Web Content Filtering Through Knowledge Distillation of Large Language Models

Tamás Vörös[1], Sean Bergeron[1] and Konstantin Berlin[1]

[1]*Sophos AI*

Abstract

We introduce a state-of-the-art approach for URL categorization that leverages the power of Large Language Models (LLMs) to address the primary objectives of web content filtering: safeguarding organizations from legal and ethical risks, limiting access to high-risk or suspicious websites, and fostering a secure and professional work environment. Our method utilizes LLMs to generate accurate classifications and then employs established knowledge distillation techniques to create smaller, more specialized student models tailored for web content filtering. Distillation results in a student model with a 9% accuracy rate improvement in classifying websites, sourced from customer telemetry data collected by a large security vendor, into 30 distinct content categories based on their URLs, surpassing the current state-of-the-art approach. Our student model matches the performance of the teacher LLM with 175 times less parameters, allowing the model to be used for in-line scanning of large volumes of URLs, and requires 3 orders of magnitude less manually labeled training data than the current state-of-the-art approach. Depending on the specific use case, the output generated by our approach can either be directly returned or employed as a pre-filter for more resource-intensive operations involving website images or HTML.

Keywords

Machine Learning, Web Content Filtering, Large Language Models

1. Introduction

Web content filtering is crucial for maintaining network security and regulatory compliance in organizations[1, 2]. The aim of a web content filtering system is to prevent employees from accessing inappropriate content that violates regulatory requirements or company policies, and by filtering out high-risk content categories, such as pornography and weapons, it helps to avoid legal liability, reduces the risk of legal or ethical issues arising from exposure to unsuitable content, and promotes a professional work environment. Unlike security classification, which detects hosted malware and phishing attacks, content filtering models address a more general problem that is independent of the attack mechanism. In this work, we address the problem of web content categorization.

Traditional approaches to website categorization have relied upon creating and maintaining domain-to-category mappings, which are lists of domains grouped by their manually assigned categories [3]. A natural extension to list-based URL categorization is to en-

CAMLIS'23: Conference on Applied Machine Learning for Information Security, October 19–20, 2023, Arlington, VA

tamas.voros@sophos.com (T. Vörös); sean.bergeron@sophos.com (S. Bergeron); konstantin.berlin@sophos.com (K. Berlin)

hance them with signatures created by analysts, that would generalize better than exact string matching. In the case of web content filtering, the most straightforward signature-based approach is to propagate labels based on domains and subdomains, although more complex rules may be applied [4, 5, 6]. An example of this kind of label propagation is to maintain a list of known domains with predetermined labels, such as labeling "online-shop.com" as an e-commerce site and "news-site.com" as a news site. All URLs under these domains inherit the label. For instance, any URL under "online-shop.com" such as "online-shop.com/products/clothing", "online-shop.com/products/electronics", and "online-shop.com/cart" can be labeled as e-commerce. Similarly, any URL under "news-site.com", such as "news-site.com/politics", "news-site.com/technology", and "news-site.com/entertainment" can be labeled as news. In the manuscript, we focus on domain label propagation signatures for acquiring ground truth for the sake of simplicity, but it could be trivially extended to longest prefix matching of the URL for ambiguous websites. To provide comprehensive customer telemetry coverage for organizations, one of the most resource-effective manual methods involves ranking domains by frequency and labeling them in descending order. This approach maximizes coverage by prioritizing the labeling of a single domain. As new websites emerge daily and with over a billion existing websites, maintaining and scaling signature approaches manually for the long tail has become increasingly challenging. This necessitates the integration of machine learning into the classification pipeline[7, 8, 9]. Figure 1 illustrates the telemetry coverage of a large security vendor, with the space above the bars representing the infrequently seen long tail distribution of domains not already covered by domain labeling and label propagation signatures.

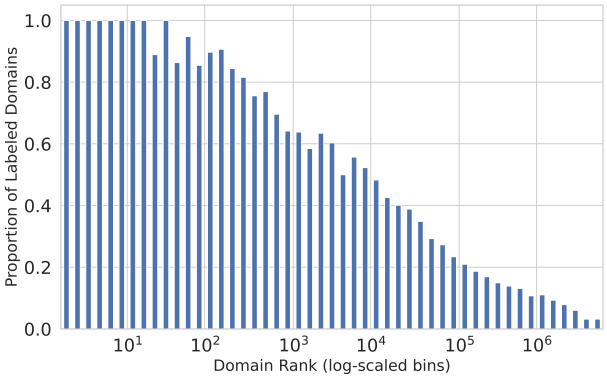

Figure 1: Labeling drop-off. The plot visualizes the proportion of analyst-labeled domains across different popularity levels of a large security vendor. The domain popularity is represented on the x-axis using a logarithmic scale, where higher values indicate more popular domains. Each bar in the plot corresponds to a specific popularity bin, with the height of the bar illustrating the proportion of labeled domains within that bin.

Maintaining domain-to-category mapping lists and extending them with signatures remains critical in the early stages of security pipelines [10]. These labels serve as initial shortcuts in the filtering pipeline to prevent catastrophic false positives, and provide low latency on more commonly seen websites. Websites like 'stackoverflow.com' are well-known and need not be

evaluated by a model whereas a potential false positive would translate to a negative impact on the productivity of an organization. In this work, we focus our evaluations on the long tail of the distribution, which aligns with actual deployment scenarios and emphasizes the need for machine learning to address the challenges associated with classifying this ever-growing subset of domains.

In addition to acting as a pre-filter, domain-to-category mapping lists and label propagation signatures are often used to create the training sets for machine learning models. However, machine learning algorithms tend to memorize patterns rather than understand underlying concepts [11, 12], thus learning from already labeled URLs is insufficient for accurate content classification in the long tail of the URL distribution. A model whose parameters are configured to memorize the head of the distribution is undesired as signatures already cover such domains without risking false positives. Therefore, our objective is to identify models with superior generalization capabilities for out-of-distribution samples.

For unknown or new domains, the model must infer a description from the URL. It is useful to view URL classification, especially for web content filtering, as a natural language processing task, considering URLs as semi-sentences. For a fair amount of our categories, the URL will frequently have explicit words to advertise its content, specifically semantically related keywords for the given category. For example, a site selling weapons will often contain keywords such as “armaments” or “glock” or “gun”. The current state-of-the-art in URL detection and our chosen baseline, URLTran [9], frames URL detection as a natural language processing task and fine-tunes a pre-trained BERT model [13] to detect phishing URLs. The BERT model is an early example of the transformer architecture [14] which has since been refined and scaled, giving rise to large language models. Large language models (LLMs) are state-of-the-art on natural language tasks [15]. LLMs are first pre-trained on large amounts of unlabeled textual data in a task-agnostic manner, learning a general understanding of language such as syntax and semantics [15]. Once pre-trained LLMs can effectively generalize to new tasks upon fine-tuning or few-shot prompting with much smaller amounts of data [15]. The amount of data needed for LLMs to generalize to new tasks is often several orders of magnitude less than the amount of data needed to fully train a smaller model.

