules (`lm_head`, `embed_tokens`).
- A sequence length of 8192 tokens to enable long-context processing due to the datasets containing texts upto 140k tokens (PRIMUS); and maximum of 120k tokens for Alpha-Root.

5.1.3 Distributed Training

Training was conducted on **two NVIDIA GH200 nodes**, leveraging their unified CPU-GPU memory. Each GH200 node contains 95GB of VRAM available for training; we were able to maintain 80-85% effective utilization for the training runs. The 1-epoch training schedule (cosine LR, 5e-4 peak) prioritized rapid convergence, with a 1% warmup ratio to stabilize early optimization.

5.2 Evaluation

We used the lm-eval [**?**] library for our evaluations.

Category	Parameter	Value
Precision & Quant.	bf16	true
	load_in_4bit	true
	bnb_4bit_quant_type	nf4
	bnb_4bit_compute_dtype	bfloat16
	bnb_4bit_quant_storage	uint8
Architecture	hidden_size	2048
	intermediate_size	8192
	num_hidden_layers	24
	num_attention_heads	32
	max_seq_length	8192
Optimizer & LR	optim	adamw_8bit
	learning_rate	5e-4
	lr_scheduler_type	cosine
	warmup_ratio	0.01
Training Loop	num_train_epochs	1
	per_device_train_batch_size	8
	gradient_accumulation_steps	8
LoRA & PEFT	rank (r)	128
	lora_alpha	256
	lora_dropout	0.05
	target_modules	q_proj, k_proj, v_proj, up_proj, down_proj, gate_proj, o_proj
	modules_to_save	lm_head, embed_tokens
Model Parameters	total	2 057 406 464
	trainable	346 030 080

Table 2: Key hyper-parameters for training SmolLM-1.7B with LoRA adapters

5.2.1 MMLU: Massive Multitask Language Understanding

The **Massive Multitask Language Understanding (MMLU)** benchmark [16] is a comprehensive evaluation framework designed to assess the knowledge and reasoning capabilities of machine learning models across diverse domains. It consists of multiple-choice questions spanning 57 subjects, including STEM, humanities, social sciences, and professional disciplines. MMLU evaluates both *factual knowledge* and *conceptual understanding*, making it a robust measure of a model's generalization ability and multitask proficiency. We specifically chose MMLU for our evaluation due to the benchmark's compatibility with models that have not been instruct-finetuned.

A specialized subset of MMLU, **MMLU_COMPUTER_SECURITY**, focuses on assessing a model's understanding of cybersecurity principles. This subset includes questions on topics such as:

- Cryptographic protocols (e.g., AES, RSA),

- Network security (e.g., firewalls, intrusion detection),
- Software vulnerabilities (e.g., buffer overflows, SQL injection),
- Access control models (e.g., RBAC, MAC).

Performance on this subset indicates a model's ability to reason about technical and theoretical aspects of computer security, which is critical for applications in automated threat analysis, secure code generation, and adversarial robustness.

6 Experiment Results

As explained in 5.2, we opted to use MMLU:Computer_Security as our evaluation metric due to the log likelihood [16] calculations for the metrics. We performed 5 n_shot evaluations for the models with 2 sets of floating point precision - this decision was made due to the usage of lower quantization in the training phase since we wanted to perform fair evaluations of the trained and the base models.

We compared our trained models against the base SmolLM-1.7B model as well as the pretrained SmolLM model later continually pre-trained on Primus-FineWeb. We notice that our models, whether pretrained with a higher batch size or with the same batch size as the Primus-FineWeb, consistently performed on the same level as the Primus-FineWeb. The **checkpoint-5700** was included to assess whether a lower amount of training steps - coupled with a dataset with duplicates in case of Alpha-Root - would skew the evaluations.

We notice irregularities in the few shot evaluations for both floating point variations of the evaluations. However, we also note that our dataset consistently performs on the same level as Primus-FineWeb - and even better in a lot of the scenarios - while requiring lesser computational resources for the extraction of the corpus. We also performed the evaluations with different seeds for a fairer comparison but it resulted in the same scores for all of the experiments we ran so we left it as **1234**. We used the *lm-eval* [**?**] library from EleutherAI[11] for generating the evaluation scores.

Model	**mmlu_computer_security**				
	0-shot	**1-shot**	**2-shot**	**3-shot**	**5-shot**
SmolLM-1.7B (Base)	0.28	0.32	**0.37**	0.26	**0.33**
Primus-FineWeb	**0.32**	<u>0.33</u>	0.29	<u>0.31</u>	0.28
Alpha-Root (256bsz, Full Training)	0.26	**0.35**	0.27	**0.32**	<u>0.30</u>
Alpha-Root (checkpoint-5700)	<u>0.30</u>	0.26	0.30	0.28	0.29
Alpha-Root (Full Training)	0.25	0.32	<u>0.31</u>	0.28	0.29

Table 3: Performance on the `mmlu_computer_security` benchmark with 0-, 1-, 2-, 3-, and 5-shot evaluations. BF16 inference. Highest scores are **bolded**, second-highest are <u>underlined</u>.

