Was My Data Used for Training? Membership Inference in Open-Source LLMs via Neural Activations

Xue Tan*‡, Hao Luan*‡, Mingyu Luo*‡, Zhuyang Yu*‡, Jun Dai†✉, Xiaoyan Sun†✉, and Ping Chen*§✉
*Institute of Big Data, Fudan University, Shanghai, China
†Department of Computer Science, Worcester Polytechnic Institute, MA, USA
‡College of Computer Science and Artificial Intelligence, Fudan University, Shanghai, China
§Purple Mountain Laboratories, Nanjing, China

Abstract—**With the rapid development of Large Language Models (LLMs), their applications have expanded across various aspects of daily life. Open-source LLMs, in particular, have gained popularity due to their accessibility, resulting in widespread downloading and redistribution. The impressive capabilities of LLMs results from training on massive and often undisclosed datasets. This raises the question of whether sensitive content such as copyrighted or personal data is included, which is known as the membership inference problem. Existing methods mainly rely on model outputs and overlook rich internal representations. Limited access to internal data leads to suboptimal results, revealing a research gap for membership inference in open-source white-box LLMs.**

In this paper, we address the challenge of detecting the training data of open-source LLMs. To support this investigation, we introduce three dynamic benchmarks: *WikiTection*, *NewsTection*, and *ArXivTection*. We then propose a white-box approach for training data detection by analyzing neural activations of LLMs. Our key insight is that the neuron activations across all layers of LLM reflect the internal representation of knowledge related to the input data within the LLM, which can effectively distinguish between training data and non-training data of LLM. Extensive experiments on these benchmarks demonstrate the strong effectiveness of our approach. For instance, on the *WikiTection* benchmark, our method achieves an AUC of around 0.98 across five LLMs: *GPT2-xl*, *LLaMA2-7B*, *LLaMA3-8B*, *Mistral-7B*, and *LLaMA2-13B*. Additionally, we conducted in-depth analysis on factors such as model size, input length, and text paraphrasing, further validating the robustness and adaptability of our method.

I. Introduction

Recent advances in Large Language Models (LLMs) have significantly enhanced performance across a diverse set of Natural Language Processing (NLP) tasks. Among the family of LLMs, open-source models, such as GPT-2 [1], [2], [3], BLOOM [4], Mistral [5], and LLaMA 1/2/3 [6], [7], [8], hold a crucial position due to their large numbers, exceptional performance, and wide-ranging impact. Currently, the Hugging Face Hub hosts over 900,000 publicly available models, spanning a wide array of architectures, including language models, vision models, and multimodal systems [9]. According to a Meta report, by early 2025, cumulative downloads of its LLaMA series models had reached 1 billion, highlighting the widespread adoption of open-source models [10].

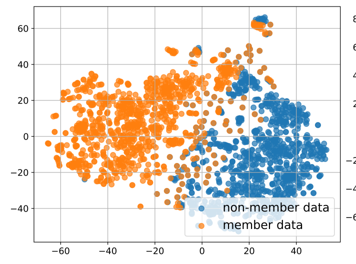
✉ Corresponding authors: Ping Chen <pchen@fudan.edu.cn>, Jun Dai <jdai@wpi.edu>, and Xiaoyan Sun <xsun7@wpi.edu>

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240474
www.ndss-symposium.org

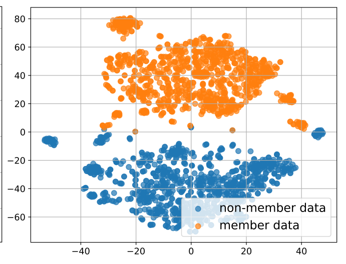
The rapid advancement of LLMs has been primarily driven by extensive training on massive datasets. For instance, the training corpus for Meta's first-generation LLaMA model contains up to 1.4 trillion tokens [6]. These datasets include not only vast amounts of general web content but also sensitive information across various domains, including personal data, financial records, and copyrighted materials [11]. With the widespread deployment of LLMs, increasing attention is being paid to the sources of training data and the associated privacy risks. Notably, while model developers often release their models to the open-source community, they are typically reluctant to disclose detailed information about the training corpora [12], [13], [14]. This lack of transparency complicates ethical and legal compliance and raises concerns about privacy protection. The corpora used for training LLMs may contain unauthorized personal data or copyrighted content [15], [16], potentially resulting in a series of legal disputes. For instance, *The New York Times* filed a copyright infringement lawsuit against OpenAI and Microsoft, accusing the companies of using millions of articles without permission to train the ChatGPT model [17]. Furthermore, due to the opacity of training data, it is impossible to confirm whether the performance of LLMs stems from genuine task understanding or simply from prior exposure to test data.

Since the training data for LLMs is not publicly disclosed, how can we detect whether an open-source LLM has been trained on a given text?

In this paper, we focus on the above question. The task of detecting training data is essentially a membership inference attack (MIA) [18], which seeks to determine whether a given text was included in the LLM's training corpus by analyzing its behavior. Research in this field has made significant progress, with most existing methods relying on the probability distribution of text tokens under the assumption that such distributions



(a) LLaMA3-8B



(b) Mistral-7B

Fig. 1: t-SNE visualizations of activations for member and non-member data in the WikiTection dataset.

serve as indicators of whether a text was included in the training set [19], [20], [21], [22], [23]. It is worth noting that these methods are primarily designed for black-box LLMs, which do not permit access to internal model information such as gradients or activations. Due to limited access to internal model information, their accuracy in membership inference tends to be relatively low. Privacy auditing research on white-box LLMs has explored the use of token embeddings, attention weights, and gradients to perform MIAs [24]. However, due to performance bottlenecks, the detection results of these methods have consistently failed to be fully convincing.

In this paper, our primary focus is on open-source LLMs. As these LLMs are white-box models, we can design effective methods to fully leverage their internal information, enabling high-accuracy training data detection. As the direct response of an LLM to its input, neural activations may more accurately reflect the membership status of the input. Our preliminary experiments indicate that *members*, texts that were likely seen by the model during training, and *non-members*, texts that are unlikely to have been seen by the model, exhibit significant differences in activations, as shown in Figure 1. Building on this key finding, we propose NART (**N**eural **A**ctivations **R**eveal **T**raining), a training data detection pipeline based on LLM activations. This pipeline consists of four main stages: **Data Collection**, **Activation Extraction**, **Preprocessing**, and **NART Model Design**.

A key challenge in evaluating MIAs on open-source LLMs is ensuring that member and non-member datasets come from the same distribution. Otherwise, models may exploit distributional differences, leading to inflated and misleading performance. Prior work [25] shows that even simple baselines can take advantage of such shifts. Therefore, the widely adopted strategy of using the model's knowledge cutoff date to distinguish between member and non-member data [26], [20], [22], [24] is not a reliable approach. To mitigate this issue, we fine-tune LLMs on curated datasets, treating included samples as members and excluded ones as non-members. However, since LLMs are pretrained on large-scale internet corpora, many public datasets may already be part of their training sets. To mitigate the risk of such data overlap, we additionally construct three dedicated datasets: *WikiTection*, *NewsTection*, and *ArXivTection*, sourced respectively from Wikipedia, Common Crawl, and ArXiv. Importantly, all data in these collections first appeared after the latest known training cutoff date of the target LLMs, significantly reducing the likelihood of unintentional data leakage. To better simulate real-world scenarios where texts of varying lengths may need to be detected, the data we collected is not of uniform length. We first split each text in the dataset to ensure it meets the input length constraints of the LLM. Then, we construct prompts from the resulting subsequences and retrieve the activation values for the last token in each prompt across all layers in the LLM, and normalize them. Next, we introduce three preprocessing strategies to effectively represent the diverse activation patterns. ***NonFE*** directly uses the normalized activation matrix of the last token as the subsequence feature, preserving full activation details but with high computational cost. ***StatFE*** summarizes activations from each layer using statistical descriptors such as the minimum, maximum, and mean, thereby enhancing robustness and reducing dimensionality. ***HistFE*** captures activation distribution characteristics through histograms, offering an efficient and noise-tolerant representation. While these strategies yield informative activation representations, activation differences between member and non-member samples remain subtle and dispersed across layers. To address this challenge, we employ a triplet network that learns a discriminative metric space by optimizing relative distances between samples, embedding member activations closer together while pushing non-member activations farther apart. This metric-learning framework enables the model to capture fine-grained relational beyond absolute activation values, and improving discrimination with limited labeled data.

We evaluate the effectiveness of NART across various model architectures. Specifically, we fine-tune models such as GPT2-xl [3], LLaMA2-7B [6], LLaMA2-13B, Mistral-7B [27], and LLaMA3-8B on our constructed datasets. In addition, we evaluate NART on the pretrained Pythia-12B [28] and DCLM-1B [29] models using their publicly available training and test sets from the Pile [30], MIMIR [31], and DCLM [29] datasets, further demonstrating the method's generalizability and robustness. Furthermore, we evaluate the robustness of NART under varying model sizes and input lengths.

In summary, our main contributions are as follows:

1) Since it has been observed that LLM activations encode membership-related information, we propose NART, a novel and automated membership inference pipeline for open-source LLMs that is designed to detect whether a given text was included in the training corpus of a language model.
2) We introduce three new benchmarks for identifying whether a text was used in the training of LLMs: *WikiTection*, *NewsTection*, and *ArXivTection*, each comprising 6,000 text samples.
3) Our method has been empirically validated across diverse LLM architectures and experimental settings. It consistently achieves an AUC around 98% on our curated benchmark datasets.

II. Background and Related Work

A. Language Modeling

The primary objective of Large Language Models (LLMs) is to understand and generate human-like text [32], [33]. In neural language modeling, a tokenizer T is first used to decompose the input text into a sequence of tokens $D = t_1, \ldots, t_N$, where each token t_i is an element of the vocabulary $\mathcal{V}$. Given the generated token sequence, a neural network [34] is trained to estimate the probability of the next token via maximum likelihood estimation. Given the model parameters θ and the training set $\mathcal{X}$ containing N tokens, the training objective of the model LLM_θ is to minimize the following loss function [35], [36]:

$$\mathcal{L}(\theta) = -\log \prod_{i=1}^{N} LLM_\theta(x_i \mid x_1, \ldots, x_{i-1}), \tag{1}$$

Due to computational constraints, the model's maximum context length is limited to a fixed value $C_{\max}$. Therefore, for any token t_i, the model is conditioned only on the previous C tokens (where $C \leq C_{\max}$), i.e., $LLM_\theta(t_{i+1} \mid t_{i-C+1}, \ldots, t_i)$ which is denoted as $LLM(t_{i+1})$.