Direct use of LLMs for URL content classification in production is prohibitive due to cost considerations at scale [16]. Fine-tuning smaller LLMs that have lower inference costs results in a loss of performance. Through knowledge distillation [17], the LLM-labeled long tail data enables a smaller student model to improve its performance while maintaining the necessary computational efficiency for production. Turc et al. [18] proposed an approach that utilizes knowledge distillation from the teacher’s predictive distribution (soft labels) followed by supervised fine-tuning of the student model. In the domain of web content classification, we combine the steps of distillation and fine-tuning and our computationally efficient student matches the performance of the teacher model. Instead of a predictive distribution, we distill the teacher using hard labels. The student model has a low inference cost and is well-suited for the purposes of web content filtering in production.

The main contributions of this paper are as follows:

- We demonstrate that when fine-tuned on data labeled with domain propagation signatures, large language models outperform standard deep learning models by 9% in terms of

accuracy on the long tail categorization problem.

- We demonstrate that we can fine-tune a large language model using 10000 samples to achieve better performance than the current state-of-the-art approach trained on 10 million samples.
- We showcase the effective application of knowledge distillation from a fine-tuned LLM to boost the performance of a smaller, more computationally efficient model, specifically for web content filtering tasks. We attain performance levels comparable to the original LLM using a model that is 175 times smaller, decreasing from 770 million parameters to just 4 million. This reduction in size makes the model more suitable for production and enables practical deployment across various contexts, such as serving as a general pre-filter for all incoming network traffic in firewalls.
- We propose a novel validation approach for the community to adopt, which more accurately assesses model performance in realistic scenarios where it works alongside a domain-to-category mapping list of ground truth labels, extended via domain label propagation signatures. In this setting, the model analysis focuses on labeling the long tail, focusing on a more relevant metric.

Our paper is structured as follows: In Section 1 we introduce the research problem and elucidate the motivation behind our proposed approach. In Section 2 we review relevant literature and prior work in the field. In Section 3, we provide a comprehensive description of our methodology, encompassing the dataset and experimental setups. In Section 4 we present our results, which include a comparison of our approach's performance against the current state-of-the-art, an analysis of the benefits of LLMs in terms of accuracy and sample efficiency, as well as an exploration of deployment challenges and our proposed solution utilizing knowledge distillation for more compact and computationally efficient models. Lastly, In Section 5 we conclude the paper, outlining potential avenues for future research in this domain.

2. Related Work

Previous work in this field has primarily focused on security classification rather than content classification and filtering. Since machine learning approaches to security classification can be readily reformulated from binary classification to multi-class classification through modification of the last layer in the neural network, approaches to security classification are relevant to the task of content classification. We will compare and build upon security publications as they are better studied.

Early work on URL-only classification for phishing detection using manually derived feature sets employed both generic features and features meant to detect certain obfuscation techniques such as obfuscation of the host with another domain [19]. The features were divided into four groups: Page Based, Domain Based, Type Based, and Word Based. The authors focused on manual feature engineering and only applied logistic regression as their classifier. A range of machine learning models, including Random Forests, Logistic Regression, Support Vector Machines, Naive Bayes, and Gradient Boosting, have been applied to detect phishing URLs using manually extracted feature sets [7, 20]. Feature sets may be entirely lexically derived

such as the length of the URL, the number of digits in the primary domain, and the number of special characters in the path [21]. In addition to lexical features, domain-specific features such as the number of passive DNS changes or the remaining time of the SSL certificate may be incorporated [22]. Manual features may also be extracted from the retrieved information of lookups (Whois, GSB Reporting, Google Ranking, and Selenium Rendering) [23].

The manual feature extraction approach is difficult to maintain as adversaries tend to adapt obfuscation methods to avoid detection so models have shifted to a featureless approach based on the raw string as input. Deep learning methods learn and then automatically extract the feature set from the raw URL during training. The use of automatically extracted features does not preclude the inclusion of manual features however as the optimal input combination of manual and automatic features can be optimized with genetic algorithms [24, 25].

Automatic feature extraction can be done on various levels of granularity starting at the character level. Saxe et al. [8] encode a URL by replacing each character with its corresponding ID whereby features are extracted from the encoded URL with sequential embedding and convolutional layers. This approach outperformed a baseline which uses a manual feature set. Learning meaningful context-independent representations is difficult when using character-level tokenization as a character token doesn't carry the same meaning that a word does. More recent approaches like subword-level and word-level tokenization have been developed in natural language processing in order to make it easier for models to maintain semantic meaning in common subwords and learn more meaningful context-independent representations.

The application of word-level tokenization to URL classification was first proposed by Le et al. [26] who extracted both character-level and word-level features. Each feature set is fed through its own series of sequential embedding and convolutional layers before being fused. Tajaddodianfar et al. [27] expand on this approach by first training the word embeddings in an unsupervised manner via FastText [28]. The word and character convolutional stems include several convolutional layers in parallel with dilated convolutions allowing the model to adaptively grow in depth and width, extracting N-grams of various lengths. In addition to using both character and word-level feature models, Bu et al.[29] apply a triplet network structure in order to address class imbalances and better learn the similarity between URLs.

In addition, to feature set selection, the choice of model architecture plays a large role in the performance of a URL classification model. Transformers have achieved state-of-the-art results in many natural language processing tasks making them a good candidate for URL classification after fine-tuning or even custom pre-training [30, 31, 9, 32, 33]. In addition to the URL, a transformer can leverage tokenized features of the HTML [34]. A URL classification system might employ different architectures in parallel, fusing the output of models with a convolutional architecture and a transformer architecture [35]. Instead of fusing model outputs, a system may employ an ensemble of different architectures including Decision Trees, LSTMs, and transformers for URL classification [36].

Other architectures applied to URL classification include graphical networks [37, 38] and GANs [39, 40]. AutoEncoders have proven useful against zero-day attacks [41]. In addition to the URL and HTML sequences, but beyond the scope of this paper, images of the webpage may be incorporated [42, 43]. The task of classification may be reformulated by approaching detection from a reinforcement learning perspective [44] or from the perspective of thwarting an adversarial opponent [45, 46].

The current state-of-the-art for URL-only classification for phishing detection, URLTran [9], utilizes the transformer architecture underpinning LLMs. Maneriker et al. fine-tune a pre-trained BERT model on Microsoft's Edge and Internet Explorer production browsing telemetry. Parallel to URLTRan is the Unified Text-to-Text Cybersecurity (UTS) model. Pal et al. [47] train a multi-task encoder-decoder LLM on cybersecurity data that includes URL phishing detection. Although Pal et al. introduce LLMs to URL phishing detection, they do not explore the few shot capabilities of LLMs in the URL domain nor test the capabilities of LLMs at scale. Compared to URLTran, UTS does not consider a methodology which would allow its large model to be used in production and reports a lower F1 score on a random split compared to URLTran's evaluation on the industry standard time split. Therefore, URLTran will act as our baseline to which all of our results will be compared.

