Matching Pairs: Attributing Fine-Tuned Models to their Pre-Trained Large Language Models

Myles Foley[*1], **Ambrish Rawat**[2], **Taesung Lee**[2],
Yufang Hou[2], **Gabriele Picco**[2], **Giulio Zizzo**[2]
[1]Imperial College London, [2]IBM Research
m.foley20@imperial.ac.uk
{ambrish.rawat, yhou}@ie.ibm.com
{taesung.lee, gabriele.picco, giulio.zizzo2}@ibm.com

Abstract

The wide applicability and adaptability of generative large language models (LLMs) has enabled their rapid adoption. While the pre-trained models can perform many tasks, such models are often fine-tuned to improve their performance on various downstream applications. However, this leads to issues over violation of model licenses, model theft, and copyright infringement. Moreover, recent advances show that generative technology is capable of producing harmful content which exacerbates the problems of accountability within model supply chains. Thus, we need a method to investigate how a model was trained or a piece of text was generated and what their pre-trained base model was. In this paper we take the first step to address this open problem by tracing back the origin of a given fine-tuned LLM to its corresponding pre-trained base model. We consider different knowledge levels and attribution strategies, and find that we can correctly trace back 8 out of the 10 fine tuned models with our best method.

1 Introduction

Recent advancements in pre-trained large language models (LLMs) have enabled the generation of high quality texts that humans have difficulty identifying as machine generated (Wahle et al., 2022). While these pre-trained models can perform many tasks in the zero-shot or few-shot settings (Brown et al., 2020; Schick and Schütze, 2021), such models are often fine-tuned to improve their performance on various downstream applications (Peters et al., 2019; Pfeiffer et al., 2020). As of May 2023, there are more than 209,000 models hosted on Huggingface[1] and more than 12,000 of them belong to the "*text generation*" category. Many generation models are fine-tuned from the open-access pre-trained base models such as XLNet (Yang et al., 2019), BART (Lewis et al., 2020), or GPT-J (Wang and Komatsuzaki, 2021) whose training typically requires significant computational resources.

While the proliferation of text generation models has led to the performance improvement for a wide range of downstream applications such as text summarization and dialogue systems, it has also been repeatedly shown that these pre-trained or fine-tuned LLMs can facilitate the creation and dissemination of misinformation at scale (Weidinger et al., 2021), and the manipulation of public opinion through false "majority opinions" (Mann, 2021). In response, laws like the EU's Digital Services Act (DSA)[2] aim at tackling these issues by enforcing procedural accountability and transparency for responsible use of AI-based technologies. These growing demands for AI forensics require the development of methods for establishing model ownership, protecting intellectual property, and analyzing the accountability of any violations.

In this work, we systematically investigate *LLM attribution*, a novel task recently proposed at the first "Machine Learning Model Attribution Challenge (MLMAC)"[3], which aims to link an arbitrary fine-tuned LLM to its pre-trained base model using information such as generated responses from the models. Through LLM attribution, regulatory bodies can trace instances of intellectual property theft or influence campaigns back to the base model. However, determining attribution for fine-tuned LLMs can be challenging as base models often have similar architectures and overlapping training data. For instance, THEPILE (Gao et al., 2020a), a large data set that consists of 22 smaller, high-quality datasets, with a total size of 825 GB, was included into the training data for both GPT-J (Wang and Komatsuzaki, 2021) and OPT (Zhang et al., 2022).

[*] Work done during internship at IBM Research.
[1] https://huggingface.co/models
[2] https://eur-lex.europa.eu/eli/reg/2022/2065/oj
[3] https://mlmac.io/

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7423–7442

In this paper, we cast LLM attribution as a classification problem under two different conditions and develop machine learning solutions to tackle the task. Through extensive experimentation, we examine the trade-offs of different approaches and analyze the influence of the fine-tuning datasets and training iterations on LLM attribution. To the best of our knowledge, this is the first comprehensive study for LLM attribution. We also make the code and models public available for future research[4].

2 Related work

While there is no systematic study of the fine-tuned LLM attribution task, there are related streams of work in the literature.

Watermarking Watermarking is the most prevalent technique to ease the problem of proving model ownership by the first party (Tuan et al., 2021; Uchida et al., 2017; Chen et al., 2019; Le Merrer et al., 2020). When given a particular input, a watermarked model returns a specific output. This allows for protection of proprietary models by having a designed input-output to always attribute the model to its original source. However, such techniques require specific training modifications which might not be readily available to a third party auditor, but they have given rise to many attacks that can evade or render such measures ineffective (Wang and Kerschbaum, 2019; Quiring et al., 2018; Hitaj et al., 2019).

Attribution to training data Generative models re-use aspects of their training data in their output at inference (Shokri et al., 2017; Hisamoto et al., 2020; Somepalli et al., 2022). This has given rise to membership inference attacks, and concerns over the re-use of training data in applications like image generation, where copyright infringement may occur (Somepalli et al., 2022).

Attribution of LLM outputs Another form of attribution attempts to map LLM outputs to identifiable sources (Rashkin et al., 2022). This is crucial when models are required to output factual information such as in search engines (Nakano et al., 2022). As such, (Rashkin et al., 2022; Menick et al., 2022) provide a benchmarks and evaluation frameworks for attribution of LLM outputs that make use of human annotations.

[4] https://github.com/IBM/model-attribution-in-machine-learning

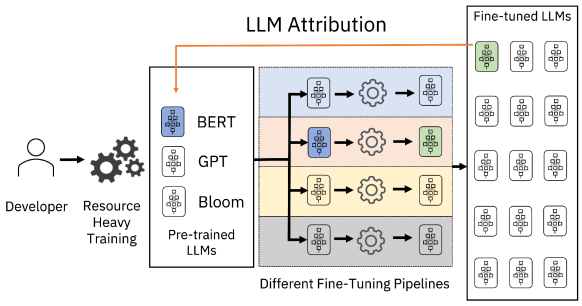


Figure 1: Example of the LLM Attribution Task, illustrating the pipeline of developing B, fine-tuing to F and attribution of m_f to m_b.

3 LLM Attribution

Definitions Formally, given a vocabulary Σ, an LLM m is a function from Σ^N to Σ^{N+1} where N is the input length. With auto-regression, m can generate an output of the arbitrary length. In this paper, we use the "generate" functionality implemented in Huggingface transformers library that leverages heuristic generation rules to reduce repeated words and determine when to stop. The input text expecting a certain type of output is often called a *prompt* (*e.g.*, "The capital of Canada is"), and the *response* completes the prompt (*e.g.*, "Ottawa"). Thus, for a prompt $p \in \Sigma^N$, an LLM m can be used to obtain a response $m(p)$.

In this work, we consider two collections of LLMs — the first one is a set B of pre-trained base LLMs, and the second one is a collection F of fine-tuned LLMs. We assume that every model $m_f \in F$ was effectively obtained by fine-tuning a model $m_b \in B$.

Problem Formulation & Challenges The goal of LLM attribution is to design a function $f : F \rightarrow B$ that maps a given fine-tuned model $m_f \in F$ back to its corresponding base model $m_b \in B$ (Figure 1). Fine-tuning a given base model m_b involves making the choices regarding the fine-tuning dataset and approach. This can blur the commonalities between m_f and m_b making attribution an inherently challenging task. For example, a fine-tuning dataset can be too dissimilar from the pre-training dataset of the base model, or too similar to that of another base model. Similarly, the choice of a fine-tuning algorithm and its hyperparameters, like the number of iterations, can result in catastrophic forgetting (Chen et al., 2020).

The difficulty in designing a mapping f is also tightly linked to the type and degree of allowed

access to the models in B and F and the amount of resources available for developing the method. In general, we assume that the developer of an attribution system can only query the LLMs as a black box to obtain the generated responses, and has limited access to models in F. We speculate this to be true for real-world settings where the producers of pre-trained base models, maintainers of model zoos, or an external auditor are incentivised to develop such attribution systems. In such scenarios, they may only have limited access to the API of fine-tuned models which will typically be owned by a third-party. Other constraints may arise from the amount of resources available for developing attribution systems. For instance, an external auditor may not have the domain expertise or computation resources to benefit from the insights from other fine-tuning pipelines. Similarly, the developer is assumed to have no knowledge of the fine-tuning datasets and approaches used to obtain the models in F, as in these cases attribution may be easily achieved by replicating the setup locally and comparing the obtained models with those in F. In addition to these assumptions, we consider the following two knowledge levels available with the developer of an attribution system.