Pretraining and supervised fine-tuning are key stages in the training process of LLMs. The pretraining phase employs an unsupervised learning approach [37], enabling the model to train on large-scale, diverse datasets. Through exposure to vast amounts of textual data, the model gradually learns intricate language patterns and nuances, fostering a deeper understanding of language. During the supervised fine-tuning phase, the LLM is further trained on task-specific or domain-specific datasets to precisely adapt to target applications. By optimizing its parameters, fine-tuning transforms the broad language understanding gained during pretraining into specialized capabilities, such as translation and question answering.

B. Membership Inference Attack

Membership Inference Attack (MIA) has gained widespread attention in the field of machine learning as a representative privacy attack technique, due to the potential negative consequences of leaking membership information [38], [39], [40], [41]. In the field of LLMs, this privacy risk has similarly attracted attention, as the specific outputs of LLMs may lead to the leakage of private information.

The theoretical foundation of existing MIAs primarily lies in the observation that models tend to generate more confident and stable predictions for samples encountered during training. Some methods utilize probability-based metrics, such as loss [42] and perplexity [19], to differentiate between members and non-members. However, MIAs based solely on perplexity may incorrectly identify simple or highly predictable sequences as members, even if they were never seen during training. The Neighboring-based Attack [21] generates text similar to the target sample through data augmentation and compares the perplexity scores between the target sample and its neighboring samples. LiRA does not rely on log perplexity; instead, it utilizes likelihood ratios to perform MIAs. Min-k%Prob [20] determines whether a target response belongs to the training data by computing the log-likelihood of the least probable token within the response. DE-COP [26] conducts MIAs by combining original texts with their rewritten versions and prompting the model to distinguish between them via multiple-choice questions. However, in white-box settings, such methods do not fully exploit internal model information, limiting the potential improvement in MIA accuracy. PARSING [24] leverages both forward (e.g., token embeddings and attention) and backward (e.g., gradients) signals of LLMs in a fine-tuning setting for membership inference, but achieves a modest AUC of approximately 0.75. Probe [43] trains autoregressive models for membership inference using internal representations from individual layers, but the approach achieves limited effectiveness (maximum AUC of 0.698 and TPR@5%FPR of 0.167) and incurs substantial computational cost, as a separate probe must be trained for each layer. Moreover, the dataset used in their experiments contains samples of at most 143 tokens, further limiting the method's effectiveness. LUMIA [44] leverages existing datasets and pretrained models to train probes on layer-wise internal activations for membership inference, extending this approach to multimodal large models; however, it also requires training individual probes for each layer. In contrast, our method aggregates activations across all layers and employs a non-linear triplet network built upon a Siamese architecture to capture global membership patterns, enabling more effective discrimination between training and non-training samples. This design achieves AUC and TPR@5%FPR scores close to 0.98 while keeping computational overhead low.

C. Siamese Networks

The Siamese network is a neural architecture widely used in metric learning. It consists of two identical, weight-sharing subnetworks that process separate inputs to produce feature vectors. The similarity between the two outputs is then computed using a distance metric such as the Euclidean distance.

Siamese networks have been widely applied in the field of image processing. For instance, HybridCNN [45] proposes an image matching method based on a Siamese network, and OSNV [46] employs a lightweight Siamese network for feature extraction. Moreover, this architecture has also been extensively used in NLP tasks. The study [47] proposes a model that uses Siamese networks to differentiate sentence attributes. During the training process, the model simultaneously inputs the main sentence along with its similar or dissimilar parts and optimizes by measuring their similarity. The study [48] addresses the problem of building text classifiers with limited or no training data by proposing a few-shot learning approach based on Siamese networks. This approach significantly reduces inference costs. Siamese networks, as a typical few-shot learning method, is based on comparing the similarity between samples, rather than relying on large amounts of labeled data like traditional classification networks. In MIAs on open-source models, where acquiring a large number of training samples is challenging, the few-shot learning approach based on Siamese networks is particularly effective.

III. Problem Statement

Capability of the Auditor: In this work, we define text D consisting of tokens t_i, where $i \in 1, \ldots, N$. We consider an auditor $\mathcal{A}$ with white-box access to an LLM, which enables the auditor to query the model using a sequence of tokens and to examine internal parameters or hidden states (e.g., activation maps). Our study focuses on open-source models that provide white-box accessibility, making this threat model both practical and realistic. Such models may inadvertently incorporate unauthorized or sensitive information during training, thereby introducing potential privacy and compliance concerns. Notably, many models such as LLaMA-2 [6], Mistral-7B [5], and GPT-2 [3] are publicly available and fully accessible through platforms like the Hugging Face Hub. Given their widespread distribution and the inherent difficulty in monitoring their downstream usage, the potential privacy risks associated with these models merit heightened attention. The auditor may represent one of three user groups: (1) individuals or organizations seeking to verify whether their private data have been used in model training, (2) regulatory authorities auditing data usage compliance under frameworks such as the GDPR, and (3) model developers aiming to ensure that their training data are free from sensitive information and fully compliant with relevant data protection standards.

In addition to having access to the LLM, the auditor is also assumed to possess a collection of texts D that follows the same distribution as the model's training corpus. Specifically, D_M denotes the subset of texts used during the LLM's training, referred to as ***members***, $\forall D \in D_M, D \in \mathcal{D}_{train}$. Conversely, let D_{NM} denote the texts that were not used during training, referred to as ***non-members***, i.e., $\forall D \in D_{NM}, D \notin \mathcal{D}_{\text{train}}$. It is important to note that both D_M and D_{NM} are obtainable in realistic settings. First, existing LLMs are commonly trained on textual corpora collected from publicly available sources such as Common Crawl [49], Project Gutenberg [50], and similar platforms. Second, these datasets are continuously updated over time. By comparing the timestamps of data updates with the timeline of model training, we can differentiate between D_M and D_{NM} [22], [26], [20]. For instance, data published prior to the model's training phase can be treated as D_M, whereas data updated afterward can be regarded as D_{NM}. This temporal distinction forms the basis of our dataset construction.

Objective of the Auditor: The auditor's primary objective is to determine whether a given target sample D was included in the training data of the LLM. Given an LLM trained on a dataset D_{train}, the auditor is assumed to have access to auxiliary knowledge $\mathcal{K}$. For a given target sample D, the auditor aims to construct a membership inference model $\mathcal{M}$ that maps the input triplet $(D, LLM, \mathcal{K})$ to a binary decision, formally defined as:

$$\mathcal{M}\left(D, LLM, \mathcal{K}\right) \rightarrow \{0, 1\}, \tag{2}$$

where 0 denotes that D was not included in the training set, while 1 indicates that D was part of the model's training data.

IV. Methodology

The goal of the auditor $\mathcal{A}$ is to determine whether a given text D was included in an LLM's training data. We extract activations from the response generation process of open-source models as the basis for making this determination.

Intuition: Prior works on the privacy of LLMs [19], [22], [26], [20] suggest that the model's prediction confidence is typically higher for training data compared to unseen data. This observation motivates the hypothesis that auditors can effectively leverage this behavior for membership inference. LLM activations, which encode input data across multiple levels of abstraction, enable the model to progressively extract high-level semantic features from low-level representations. The rich information encoded in these activations reflects both the model's end-to-end decision-making process [51] and its internal representations of knowledge associated with the input. Therefore, we hypothesize that the confidence gap between training and non-training samples is effectively manifested in the corresponding activation patterns. In summary, we propose that auditors can leverage the information embedded in LLM activations to perform membership inference.

Figure 1 shows t-SNE (t-distributed stochastic neighbor embedding) visualizations of LLM activations for both member and non-member texts. t-SNE is a non-linear dimensionality reduction technique that preserves local neighborhood structures in high-dimensional data. It visualizes the mean activations across all layers of two LLMs, LLaMA3-8B and Mistral-7B, captured during the generation of tokens associated with both training and non-training texts. The activations reveal a noticeable separation between training and non-training samples, providing empirical support for our hypothesis. These results suggest that activations at various layers of an LLM can serve as informative features for membership inference.

A. Approach Overview

Figure 2 and Algorithm 1 illustrates the steps of NART to leverage LLM activations for membership inference.

1) We first collect data from public platforms and construct corresponding datasets based on the publication time of the data and the training period of LLMs. We then query the LLM with each subsequence of text D from the dataset, setting the text length to C. We also evaluate the impact of varying C on the model's performance.
2) Next, we collect the activations of the last token in the subsequence and normalize them to highlight the pattern differences between members and non-members. This process enhances the accuracy and robustness of auditor $\mathcal{A}$ in performing membership inference.
3) Subsequently, we apply various strategies to process the activations of the last token across all layers of the LLM into a subsequence-level activation feature.
4) Finally, we use the obtained subsequence-level activation features to train a model based on a Siamese network.

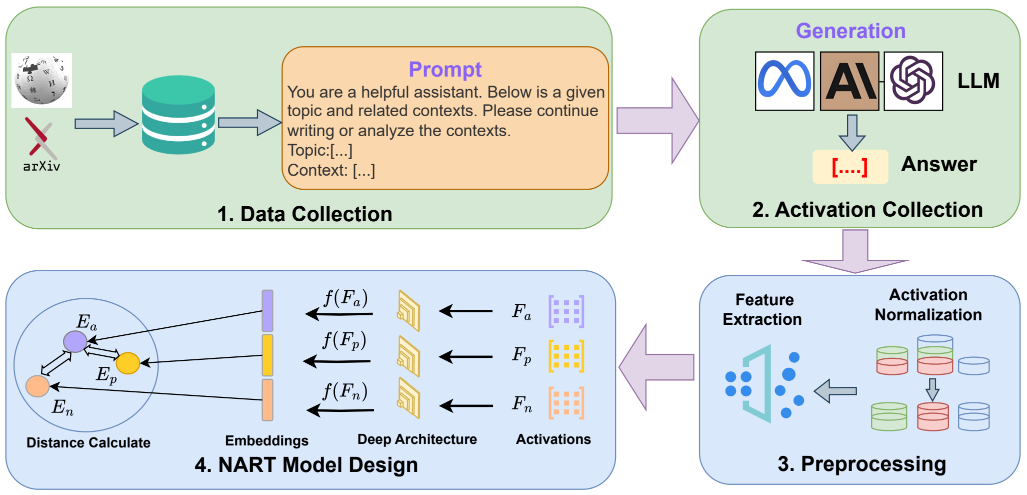


Fig. 2: The workflow of NART.