3. Methodology

In this section, we describe our methodology for collecting data and constructing training, validation, and test sets. We also explain our experimental setup and provide a detailed account of how we trained our model.

3.1. Data

We obtained our dataset from a large security vendor's customer telemetry data sourced from its firewall and endpoint products over a period spanning July 1, 2022 to December 23, 2022.

We track 30 categories in our dataset. These categories were defined by a team of expert analysts to be representative of the most common internet content categories as well as the most impactful, which we define as the potential to impact productivity, the degree of liability for the organization, and the degree of associated ethical concerns. The categories include: "Chat", "Games", "Shopping", "Sports", "News", "Job Search", "Search Engines", "Alcohol", "Gambling", "Weapons", "Porn", "Banking", "Business", "Education", "Entertainment", "Food and Dining", "Government", "Health and Medicine", "Motor Vehicles", "Peer to Peer", "Real Estate", "Religion", "Travel", "Translators", "Computer and Internet", "Hunting and Fishing", "Marijuana", "Radio and Audio Hosting", "Social Networking", and "Video Hosting". The majority of websites in our dataset belong to categories such as "Computer and Internet", "Search Engines", and "Business", while niche categories such as "Hunting and Fishing" and "Marijuana" have fewer instances. Figure A5 shows the distribution of categories in our dataset. We define our categorization task as a closed-world problem, meaning every URL belongs to one of the 30 categories. It's important to note that, due to limitations in the domain-to-category database, we only consider a single category per URL, even though some pages may realistically have multiple category labels.

3.2. Training Sets

To construct our training dataset which spans the period from July 1, 2022 to August 19, 2022, we uniformly sampled 10 million distinct URLs, out of the billions of URL lookups, that have been labeled using a domain-to-category mapping database with label propagation. Additionally, we

sampled 10 million URLs from this period that did not correspond to a signature (unlabeled). The unlabeled URLs were set aside for training augmentation purposes.

3.3. Validation and Test Sets

We sampled an evaluation dataset spanning from August 19, 2022 to December 23, 2022 and divided it into two validation and test sets to assess our model's performance in different scenarios. The validation sets were based on data first seen between August 19, 2022 and November 24, 2022 while the test sets included data first seen between November 24, 2022 and December 23, 2022.

We created a domain and time split to simulate a long tail deployment setting. We separated the data based on the first-seen time of the URL and the first-seen time of the URL's domain. The first-seen time of a domain refers to the earliest instance of a URL from that domain. Meaning, there is no domain overlap between the training, validation, and test sets. This approach allowed us to better approximate the unlabeled part of the telemetry.

To compare our results with the industry-standard evaluation methodology we also created a time split. This split was sampled from the same time span as the domain and time split but without the constraint of dividing based on the domain's first-seen time.

For the domain and time split, the validation set was comprised of 79,313 unique URLs from 30,897 unique domains, with a maximum of 5 URLs per domain. The test set included 110,624 unique URLs from 43,996 domains. For the time split, we sampled 183,935 URLs from 62,961 domains.

To compare the various splits, we display the most common domains and their frequencies for the labeled training data, both test splits, and the unlabeled training data in Table 1. The labeled training set and the time split are dominated by common domains such as "google.com". The domain and time split is most similar to the unlabeled long tail of the data where the desired value of machine learning resides.

Table 1
Domains and their frequencies in the train and test sets.

Training Set		Time Split Test Set		Domain and Time Split Test Set		Unlabeled Data	
Domain	**Frequency %**	**Domain**	**Frequency %**	**Domain**	**Frequency %**	**Domain**	**Frequency %**
google.com	20	google.com	33	tomcleaneraddon.com	<1	wymondhamcollege.org	2
microsoft.com	6	microsoft.com	4	ammdx.com	<1	mfa.cloud	1
googleapis.com	5	gstatic.com	2	ogp.me	<1	dimmittisd.net	<1
cedexis-radar.net	3	googlesyndication.com	2	officested.com	<1	qq.com.cn	<1
gvt1.com	3	doubleclick.net	2	dimensionu.com	<1	gnsmat.co.uk	<1
facebook.com	2	msn.com	2	shreemaruti.com	<1	headlandentertainment.com	<1
zeotap.com	2	googleusercontent.com	2	wbe-eindhoven.nl	<1	murray.edu	<1
youtube.com	1	youtube.com	1	vapornodes.finance	<1	pcbid.top	<1
amazonaws.com	1	amazonaws.com	1	trendingtrck.com	<1	stoughtonwi.com	<1
sharepoint.com	1	cloudfront.net	1	bluedrop360.com	<1	aveha.com	<1

To quantitatively assess the disparities between the domain distributions of the time split and the domain and time split, which more closely models the long tail, we employed the Kullback-Leibler (KL) divergence as a metric for measuring the dissimilarity between token distributions. The KL divergence values were calculated between each validation split and the training dataset as the reference. We tokenized all the URLs in the training dataset using BERT tokenization and then combined all of the tokens to define the distribution of the base

training dataset. We tokenized all the URLs in both validation splits using BERT tokenization and each token sequence was converted into probability distributions by computing normalized histograms. The KL divergence between the token probability distribution of each URL and the reference distribution was then determined using the entropy function. Figure 2 illustrates that the token distribution of the domain and time split displays substantially higher KL divergence values from the reference compared to the time split. This observation highlights the distinct nature of the domain distributions in the two validation splits and the similarity between the unlabeled part of the customer telemetry and the domain and time split.

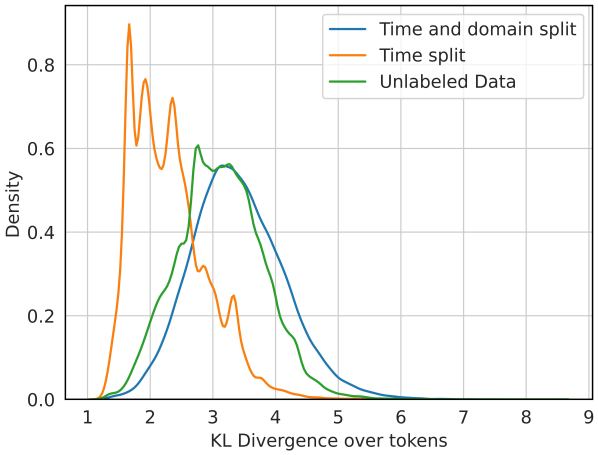

Figure 2: KL Divergence over BERT tokens. The x-axis represents the possible range of KL divergence values over BERT tokens, while the y-axis represents the estimated probability density of these values. The plot quantifies the difference between validation split URLs as compared to the training set token distribution, with higher KL divergence values indicating greater differences between the BERT token distributions of the base training set and validation split.

3.4. Experiments

The primary objective of our experiments was to identify the best-performing model in terms of accuracy on our dataset, while using as few training labels and being as small as possible. To achieve this, we varied the training set size as a hyperparameter for each LLM, compact model, and the baseline. We explored training set sizes ranging from few-shot to large-scale learning, increasing the sample size from 10 samples per category to 5 million total samples, growing by an order of magnitude at each step. For a given sample step size N, the exact samples per category were determined by the minimum of N and the total labeled instances in that category.