- **Universal knowledge** K_U: This allows the developer access to universal knowledge about models in B. This allows the analysis by a human expert, as well as computing the perplexity of the input. Moreover, the developer can build an additional set of fine-tuned models A, or even the capability to train such models. This enables building a supervised attributor using A as a training dataset.

- **Restricted knowledge** K_R: We do not have access to A, and can only query the models in B as a black box to get the responses.

4 Attribution Methods

We approach the LLM attribution problem as a classification task. Essentially, LLM attribution requires identifying the certain robust or latent characteristics of a pre-trained base model within the given fine-tuned model. The fine-tuned model may retain unique aspects in the pre-training data like events and vocabulary of a specific time period, or the fluency of the base model in a certain domain.

In particular, as shown in Figure 2 we build a classifier h_{m_b} testing a response for each pre-trained base model m_b to decide if a given fine-tuned model m_f retains the characteristics of m_b, following the one-vs-rest (m_b or others) scheme. Then, we aggregate the result to pick the top-1 base model with the majority voting method. In other words, we take m_f such that $\sum_{p \in P} h_{m_b}(m_f(p))$ is maximized, where P is a set of prompts.

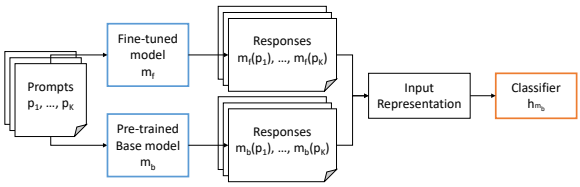


Figure 2: An example configuration of a one-vs-rest classifier h_{m_b} using both base model m_b and fine-tuned model m_f.

The task can be further broken down into two steps for each base model m_b and its classifier h_{m_b} including (1) characterizing the target base model m_b and representing the input to the classifier (Section 4.1.1), (2) selecting the prompts (Section 4.1.2), and (3) designing the classifier (Section 4.2).

4.1 Model Characterization and Input Representation

In this step, we characterize an LLM (fine-tuned or base model), and prepare the input to the classifier h_{m_b}. One piece of evidence of attribution lies in exploiting the artefacts of a pre-trained LLM that are expected to persist through the fine-tuning process and are inherited by their fine-tuned counterparts. For instance, a distinctive feature of RoBERTa (Liu et al., 2019) is the sequence length limit of 512 which is often inherited by its fine-tuned versions. The task characteristics and associated training data may also help distinguish different LLMs. For example, LLMs trained for specific tasks like chat bots or code generation will have characteristically different output spaces. They may also have unique aspects in their training data like a specific language or markers such as data collected over specific time period.

While feature engineering can extract a usable set of features, it is prone to bias, and less adaptable, and it also requires deep knowledge about B. Thus, we leverage the embeddings of the prompts and responses to learn and exploit such knowledge.

4.1.1 Input Representation

Our goal is to train a classifier to capture the correlations between an arbitrary response and the base model m_b. For example, with a prompt p, this could capture the relationship between a response $m_b(p)$ and m_b. Similarly, we can capture the relationship between a response $m_f(p)$ and m_b where m_f is obtained by fine-tuning m_b. Assuming that such correlations are preserved in a base model and fine-tuned model pair, we use it to determine the attribution of a fine-tuned LLM.

Given a set of prompts $p_1, \ldots, p_K$, there are multiple ways to prepare them for the classifier. We can apply the target base model, or fine-tuned model to get the responses, and concatenate the prompt and its response. Specifically, we list the input representations we consider as follows:

- Base model only ($\mathbf{I_B}$): "$p_i\ m_b(p_i)$"
- Fine-tuned model only ($\mathbf{I_F}$): "$p_i\ m_f(p_i)$"
- Base model + fine-tuned model ($\mathbf{I_{B+F}}$): "$p_i\ m_b(p_i)$ <SEP> $p_i\ m_f(p_i)$"
- Separate embeddings for base model and fine-tuned model.

We embed these concatenated sentences using BERT computed by a `best-base-multilingual-cased` model[5] except for the last approach that embeds the components separately for margin-based classifier TripletNet described in Section 4.2. Note that all reference to a fine-tuned model m_f during training are actually sampled from another set A of fine-tuned models under K_U assumption as we assume only sparse access to m_f. Also, I_B takes the responses from m_f during prediction to test if the responses share the same characteristics that this classifier learned about m_b.

4.1.2 Prompt Selection

While many corpora to pre-train LLMs provide prompts, they might not be all useful to predict the base model. Thus, we aim to test and select prompts with more distinctive outcome. Our prompt selection strategy is driven to help best characterise base models. We first collect the list of datasets used in training each base model, identifying unique aspects of datasets that can help identify a base model. Intuitively, one might expect such unique prompts or 'edge cases' to bring out the distinctive aspects in the subsequent fine-tuned models. Specifically, we first identify the unique categories of prompts (e.g. different languages) present in different datasets and sample from this set.[6]

More specifically, we consider three approaches: a small set (**P1**) of *edge cases* that are distinct to each corpus, a naive collection (**P2**) of prompts, and reinforcement learning to select a subset (**P3**) from the edge cases.

While the naive collection of the 10,000 prompts from ThePile corpus and manually selecting a set of prompts unique to each training dataset is clear, we can also use reinforcement learning to optimize the selection using the classification result. More specifically, we train an agent for each h_{m_b} that can supply prompts for attribution inference. During the training episodes, the agent is rewarded for prompts whose responses lead to correct attribution. The reinforcement learning setup for this problem is defines as follows:

- **State.** A feature space consisting of the classification of the prompt, and an embedding of the prompt response computed by `best-base-multilingual-cased`.
- **Action.** Selecting one of the prompts from **P1**.
- **Reward.** Using a sparse reward function we reward (+1) for correct classification and penalise (-10) for incorrect classification.
- **Episode.** 20 possible actions.

At the start of each episode we are able to randomly select one of the models that the classifier was trained on, thus the RL agent learns to generalise to a variety of different models. We implement the RL agent using the Proximal Policy Optimisation (PPO) method (Schulman et al., 2017).

We can use these collected prompts in a few different ways. A simplistic approach is using each set **P1**, **P2** or **P3** individually. Another approach **P1+P2** trains the classifier with **P2**, and then fine-tune with **P1** to leverage both of them (**P3** is already a subset of **P2**) and we find this is promising in our experiments. See Appendix D for details of the prompts used from THEPILE for this combination approach.

[5] https://huggingface.co/bert-base-multilingual-cased

[6] Prompt selection is complex and the numerous dimensions of fairness and robustness of such schemes are useful for further investigation of the LLM attribution problem. However, we believe them to be out of scope of this first systematic study on LLM attribution.

4.2 Classifier Architecture

We consider a one vs rest setup where for each base model m_b we train a binary classifier $h_{m_b} : \Sigma^M \rightarrow \{0, 1\}$ which takes as input a response $s \in \Sigma^N$, optionally with additional tokens, and predicts a score that reflects its association to the based model m_b. Single embeddings prepared in Section 4.1.1 can be straightforwardly used in a simple classifier. We fine-tune the BERT model used for the embedding to make the binary prediction with cross-entropy loss. Given the one-vs-rest approach the positive samples for an h_{m_b} are repurposed as negative ones for the rest of the classifiers h_{m_l} for $m_l \in B \backslash \{m_b\}$. The best average score thus obtained is used to establish the attribution for m_f.

We also consider TripletNet (Wei et al., 2021) based classifiers that use a margin-based loss function using the separate embeddings of the base and fine-tuned model responses. The TripletNet is able to make predictions by taking in a single sentence, computing the output embedding, and finding the closest embedding from the training set and using the label of the training sentence as a prediction. The cosine distance between the anchor input, positive example, and negative example are then computed as the loss. We adopt the margin parameter 0.4 from the original paper (Wei et al., 2021).