For each input text D_{test}, NART can determine whether D_{test} belongs to the training set of the LLM.

You are a helpful assistant. Below is a given topic and related contexts. Please continue writing or analyze the contexts.
Topic: [topic]
Context: [contexts]

Fig. 3: Prompt used to guide the LLM in generating a response given the context.

B. Querying the Model

When querying the model, the length N of the input text D may exceed the model's maximum context length $C_{\max}$. Therefore, we divide D into N_s subsequences S_j of length C, where $j = 1, \ldots, N_s, C \leq C_{max}, N_s = \lceil \frac{N}{C} \rceil$. For the last subsequence S_{N_s}, its length is denoted as C', satisfying $C' \leq C$. This reformulates the inherently complex long-text membership inference problem into a more tractable short-text detection task. Each subsequence S_j is then used to construct a prompt following the format illustrated in Figure 3. By aggregating the detection results from these fine-grained subsequences, we make a final determination on whether D belongs to the training set. This strategy not only improves the model's capability to handle long texts but also enhances the overall stability and robustness of the detection results. The context length C is a predefined hyperparameter that will be evaluated in subsequent experiments to assess its effect on model performance.

C. Activation Collection and Normalization

The activation of the last token integrates the semantic information of the entire input, representing the LLM's internal encoding of the input and its associated knowledge. As the model processes the input, information propagates through layers to retrieve relevant knowledge for answer generation [52]. When the generated content matches the model's learned knowledge, retrieval succeeds; mismatches indicate missing knowledge and produce distinct last-token activation patterns. To collect these activations, for each subsequence S_j of the text D, we extract the activation Act_j of the last token across all layers of the LLM, where $j \in \{1, \ldots, N_s\}$. Next, we introduce activation normalization to facilitate its effective integration into the training process. Specifically, we compute the mean μ and standard deviation σ of the activations across all texts in the dataset. These statistics are then used to normalize the activations according to the following formula:

$$Nor_j = (Act_j - \mu) / \sigma. \tag{3}$$

D. Feature Extraction

We employ three different methods to construct the feature representation of each subsequence from the normalized activation Nor_j. We first directly use the normalized activation Nor_j of the last token in subsequence S_j as the feature representation of that subsequence, referred to as ***NonFE***. However, as LLMs generate high-dimensional activations for each token across multiple layers (e.g., 48 layers × 6,400 dimensions in GPT2-xl), utilizing them for inference incurs substantial computational cost, although it retains fine-grained activation details beneficial for discrimination. To address this, we further introduce two feature extractors, ***StatFE*** and ***HistFE***, to derive more discriminative representations from the normalized activations. ***StatFE*** extracts statistical summaries of layer-wise activations rather than full activation details, enhancing robustness and reducing computational cost, while ***HistFE*** models activation distributions via histograms, which compress high-dimensional activations into compact statistical representations, thereby improving efficiency under low-resource or noisy conditions.

Statistical Feature Extractor (*StatFE*). We compute statistical features from each layer of the normalized activations Nor_j of the last token in subsequence S_j to capture the overall distribution patterns, including the minimum, maximum, mean, and standard deviation. Additionally, a set of x-percentile values is calculated, where $x \in X_{\text{perc}}$, to further capture the characteristics of the activation distribution. These statistical features are then concatenated to form the feature representation of subsequence S_j.

Algorithm 1: The Procedure of **NART**

```
Input : A text to be detected $D = t_1 t_2 \dots t_N$, a
        target $LLM$, the length of the subsequence
        $C$, embedding dimension $d$, the margin of the
        triplet margin loss $\alpha$, support samples
        $\{D_{s_i}\}_{i=1}^{M} \in T$ with label $\{L_{sup_i}\}_{i=1}^{M}$;
1  Divide $D$ into $N_s$ subsequences $S_j$ of length $C$, where
     $j = 1, \dots, N_s, C \le C_{max}, N_s = \lceil \frac{N}{C} \rceil$;
2  Query the $LM$ with each subsequence $S_j$;
3  Normalization:
4    Collect activation $Act_j$ from all layers of $LLM$ for
       the last token in each subsequence $S_j$;
5    Normalize the $Act_j$ with the mean $\mu$ and standard
       deviation $\sigma$, w.r.t. Eq. 3
6  Feature Extraction:
7    Preprocess the activation $Act_j$ to obtain a
       subsequence-level feature representation:
       $F_{S_j} = FEATURE\_EXTRACTOR\{Nor_j\}$,
       which contains three different strategies: NonFE,
       StatFE and HistFE , as shown in Sec. IV-D;
8  Divide the training set, processed as above and
     containing both member and non-member samples,
     into multiple triplets $\{F_a, F_p, F_n\}$;
9  Model Training:
10   Initialize Siamese Network $\mathcal{M}$ with ResNet18,
       replace fully-connected layer with Linear $(512, d)$;
11   for all each triplet $\{F_a, F_p, F_n\}$ do
12       Compute embeddings $E_a = \mathcal{M}(F_a)$,
           $E_p = \mathcal{M}(F_p)$, $E_n = \mathcal{M}(F_n)$;
13   Define loss function:
       $L = \max(\text{Dist}(E_a, E_p) - \text{Dist}(E_a, E_n) + \alpha, 0)$;
14   Train $\mathcal{M}$ on each triplet $\{E_a, E_p, E_n\}$;
15 Sample Test:
16   for all support sample $D_{sup_i}$ in $\{D_{sup_i}\}_{i=1}^{M}$ do
17       Compute embedding $E_{D_{sup_i}} = \mathcal{M}(D_{sup_i})$
18   for all subsequences in the test sample $D_t$ do
19       Compute embedding of each subsequence:
           $E_{S_j} = \mathcal{M}(S_j)$, $j \in \{1, \dots, N_s\}$.
20   for all subsequence embedding $E_{S_j}$ in $\{E_{S_j}\}_{j=1}^{N_s}$ do
21       for all support embedding $E_{D_{sup_i}}$ in
           $\{E_{D_{sup_i}}\}_{i=1}^{M}$ do
22           Compute distance $d_i = \|E_{S_j} - E_{D_{sup_i}}\|_2$
23       Predict $L_{S_j} = L_{D_{sup_K}}$, where $K = \arg\min_i d_i$
24   The lable of the test sample $D_t$:
       $L(D_t) = \mathbb{1}\left[\sum_{j=1}^{N_s} L_{S_j} \ge \frac{N_s}{2}\right]$ , $L(D_t) = 1$
       indicates that $D_t$ is a member sample, whereas it
       is a non-member sample.
```

Histogram Feature Extractor (*HistFE*). HistFE characterizes the activation distribution of subsequence S_j in D by constructing a histogram. Specifically, it first bins each layer of the normalized activations Nor_j into N_b equal-sized intervals (bins). Then, it computes both the count and proportion of activation values falling into each interval. Finally, the proportions of all N_b bins are aggregated to form a feature vector that represents the subsequence-level characteristics of S_j for membership inference.

E. NART Model Design

We utilize the NART model to perform membership inference on the subsequences S_j of text D, where $j \in \{1, \dots, N_s\}$. We employ Convolutional Neural Networks (CNNs) with the ResNet18 architecture [53] to efficiently capture the relationships between and within different layers of the LLM. Drawing inspiration from few-shot learning and Siamese networks, we utilize triplet networks that share the same architecture and weights to learn the activation features of member and non-member texts, as shown in Figure 2. In this membership inference task, labeled samples are typically limited, and training a classifier directly on raw activations is vulnerable to noise, inter-layer inconsistencies, and subtle differences between member and non-member samples, leading to limited effectiveness. Contrastive learning based on Siamese networks adopts a self-supervised objective that pulls similar samples closer and pushes dissimilar ones apart, enabling the extraction of discriminative features without explicit labels and enhancing the separability between member and non-member representations. In addition, it amplifies the supervision signal under limited data conditions by constructing abundant sample pairs, which is particularly valuable since obtaining sufficient member and non-member samples is often difficult in practice. Thus, the proposed contrastive learning model is particularly well suited for membership inference under limited labeled data, as it effectively preserves and enhances the model's discriminative power.

We employ the triplet margin loss as the model's training objective [54], a loss function based on "triplets," commonly used in metric learning tasks. A triplet consists of three components:

- **Anchor:** The reference sample, typically serving as the benchmark in the triplet.
- **Positive:** A sample from the same class as the anchor, with the goal of minimizing the distance between positive sample and the anchor.
- **Negative:** A sample from a different class than the anchor, with the goal of maximizing the distance between negative sample and the anchor.

During training, we partition the dataset into a training set, a test set, and a support set. Each text in the training set is first divided into multiple subsequences based on a given length C, and then further divided into D_a, D_p, and D_n, which represent the Anchor, Positive, and Negative samples, respectively. Subsequently, we feed the aforementioned samples into the LLM to extract the activation feature vectors corresponding to the subsequence, denoted as F_a, F_p, and F_n. These activation features are mapped into embeddings E_a, E_p and E_n through the Siamese network $\mathcal{M}$. The triplet loss ensures that the distance between the anchor and the positive sample, $\text{Dist}\left(E_a, E_p\right)$, is smaller than the distance between the anchor and the negative sample, $\text{Dist}\left(E_a, E_n\right)$, by at least a specified margin α. The loss function is formally defined as:

$$L = \max(\text{Dist}\left(E_a, E_p\right) - \text{Dist}\left(E_a, E_n\right) + \alpha, 0), \qquad (4)$$

where $\text{Dist}(\cdot, \cdot)$ denotes a distance metric (typically the Euclidean distance), and α is a positive constant. If α is too large, the loss remains high and convergence is difficult, but it improves the model's capability to distinguish between similar samples. If α is too small, the loss quickly approaches zero, making training easier but less effective at differentiating between E_p and E_n. Choosing an appropriate α enables a balance between discriminative capability and convergence speed. The goal of the training process is to minimize the loss, thereby enhancing the model's capability to distinguish between member and non-member samples of the LLM.