Our next goal was to refine the top-performing large language model (LLM) configuration into a more compact student model. We achieved this by using labels generated by the best-performing LLM to train smaller models.

We labeled 10 million unlabeled URLs from our dataset using the best-performing LLM, utilizing them as hard labels. This resulted in a total of 20 million training set with the 10 million signature-labeled base training set and an additional 10 million labels generated by the

LLM. We then investigated the impact of combining these labels using various mixing ratios of labeled samples from the base training set and LLM-labeled samples. Each compact student model and baseline were trained on a variety of dataset configurations, each containing a total of 10 million samples.

We began with a 10-million base training set, incorporating LLM-generated labels at 0.0, 0.25, 0.50, 0.75, and 1.0 ratios. The 0.0 ratio used only the base training set, while the 0.25 ratio included 7.5 million base URLs and 2.5 million LLM-generated. At 0.5, the sources were evenly split with 5 million each. The 0.75 ratio contained 2.5 million base and 7.5 million LLM URLs, and the 1.0 ratio relied solely on LLM-generated labels. By varying the mixing ratios, we were able to assess the effectiveness of our knowledge distillation process and compare the contributions of LLM-generated labels to simply using signature-generated labels.

We trained and compared the performance of five models: BERT-based URLTran as the baseline[9], which demonstrated state-of-the-art performance for URL classification, eXpose [8] and BERTiny [48] as the student models, and T5 Large [49] and GPT-3 Babbage [15] as the teacher models. The size configurations of our teacher models were limited by budgetary constraints, precluding larger configurations such as GPT-3 Davinci and T5-11B. Our student models were chosen for the following reasons: BERTiny is the smallest pre-trained configuration of the baseline and the inclusion of eXpose allows us to demonstrate the improvements of the transformer architecture over convolutional models for natural language tasks, specifically web content categorization. Unless otherwise noted, all experiments were evaluated on the test set of both validation splits. The GPT-3 Babbage model was not fine-tuned on 5 million samples due to cost considerations.

3.5. Training

For all models, we pre-processed the data by splitting at the first occurrence of the "?" character and removing the query parameters. The query is assumed to be noisy and without any meaningful information. All URLs were truncated to a fixed length of 128 characters as we have seen no improvement in further increasing the size. The base pre-trained models and tokenizers for all T5 Large, BERT, and BERTiny configurations were the HuggingFace defaults [50].

For all reported T5 configurations, we fine-tuned all weights of a pre-trained T5 Large model using the Adafactor optimizer [51]. Early stopping was applied by monitoring performance on the validation set of the domain and time split. For all reported GPT-3 configurations, we fine-tuned the Babbage model using the OpenAI API.

T5 and GPT-3 are generative models that can utilize semantic relationships between class labels and keywords in a URL for making predictions. Consequently, we employed literal class labels as our prediction target. When reporting aggregate metrics, out-of-vocabulary (OOV) predictions are not considered as a separate class, and they were considered as misclassfication for every class. Additionally, any unlabeled data for which LLM generates an OOV prediction is excluded from the distillation process.

For GPT3 the temperature was set to 0 to ensure deterministic results upon inference. The logit bias for tokens associated with the class labels were set to 100 to ensure exclusive selection of expected tokens. Finally, the stop token was set to the stop sequence seen during training.

For the student models, we trained a 1D convolutional eXpose model and fine-tuned all

weights of a pre-trained BERTiny model. We fine-tuned all weights of a pre-trained BERT model to reproduce the architecture of URLTran as our baseline. No custom vocabulary was created for the BERT-based models. Hyperparameter configurations for T5, BERTiny, BERT, and eXpose may be found in Tables A7, A4, A5, and A6 respectively.

4. Results

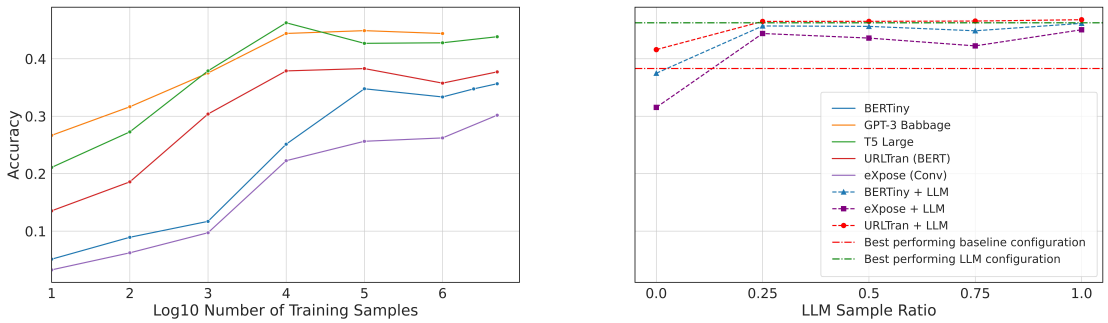

Figure 3: The results for scaling and augmentation are dependent on the domain and time split. Left: Scaling results: illustrates the performance of various models in relation to the logarithm of training sample size. Right: Augmentation results: compares the top-performing LLM and baseline configurations with the performance of different student models as a function of the mixing ratio for LLM-generated labels. The GPT-3 Babbage model was not fine-tuned on 5 million samples due to cost considerations.

In this section, we present the key findings and results of our two sets of experiments. We report the results in terms of accuracy, with additional metrics for both experiments provided in the Appendix.

The performance of various models as a function of the log of the training sample counts is displayed in Figure 3. The top-scoring configuration for each model is detailed in Table 2. On the domain and time split, the best performing model, T5 Large, achieves 46.3% accuracy after being fine-tuned on 10,000 samples. GPT-3 Babbage attains 44.4% accuracy after fine-tuning on 10,000 samples. Both LLMs surpass the best baseline configuration, which achieves 38.3% accuracy. BERTiny and eXpose achieve 35.7% and 30.2% accuracy, respectively, when trained on 5 million samples.

On the time split, eXpose achieves 92.8% accuracy when trained on 5 million samples. BERTiny, fine-tuned on 5 million samples, attains 97.6% accuracy. The best configurations for the baseline, GPT-3 Babbage, and T5 Large achieve 97.1%, 98.14%, and 97.5% accuracy, respectively. Additional metrics for the time split are reported in Table A10, and for the domain and time split in Table A9 for all experiments.

On the domain and time split, the best performance was achieved with T5 on 10,000 training samples, so we selected it as our teacher model. For the domain and time split, we found the best ratio to be 1.0, where every training sample had 10 million previously unlabeled URLs labeled by T5. Training eXpose on all of them increases the accuracy from 31.5% to 45%. Fine-tuning BERTiny on all 10 million LLM labels, compared to the 10 million base training set, improves

the accuracy from 37.5% to 46.2%. Finally, fine-tuning URLTran on all 10 million LLM labels, compared to the 10 million base training set, raises the accuracy from 41.6% to 46.8%.