5 Experiments

5.1 Experiment Setup

For training the attribution models h_{m_b} we make use of popular text corpora including: GitHub, The BigScience ROOTS Corpus (Laurençon et al., 2022), CC-100 (Conneau et al., 2020), Reddit (Hamilton et al., 2017), and THEPILE (Gao et al., 2020b).

We also use a variety of prompt sizes for attribution (150 to 10,000), and datasets (IMDB Reviews (Maas et al., 2011), GLUE (Wang et al., 2018), Tajik OSCAR (Abadji et al., 2022), and Amazon Multilingual (Keung et al., 2020).

To provide a wide didactic range of models for our approaches we utilise 10 pre-trained LLMs to create B and corresponding fine-tuned models (Table 1): Bloom (Scao and et al., 2022), OPT (Zhang et al., 2022), DialoGPT (Zhang et al., 2020), DistilGPT2 (Sanh et al., 2020), GPT2 (Radford et al., 2019), GPT2-XL (Radford et al., 2019), GPT-NEO (Black et al., 2021), CodeGen (Nijkamp et al., 2023), XLNET, MultiLingual-MiniLM (Wang et al., 2021). These models provide different architectures, parameter sizes, and tasks to offer a variety of model behaviors.

$m_\#$	Base Model	Fine-tuning dataset
0	bloom-350m	common_gen (Lin et al., 2020)
1	OPT-350M	Pike, CYS, Manga-v1
2	DialoGPT-large	Persuasion For Good Dataset (Wang et al., 2019)
3	distilgpt2	wikitext2 (Merity et al., 2016)
4	GPT2-XL	the Wizard of Wikipedia dataset (Dinan et al., 2019)
5	gpt2	Wikipedia dump, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl and News Crawl.
6	GPT-Neo-125m	Cmotions - Beatles lyrics
7	xlnet-base-cased	IMDB (Maas et al., 2011)
8	multilingual-MiniLM-L12-v2	Unknown
9	codegen-350M	Zhu et al. (2022)

Table 1: Fine-tuned models, their original base models and the datasets they are fine-tuned on.

We consider a one-to-one mapping from B to F (and A), thus F and A contain ten models each. We utilise open-source models that are implemented in the Huggingface library to form the sets of F and A. We select A and F such that the fine-tuning dataset, and model configuration are known to us, of these we select the most popular by number of downloads. We provide further details of these in Appendix B.

We take the top-1 result for each m_b as mentioned in Section 4 and check its correctness. We use F1 and ROC curves as additional metrics. These are calculated using prompt-level attribution calculated per m_b (as in Figure 8), and we use an average per h_{m_b} (as in Figure 3). Each of the attributors h_{m_b} described is ran once to determine the attribution of m_f to m_b. Training is conducted using a single NVIDIA A100 GPU.

5.2 Compared Approaches

We consider different configurations for BERT classifiers based on the input representations $\mathbf{I_B}$, $\mathbf{I_F}$ or $\mathbf{I_{B+F}}$, and the prompts used **P1**, **P2**, **P3** or **P1+P2** described in Section 4.1.1.

We also consider the margin classifier TripleNet (Section 4.2), and the following heuristic approaches.

- *Perplexity*: A measure of how confident a model is at making predictions, this can be

leveraged for measuring attribution by computing the perplexity of m_b relative to the response of m_f to prompt p.

- *Heuristic Decision Tree (HDT)*: Using K_U we can use knowledge of B to create a series of discriminative heuristics to categorise F as used by the winning solution to the first MLMAC[7].

- *Exact Match*: Attribute responses m_f to m_b when both models respond the same to a prompt. Using the argmax of these attributions to attribute m_f to m_b.

For detailed descriptions of the heuristic approaches, please refer to Appendix A.

5.3 Attribution Accuracy

Here, we examine the attribution abilities of the compared approaches shown in Table 2. Under K_U conditions the baselines of Perplexity and HDT are only able to correctly attribute 1 and 5 models respectively. Perplexity fails to capture the subtly of attribution, as repetitive responses lead to lower perplexity and so incorrect attribution. The HDT particularly fails to account for overlap in pre-training and fine-tuning. For instance, DialoGPT-Large and m_{f3} (fine-tuned version of distilgpt2) respond in similar short sentences that leads to incorrect attribution. The TripletNet baseline performs poorly, only correctly attributing 3 of the models. Both BERT based attributors are able to attribute more models correctly in comparison to the baselines.

Examining the models at K_R shows similar performance. The exact match correctly attributes 5 models and BERT+I_B identifies 6 models. BERT+I_B+$P1+P2$ attributor is the most successful by correctly attributing 8 models. Note that this model is the most expensive to train as we have to query a large number of prompts.

We compare the ROC curves for BERT based attributor defined under each K in Figure 3. We provide plots of h_{m_b} in each variant in Appendix C. It is interesting to note that the models under K_R have shallower curves than their K_U counterparts, yet these K_R models lead to the same or higher number of correct attributions. This is likely due to the 'noise' that gets added to responses of A from their separate fine-tuning task, $\mathcal{T}_A$. This noise moves the responses of m_a further from m_f (and by extent m_b). As such responses from m_b are closer to m_f than m_a. This makes the attributors predict more negative samples correctly under K_U as there is greater disparity in response between m_a and m_f, leading to a higher AUC; but also to miss-attribution of m_f at inference. Hence, it is unsurprising that the pretrained K_R has the lowest AUC of any model, yet it leads to the highest attribution accuracy in Table 2 as it is trained on responses of m_b which is closer in the latent space to responses of m_f than m_a.

[7] https://pranjal2041.medium.com/identifying-pretrained-models-from-finetuned-lms-32ceb878898f

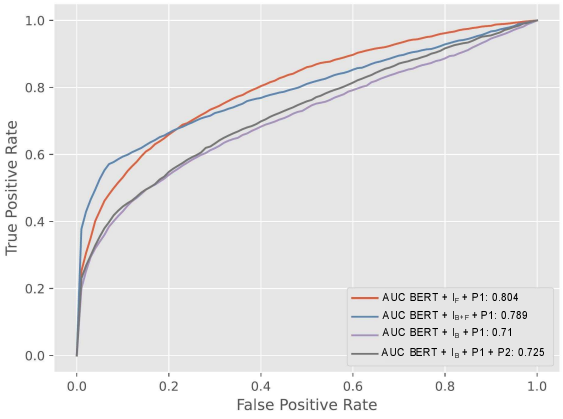


Figure 3: Average ROC plots for each classifier at each knowledge level.

Attribution Method	K	$m_\#$ 0	1	2	3	4	5	6	7	8	9	TP
HDT	K_U	✓	✓	✗	✗	✓	✗	✗	✓	✗	✓	5
Perplexity	K_U	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	1
TripletNet + $P1$	K_U	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	3
BERT + I_F + $P1$	K_U	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	6
BERT + I_{B+F} + $P1$	K_U	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	6
Exact matching	K_R	✓	✓	✓	✗	✗	✗	✓	✗	✗	✓	5
BERT + I_B + $P1$	K_R	✓	-	✓	-	✗	✗	✓	✓	✓	✓	6
BERT + I_B + $P3$	K_R	✓	✗	-	✗	✗	✓	✓	✓	-	✓	5
BERT + I_B+$P1$+$P2$	K_R	✓	✓	✓	-	✗	✓	✓	✓	✓	✓	**8**

Table 2: Model Attributions on $m_\#$ from the different methods. Dashes (–) are used when multiple models (m_f) are attributed to m_b. TP denotes True Positives.

> ***Lesson Learned:*** *Even under reduced knowledge level, pre-training was found to be the factor that contributed to the highest attribution performance.*

5.4 Effects of Prompt usage

The number of prompts available to an attributor for classification can have an influence on the attribution performance: we hypothesize that increasing the number of prompts used results in a clearer signal as to the finetuned to base model attribution.