During testing, for a given test text D_t, it is first divided into a set of subsequences $\{S_j\}$ with fixed lengths, where $j \in \{1, \ldots, N_s\}$. For each subsequence S_j, we compute the distance between its embedding E_{S_j} and that of support samples $E_{D_{sup}}, D_{sup} \in T$, using the trained model. The support set T refers to a constructed dataset comprising labeled samples, denoted as $\{D_{sup_1}, ..., D_{sup_M}\}$, and their corresponding labels are $\{L_{sup_1}, ..., L_{sup_M}\}$. The label of S_j is determined according to the label of the support sample D_{sup} that is closest to it. That is, S_j is assigned the label of D_{sup}, meaning $L_{S_j} = L_{D_{sup_K}}$, where $K = \arg\min_i Dist(E_{S_j}, E_{D_{sup_i}})$ and D_{sup_K} is the nearest support data to the S_j. $L_{S_j} = 1$ indicates that the subsequence is a member sample, while $L_{S_j} = 0$ indicates that it is a non-member sample. Finally, the label $L(D_t)$ of text D_t is aggregated based on the membership of all subsequences:

$$L(D_t) = \mathbb{1}\left[\sum_{j=1}^{N_s} L_{S_j} \geq \frac{N_s}{2}\right]. \qquad (5)$$

The indicator function $\mathbb{1}$ returns 1 if a majority of subsequences are classified as members; otherwise 0.

V. Experimental Setup

A. Model

Our experiments focus on several popular open-source models, including GPT-2 [3], LLaMA2-7B [6], Mistral-7B [27], LLaMA3-8B, GPT-OSS-20B [55], Qwen3-8B [56], Pythia-12B [28], and DCLM-1B [29]. We chose these LLMs because they are widely used across various tasks and exhibit different structures and characteristics, making them well-suited for studying MIAs on open-source models. We use the prompt shown in Figure 3 to guide the LLMs in generating responses to input questions. GPT-2 employs a vocabulary of 50K tokens and supports an input length of up to 1K tokens. LLaMA2-7B and LLaMA3-8B both utilize a 32K-token vocabulary, with maximum context lengths of 4K and 8K tokens, respectively. Mistra-7B also adopts a 32K-token vocabulary and supports an 8K-token context window. GPT-OSS-20B supports a maximum context window of approximately 131K tokens, whereas Qwen3-8B has a context window size of 32K tokens. Pythia-12B is a decoder-only Transformer comprising 12 billion parameters and a 2K-token context window. DCLM-1B is a 1.4-billion-parameter language model trained on the DCLM-Baseline dataset, with a context length of 2K tokens.

B. Dataset for Membership

To rigorously evaluate the effectiveness of our method, we carefully constructed three datasets: WikiTection, NewsTection, and ArXivTection. Each dataset comprises internet content that was published after the most recent update of the LLMs, ensuring that the data were not accessible during the model's pretraining phase and thereby minimizing the risk of data leakage. We fine-tuned the LLM on a subset of each dataset, with fine-tuning details provided in Section A-A. In the subsequent membership inference task, the data used for fine-tuning are designated as positive samples, while the remaining data that were not involved in fine-tuning are treated as negative samples. We further evaluated our method on the pretrained Pythia-12B and DCLM-1B models, which were trained on the Pile [30] and DCLM [29] datasets, respectively. The results are presented in Section B-D.

- **WikiTection:** We used the Wikipedia API to collect web pages that were first published between October 2024 and February 2025. This time frame was chosen because most LLMs, including those used in our experiments, were last updated prior to this period. We used webpage titles as the topic for each data sample and cleaned the content by removing noise and non-informative characters, retaining only meaningful plain text. The constructed WikiTection dataset consists of 6,000 samples. These samples vary significantly in length, ranging from 128 to 512 tokens. The same construction principle is applied to the two subsequent datasets.
- **NewsTection:** News data is inherently time-sensitive, making it well-suited to our dataset construction strategy. Accordingly, we adopted the same cutoff date as used in the WikiTection dataset and selected news webpage links from the Common Crawl [49] dataset that satisfied the temporal criteria. We then crawled the corresponding webpages and extracted the meaningful body text along with the headlines. The lengths of these texts also ranges from 128 to 512 tokens.
- **ArXivTection:** ArXiv is a major platform for academic research dissemination, and we utilized papers from this platform to construct our dataset. RedPajama-Data [57] notes that the ArXiv maintainers provide an Amazon S3 bucket containing monthly updates of all source LaTeX files. We retrieved papers from this bucket that met our temporal criteria, applied similar preprocessing steps. In this dataset, the text length is fixed at 2048 tokens, as academic papers are typically long-form texts. In contrast, the text length in the other two datasets is not fixed, and varies.

TABLE I: The performance of NART on three datasets across different LLMs.

Dataset	**FeaEXTRACT**	**Metrics**	***LLMs***						
			GPT2-xl	LLaMA2-7B	LLaMA3-8B	Mistral-7B	LLaMA2-13B	GPT-OSS-20B	Qwen3-8B
WikiTection	***NonFE***	TPR@5%FPR	0.991	0.941	0.991	0.997	0.989	0.990	0.996
		AUC	0.992	0.981	0.998	0.996	0.997	0.994	0.999
	StatFE	TPR@5%FPR	0.986	0.991	0.996	0.992	0.987	0.988	0.990
		AUC	0.997	0.994	0.991	0.998	0.997	0.993	0.992
	HistFE	TPR@5%FPR	0.995	0.982	0.991	0.995	0.991	0.990	0.993
		AUC	0.995	0.996	0.995	0.997	0.996	0.994	0.998
NewsTection	***NonFE***	TPR@5%FPR	0.918	0.900	0.941	0.959	0.982	0.972	0.984
		AUC	0.977	0.976	0.988	0.985	0.986	0.980	0.988
	StatFE	TPR@5%FPR	0.982	0.972	0.918	0.977	0.964	0.980	0.978
		AUC	0.994	0.988	0.971	0.996	0.989	0.989	0.984
	HistFE	TPR@5%FPR	0.982	0.941	0.939	0.973	0.964	0.986	0.990
		AUC	0.988	0.985	0.965	0.991	0.984	0.990	0.994
ArXivTection	***NonFE***	TPR@5%FPR	0.973	0.941	0.988	0.982	0.977	0.984	0.982
		AUC	0.986	0.967	0.994	0.989	0.993	0.990	0.986
	StatFE	TPR@5%FPR	0.982	0.941	0.982	0.973	0.991	0.990	0.988
		AUC	0.998	0.984	0.993	0.994	0.996	0.994	0.990
	HistFE	TPR@5%FPR	0.986	0.961	0.982	0.964	0.996	0.990	0.992
		AUC	0.991	0.985	0.997	0.992	0.999	0.995	0.997

C. Baselines

To validate the effectiveness of our method, we conducted comparisons with several representative baselines:

• The ***Loss attack*** [42] infers the membership status of a given example by checking whether the target model's loss on it exceeds a predefined threshold.

• ***Zlib*** and ***Lowercase*** are two perplexity-based membership inference methods [38]. ***Zlib*** determines membership by comparing the perplexity of an example to its zlib compression entropy. In contrast, ***Lowercase*** evaluates membership by comparing the perplexity of the original example to that of its lowercased version.

• ***Min-K% Prob*** [58] makes predictions based on the log-likelihood of the lowest-probability token in the target sample. It assumes that the model exhibits higher confidence across all tokens for training members; therefore, the log-likelihood of low-probability tokens can serve as a discriminative signal to distinguish between members and non-members.

• The ***Neighborhood*** attack [21] leverages probability curvature to detect membership. In the experiment, we use the RoBERTa masked language model [59] to replace a token in the original sample, generating multiple neighboring samples. We then compare the loss of the target language model on the original sample with the average loss on the neighboring samples, thereby determining the membership status of the original sample.

• ***PARSING*** [24] proposes a white-box membership inference method that combines forward information (such as intermediate representations) and backward information (such as gradients and losses) from the model. By constructing attribute embeddings and applying discrepancy metrics, the method effectively distinguishes member samples from non-member ones, enabling dynamic monitoring of privacy risks.

• ***Probe*** [43] trains linear probes on activations from each layer of the LLM and selects the layer with the best validation performance as the final detection layer. The corresponding probe is then used to determine whether a test sample belongs to the member set.

D. Evaluation Metrics

To comprehensively evaluate the effectiveness of NART, we adopt two metrics: True Positive Rate at low False Positive Rate (TPR at low FPR) and Area Under the Curve (**AUC**). In MIA tasks, the primary objective is to accurately identify member samples while minimizing the impact of false positives. Therefore, TPR at low FPR is a more appropriate evaluation metric, as it measures the true positive rate under a fixed low false positive rate, reflecting the practical effectiveness of the attack. In this work, we adopt **TPR@5%FPR** and **TPR@10%FPR** as our evaluation metrics [58], [22]. In addition, Table XI in Section B-A further validates the model's performance under TPR@3%FPR and TPR@1%FPR.

E. Methodology Parameters

In the experiments, since the collected texts originate from different platforms and vary in length, they often do not meet the maximum input length requirements of LLMs. Therefore, we divide each text D into multiple subsequences $\{S_j\}$ with the fixed length $C = 128$. We randomly selected 1,000 member samples and 1,000 non-member samples from each dataset for the experiment. For the feature extractor ***StatFE***, we uniformly selected 50 percentiles within the range of 0 to 100. In ***HistFE***, the number of bins was set to 200. ***NonFE*** is the default method we employed. The ratio of the training set to the test set was 3:1, and the number of samples in the support set is set to 100 by default. The margin of the triplet margin loss was set to $\alpha = 1.0$. All of our experiments and language model queries were conducted on a set of A100 NVIDIA GPUs.

VI. Results

A. Evaluation across LLMs and Datasets

Table I summarizes the experimental results of NART using the ***NonFE***, ***StatFE***, and ***HistFE*** feature extraction methods on the WikiTection, NewsTection, and ArXivTection datasets. A notable distinction is that the Qwen3-8B model used in our experiments corresponds to the July 2025 update.

Consequently, we augmented all three datasets with additional samples collected between August and November 2025 to ensure that the fine-tuning data for Qwen3-8B did not appear in the model's training corpus. The results in Table I provide a comprehensive evaluation of NART's performance in training data detection across diverse domains and LLMs. For instance, under Mistral-7B, NART achieves an AUC of around 0.98 across all three datasets with both ***NonFE*** and ***HistFE***, validating the stability and superior performance of NART under diverse data distributions. The performance of NART with ***StatFE*** may fluctuate due to the characteristics of the dataset and the choice of parameters. On the WikiTection dataset, NART with all three feature extraction methods achieves a TPR@5%FPR of 0.99 on nearly all of the seven mainstream LLMs, including GPT2-xl, LLaMA2-7B, LLaMA3-8B, Mistral-7B, LLaMA2-13B, GPT-OSS-20B, and Qwen3-8B. In addition, Table XI in Section B-A further validates the model's performance under more stringent false positive rate conditions, specifically TPR@3%FPR and TPR@1%FPR. In Section B-E , we also evaluate the effectiveness of our method when the fine-tuning dataset consists of tens of thousands of samples. These results demonstrate that the method maintains exceptional training data identification capability, even at extremely low false positive rates, which is crucial for practical applications. Furthermore, NART demonstrates consistently strong performance across both smaller models (e.g., GPT2-xl with approximately 1.5B parameters) and larger models (e.g., LLaMA2-13B with approximately 13B parameters), further confirming its robustness and generalizability. Overall, the experimental results highlight NART's effectiveness in identifying training data, thereby validating the proposed approach.