For the traditional time split, the augmentation at a ratio of 0.75 also increased the performance, albeit marginally.

The performance of the students and the baseline trained via knowledge distillation is shown in the augmentation plot of Figure 3 as a function of the LLM label ratio in the training data. The top-scoring distillation configuration for each student model is detailed in Table 2.

Table 2
The best performing model configurations are presented, including the distilled versions of the students and baseline. The accuracy for each model's top configuration is displayed, along with the model's parameter count relative to the best performing LLM. For the Time and Domain split, the LLM label ratios correspond to 1.0, and for the Time split, the ratio is 0.75, as these were consistently the best across the models. More detailed results can be found in Table A9 and Table A10.

Model	Accuracy Time and Domain Split	Accuracy Time Split	Parameter Count in millions	Parameter Count Relative to the Teacher (%)	Training Samples Count
eXpose (Conv)	0.30	0.93	3.3		5×10^6
BERTiny	0.36	0.97	4.4		5×10^6
URLTran (BERT)	0.38	0.97	110		1×10^5
T5 Large	**0.46**	0.97	770		1×10^4
GPT3 Babbage	0.45	**0.98**	6700		1×10^5
eXpose + T5 Labels	0.45	0.98	3.3	0.42	1×10^7
BERTiny + T5 Labels	0.46	0.98	4.4	0.57	1×10^7
URLTran + T5 Labels	**0.47**	**0.99**	110	14.29	1×10^7

4.1. Discussion

A comparison of the models' performance on the two evaluation splits reveals that results on the time split, the traditional validation approach, are overly optimistic. Small models such as BERTiny, trained merely on signature-driven data, exhibit performance comparable to T5 and GPT-3. The disparity in model performance between the domain and time split versus the time split, particularly for small models, underscores that signature-sourced data is repetitive and can be memorized with just a few million parameters. Time split validation measures a model's ability to match the signature distribution while in a production setting the primary concern within the context of the overall pipeline is a model's capacity to generalize to new data from the long tail that falls outside the coverage of signatures.

When considering the domain and time split—which aligns more closely with real-world performance on unlabeled data—small models no longer match the performance of LLMs, as seen in Figure 3. Beyond 10,000 samples, LLMs show minimal to no performance gains when scaling up further. Conversely, the performance of small models and the baseline has not yet converged at 5 million training samples. This demonstrates the sample-efficiency of LLMs in the domain of website content categorization.

LLMs outperform student models in terms of performance, but they still fall short of perfection when applied to domain and test splits. This discrepancy can be attributed to two main factors. First, due to dataset limitations, web content classification is framed as a single-label classification problem. Table 3 displays a set of LLM misclassifications on domain and time split, highlighting

Table 3
Examples of LLM (T5) misclassifications. Comparison of LLM performance on the domain and time split, highlighting the impact of the single-label strategy and keyword-absent URLs.

Domain	LLM Label	True Label
citytocoastneurosurgery.com.au	HEALTH AND MEDICINE	BUSINESS
twittodon.com	SOCIAL NETWORKING	COMPUTER AND INTERNET
robinsonmalls.com/mall-info	SHOPPING	BUSINESS
online-weinshop.at	SHOPPING	ALCOHOL
www.fourbakery.com	BUSINESS	FOOD
sargenttoolsonline.com	BUSINESS	SHOPPING
praeyforthegods.com	RELIGION	GAMES
www.hygiene-3d.com	HEALTH AND MEDICINE	SHOPPING
beta.x9zb.live	COMPUTING AND INTERNET	GAMBLING
www.857zb6.com	ENTERTAINMENT	SPORTS
www.lxf.cz	BUSINESS	SHOPPING
g11.178tiyu.com	ENTERTAINMENT	SPORTS

that a URL could potentially belong to multiple categories. In the first three samples, the analyst opted for the more generic label, while the model choose the more generic labels in the following three samples. Both predicted and true labels could be considered correct in all six cases, suggesting that the true performance is likely better than the metrics indicate because of the single-label limitation. This trade-off between equally correct specific and general labels becomes evident when examining the confusion matrix, displayed in Figure A4 in the Appendix, for a T5 Large model's performance on the domain and time split. As we can see on the confusion matrix, the LLM tends to generate class labels that are more specific than manual labels.

The second factor occurs when a URL lacks keywords or context related to its category, as demonstrated by the last six entries in Table 3. For the middle two URLs, the model was misled by a prominent keyword in the URL, which was unrelated to its content. The final four URLs contain no apparent signal. Consequently, the large-scale pre-training of LLMs struggles to effectively transfer knowledge to a URL from the long tail. This means that if a URL lacks clear or strong indicators of its category, the LLM may not accurately classify it, leading to misclassifications.

Our results reveal that mixing in the LLM-generated labels significantly enhances the performance of student models BERTiny and eXpose as we can see on Figure 3. Through this simple form of augmentation, we nearly matched the 46.3% accuracy of T5 Large, the best-performing LLM, with a transformer model that has a parameter count several orders of magnitude smaller (0.57% of the teacher) and could reasonably be deployed in-line in production.

5. Conclusion

In conclusion, our paper contributes to the field of web content classification with the development of lightweight models distilled from fine-tuned LLMs. We have demonstrated that LLMs,

when fine-tuned on data labeled with domain propagation signatures, significantly outperform the current state-of-the-art approach on the long tail categorization problem. Our teacher-student training approach enables the distillation of LLMs into models 175 times smaller without sacrificing accuracy, thus making deployment practical in a wide variety of new contexts. The amount of manual labels required to finetune the teacher LLM is orders of magnitude smaller than what is required for convergence of the current state-of-the-art approach. Furthermore, we have proposed a new validation approach that better measures model performance in more realistic scenarios, which should be adapted by the community to improve generalization capabilities to unseen data.

Expanding beyond web content classification, the cybersecurity field could greatly benefit from proven methods of distilling large language models (LLMs) into more compact versions. This approach is particularly valuable when dealing with large data volumes and expensive training samples, especially when the model is applied to out-of-distribution cases. For addressing web content classification tasks specifically, we suggest future work should focus on augmenting the training data and feature space with HTML and image data, utilizing GPT-4 as a teacher, allowing URLs to have more than one label, and re-working signatures for the assignment of general categories.

References

[1] F. García, Web content filtering. advances in computers., 2009. https://www.academia.edu/11471179/Web_Content_Filtering.

[2] D. S. K. Ankur Baishya, A review on web content filtering, its technique and prospects, http://www.ijcstjournal.org/volume-7/issue-3/IJCST-V7I3P5.pdf (2022).

[3] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, C. Zhang, An empirical analysis of phishing blacklists, roceedings of Sixth Conference on Email and Anti-Spam (CEAS) (2009).

[4] Q. Chen, P. Snyder, B. Livshits, A. Kapravelos, Improving web content blocking with event-loop-turn granularity javascript signatures, arXiv preprint arXiv:2005.11910 (2020).

[5] C.-Y. Huang, S.-P. Ma, W.-L. Yeh, C.-Y. Lin, C.-T. Liu, Mitigate web phishing using site signatures, in: TENCON 2010-2010 IEEE Region 10 Conference, IEEE, 2010, pp. 803–808.