[11] https://github.com/EleutherAI/lm-evaluation-harness

Model	mmlu_computer_security				
	0-shot	**1-shot**	**2-shot**	**3-shot**	**5-shot**
SmolLM-1.7B (Base)	<u>0.27</u>	0.34	**0.32**	**0.30**	**0.31**
Primus-FineWeb	**0.31**	<u>0.35</u>	0.28	**0.30**	<u>0.29</u>
Alpha-Root (256bsz, Full Training)	0.26	**0.38**	0.28	0.26	<u>0.29</u>
Alpha-Root (checkpoint-5700)	**0.31**	0.27	<u>0.30</u>	<u>0.28</u>	0.28
Alpha-Root (Full Training)	0.26	0.34	**0.32**	**0.30**	<u>0.29</u>

Table 4: Performance on the `mmlu_computer_security` benchmark with 0-, 1-, 2-, 3-, and 5-shot evaluations. FP32 inference. Highest scores are **bolded**, second-highest are <u>underlined</u>.

7 Discussion

In this section, we are going to share our findings in data collection and pretraining:

7.1 Data Collection at Scale is Hard

When we initially started to collect the domains by leveraging the web graph, we had to find workarounds for managing the memory and storage requirements since creating an adjacency list out of a large graph like the commoncrawl web graph – with >100 million nodes and >1.8 billion edges – would require petabyte scale of memory just for storage. We found that using memory mapped file for the sparse matrix and splitting the original edge list into several small files for processing the small *splits* was sufficient. We also found the UNIX program *split* to be sufficient for the task while the splits themselves were processed using a custom Python script.

The memory and storage impediments were persistent during the text extraction phase too. Due to a non-trivial amount of web pages (70M) to extract from CommonCrawl, we opted to use FineWeb-Edu as our data source. We faced 3 hurdles during this phase,

- **Downloading Web Pages**: The webpages from commoncrawl were hosted in AWS S3 which throttled our requests to 30 iterations per second; this made it impossible to download the web pages in a reasonable time - with the estimated time being 30-36 days for downloading the WARC records alone.
- **Scoring the text from Web Pages**: We had initially planned to score the texts by fine-tuning a scoring model trained on a FineWeb-Edu inspired LLM-as-a-Judge dataset for filtering out high quality cybersecurity content. While the training of the model was completed, the resultant model could not be used for scoring the texts due to computational restraints.

7.2 Common Crawl is Massive

During our data collection phase, we noticed that several websites now disallow the common crawl bots from crawling their websites.[12] Even with the restrictions, each individual release of the official crawls contain terabytes of data - the latest April 2025 crawl contains 468 TiB of data. [13]

The scale of the available data makes it infeasible for domain-specific model authors to naively collect training data using scoring based models which requires the processing of every example of text in the crawls. Alternatively, our

[12] https://www.bleepingcomputer.com/robots.txt
[13] https://commoncrawl.org/blog/april-2025-crawl-archive-now-available

method provides a leaner approach to the problem by directly finding good domains for crawling and text extraction by exploiting the graph features of the commoncrawl web graph.

We found that out of the **15240** unique domains present in Alpha-Root, **9250** are also present in the PRIMUS dataset. This intersection of common data sources also signals that our method is able to mine sources of good data without directly inferring their relevance from processing their content - which is both computationally expensive and time consuming.

7.3 Text Scoring Models are Good Too

While our method focuses on extracting domains and web pages from those domains using the web graph, the extracted web pages can undoubtedly be filtered even further by using a scoring model. The text-scoring model can be integrated into our method after the URLs have been collected from the domains in the mined cybersecurity community.

8 Future Work

There are a few unfinished tasks directly related to our work that we would like to highlight as future work. We list some of them here as:

- **Evaluate the domain extraction phase.** In this paper we just used the extracted domains and constructed our dataset, for a deeper analysis we could extract the domains at multiple configurations of the modularity maximization algorithm.
- **Train on the de-duplicated Alpha-Root dataset.** We haven't had the chance to perform the training and evaluations on the de-duplicated dataset.
- **Download and filter the original 70 million URLs links using the scoring model.** Although this would require a lot of processing and storage, this would be an interesting direction to pursue.
- **Continue Pretraining without using LoRA adapters.** We can see that there are some fluctuations on the MMLU evaluations based on the floating point type used during inference. It would be a natural direction for us to continue pretrain in a higher precision for more rigorous comparisons.

Acknowledgments

Thanks to Prof. Nidhi Rastogi, and Prof. Nishant Malik. Thank you to Prof. Matthew Wright for providing me with RC@[26] access. In remembrance of https://github.com/rootnp (alpha-root).

References

[1] Garima Agrawal, Kuntal Pal, Yuli Deng, Huan Liu, and Ying-Chih Chen. Cyberq: Generating questions and answers for cybersecurity education using knowledge graph-augmented llms. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(2121):23164–23172, March 2024.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. (arXiv:1409.0473), May 2016. arXiv:1409.0473 [cs].