We train BERT attributors under the K_R con-

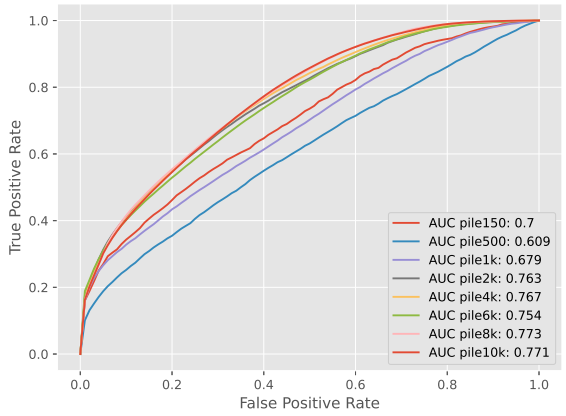


Figure 4: Mean ROC for varying quantities of prompts.

Number of Prompts	$m_\#$ 0	1	2	3	4	5	6	7	8	9	TP
150	✗	✓	✗	-	-	-	✓	✗	-	✓	3
500	✗	✗	✗	-	-	-	-	-	✓	-	1
1000	✗	✗	-	-	-	-	-	✗	✓	-	1
2000	✗	✓	✓	✓	-	-	-	✓	✓	✓	6
4000	✗	✓	✓	✓	✗	✗	✗	✗	✓	✓	5
6000	✗	✓	✓	-	-	-	-	✗	✓	✓	4
8000	✗	✓	✗	-	-	-	-	✗	✓	-	2
10000	-	✓	✗	-	-	-	-	✓	✓	✓	4
BERT + I_B + $P1$ + $P2$	✓	✓	✓	-	✗	✓	✓	✓	✓	✓	8

Table 3: Model Attributions on F using a varying number of prompts from The Pile.

dition, as the K_R pretrained model performed the strongest. For these experiments we do not use RL prompt selection.

The results of this experiment are shown in Figure 4. By increasing the number of prompts that a classifier is able to use for classification, we see that there is an improvement in the AUC, with diminishing returns from 6,000 prompts onward.

Increasing the number of prompts improves the AUC, yet does not lead to direct improvement in terms of the attribution accuracy as shown in Table 3. In fact, increasing the number of prompts used for classification leads to a highly variable performance. None of the models that directly use these prompts (150 - 10K prompts from the pile) are able to improve or even match that of the pretrained model using 150 prompts from Table 1.

> ***Lesson Learned:*** *Increasing the number of prompts for attribution does not lead to reliable improvements in the number of models correctly attributed.*

5.5 Effects of pretraining attributors

We next aim to investigate how the size of the pretraining data effects the performance of the attribution, as while using increasingly large data for direct attribution may not improve performance, Section 5.3 shows that using it as pretraining data does improve attribution.

Number of Prompts	$m_\#$ 0	1	2	3	4	5	6	7	8	9	TP
150	✗	-	✓	✓	✗	✗	✓	✓	✗	✓	5
500	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	6
1000	✗	✓	✓	✓	✗	✓	✗	-	✓	✓	6
2000	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	7
4000	✓	✓	✓	-	✓	✓	✓	✓	✗	✓	8
6000	✓	✓	✓	-	✗	✓	✓	✓	✓	✓	8
8000	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	7
10000	✓	✓	✓	-	✗	✓	✓	✓	✓	✓	8
BERT + I_B + $P1$ + $P2$	✓	✓	✓	-	✗	✓	✓	✓	✓	✓	8

Table 4: Model Attributions on F from the models pretrained on different portion from P2, and then finetuned with P1.

To this end each model discussed in Section 5.4 is finetuned under K_R, varying the size of pretraining data from 150 prompt responses to 10,000.

We report the results of the experiment in Figure 5. In Figure 5a we see that the finetuned models are able to improve over the equivalent models in Figure 4. Yet they do not improve on the AUC of models trained under K_U conditions.

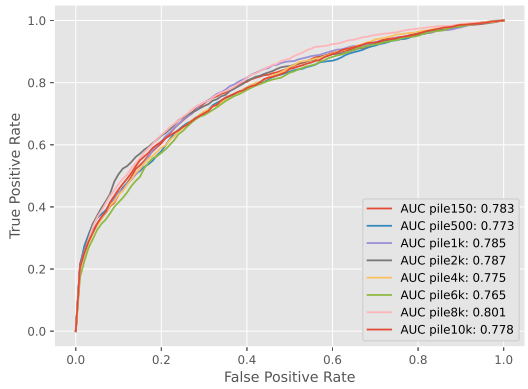
We see from Figure 5b that increasing the number of prompts minimally improves the precision and recall of attribution, with little correlation between number of prompts, even of a varied set like THEPILE. Whilst these pretrained-finetuned attributors are able to improve on the precision of the attributor using manual selected prompts, however they are unable to improve on the recall.

What is most important for this task, however, is the ability of attribution, hence we also determine the model attributions for each model in Table 4. The models that have been pretrained on a larger number are able to outperform the K_R model of Section 5.3 attributing 8 models correctly in the the models pretrained on 4k and 6k prompts.

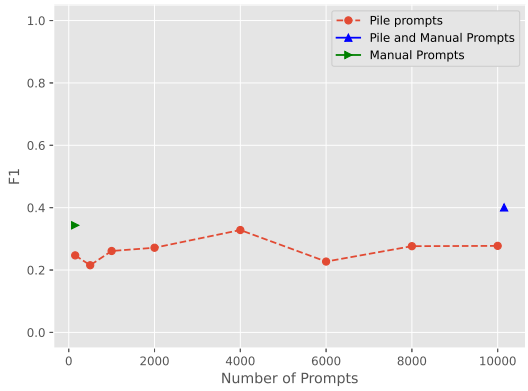
> ***Lesson Learned:*** *Pretraining attributors is vital to improve the attribution performance. However, this has to diminishing returns in terms of correct attributions and AUC.*

5.6 Effects of Finetuning on Attribution

The type and duration of the finetuning conducted on a base model B can effect attribution performance. To investigate this we use of two base models: distilgpt2 and Multilingual-MiniLM and finetune them using three datasets: IMDB (Maas et al., 2011), GLUE (Wang et al., 2018), Amazon



(a) ROC and AUC of attributors with pretrained models using different pretraining data sizes.



(b) F1 score of attributors with pretrained models using different pretraining data sizes. Manual=150 selected.

Figure 5

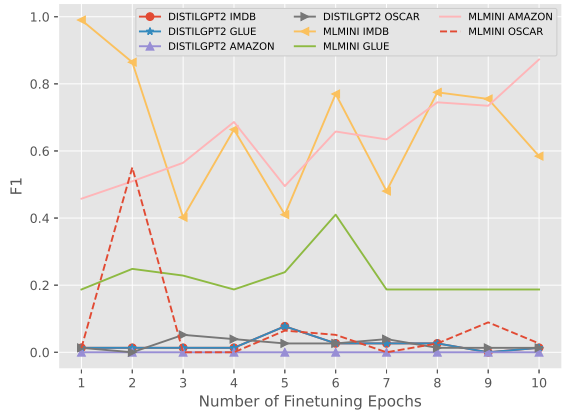


Figure 6: F1 scores of DistilGPT2 and MLMINI attributors.

reviews Multilingual (Keung et al., 2020), and the Tajik language subset of OSCAR (Abadji et al., 2022).

Using such datasets more closely models the realistic attack scenario where common pre-training prompt sets are used in an attempt to determine attribution, and fine-tuning datasets are often proprietary and/or unique to the application. Conducting experiments in this scenario in a controlled setting allows us to study the effect of finetuning on attribution in terms of (a) number of epoch and (b) size of dataset.

Effect of Finetuning Epochs: Firstly, we study the effect of the number of finetuning epochs has on attribution. Figure 6 shows the F1 score of the MLMini and distilgpt2 attributors when trying to attribute the finetuned base models.

The MLMini attributor is greatly affected initially by MLMini being finetuned on IMDB, however as with the model finetuned on Amazon reviews there is an increase in attribution performance with increasing finetuning epochs. Conversely, the MLMini model finetuned on GLUE MNLI had minimal change in performance only with anomalous increased F1 score at epoch 6.