[6] S. Haruta, F. Yamazaki, H. Asahina, I. Sasase, A novel visual similarity-based phishing detection scheme using hue information with auto updating database, in: 2019 25th Asia-Pacific Conference on Communications (APCC), IEEE, 2019, pp. 280–285.

[7] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Beyond blacklists: learning to detect malicious web sites from suspicious urls, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 1245–1254.

[8] J. Saxe, K. Berlin, expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys, arXiv preprint arXiv:1702.08568 (2017).

[9] P. Maneriker, J. W. Stokes, E. G. Lazo, D. Carutasu, F. Tajaddodianfar, A. Gururajan, Urltran: Improving phishing url detection using transformers, in: MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM), IEEE, 2021, pp. 197–204.

[10] G. Apruzzese, H. S. Anderson, S. Dambra, D. Freeman, F. Pierazzi, K. A. Roundy, "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice, https://doi.org/10.48550/arXiv.2212.14315 (2022).
[11] J. Zhang, H. Chao, A. Dhurandhar, P. Chen, A. Tajer, Y. Xu, P. Yan, When neural networks fail to generalize? a model sensitivity perspective, https://doi.org/10.48550/arXiv.2212.00850 (2022).
[12] P. Bhargava, A. Drozd, A. Rogers, Generalization in nli: Ways (not) to go beyond simple heuristics, 2021. arXiv:2110.01518.
[13] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
[16] , How many websites are there, 2023. https://siteefy.com/how-many-websites-are-there/.
[17] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 (2015).
[18] I. Turc, M. Chang, K. Lee, K. Toutanova, Well-read students learn better: The impact of student initialization on knowledge distillation, CoRR abs/1908.08962 (2019). URL: http://arxiv.org/abs/1908.08962. arXiv:1908.08962.
[19] S. Garera, N. Provos, M. Chew, A. D. Rubin, A framework for detection and measurement of phishing attacks, in: Proceedings of the 2007 ACM workshop on Recurring malcode, 2007, pp. 1–8.
[20] A. Oshingbesan, C. Ekoh, C. Okobi, A. Munezero, K. Richard, Detection of malicious websites using machine learning techniques, arXiv preprint arXiv:2209.09630 (2022).
[21] A. Joshi, L. Lloyd, P. Westin, S. Seethapathy, Using lexical features for malicious url detection–a machine learning approach, arXiv preprint arXiv:1910.06277 (2019).
[22] C. Hajaj, N. Hason, A. Dvir, Less is more: Robust and novel features for malicious domain detection, Electronics 11 (2022) 969.
[23] A. Abuadbba, S. Wang, M. Almashor, M. E. Ahmed, R. Gaire, S. Camtepe, S. Nepal, Towards web phishing detection limitations and mitigation, arXiv preprint arXiv:2204.00985 (2022).
[24] S.-J. Bu, H.-J. Kim, Optimized url feature selection based on genetic-algorithm-embedded deep learning for phishing website detection, Electronics 11 (2022) 1090.
[25] K.-W. Park, S.-J. Bu, S.-B. Cho, Evolutionary optimization of neuro-symbolic integration for phishing url detection, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2021, pp. 88–100.
[26] H. Le, Q. Pham, D. Sahoo, S. C. Hoi, Urlnet: Learning a url representation with deep learning for malicious url detection, arXiv preprint arXiv:1802.03162 (2018).
[27] F. Tajaddodianfar, J. W. Stokes, A. Gururajan, Texception: a character/word-level deep learning model for phishing url detection, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 2857–2861.

[28] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fasttext.zip: Compressing text classification models, arXiv preprint arXiv:1612.03651 (2016).
[29] S.-J. Bu, H.-J. Kim, Learning disentangled representation of web address via convolutional-recurrent triplet network for classifying phishing urls, in: 2021 International Conference on Electronics, Information, and Communication (ICEIC), IEEE, 2021, pp. 1–4.
[30] E. M. Rudd, A. Abdallah, Training transformers for information security tasks: A case study on malicious url prediction, arXiv preprint arXiv:2011.03040 (2020).
[31] E. M. Rudd, M. S. Rahman, P. Tully, Transformers for end-to-end infosec tasks: A feasibility study, in: Proceedings of the 1st Workshop on Robust Malware Analysis, 2022, pp. 21–31.
[32] W. Chang, F. Du, Y. Wang, Research on malicious url detection technology based on bert model, in: 2021 IEEE 9th International Conference on Information, Communication and Networks (ICICN), IEEE, 2021, pp. 340–345.
[33] H. Shirazia, K. Haynesb, I. Raya, Towards performance of nlp transformers on url-based phishing detection for mobile devices, Journal of Ubiquitous Systems and Pervasive Networks, Volume 17,No. 1(2022) pp. 35-42 (2022).
[34] Q. Hu, H. Zhou, Q. Liu, Phishing website detection based on multi-feature stacking, in: 2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE), IEEE, 2021, pp. 716–720.
[35] C. Wang, Y. Chen, Tcurl: Exploring hybrid transformer and convolutional neural network on phishing url detection, Knowledge-Based Systems (2022) 109955.
[36] S. Venugopal, S. Y. Panale, M. Agarwal, R. Kashyap, U. Ananthanagu, Detection of malicious urls through an ensemble of machine learning techniques, in: 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), IEEE, 2021, pp. 1–6.
[37] S. Ariyadasa, S. Fernando, S. Fernando, Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using url and html, IEEE Access 10 (2022) 82355–82375.
[38] T. Bilot, G. Geis, B. Hammi, Phishgnn: A phishing website detection framework using graph neural networks, Conference: SECRYPT 2022At: Lisbon (2022).
[39] S. A. Kamran, S. Sengupta, A. Tavakkoli, Semi-supervised conditional gan for simultaneous generation and detection of phishing urls: A game theoretic perspective, arXiv preprint arXiv:2108.01852 (2021).
[40] J. Geng, S. Li, Z. Liu, Z. Cheng, L. Fan, Effective malicious url detection by using generative adversarial networks, in: International Conference on Web Engineering, Springer, 2022, pp. 341–356.
[41] S.-J. Bu, S.-B. Cho, Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing url detection, Electronics 10 (2021) 1492.
[42] J. Yuan, G. Chen, S. Tian, X. Pei, Malicious url detection based on a parallel neural joint model, IEEE Access 9 (2021) 9464–9472.
[43] R. Liu, Y. Lin, X. Yang, S. H. Ng, D. M. Divakaran, J. S. Dong, Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach, in: 30th {USENIX} Security Symposium ({USENIX} Security 21), 2022, p. 1.
[44] O. Lavie, A. Shabtai, G. Katz, A transferable and automatic tuning of deep reinforcement learning for cost effective phishing detection, arXiv preprint arXiv:2209.09033 (2022).
[45] Z. Peng, Y. He, Z. Sun, J. Ni, B. Niu, X. Deng, Crafting text adversarial examples to attack the