B. Comparison to Baselines

To validate the outstanding performance of NART, we compared it with several baseline methods, and the results are shown in Table II. The baseline methods are evaluated on the same test dataset as our proposed approach, with member and non-member samples each accounting for 50% of the data. As shown in the experimental results, our proposed method significantly outperforms existing methods in terms of performance. For instance, on the WikiTection dataset, the ***Min-K% Prob*** method achieves an AUC of 0.694 for training data detection on the GPT2-xl model, whereas our NART method reaches an AUC of 0.992, demonstrating a substantial performance advantage. Moreover, the performance of existing baselines shows instability when applied to different LLMs. In contrast, our proposed method maintains consistently high and stable performance across various datasets and model scales, with AUC scores consistently around 0.98. This further demonstrates the robustness and generalization capability of our approach across diverse data distributions and text types, as well as its strong compatibility with various LLMs. This further demonstrates the strong robustness and generalization of our approach under varying data distributions and text types, as well as its good compatibility with different LLMs.

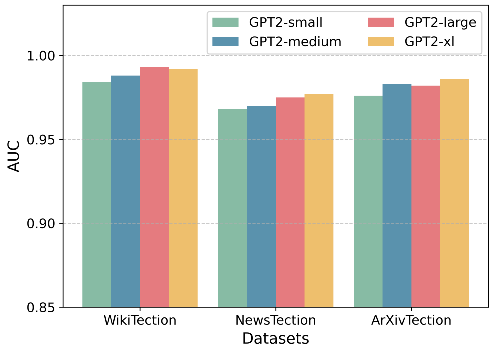


Fig. 4: AUC performance across different model sizes.

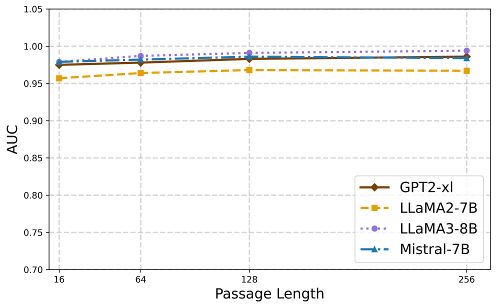


Fig. 5: AUC performance across different text lengths for texts from ArXiv papers.

C. Model Size

We evaluated the performance of our solution on the WikiTection, NewsTection, and ArXivTection datasets, using LLMs of different sizes from the GPT-2 series, specifically GPT2-small with 124M parameters, GPT2-medium with 355M parameters, GPT2-large with 774M parameters, and GPT2-xl with 1.5B parameters. Given that our previous experimental results have shown that our solution performs exceptionally well on large-scale models, in this experiment, we chose models from the GPT-2 series to explore the performance of our solution on smaller models of varying sizes.

The experimental results shown in Figure 4 demonstrate that our method delivers excellent performance on small-scale LLMs, with the overall AUC consistently exceeding 0.97. Additionally, we observe a steady improvement in detection performance as model size increases. This trend can be attributed to the larger parameter counts, which enhance the models' memorization and representation capabilities. However, the performance gain tends to plateau in medium-sized models. For instance, on GPT2-large, the AUC reaches 0.99 on the WikiTection dataset, approaching the optimal level. These findings suggest that our method can reliably and accurately detect training data.

D. Text Length

We further evaluated the performance of our method in training data detection tasks across different LLMs as the input text length varied. Figure 5 illustrates the trend of detection

TABLE II: NART outperforms baselines in terms of **AUC** across different LLMs.

Dataset	Models	Black-box Techniques					White-box Techniques		
		Loss attack [42]	*Zlib* [38]	*Lowercase* [38]	*Min-K% Prob* [58]	*Neighborhood* [21]	*PARSING* [24]	*Probe* [43]	*Our Method*
WikiTection	GPT2-xl	0.528	0.502	0.497	0.694	0.635	0.761	0.797	**0.992**
	LLaMA2-7B	0.519	0.499	0.509	0.717	0.623	0.779	0.806	**0.981**
	LLaMA3-8B	0.521	0.516	0.513	0.684	0.638	0.807	0.839	**0.998**
	Mistral-7B	0.531	0.511	0.521	0.701	0.645	0.821	0.836	**0.996**
NewsTection	GPT2-xl	0.503	0.512	0.563	0.707	0.686	0.731	0.746	**0.977**
	LLaMA2-7B	0.497	0.509	0.556	0.682	0.721	0.742	0.738	**0.976**
	LLaMA3-8B	0.516	0.491	0.547	0.686	0.752	0.759	0.815	**0.988**
	Mistral-7B	0.491	0.514	0.550	0.691	0.744	0.778	0.806	**0.985**
ArXivTection	GPT2-xl	0.516	0.477	0.534	0.571	0.672	0.756	0.753	**0.986**
	LLaMA2-7B	0.486	0.528	0.536	0.563	0.696	0.779	0.721	**0.967**
	LLaMA3-8B	0.528	0.536	0.531	0.601	0.715	0.812	0.791	**0.994**
	Mistral-7B	0.482	0.502	0.518	0.581	0.737	0.797	0.784	**0.984**

performance (AUC) on the ArXivTection dataset across different input lengths (measured in tokens). As the number of input tokens increases, the model's ability to distinguish between training and non-training data improves steadily, with AUC values gradually rising and stabilizing.

This trend suggests that longer texts generally contain richer semantic information, which makes the model's activation responses more distinguishable between member and non-member samples, thereby improving detection accuracy. We also observe that when the input text length is around 128 tokens, the detection performance of NART has essentially reached its optimum. Additionally, we further evaluate the performance of our method as the input text reaches the maximum context length of the LLMs in Section B-F.

TABLE III: The performance of NART on paraphrased datasets across different LLMs.

Paraphrased Dataset	Metrics	*LLMs*			
		GPT2-xl	LLaMA2-7B	LLaMA3-8B	Mistral-7B
WikiTection_para	TPR@5%FPR	0.978	0.942	0.969	0.987
	AUC	0.989	0.973	0.995	0.997
NewsTection_para	TPR@5%FPR	0.841	0.827	0.841	0.959
	AUC	0.969	0.953	0.965	0.988
ArXivTection_para	TPR@5%FPR	0.971	0.934	0.982	0.971
	AUC	0.983	0.968	0.997	0.989

E. Paraphrased Text Detection

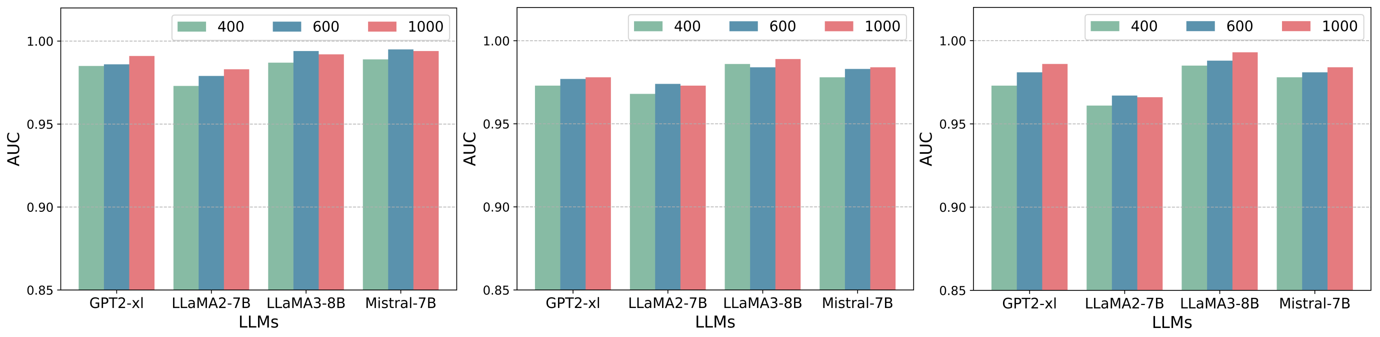
In real-world scenarios, it is often necessary to determine whether paraphrased or edited texts are part of the training data of LLMs. For instance, copyrighted content may be subtly modified without authorization and included in the training of LLMs, thereby circumventing regulatory oversight. Existing research primarily focuses on detecting instances that exactly match the training data. However, it remains unclear whether MIA methods can effectively identify semantically equivalent paraphrased instances. To explore this problem, we investigate training data detection in a paraphrased setting. We utilize GPT-4o to paraphrase the original texts of member and non-member samples while preserving the original semantic meaning, with the prompt shown in Figure 9 (Section A-B).

Observations from Table III indicate that, even after paraphrasing, the WikiTection and ArXivTection datasets can still be accurately detected by our proposed method. When evaluated across four LLMs, including GPT2-xl, LLaMA2-7B, LLaMA3-8B and Mistral-7B, the AUC values all exceeded 0.965, and the TPR@5%FPR values were also above 0.93. These results demonstrate that our method maintains strong detection performance even when handling paraphrased data. We attribute this effectiveness on paraphrased texts to its reliance on activation patterns shaped by semantics rather than lexical memorization, and the paraphrased texts preserve the original meanings.

F. Generalization

As previously discussed, our method requires a certain amount of training data to construct the detection model. However, in real-world scenarios, challenges such as limited availability of labeled data and imbalanced class distributions are common. To assess the robustness and practicality of our approach under such conditions, we further explore its applicability in data-scarce and imbalanced settings. Specifically, we combine the three datasets for joint training and evaluate performance on each dataset individually. This strategy more closely reflects real-world deployment scenarios, where training a dedicated model for each task may be infeasible due to computational or data constraints. Instead, we aim to develop a generalizable solution capable of effectively handling diverse tasks using a unified model.