However, when trying to attribute MLMINI finetuned with the Tajik subset of OSCAR we see that the F1 score is significantly worse. We speculate that AMAZON and IMDB datasets are similar to the pretraining dataset of MLMini (CC-100) and that the AMAZON reviews, with its 6 languages, are the most similar to this. In fact, the CC-100 is likely to have an overlap in the data distribution of all three of these datasets as all are openly available. As there is no Tajik in CC-100 it is out-of-distribution (OOD) of MLMINI's pretraining dataset, which leads to the poor performance in attribution.

With the attributor for distilgpt2 there is poor performance in all datasets regardless of the number of epochs. This follows due to the finetuning datasets being OOD relative the the pretraining data of distilgpt2 which used the OpenWebTextCorpus. As OpenWebTextCorpus is mostly in English, finetuning in other languages such as those in the AMAZON dataset, makes attribution harder.

Lesson Learned: *The attribution performance is dominated by the similarity of the fine-tuning dataset to the pre-training dataset, rather than the amount of fine-tuning conducted.*

Effects of Dataset Size: In addition to the number of finetuning epochs we consider the overall

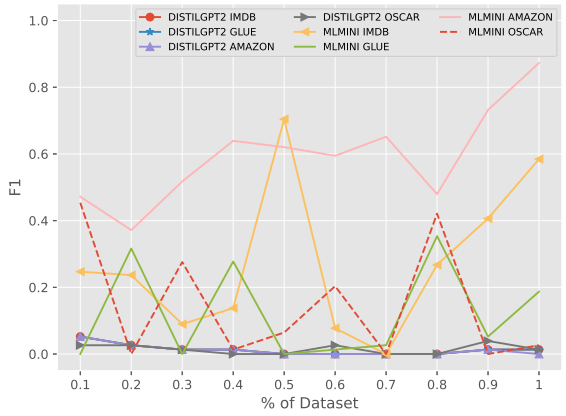


Figure 7: F1 of DistilGPT2 and MLMINI attributors under varying dataset size, relative to original dataset.

size of the finetuning set on attribution. We report the results of using a fixed 10 epochs and varying the finetuning dataset size in Figure 7. We can see similar effects as in Figure 6, that the OOD datasets for Distilgpt2 lead to poor F1 scores, and consequently, poor attribution results.

For MLMINI we see similar performance on IMDB and AMAZON (two of the in-distibution datasets) with an increased F1 as the dataset size increases. When finetuning on OSCAR and GLUE the F1 score shows a minimal correlation with dataset size. This again follows from Figure 6. OSCAR is OOD for MLMINI, which makes attribution significantly harder. Similarly GLUE offers the most varied dataset making attribution harder and giving lower F1.

> ***Lesson Learned:*** *Training on a richer dataset broadly improves results if it is within distribution.*

Effects of Dataset: Across Figures 6 and 7 we see the effect of different finetuning datasets on the ability to attribute to base models.

We can observe the effect of the finetuning datasets on the ability to attribute to base models in Figures 6 and 7. These figures show the distribution of the dataset greatly affects attribution. Finetuning datasets that are completely out of distribution in relation to the original pre-training dataset severely impact attribution performance. This is particularly apparent in MLMINI where finetuning on OSCAR leads to poor attribution performance in Figure 6 and 7.

Both base models finetuned with GLUE also make attribution harder. We reason that this is due to the broad range of prompts that are not typical of a finetuning dataset. This leads the model to produce generic responses to the targeted prompts used for attribution.

> ***Lesson Learned:*** *The most significant impact on attribution is the distribution and variety of the finetuing dataset.*

6 Conclusion

In this work we have taken initial steps in the LLM attribution problem. We study LLM attribution in K_U and K_R settings which limit access to B and F to different levels. We argue this prevents trivial solutions in white-box settings, and provides an interesting and realistic study of LLM attribution.

We have considered a variety of different LLMs that are trained on different datasets, and for different purposes. We postulate that the 10 different LLMs provide a didactic range of models for LLM attribution. In our experiments, we have used pre-existing LLMs that have been fine-tuned by the open-source community to demonstrate the applicability of our methodology. To mitigate the potential for bias this causes, we have tried out best to ensure the fine-tuning task and dataset of such models is known. In addition, we fine-tune a subset of these models in an ablation study, which demonstrates the effect that such fine-tuning has on LLM attribution in a controlled environment. Our ablation study also studies the effect that OOD fine-tuning datasets have on attribution. This mitigates the effect of only fine-tuning within distribution (of the pre-training data).

Overall, our work contributes to the growing understanding of LLM attribution, laying the foundation for future advancements and developments in this domain.

Limitations

We have considered a variety of different LLMs in order to study attribution. However we have only considered a small sample of the different LLM architectures and training strategies. This has been with a view to using a small but diverse set of LLMs. Of these 10 base models, we tested our approach to attribution on a controlled set of fine-tuned models. While a study that considers a wider variety and larger scale of fine-tuned models would be beneficial to the problem of attribution, the computation resources limited our study.

Furthermore, in our assumptions in this work we consider that there is a one-to-one mapping between m_f and m_b. However, this is not necessarily the case. There could be an m-to-n mapping and also a model may be present in one set, but not the other.

We believe there is rich space for further research in this area that can address these limitations, and further develop the problem of attribution.

Ethics Statement

In the discussion we have highlighted how the techniques for attributing fine-tuned models to their pre-trained large language models can be used as a tool to mitigate issues such as violation of model licenses, model theft, and copyright infringement, but this is only a subset of the issues related to authorship attribution. The increasing quality and credibility of LLM generated text has recently highlighted ethical issues such as plagiarism[8] or the banning of users for submitting AI generated responses to answer questions. [9] Even within the scientific community discussions are arising related to topics such as the authorship of papers or codes, who owns what is it generated? Many AI conferences have banned the submission of entirely self-generated scientific papers.[10]

These are some examples of controversial situations, but the use of AI-generated content has ethical implications in several domains that depend on the specific context and application. It is therefore crucial, as a first step to tackle these ethical issues, to ensure that any AI-generated contents are clearly labeled as such and are not presented as original work without proper attribution (whether it's a person or a base model).

Acknowledgements

This work was supported by European Union's Horizon 2020 research and innovation programme under grant number 951911 – AI4Media.

[8] New bot ChatGPT will force colleges to get creative to prevent cheating, experts say

[9] AI-generated answers temporarily banned on coding Q&A site Stack Overflow

[10] Top AI conference bans use of ChatGPT and AI language tools to write academic papers

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the Thirteenth Language journal and Evaluation Conference*, pages 4344–4355, Marseille, France. European Language Resources Association.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting Training Data from Large Language Models. In *Proceedings of the 30th USENIX Security Symposium*. arXiv.

Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. DeepMarks: A Secure Fingerprinting Framework for Digital Rights Management of Deep Learning Models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, ICMR '19, pages 105–113, New York, NY, USA. Association for Computing Machinery.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Nathan Coooper, Artashes Arutiunian, Santiago Hincapié-Potes, Ben Trevett, Arun Raja, Erfan Hossami, and Mrinal Mathur. 2021. Code Clippy Data: A large dataset of code data from Github for research into code language models.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents.

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. 2020a. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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. 2020b. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. ArXiv:2101.00027 [cs].

Shivali Goel, Rishi Madhok, and Shweta Garg. 2018. Proposing Contextually Relevant Quotes for Images. In *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 591–597, Cham. Springer International Publishing.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA. Curran Associates Inc.

Sorami Hisamoto, Matt Post, and Kevin Duh. 2020. Membership Inference Attacks on Sequence-to-Sequence Models: Is My Data In Your Machine Translation System? *Transactions of the Association for Computational Linguistics*, 8:49–63. Place: Cambridge, MA Publisher: MIT Press.

Dorjan Hitaj, Briland Hitaj, and Luigi V. Mancini. 2019. Evasion Attacks Against Watermarking Techniques found in MLaaS Systems. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, pages 55–63.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual Amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4563–4568, Online. Association for Computational Linguistics.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben Allal, Francesco De Toni,

Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Romero Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Vu Minh Chien, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Ifeoluwa Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Luccioni, and Yacine Jernite. 2022. The BigScience ROOTS Corpus: A 1.6TB Composite Multilingual Dataset. In *Proceedings of the 36st International Conference on Neural Information Processing Systems*.