deep-learning-based malicious url detection, in: ICC 2022-IEEE International Conference on Communications, IEEE, 2022, pp. 3118–3123.
[46] T. Kim, N. Park, J. Hong, S.-W. Kim, Phishing url detection: A network-based approach robust to evasion, arXiv preprint arXiv:2209.01454 (2022).
[47] K. K. Pal, K. Kashihara, U. Anantheswaran, K. C. Kuznia, S. Jagtap, C. Baral, Exploring the limits of transfer learning with unified model in the cybersecurity domain, arXiv preprint arXiv:2302.10346 (2023).
[48] P. Bhargava, A. Drozd, A. Rogers, Generalization in nli: Ways (not) to go beyond simple heuristics, 2021. arXiv:2110.01518.
[49] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, The Journal of Machine Learning Research 21 (2020) 5485–5551.
[50] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, 2020, pp. 38–45.
[51] N. Shazeer, M. Stern, Adafactor: Adaptive learning rates with sublinear memory cost, in: International Conference on Machine Learning, PMLR, 2018, pp. 4596–4604.

A. Supplementary Plots and Tables

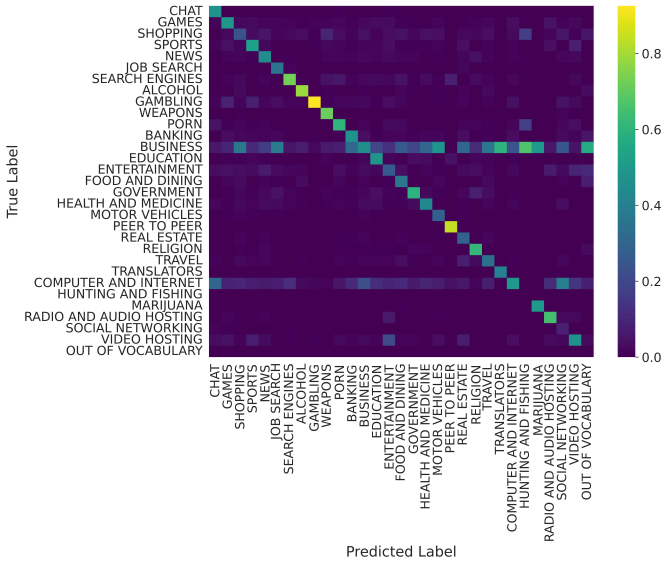

Figure A4: Normalized confusion matrix on the domain and time split for the best T5 Large configuration

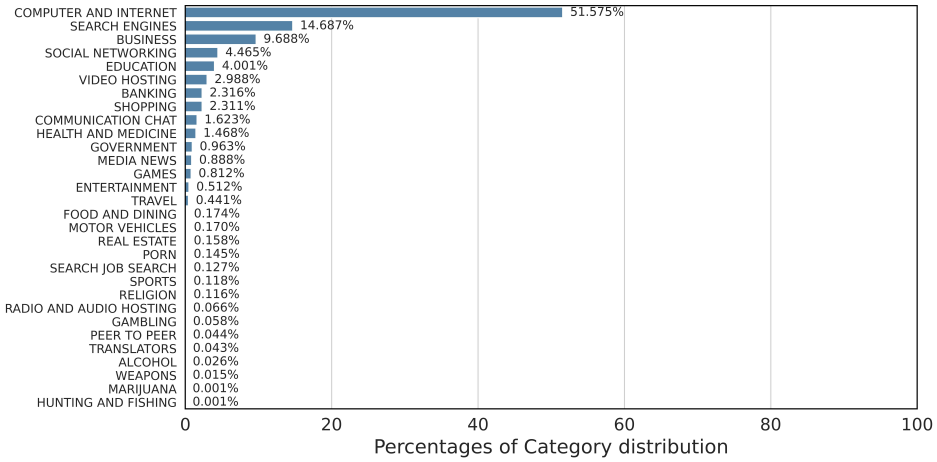

Figure A5: Class Distribution. Displaying the 30 classes and their distribution over the datasets.

Table A4
BERTiny Parameters

Parameter	Value
model_name	prajjwal1/bert-tiny
hidden_size	128
hidden_act	gelu
initializer_range	0.02
vocab_size	30522
hidden_dropout_prob	0.1
num_attention_heads	2
type_vocab_size	2
max_position_embeddings	512
num_hidden_layers	2
intermediate_size	512
attention_probs_dropout_prob	0.1
maximum sequence length	128
learning rate	1e-4
batch size	49160
optimizer	Adam
maximum training epochs	20

Table A5
BERT Parameters

Parameter	Value
model_name	bert-base-uncased
hidden_size	768
hidden_act	gelu
initializer_range	0.02
vocab_size	30522
hidden_dropout_prob	0.1
num_attention_heads	12
type_vocab_size	2
max_position_embeddings	512
num_hidden_layers	12
intermediate_size	3072
attention_probs_dropout_prob	0.1
maximum sequence length	128
learning rate	1e-4
batch size	2048
optimizer	Adam
maximum training epochs	20

Table A6
eXpose Parameters

Parameter	Value
vocab_size	76
filter_size	128
dropout	0.05
learning rate	1e-3
batch size	49160
optimizer	Adam
maximum training epochs	20

Table A7
T5 Parameters

Parameter	Value
model_name	t5-large
d_ff	4096
d_kv	64
d_model	1024
dropout_rate	0.1
initializer_facto	1.0
layer_norm_epsilon	1e-06
n_positions	512
num_heads	16
num_layers	24
relative_attention_num_buckets	32
vocab_size	32128
maximum sequence length	128
learning rate	1e-3
batch size	60
optimizer	Adafactor
Adafactor scale_parameter	False
Adafactor relative_step	False
Adafactor warmup_init	False
maximum training epochs	15