Less Training Data. To simulate data-scarce scenarios, we design three experimental settings with varying amounts of training data. The first setting includes 1,000 samples per dataset, resulting in 3,000 samples in total, with an equal split between training and non-training data for each LLM. The second setting uses 600 samples per dataset, yielding a total of 1,800 samples. The third setting contains 400 samples per dataset, amounting to 1,200 samples in total. The test set and support set are fixed at 400 and 100 samples, respectively. As shown in Figure 6, our method consistently achieves strong performance, with AUC scores exceeding 0.95 across different training data scales. Although a slight decline in AUC is observed as the training data size decreases, the overall performance remains within an acceptable range. In the experiments presented in Section B-B, we further reduced the size of the training data by setting the training set to 50, 100, and 200 samples. The results are reported in Table XII. These results show our approach remains robust and practical even with limited data.



(a) WikiTection (b) NewsTection (c) ArXivTection

Fig. 6: AUC scores of NART across different test datasets under varying training data sizes.

TABLE IV: Evaluating NART under imbalanced datasets.

Testing Dataset	Metrics	*LLMs*			
		GPT2-xl	LLaMA2-7B	LLaMA3-8B	Mistral-7B
WikiTection	TPR@5%FPR	0.986	0.934	0.983	0.975
	TPR@10%FPR	0.988	0.965	0.991	0.980
	AUC	0.989	0.976	0.989	0.986
NewsTection	TPR@5%FPR	0.911	0.902	0.938	0.955
	TPR@10%FPR	0.932	0.965	0.973	0.969
	AUC	0.973	0.977	0.985	0.981
ArXivTection	TPR@5%FPR	0.971	0.940	0.988	0.975
	TPR@10%FPR	0.979	0.956	0.990	0.979
	AUC	0.978	0.966	0.989	0.981

Imbalanced Training Data. In this experiment, we focus on scenarios involving imbalanced training data across different categories. The data distribution is configured in a 1:2:3 ratio across the WikiTection, NewsTection, and ArXivTection datasets, corresponding to 400, 800, and 1,200 samples, respectively. As shown in Table IV, our method continues to exhibit strong performance despite the imbalanced training data distribution. The AUC scores for all datasets exceed 0.96, and the TPR@10%FPR values remain consistently above 0.95, with the highest reaching 0.991. These results further validate the robustness and practical applicability of our approach in imbalanced data scenarios.

G. Impact of Mislabeled Samples

In practical settings, the training dataset may include a notable proportion of mislabeled instances. For example, samples labeled as members may actually be non-members, and vice versa. Table V presents the impact of such mislabeled membership samples. We simulated three levels of label noise with error ratios of 5%, 10%, and 20%. The results show that when the error ratio is low (e.g., 5%), the proposed method maintains strong detection performance. However, as the error ratio increases to 20%, the effectiveness of the method declines noticeably, though the AUC still reaches up to 0.946 and the TPR@5%FPR remains at 0.733. These findings highlight the importance of accurate sample labeling for ensuring the reliability of the detection model.

TABLE V: Performance of NART under different mislabeled data ratios across multiple LLMs.

Dataset	Mislabeled Ratio	GPT2-xl		LLaMA3-8B		Mistral-7B	
		AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
WikiTection	5%	0.988	0.941	0.986	0.961	0.939	0.686
	10%	0.956	0.757	0.923	0.757	0.872	0.398
	20%	0.861	0.486	0.863	0.438	0.768	0.181
NewsTection	5%	0.982	0.961	0.945	0.853	0.968	0.951
	10%	0.971	0.893	0.894	0.427	0.940	0.748
	20%	0.946	0.733	0.808	0.405	0.818	0.259
ArXivTection	5%	0.991	0.941	0.989	0.971	0.970	0.951
	10%	0.962	0.874	0.959	0.709	0.938	0.689
	20%	0.814	0.143	0.829	0.171	0.848	0.295

TABLE VI: Training and inference time comparisons across datasets and models.

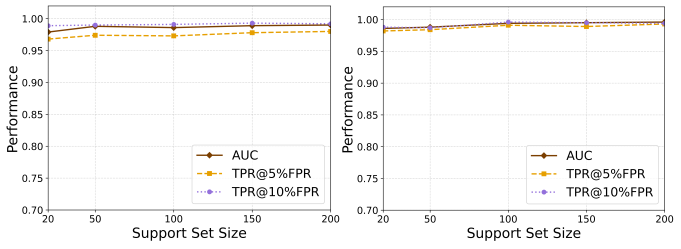
Model	Training Time per Epoch			Inference Time per Sample		
	WikiTection	NewsTection	ArXivTection	WikiTection	NewsTection	ArXivTection
PARSING	11.15s	10.76s	10.89s	0.0019s	0.0015s	0.0018s
NART_NonFE	18.34s	18.52s	18.62s	0.0060s	0.0059s	0.0061s
NART_StatFE	0.58s	0.56s	0.58s	0.0054s	0.0034s	0.0035s
NART_HistFE	1.99s	1.21s	1.66s	0.0098s	0.0098s	0.0056s

H. Efficiency

Table VI compares the time overhead of NART with different feature extraction methods and the white-box membership inference method PARSING [24], using LLaMA3-8B as the target LLM. The comparison includes the average training time per epoch and the average inference time per test sample. The experiment was conducted with 1,500 training samples and 500 test samples, where member and non-member samples each accounted for 50%. Experimental results show that NART exhibits minimal variation in inference time per test sample across different feature extraction strategies, while the training time varies significantly. This is primarily because the ***StatFE*** and ***HistFE*** methods substantially reduce the dimensionality of the training data, thereby lowering computational overhead. In addition, NART outperforms PARSING in terms of efficiency. This is because PARSING requires extensive access to the LLM's internal information to train the membership inference classifier.

I. Ablation Study

Effects of Different Support Set Size. In our proposed framework, the support set D_{sup}, introduced during the testing phase serves as a reference for each test instance. Specifically, the model computes the distance between the test sample and category-specific samples in the support set, assigning the label of the closest category. The support set effectively acts as a set of "prototypes," enabling reliable comparisons under few-shot conditions. This allows the model to perform accurate class discrimination in the embedding space based on the learned representations. We conducted experiments with varying support set sizes to investigate their impact on the performance of NART. Specifically, we gradually increased the number of support samples from 20 to 200 and evaluated the resulting changes in detection performance. As shown in Figure 7, we performed sensitivity experiments on support set size using the ArXivTection dataset with two LLMs: GPT2-xl and LLaMA3-8B. Across all performance metrics, the variation in support set size had minimal impact on the overall effectiveness of NART. Notably, even with a minimal support set of just 20 samples, the model maintained excellent detection performance, with AUC values consistently around 0.99. These results demonstrate that NART exhibits strong robustness with respect to support set size.



(a) GPT2-xl (b) LLaMA3-8B

Fig. 7: Effects of support set size.

Activations from Specified Layers. In previous experiments, we consistently used activations from all layers of the LLMs as input features to comprehensively capture the activation differences when processing training and non-training data. Existing experimental results indicate that this approach helps retain the model's internal representational information to the fullest extent, thereby enhancing the accuracy and robustness of training data detection. Here, we explore whether using activations from specific layers can yield the desired results. The results are shown in Table VII, with Table VIIa and Table VIIb presenting the AUC scores obtained using activations from a single layer and activations from a subset of layers, respectively. The LLM used in this experiment is LLaMA3-8B, which has a 32-layer network architecture.

We found that, in both settings, our approach still achieves good AUC scores in most cases. However, some fluctuations were occasionally observed. For instance, in Table VIIa, the AUC score for the NewsTection dataset exhibited significant variation, with the lowest value being 0.507. In Table VIIb, the lowest AUC score for the NewsTection dataset was 0.581. Notably, these fluctuations typically occurred in the earlier or middle layers of the LLM, whereas using the last few layers of the model resulted in stable and satisfactory AUC scores. The fluctuations in AUC scores are primarily attributed to the characteristics of different layers in the large model. The earlier and intermediate layers capture low-level features, which are relatively unstable, leading to fluctuations in some datasets. In contrast, the later layers, which integrate higher-level information, provide more stable features, thus enhancing the stability of the AUC scores. Furthermore, the characteristics of different datasets exacerbate these fluctuations. For example, the feature distribution of the NewsTection dataset may be more complex, leading to larger fluctuations in the earlier layers, while datasets like WikiTection, with greater structure and information density, tend to be more stable.

TABLE VII: AUC scores of NART using activations from different transformer layers of LLaMA3-8B.

(a) AUC scores on Specified Layer

Dataset	layer 1	layer 7	layer 12	layer 18	layer 25	layer 32
WikiTection	0.993	0.996	0.993	0.999	0.997	0.998
NewsTection	0.507	0.987	0.779	0.569	0.991	0.989
ArXivTection	0.989	0.990	0.998	0.991	0.995	0.997

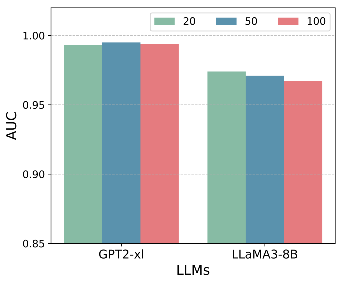
(b) AUC scores on Specified Layers

Dataset	layers 1–3	layers 6–9	layers 11–14	layers 17–22	layers 24–28	layers 29–32
WikiTection	0.997	0.995	0.999	0.998	0.997	0.998
NewsTection	0.581	0.987	0.630	0.677	0.987	0.976
ArXivTection	0.993	0.989	0.998	0.993	0.996	0.998

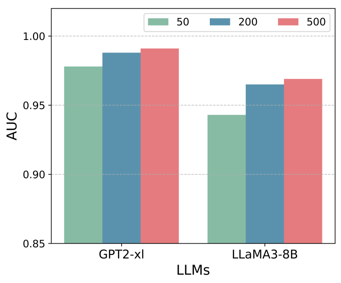
Quantized Models. All the above experiments were conducted with each LLM configured to use floating point precision of 32 bits. To further assess the robustness of our proposed approach in real-world deployment scenarios, we conduct experiments on the WikiTection and ArXivTection datasets to investigate whether changes in model activations under different numerical precision settings (e.g., float32, float16, int8, int4) affect membership inference performance. This experiment is designed to simulate practical deployment conditions where LLMs may be quantized for resource-constrained environments, such as edge devices or mobile platforms. Our goal is to evaluate the applicability and stability of the proposed method under low-precision computation. As shown in the results of Table VIII, when model inference is performed at lower precisions (e.g., float16 or int8), the activations produced by the LLM still support our approach in achieving satisfactory membership inference performance, despite a slight fluctuations in AUC. This indicates that changes in model precision have minimal impact on the performance of our method, further demonstrating its robustness.