Erwan Le Merrer, Patrick Pérez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv:1907.11692 [cs].

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Christopher B. Mann. 2021. Can conversing with a computer increase turnout? mobilization using chatbot communication. *Journal of Experimental Political Science*, 8(1):51–62.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes. ArXiv:2203.11147 [cs].

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. In *Proceedings of the 5th International Conference on Learning Representations*.

Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. 2022. Memorization in NLP Fine-tuning Methods. ArXiv:2205.12506 [cs].

Nur Azmina Mohamad Zamani, Jasy Suet Yan Liew, and Ahmad Muhyiddin Yusof. 2022. XLNET-GRU sentiment regression model for cryptocurrency news in English and Malay. In *Proceedings of the 4th Financial Narrative Processing Workshop @LREC2022*, pages 36–42, Marseille, France. European Language Resources Association.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. WebGPT: Browser-assisted question-answering with human feedback. ArXiv:2112.09332 [cs].

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis. *ICLR*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Erwin Quiring, Daniel Arp, and Konrad Rieck. 2018. Forgotten Siblings: Unifying Attacks on Machine Learning and Digital Watermarking. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 488–502.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. 2022. Measuring Attribution in Natural Language Generation Models. ArXiv:2112.12870 [cs].

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv:1910.01108 [cs].

Teven Le Scao and et al. 2022. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. ArXiv:2211.05100 [cs].

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, Los Alamitos, CA, USA. IEEE Computer Society.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2022. Diffusion Art or Digital Forgery? Investigating Data Replication in Diffusion Models. ArXiv:2212.03860 [cs].

Kai-Wen Tuan, Yi-Jyun Chen, Yi-Chien Lin, Chun-Ho Kwok, Hai-Lun Tu, and Jason S. Chang. 2021. Learning to find translation of grammar patterns in parallel corpus. In *Proceedings of the 33rd Conference on Computational Linguistics and Speech Processing (ROCLING 2021)*, pages 301–309, Taoyuan, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding Watermarks into Deep Neural Networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ICMR '17, pages 269–277, New York, NY, USA. Association for Computing Machinery.

Jan Philip Wahle, Terry Ruas, Frederic Kirstein, and Bela Gipp. 2022. How large language models are transforming machine-paraphrase plagiarism. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Tianhao Wang and Florian Kerschbaum. 2019. Attacks on Digital Watermarks for Deep Neural Networks. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. ISSN: 2379-190X.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649, Florence, Italy. Association for Computational Linguistics.

Jason Wei, Chengyu Huang, Soroush Vosoughi, Yu Cheng, and Shiqi Xu. 2021. Few-Shot Text Classification with Triplet Networks, Data Augmentation, and Curriculum Learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5493–5500, Online. Association for Computational Linguistics.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. ArXiv:2205.01068 [cs].

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Ming Zhu, Aneesh Jain, Karthik Suresh, Roshan Ravindran, Sindhu Tipirneni, and Chandan K. Reddy. 2022. XLCoST: A Benchmark Dataset for Cross-lingual Code Intelligence. ArXiv:2206.08474 [cs].

A Heuristic Approaches

A.1 Perplexity

Using the response of F we can calculate the perplexity of B relative to F. This can then be used as a measure of how confident B is in predicting F, where a lower perplexity would indicate higher confidence and attribution. In our initial experiments, we found this to be loose approximation of similarity between models in B and F. Moreover, this approach assumed stronger access which is typically not available in real-world settings as we discussed in Section 3.

Perplexity is a measure of how well a model is able to predict a sample. It has previously been used in analogous settings for extracting training data from language models (Carlini et al., 2021; Mireshghallah et al., 2022) to determine if a model is confident in its prediction of a sample. It is possible to leverage this for the purpose of attributing $\mathbb{F}$ to $\mathbb{B}$. By collecting responses of $\mathbb{F}$ to prompts we can calculate the perplexity of $\mathbb{B}$ relative to $\mathbb{F}$. Thus we can take the perplexity score as a measure of how confident $\mathbb{B}$ is in predicting the response of $\mathbb{F}$, we would expect lower perplexity to be an indication of higher confidence and therefore higher chances of attribution.

A.2 Heuristic Decision Tree

When it comes to generalisation, many LLMs share an equal footing owing to the massive size and intensive training backing their capabilities. However, when examined closely there are distinctive features that set them apart which can be detected via static or dynamic inspection of the model. For instance, LLMs with a larger number of parameters tend to take longer for inference. Similarly, length of response varies across LLMs, and some are prone to repetition (such as XLNET (Mohamad Zamani et al., 2022)). The task characteristics and associated training data may also help distinguish different LLMs. For example, LLMs trained for specific tasks like chat bots or code generation will have characteristically different output spaces. They may also have unique aspects in their training data like a specific language or markers such as data collected over specific time period. Much like watermarking, these can be used to craft prompts that can help reveal these unique artefacts [11].

While in principle many of these heuristics can be used for attribution, the practical development of such systems faces a range of challenges. First, these properties may not be preserved across the fine tuning process and therefore provide no meaningful insight for attribution. Second, these heuristics require a high level of expertise and knowledge which may not always be available. An external auditor working with the restricted knowledge of K_R may not be able to develop such solutions. Third, many of the properties of models in F can be easily obfuscated by the exposed API. For example it is fairly easy to normalise response times or post-process the responses to account for repetition. Moreover, an API may be simultaneously backed by multiple different models which would make the attribution even more challenging. Finally, LLMs often have overlapping datasets which can dilute many of the subtleties underlying these heuristics. This limits the applicability and scalability of such approaches for larger collections of B and F.

B Fine-tuned model Details

Here we provide details of the fine-tuned LLMs we use in sets A and F. Each of the LLMs is an open source implementation hosted on the Huggingface, we provide the link to the fine-tuned model. In Table 5 we show set F as FT models 0-9 inclusive, and set A from 10-19 inclusive. For each model we also provide the dataset used to fine-tune each of the LLMs.