Table A8
Adafactor Parameters

Parameter	Value
scale_parameter	False
relative_step	False
warmup_init	False

Table A9
Model performance on Domain and Time Split

Model	Samples	Accuracy	Macro F1	Macro Recall	Macro Precision
BERTiny	10	0.051	0.045	0.077	0.06
BERTiny	100	0.089	0.083	0.152	0.089
BERTiny	1000	0.117	0.079	0.123	0.119
BERTiny	10000	0.251	0.098	0.109	0.137
BERTiny	100000	0.348	0.208	0.185	0.317
BERTiny	1000000	0.333	0.213	0.196	0.279
BERTiny	2500000	0.348	0.232	0.208	0.313
BERTiny	5000000	0.357	0.241	0.221	0.304
BERTiny + LLM Labels (0.0 Ratio)	10000000	0.375	0.253	0.23	0.335
BERTiny + LLM Labels (0.25 Ratio)	10000000	0.457	0.34	0.296	0.456
BERTiny + LLM Labels (0.5 Ratio)	10000000	0.456	0.336	0.293	0.452
BERTiny + LLM Labels (0.75 Ratio)	10000000	0.449	0.323	0.279	0.442
BERTiny + LLM Labels (1.0 Ratio)	10000000	0.462	0.347	0.299	0.473
GPT-3 Babbage	10	0.267	0.235	0.362	0.214
GPT-3 Babbage	100	0.316	0.265	0.384	0.232
GPT-3 Babbage	1000	0.375	0.315	0.379	0.294
GPT-3 Babbage	10000	0.444	0.364	0.354	0.402
GPT-3 Babbage	100000	0.449	0.36	0.328	0.428
GPT-3 Babbage	1000000	0.444	0.364	0.33	0.465
T5 Large	10	0.211	0.254	0.284	0.324
T5 Large	100	0.273	0.277	0.374	0.282
T5 Large	1000	0.379	0.34	0.382	0.367
T5 Large	10000	0.463	0.368	0.332	0.477
T5 Large	100000	0.427	0.372	0.353	0.426
T5 Large	1000000	0.428	0.296	0.256	0.434
T5 Large	5000000	0.438	0.329	0.286	0.454
URLTran (BERT)	10	0.135	0.055	0.092	0.096
URLTran (BERT)	100	0.186	0.176	0.285	0.168
URLTran (BERT)	1000	0.304	0.261	0.308	0.261
URLTran (BERT)	10000	0.379	0.278	0.266	0.322
URLTran (BERT)	100000	0.383	0.246	0.224	0.325
URLTran (BERT)	1000000	0.358	0.264	0.254	0.333
URLTran (BERT)	5000000	0.377	0.288	0.278	0.357
URLTran + LLM Labels (0.0 Ratio)	10000000	0.416	0.317	0.307	0.364
URLTran + LLM Labels (0.25 Ratio)	10000000	0.465	0.357	0.326	0.446
URLTran + LLM Labels (0.5 Ratio)	10000000	0.465	0.364	0.338	0.443
URLTran + LLM Labels (0.75 Ratio)	10000000	0.466	0.364	0.329	0.453
URLTran + LLM Labels (1.0 Ratio)	10000000	0.468	0.375	0.338	0.496
eXpose (Conv)	10	0.033	0.023	0.05	0.049
eXpose (Conv)	100	0.062	0.052	0.107	0.062
eXpose (Conv)	1000	0.097	0.059	0.093	0.074
eXpose (Conv)	10000	0.223	0.051	0.063	0.105
eXpose (Conv)	100000	0.256	0.042	0.049	0.129
eXpose (Conv)	1000000	0.262	0.107	0.103	0.175
eXpose (Conv)	5000000	0.302	0.153	0.143	0.229
eXpose + LLM Labels (0.0 Ratio)	10000000	0.315	0.168	0.157	0.237
eXpose + LLM Labels (0.25 Ratio)	10000000	0.444	0.316	0.266	0.45
eXpose + LLM Labels (0.5 Ratio)	10000000	0.436	0.308	0.259	0.445
eXpose + LLM Labels (0.75 Ratio)	10000000	0.422	0.28	0.231	0.426
eXpose + LLM Labels (1.0 Ratio)	10000000	0.45	0.318	0.268	0.453

Table A10
Model performance on Time Split

Model	Samples	Accuracy	Macro F1	Macro Recall	Macro Precision
BERTiny	10	0.206	0.104	0.27	0.128
BERTiny	100	0.307	0.155	0.401	0.15
BERTiny	1000	0.355	0.135	0.301	0.176
BERTiny	10000	0.432	0.177	0.228	0.197
BERTiny	100000	0.912	0.713	0.722	0.714
BERTiny	1000000	0.967	0.78	0.765	0.802
BERTiny	2500000	0.973	0.821	0.804	0.844
BERTiny	5000000	0.976	0.856	0.838	0.882
BERTiny + LLM Labels (0.0 Ratio)	10000000	0.987	0.867	0.858	0.878
BERTiny + LLM Labels (0.25 Ratio)	10000000	0.971	0.835	0.814	0.86
BERTiny + LLM Labels (0.5 Ratio)	10000000	0.98	0.858	0.841	0.878
BERTiny + LLM Labels (0.75 Ratio)	10000000	0.982	0.877	0.859	0.902
BERTiny + LLM Labels (1.0 Ratio)	10000000	0.608	0.546	0.546	0.605
GPT-3 Babbage	10	0.429	0.304	0.593	0.269
GPT-3 Babbage	100	0.603	0.409	0.716	0.343
GPT-3 Babbage	1000	0.771	0.62	0.793	0.534
GPT-3 Babbage	10000	0.969	0.818	0.831	0.809
GPT-3 Babbage	100000	0.973	0.837	0.829	0.849
GPT-3 Babbage	1000000	0.984	0.855	0.839	0.874
T5 Large	10	0.394	0.35	0.497	0.344
T5 Large	100	0.475	0.375	0.61	0.328
T5 Large	1000	0.688	0.542	0.715	0.489
T5 Large	10000	0.927	0.794	0.796	0.8
T5 Large	100000	0.959	0.821	0.824	0.821
T5 Large	1000000	0.966	0.772	0.73	0.838
T5 Large	5000000	0.975	0.835	0.807	0.874
URLTran (BERT)	10	0.122	0.093	0.192	0.166
URLTran (BERT)	100	0.396	0.272	0.598	0.238
URLTran (BERT)	1000	0.636	0.508	0.773	0.432
URLTran (BERT)	10000	0.944	0.757	0.812	0.72
URLTran (BERT)	100000	0.947	0.778	0.791	0.774
URLTran (BERT)	1000000	0.971	0.817	0.806	0.836
URLTran (BERT)	5000000	0.956	0.82	0.802	0.873
URLTran + LLM Labels (0.0 Ratio)	10000000	0.992	0.914	0.897	0.94
URLTran + LLM Labels (0.25 Ratio)	10000000	0.985	0.859	0.851	0.869
URLTran + LLM Labels (0.5 Ratio)	10000000	0.988	0.889	0.884	0.896
URLTran + LLM Labels (0.75 Ratio)	10000000	0.991	0.908	0.896	0.927
URLTran + LLM Labels (1.0 Ratio)	10000000	0.74	0.676	0.672	0.706
eXpose (Conv)	10	0.125	0.075	0.165	0.116
eXpose (Conv)	100	0.296	0.168	0.362	0.164
eXpose (Conv)	1000	0.306	0.193	0.327	0.215
eXpose (Conv)	10000	0.565	0.18	0.204	0.26
eXpose (Conv)	100000	0.549	0.156	0.147	0.399
eXpose (Conv)	1000000	0.904	0.539	0.478	0.744
eXpose (Conv)	5000000	0.928	0.662	0.601	0.784
eXpose + LLM Labels (0.0 Ratio)	10000000	0.978	0.838	0.809	0.878
eXpose + LLM Labels (0.25 Ratio)	10000000	0.964	0.78	0.748	0.823
eXpose + LLM Labels (0.5 Ratio)	10000000	0.971	0.823	0.796	0.858
eXpose + LLM Labels (0.75 Ratio)	10000000	0.976	0.842	0.811	0.886
eXpose + LLM Labels (1.0 Ratio)	10000000	0.49	0.413	0.397	0.477