TABLE VIII: TPR@5%FPR of NART on WikiTection and ArXivTection with LLaMA3-8B and Mistral-7B under varying precision.

Dataset	Precisions	LLaMA3-8B	Mistral-7B
WikiTection	float32	0.991	0.977
	float16	0.994	0.981
	int8	0.989	0.975
	int4	0.993	0.979
ArXivTection	float32	0.988	0.982
	float16	0.986	0.979
	int8	0.993	0.981
	int4	0.984	0.983



(a) Percentiles of ***StatFE***



(b) Bins of ***HistFE***

Fig. 8: Impact of percentile selection in ***StatFE*** and bins count in ***HistFE*** on AUC scores for NewsTection.

Impacts of Parameters in *StatFE* and *HistFE* Methods. In the two feature extraction methods we propose, ***StatFE*** and ***HistFE***, the number of percentiles determines the dimensionality of the ***StatFE*** feature vector, while the number of bins directly affects the accuracy of the histogram in ***HistFE***, thus influencing the final feature dimensions. To better understand the impact of these parameters on feature extraction performance, we conducted ablation experiments focusing on these two key factors. The experimental results are shown in Figure 8, where Figure 8a illustrates the performance of ***StatFE*** with 20, 50, and 100 percentiles uniformly selected from the range of 0 to 100. We observed that the choice of percentiles impacts the performance of our method, and the optimal selection varies across different models. Figure 8b shows the effect of different bin counts in ***HistFE*** on the final performance. As the number of bins increases, the data range represented by each bin becomes narrower, enabling the model to capture more refined features and variations.

VII. Discussion

Next, we present an in-depth discussion on the applicability and potential limitations of the proposed approach.

General Applicability. In this work, we address the problem of detecting training data in LLMs within a white-box setting. To achieve this, we design and implement a comprehensive detection pipeline: we first extract deep representation information from the model's intermediate activations, then apply carefully designed feature extraction methods, such as ***NonFE***, ***StatFE***, and ***HistFE***, to convert these activations into vector representations suitable for classification. Finally, we employ a framework to facilitate efficient and robust membership inference in a few-shot setting. We systematically evaluate our approach across various language model architectures and diverse datasets. The results demonstrate that our method consistently delivers strong performance and exhibits excellent generalization across multiple real-world tasks. Even with a limited number of textual examples and significant class imbalance, our method consistently demonstrates robust and reliable performance (as shown in Section VI-F). Moreover, to address the variability of text lengths encountered in real-world scenarios, our approach improves adaptability and generality by segmenting long texts and aggregating predictions from their subsequences. Table VI in Section VI-H demonstrates that the ***StatFE*** and ***HistFE*** feature extraction methods significantly reduce training time. Meanwhile, in Section B-I, we performed blind attacks on our dataset, which demonstrated that there is no distribution shift, further validating the effectiveness and generalizability of our approach. Overall, we provide a comprehensive solution for stakeholders in the open-source LLM ecosystem, addressing privacy protection and copyright compliance challenges.

Limitations. In our experiments, we observed that the proposed approach performs suboptimally in cross-category tasks. Specifically, when the training and testing data originate from different datasets, such as training on WikiTection and testing on ArXivTection, the model exhibits poor performance. Section B-B considers the scenario where training and testing samples are drawn from different datasets. The corresponding experimental results are presented in Table XIII. We attribute this issue to differences in textual characteristics across datasets from different platforms. For example, Wikipedia texts are more standardized and clearly structured, intended for a general audience and written in relatively simple language. In contrast, ArXiv texts are academic in nature, rich in technical terminology and complex syntactic constructions. Our experimental results indicate that as long as the training set includes samples from a particular category, even in small quantities, the model can still achieve strong performance. Moreover, acquiring a small number of representative text samples is often feasible, ensuring that our approach remains effective even under data-scarce conditions.

VIII. Conclusion

In this paper, we leverage the white-box nature of open-source LLMs and observe that their internal activation patterns exhibit significant differences when queried with training versus non-training data. Building on this insight, we propose a method that utilizes model activation information to detect training data, thereby enabling MIAs against LLMs. We fine-tune LLMs using carefully selected datasets to determine the membership status of data samples. We collect texts from multiple platforms and construct three datasets, namely WikiTection, NewsTection, and ArXivTection. Each dataset contains data released after the latest known training cutoff dates of the LLMs, ensuring that all training samples are genuinely novel to the models. Next, we generate prompts from the constructed datasets and input them into the LLM to obtain the corresponding activation values. These activations are then normalized, and three feature extraction methods, namely ***NonFE***, ***StatFE***, and ***HistFE***, are employed to extract feature vectors that represent the input texts. Finally, we train a triplet network to learn activation features of member and non-member texts, enabling the model to capture subtle distinctions and substantially improving the accuracy and robustness of training data detection. Extensive experiments across various model architectures and datasets demonstrate that our approach not only achieves outstanding performance but also exhibits strong robustness and wide applicability.

ACKNOWLEDGMENT

Xue Tan, Hao Luan, Mingyu Luo, Zhuyang Yu, and Ping Chen were funded in part by the National Key R&D Program of China under Grant No. 2022YFB3102902.

REFERENCES

[1] 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," *NeurIPS*, vol. 33, pp. 1877–1901, 2020.

[2] OpenAI2023, "Gpt-4 technical report," https://cdn.openai.com/papers/gpt-4.pdf, 2023.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé *et al.*, "Bloom: A 176b-parameter open-access multilingual language model," *arXiv preprint arXiv:2211.05100*, 2022.

[5] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

[6] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[7] ——, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[8] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[9] H. Face, "Hugging face models," https://huggingface.co/models, 2025, accessed: 2025-03-25.

[10] M. AI, "Meta ai blog," https://ai.meta.com/blog/, 2025, accessed: 2025-03-25.

[11] X. Zhao, L. Li, and Y.-X. Wang, "Provably confidential language modelling," in *NAACL*, 2022, pp. 943–955.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[13] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[14] A. Yang, B. Xiao, B. Wang, B. Zhang, C. Bian, C. Yin, C. Lv, D. Pan, D. Wang, D. Yan *et al.*, "Baichuan 2: Open large-scale language models," *arXiv preprint arXiv:2309.10305*, 2023.

[15] K. Chang, M. Cramer, S. Soni, and D. Bamman, "Speak, memory: An archaeology of books known to chatgpt/gpt-4," in *EMNLP*, 2023.

[16] M. Mozes, X. He, B. Kleinberg, and L. D. Griffin, "Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities," *arXiv preprint arXiv:2308.12833*, 2023.

[17] M. M. Grynbaum and R. Mac, "The Times sues OpenAI and Microsoft over A.I. use of copyrighted work," https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html, 2023.

[18] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[19] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *USENIX Security 21*, 2021.

[20] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, D. Chen, and L. Zettlemoyer, "Detecting pretraining data from large language models," in *ICLR*, 2024.

[21] J. Mattern, F. Mireshghallah, Z. Jin, B. Schoelkopf, M. Sachan, and T. Berg-Kirkpatrick, "Membership inference attacks against language models via neighbourhood comparison," *Findings of ACL*, 2023.

[22] M. Meeus, S. Jain, M. Rei, and Y.-A. de Montjoye, "Did the neurons read your book? document-level membership inference for large language models," in *USENIX Security*, 2024, pp. 2369–2385.

[23] W. Zhang, R. Zhang, J. Guo, M. Rijke, Y. Fan, and X. Cheng, "Pretraining data detection for large language models: A divergence-based calibration method," in *EMNLP*, 2024, pp. 5263–5274.

[24] Q. Sun, H. Wu, and X. S. Zhang, "On active privacy auditing in supervised fine-tuning for white-box language models," *arXiv preprint arXiv:2411.07070*, 2024.

[25] D. Das, J. Zhang, and F. Tramèr, "Blind baselines beat membership inference attacks for foundation models," *arXiv preprint arXiv:2406.16201*, 2024.

[26] A. V. Duarte, X. Zhao, A. L. Oliveira, and L. Li, "De-cop: detecting copyrighted content in language models training data," in *ICML*, 2024.

[27] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[28] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," in *ICML*. PMLR, 2023, pp. 2397–2430.

[29] J. Li, A. Fang, G. Smyrnis, M. Ivgi, M. Jordan, S. Y. Gadre, H. Bansal, E. Guha, S. S. Keh, K. Arora *et al.*, "Datacomp-lm: In search of the next generation of training sets for language models," *NeurIPS*, 2024.

[30] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[31] M. Duan, A. Suri, N. Mireshghallah, S. Min, W. Shi, L. Zettlemoyer, Y. Tsvetkov, Y. Choi, D. Evans, and H. Hajishirzi, "Do membership inference attacks work on large language models?" in *COLM*, 2024.

[32] S. Narang and A. Chowdhery, "Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance," *Google AI Blog*, 2022.

[33] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu *et al.*, "Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," *arXiv preprint arXiv:2107.02137*, 2021.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[35] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever, "Better language models and their implications," *OpenAI blog*, vol. 1, no. 2, 2019.

[36] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[37] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," *The elements of statistical learning: Data mining, inference, and prediction*, pp. 485–585, 2009.

[38] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *SP*. IEEE, 2022, pp. 1897–1914.

[39] V. Feldman, "Does learning require memorization? a short tale about a long tail," in *STOC*, 2020, pp. 954–959.

[40] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[41] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE transactions on services computing*, vol. 14, no. 6, pp. 2073–2089, 2019.

[42] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” in *CSF*. IEEE, 2018, pp. 268–282.

[43] Z. Liu, T. Zhu, C. Tan, B. Liu, H. Lu, and W. Chen, “Probing language models for pre-training data detection,” in *ACL*, 2024, pp. 1576–1587.

[44] L. Ibanez-Lissen, L. Gonzalez-Manzano, J. M. de Fuentes, N. Anciaux, and J. Garcia-Alfaro, “Lumia: Linear probing for unimodal and multimodal membership inference attacks leveraging internal llm states,” in *ESORICS*. Springer, 2025, pp. 186–206.

[45] I. Melekhov, J. Kannala, and E. Rahtu, “Siamese network features for image matching,” in *ICPR*. IEEE, 2016, pp. 378–383.

[46] S. Chang, W. Li, Y. Zhang, and Z. Feng, “Online siamese network for visual object tracking,” *Sensors*, vol. 19, no. 8, p. 1858, 2019.