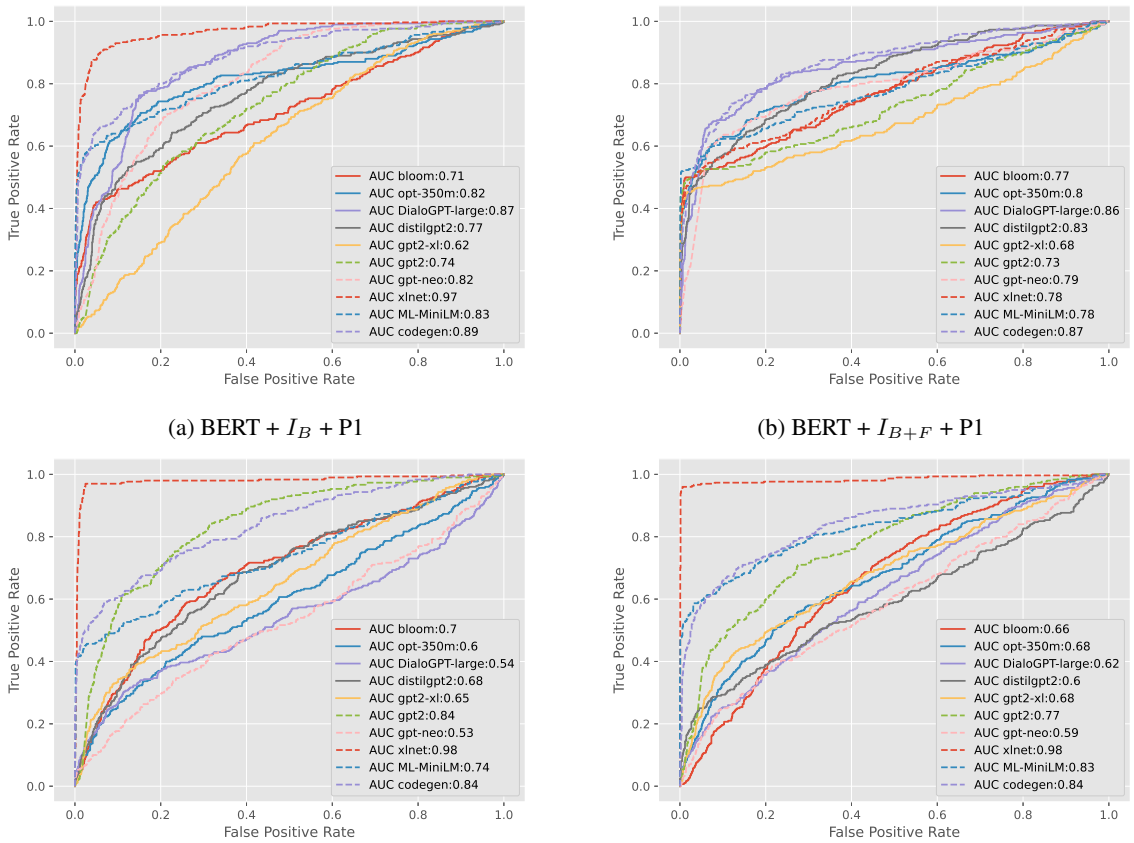
C AUC Curves

We provide the finegrained plots of how each individual h_{m_b} did in each experiment. Figure 8 shows the results from the experiment that measures the attribution accuracy under different K as discussed in Section 5.3. Figure 9 details the effect of using a different number of prompts for attribution under K_R, as discussed in Section 5.4. Finally Figure 10 shows the effect of varying the number of prompts for pretaining h_{m_b} (Section 5.5).

[11] Winning solution to the first MLMAC

FT model	Base Model	FT dataset
0	bloom-350m	common_gen (Lin et al., 2020)
1	OPT-350M	Pike, CYS, Manga-v1
2	DialoGPT-large	Persuasion For Good Dataset (Wang et al., 2019)
3	distilgpt2	wikitext2 (Merity et al., 2016)
4	GPT2-XL	the Wizard of Wikipedia dataset (Dinan et al., 2019)
5	gpt2	Wikipedia dump, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl and News Crawl.
6	GPT-Neo-125m	Cmotions - Beatles lyrics
7	xlnet-base-cased	IMDB (Maas et al., 2011)
8	multilingual-MiniLM-L12-v2	Unknown
9	codegen-350M	Zhu et al. (2022)
10	bloom-350m	Cmotions - Beatles lyrics
11	OPT-350M	GLUE (Wang et al., 2018)
12	DialoGPT-large	The complete works of Sir Arthur Conan Doyle
13	distilgpt2	Quotes-500K (Goel et al., 2018)
14	GPT2-XL	OSCAR (Abadji et al., 2022)
15	gpt2	IMDB (Maas et al., 2011)
16	GPT-Neo-125m	Code Clippy Data dataset (Coooper et al., 2021)
17	xlnet-base-cased	Rotten Tomatoes (Pang and Lee, 2005)
18	multilingual-MiniLM-L12-v2	https://www.tensorflow.org/datasets/catalog/wikipedia #wikipedia20200301bn
19	codegen-350M	BigPython dataset

Table 5: Fine-tuned models, their original base models and the datasets they are fine-tuned on.



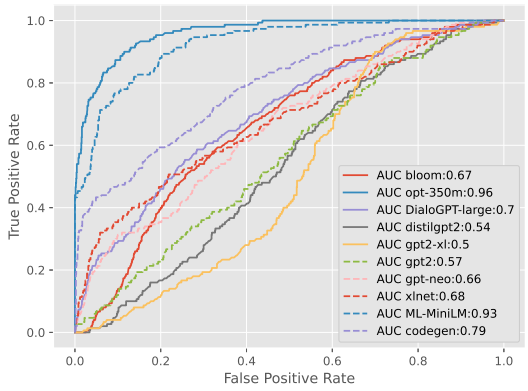
(a) BERT + I_B + P1

(b) BERT + I_{B+F} + P1

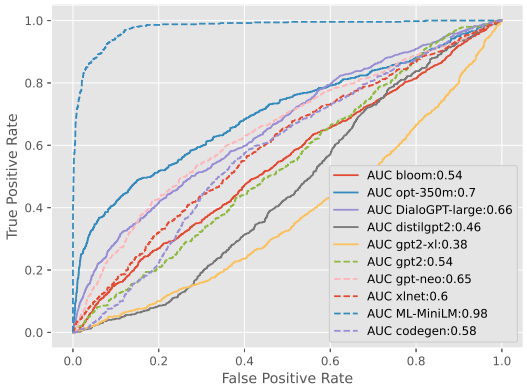
(c) BERT + I_B + P1

(d) BERT + I_B + P1+P2

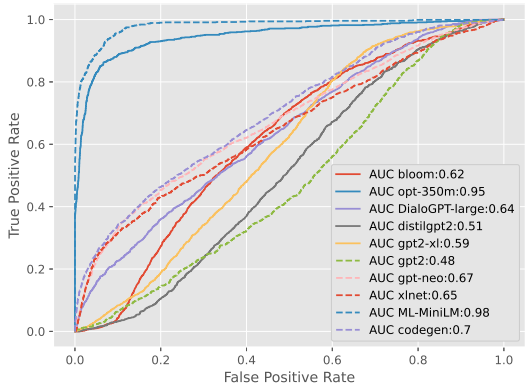
Figure 8: ROC of Individual base model classifiers, h_{m_b}, under different K