[3] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[4] Dipkamal Bhusal, Md Tanvirul Alam, Le Nguyen, Ashim Mahara, Zachary Lightcap, Rodney Frazier, Romy Fieblinger, Grace Long Torales, Benjamin A. Blakely, and Nidhi Rastogi. Secure: Benchmarking large language models for cybersecurity. (arXiv:2405.20441), October 2024. arXiv:2405.20441 [cs].

[5] Bugra Cakir and Erdogan Dogdu. Malware classification using deep learning methods. In *Proceedings of the 2018 ACM Southeast Conference*, ACMSE '18, page 1–5, New York, NY, USA, March 2018. Association for Computing Machinery.

[6] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 12 2018.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Priyanka Dixit and Sanjay Silakari. Deep learning algorithms for cybersecurity applications: A technological and status review. *Computer Science Review*, 39:100317, February 2021.

[9] István Endrédy and Attila Novák. More effective boilerplate removal-the goldminer algorithm. *Polibits*, (48):79–83, December 2013.

[10] Amir Farzad and T. Aaron Gulliver. Log message anomaly detection and classification using auto-b/lstm and auto-gru. (arXiv:1911.08744), April 2021. arXiv:1911.08744 [cs].

[11] G.W. Flake, S. Lawrence, C.L. Giles, and F.M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–70, March 2002.

[12] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. (arXiv:2101.00027), December 2020. arXiv:2101.00027 [cs].

[13] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems: links, objects, time and space—structure in hypermedia systems*, HYPERTEXT '98, page 225–234, New York, NY, USA, May 1998. Association for Computing Machinery.

[14] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[15] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 8342–8360, Online, July 2020. Association for Computational Linguistics.

[16] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31(11–16):1481–1493, May 1999.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[20] Matan Levi, Yair Alluouche, Daniel Ohayon, and Anton Puzanov. Cyberpal.ai: Empowering llms with expert-driven cybersecurity instructions. (arXiv:2408.09304), August 2024. arXiv:2408.09304 [cs].

[21] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. page 3793–3810, 2021.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. (arXiv:1907.11692), July 2019. arXiv:1907.11692 [cs].

[23] Alberto Miranda-García, Agustín Zubillaga Rego, Iker Pastor-López, Borja Sanz, Alberto Tellaeche, José Gaviria, and Pablo G. Bringas. Deep learning applications on cybersecurity: A practical approach. *Neurocomputing*, 563:126904, January 2024.

[24] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, March 2004.

[25] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[26] Rochester Institute of Technology. Research computing services, ¡YEAR$_Y OU_U SED_R C >$.

[27] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. (arXiv:1511.08458), December 2015. arXiv:1511.08458 [cs].

[28] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. (arXiv:2406.17557), October 2024. arXiv:2406.17557 [cs].

[29] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data only. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, page 79155–79172, Red Hook, NY, USA, December 2023. Curran Associates Inc.

[30] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[32] RaffelColin, ShazeerNoam, RobertsAdam, LeeKatherine, NarangSharan, MatenaMichael, ZhouYanqi, LiWei, and LiuPeter J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, January 2020.

[33] Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. Unsupervised pretraining for sequence to sequence learning. (arXiv:1611.02683), February 2018. arXiv:1611.02683 [cs].

[34] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, September 1951.

[35] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[36] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. (arXiv:1912.05911), November 2019. arXiv:1912.05911 [cs].

[37] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. (arXiv:2402.03300), April 2024. arXiv:2402.03300 [cs].

[38] Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, and Eric Xing. Slimpajama-dc: Understanding data combinations for llm training. (arXiv:2309.10818), May 2024. arXiv:2309.10818 [cs].

[39] Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, September 1953.

[40] Norbert Tihanyi, Mohamed Amine Ferrag, Ridhi Jain, Tamas Bisztray, and Merouane Debbah. Cybermetric: A benchmark dataset based on retrieval-augmented generation for evaluating llms in cybersecurity knowledge. (arXiv:2402.07688), June 2024. arXiv:2402.07688 [cs].

[41] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, March 2019.

[42] Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. (arXiv:1806.02847), September 2019. arXiv:1806.02847 [cs].

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[44] Maurice Weber, Dan Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy S Liang, Christopher Ré, Irina Rish, and Ce Zhang. Redpajama: an open dataset for training large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, page 116462–116492. Curran Associates, Inc., 2024.

[45] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France, May 2020. European Language Resources Association.

[46] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. Predicting domain generation algorithms with long short-term memory networks. (arXiv:1611.00791), November 2016. arXiv:1611.00791 [cs].

[47] Yao-Ching Yu, Tsun-Han Chiang, Cheng-Wei Tsai, Chien-Ming Huang, and Wen-Kwang Tsao. Primus: A pioneering collection of open-source datasets for cybersecurity llm training. (arXiv:2502.11191), February 2025. arXiv:2502.11191 [cs].

[48] Jie Zhang, Haoyu Bu, Hui Wen, Yongji Liu, Haiqiang Fei, Rongrong Xi, Lun Li, Yun Yang, Hongsong Zhu, and Dan Meng. When llms meet cybersecurity: a systematic literature review. *Cybersecurity*, 8(1):55, February 2025.

[49] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. (arXiv:1506.06724), June 2015. arXiv:1506.06724 [cs].