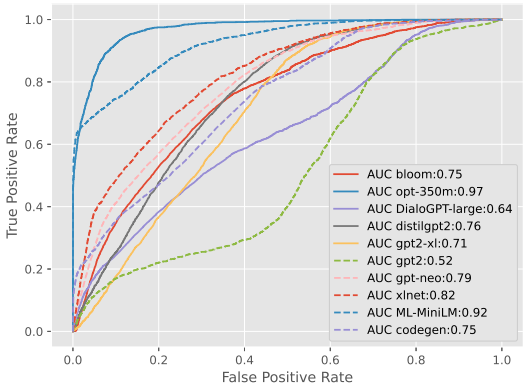
(a) pile150



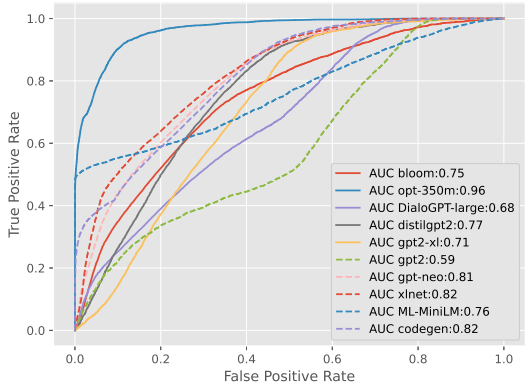
(b) pile 500



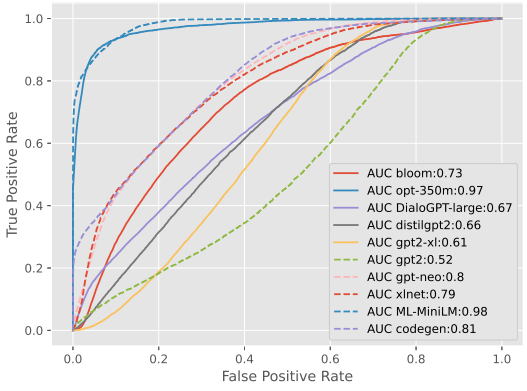
(c) pile 1k



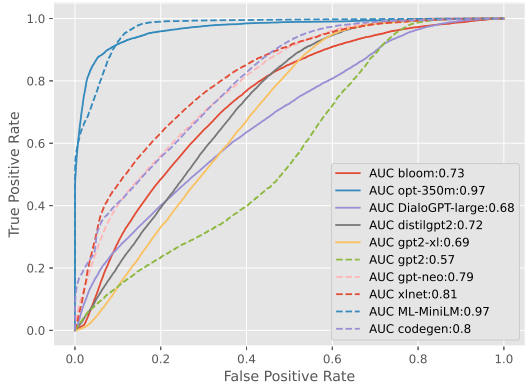
(d) pile 2k



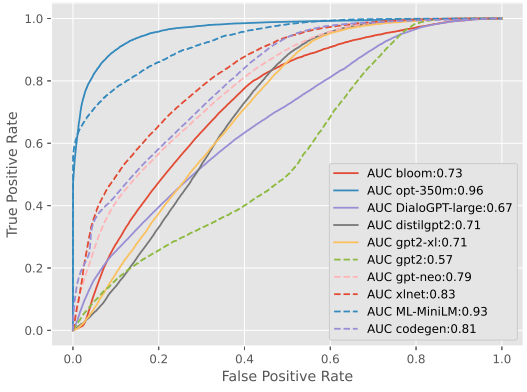
(e) pile 4k



(f) pile 6k

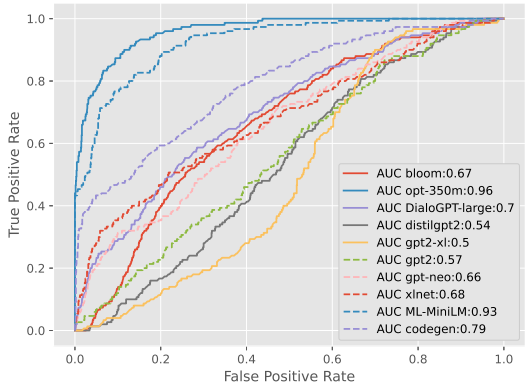


(g) pile 8k

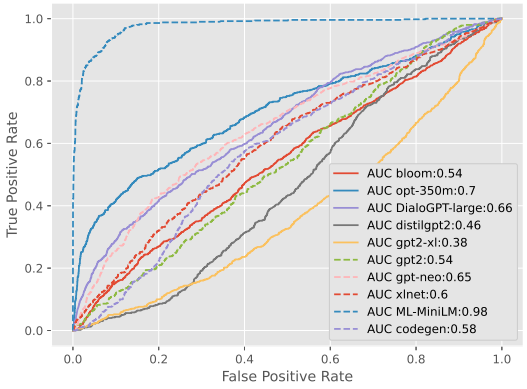


(h) pile 10k

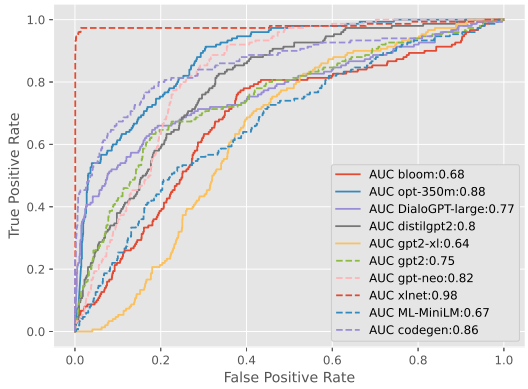
Figure 9: ROC of Individual base model classifiers, h_{m_b}, with different number of prompts used for attribution under K_R



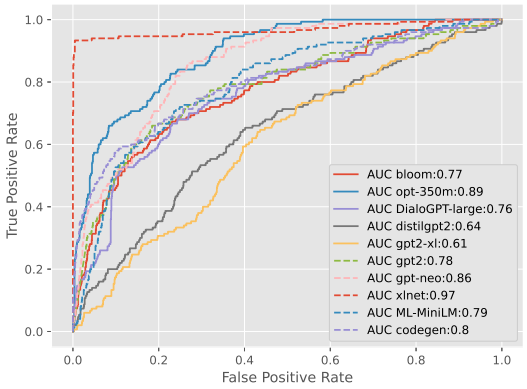
(a) pile150



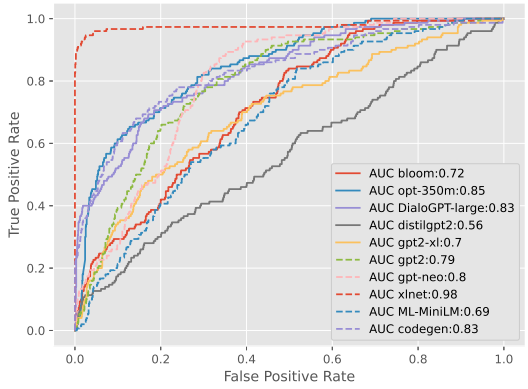
(b) pile 500



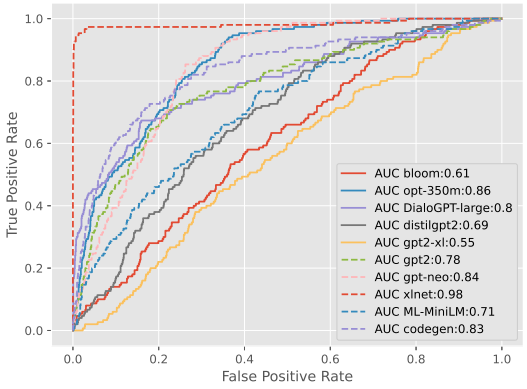
(c) pile 1k



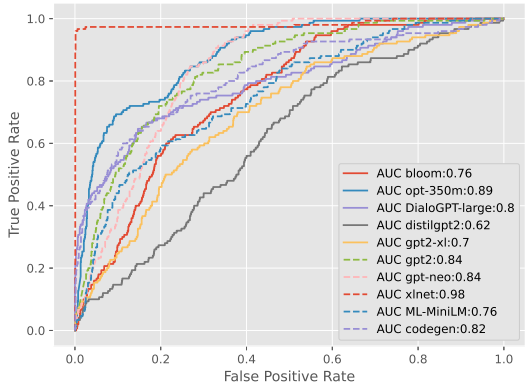
(d) pile 2k



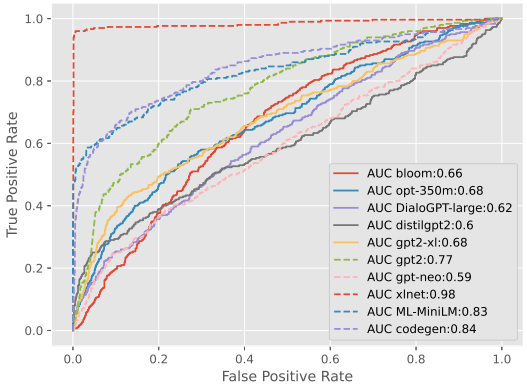
(e) pile 4k



(f) pile 6k



(g) pile 8k



(h) pile 10k

Figure 10: ROC of Individual base model classifiers, h_{m_b} using a varying number of prompts for pretraining h_{m_b}.

Dataset	Percentage of prompts in 10,000 subset of the Pile
Pile-CC	25.24
OpenWebText2	15.20
PubMed Abstracts	14.23
StackExchange	13.99
Github	8.55
Wikipedia (en)	7.79
USPTO Backgrounds	5.14
PubMed Central	2.59
FreeLaw	2.41
NIH ExPorter	1.04
DM Mathematics	0.99
ArXiv	0.91
HackerNews	0.81
Enron Emails	0.47
OpenSubtitles	0.27
YoutubeSubtitles	0.11
Books3	0.09
EuroParl	0.06
PhilPapers	0.05
BookCorpus2	0.02
Ubuntu IRC	0.02
Gutenberg (PG-19)	0.02

Table 6: Distribution of the original datasets present in the 10,000 prompt subset of The Pile

D The Pile subset

We make use of a 10,000 prompt subset of The Pile (Gao et al., 2020b), in Table 6 we report the distrubtion of the smaller datasets present in The Pile.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
7

☑ A2. Did you discuss any potential risks of your work?
7

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

5

☑ B1. Did you cite the creators of artifacts you used?
5, B

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No, the artifacts used all have open source licences, and are freely avaliable at their respective citations.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We make use of data used to train/fine-tune language models, this is consistent with their intended use. We don't discuss this use explicity relative to the intended us, but do discuss the use of these artifacts for finetuning purposes in Section 5, B.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No as none of the data used contains information that names or uniquely identifies individual people or offensive content.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
B,D

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
5

☒ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
We utilised the BERT using default parameter values used in the original paper and then used this model to develop a classifier.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
5, B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☒ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☒ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☒ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☒ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☒ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.