[47] Y. Huang, A. Li, B. Zhou, J. Huang, L. Lan, X. Yin, and Y. Jia, “Person entity attribute extraction based on siamese network,” *IEEE Access*, vol. 7, pp. 64 506–64 516, 2019.

[48] T. Mueller, G. Pérez-Torró, and M. Franco-Salvador, “Few-shot learning with siamese networks and label tuning,” in *ACL*, 2022, pp. 8532–8545.

[49] “Common crawl,” https://commoncrawl.org.

[50] M. Hart, “Project gutenberg.” [Online]. Available: https://www.gutenberg.org/

[51] J. He, Y. Gong, Z. Lin, C. Wei, Y. Zhao, and K. Chen, “Llm factoscope: Uncovering llms’ factual discernment through measuring inner states,” in *Findings of ACL*, 2024, pp. 10 218–10 230.

[52] K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau, “Mass-editing memory in a transformer,” in *ICLR*, 2023.

[53] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016, pp. 770–778.

[54] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *CVPR*, 2015, pp. 815–823.

[55] S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, E. Arbus, R. K. Arora, Y. Bai, B. Baker, H. Bao *et al.*, “gpt-oss-120b & gpt-oss-20b model card,” *arXiv preprint arXiv:2508.10925*, 2025.

[56] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, “Qwen3 technical report,” *arXiv preprint arXiv:2505.09388*, 2025.

[57] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo *et al.*, “Solving quantitative reasoning problems with language models,” *NeurIPS*, vol. 35, pp. 3843–3857, 2022.

[58] X. Li, Y. Zhang, R. Lou, C. Wu, and J. Wang, “Chain-of-scrutiny: Detecting backdoor attacks for large language models,” *arXiv preprint arXiv:2406.05948*, 2024.

[59] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

Appendix A
Additional Details of Experiment

A. *Details and Results of Finetuning*

Table IX presents the key parameters used for fine-tuning the target LLM, which are determined based on the model’s convergence behavior during the fine-tuning process. Table X presents the performance of the fine-tuned LLM on the training and test sets. We employ three metrics, namely BLEU-4, ROUGE-1, and ROUGE-L, to evaluate the quality of the generated text. The results show that the scores on the training and test sets are highly consistent across all datasets, with differences of less than 0.03. This indicates that the model maintains stable generation quality on unseen data, demonstrating strong generalization ability without signs of overfitting. The performance differences across datasets are mainly attributed to variations in domain and writing style. Specifically, the structurally regular WikiTection dataset achieves better performance, whereas the more complex ArXivTection dataset performs relatively worse.

TABLE IX: Main hyperparameters used in finetuning target model.

Dataset	**Target Model**			
	learning rate	**batch size**	**epoch**	**max-seq-len**
WikiTection	2e-5	16	3	512
NewsTection	2e-5	16	4	512
ArxivTection	2e-5	16	3	2048

TABLE X: Performance comparison of different LLMs on training and test sets across three datasets.

Dataset	**Application**	**Metrics**	**LLMs**						
			GPT2 -xl	**LLaMA2 -7B**	**LLaMA3 -8B**	**Mistral -7B**	**LLaMA2 -13B**	**GPT-OSS -20B**	**Qwen3 -8B**
WikiTection	Training Set	BLEU-4	0.736	0.755	0.820	0.816	0.753	0.569	0.727
		ROUGE-1	0.810	0.870	0.889	0.896	0.864	0.612	0.790
		ROUGE-L	0.809	0.867	0.886	0.894	0.861	0.597	0.750
	Test Set	BLEU-4	0.721	0.748	0.813	0.773	0.747	0.531	0.731
		ROUGE-1	0.794	0.858	0.879	0.889	0.859	0.578	0.784
		ROUGE-L	0.791	0.856	0.876	0.885	0.856	0.561	0.787
NewsTection	Training Set	BLEU-4	0.379	0.407	0.465	0.458	0.415	0.187	0.232
		ROUGE-1	0.363	0.387	0.444	0.456	0.403	0.263	0.341
		ROUGE-L	0.329	0.359	0.413	0.436	0.375	0.212	0.306
	Test Set	BLEU-4	0.365	0.387	0.448	0.446	0.406	0.157	0.256
		ROUGE-1	0.346	0.371	0.439	0.447	0.395	0.230	0.356
		ROUGE-L	0.317	0.334	0.404	0.415	0.357	0.181	0.324
ArXivTection	Training Set	BLEU-4	0.282	0.399	0.392	0.412	0.396	0.141	0.239
		ROUGE-1	0.228	0.240	0.244	0.244	0.239	0.127	0.215
		ROUGE-L	0.139	0.153	0.155	0.155	0.151	0.103.	0.161
	Test Set	BLEU-4	0.249	0.332	0.338	0.339	0.333	0.136	0.242
		ROUGE-1	0.190	0.196	0.198	0.197	0.198	0.117	0.234
		ROUGE-L	0.135	0.143	0.145	0.144	0.145	0.101	0.189

B. *Prompt*

You are a helpful assistant. Below is an original passage along with its corresponding topic. Please rewrite and paraphrase the passage in a way that preserves its meaning while altering the sentence structure and expression.
Original Passage:[original passage]
Topic: [topic]

Fig. 9: Prompt used to guide GPT-4o in generating semantically equivalent paraphrases of the given context.

Appendix B
Supplementary Experimental Results and Analyses

A. *The Performance of NART*

In order to provide a more comprehensive evaluation of the proposed NART method under membership inference attack (MIA) scenarios, we further introduce two additional metrics: TPR@3%FPR and TPR@1%FPR. These metrics assess the accuracy of membership inference under stricter false positive rate constraints. As shown in Table XI, NART maintains high membership inference accuracy even under stringent FPR conditions. For example, on the WikiTection dataset, when different feature extraction strategies are applied and LLMs such as GPT2-xl, LLaMA3-8B, and Mistral-7B are used, the TPR@1%FPR ranges from 0.968 to 0.994. These results further validate the effectiveness of the proposed method, indicating its ability to reliably determine whether a data sample was part of an LLM's training set.

TABLE XI: The performance of NART across different LLMs.

Dataset	FeaEXTRACT	Metrics	LLMs		
			GPT2-xl	LLaMA3-8B	Mistral-7B
WikiTection	*NonFE*	TPR@3%FPR	0.991	0.991	0.993
		TPR@1%FPR	0.968	0.990	0.989
	StatFE	TPR@3%FPR	0.982	0.995	0.995
		TPR@1%FPR	0.973	0.994	0.993
	HistFE	TPR@3%FPR	0.995	0.989	0.995
		TPR@1%FPR	0.977	0.978	0.991
ArXivTection	*NonFE*	TPR@3%FPR	0.971	0.987	0.982
		TPR@1%FPR	0.955	0.986	0.973
	StatFE	TPR@3%FPR	0.981	0.977	0.973
		TPR@1%FPR	0.955	0.971	0.965
	HistFE	TPR@3%FPR	0.982	0.978	0.964
		TPR@1%FPR	0.973	0.968	0.932

TABLE XII: AUC and TPR@5%FPR results of NART with varying training dataset sizes.

Dataset	Dataset	GPT2-xl		LLaMA3-8B		Mistral-7B	
	Size	AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
WikiTection	50	0.994	0.989	0.991	0.976	0.995	0.993
	100	0.994	0.990	0.998	0.989	0.996	0.990
	200	0.995	0.991	0.999	0.990	0.998	0.996
ArXivTection	50	0.988	0.973	0.998	0.987	0.990	0.986
	100	0.991	0.978	0.996	0.989	0.988	0.989
	200	0.989	0.980	0.993	0.991	0.991	0.983

B. Generation

Less Training Data. Table XII presents the performance of NART with limited training data. In this setting, the test set contains 400 samples, while the support set consists of 50 samples. The experimental results indicate that, due to the twin-network architecture and the use of triplet loss, NART achieves an AUC above 0.986 and a TPR@5%FPR exceeding 0.973, even with only 50 training samples. This demonstrates the robustness and applicability of NART in extreme scenarios, particularly when access to LLM training data is severely limited.

Training and Testing Data from Different Sources. To further evaluate the generalization capability of the proposed method, we simulate a realistic scenario in which the training and testing data originate from different sources. In this experiment, the training set is constructed from two datasets, while the test set is drawn from a third, distinct dataset. The training set contains 3,000 samples, the test set 1,000 samples, and the support set 200 samples. As shown in Table XIII, NART achieves the best performance when the training data come from NewsTection and ArXivTection and the test data are from WikiTection, with AUC values exceeding 0.970 and TPR@5%FPR above 0.930. In contrast, when the test set consists of samples from NewsTection or ArXivTection, the performance of NART drops noticeably. We attribute this to the higher degree of specialization and complexity in NewsTection and ArXivTection compared to WikiTection, making models trained on these datasets more adaptable to WikiTection, whereas the reverse transfer is less effective.

TABLE XIII: Generalization performance of NART. Abbreviations: Wiki (WikiTection), News (NewsTection), ArXiv (ArXivTection).

Training Dataset	Test Dataset	Metrics	LLMs		
			GPT2-xl	LLaMA3-8B	Mistral-7B
Wiki & News	ArXiv	TPR@5%FPR	0.567	0.405	0.894
		AUC	0.918	0.751	0.970
News & ArXiv	Wiki	TPR@5%FPR	0.932	0.959	0.951
		AUC	0.972	0.989	0.978
Wiki & ArXiv	News	TPR@5%FPR	0.206	0.526	0.209
		AUC	0.712	0.889	0.762

C. Comparison of Model Architecture

The proposed membership inference model is designed as a triplet network based on a Siamese architecture and few-shot learning principles. This design enables the model to maintain high performance even under extremely limited training data, as shown in Table XII. We also compare our approach with standard multilayer perceptron (MLP) and logistic regression classifiers, as summarized in Table XIV. The results show that both MLP and logistic regression models suffer substantial performance degradation and instability when trained with limited samples, as they are unable to capture the subtle activation differences between member and non-member samples. In contrast, our proposed model effectively amplifies fine-grained representation differences and consistently achieves accurate and stable membership inference under such conditions.

TABLE XIV: AUC and TPR@5%FPR results of NART across different inference model architectures on ArXivTection.

Model	Dataset	GPT2-xl		LLaMA3-8B		Mistral-7B	
	Size	AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
MLP	50	0.875	0.750	0.910	0.000	0.953	0.893
	200	0.978	0.960	0.991	0.989	0.993	0.986
Logistic Regression	50	0.882	0.702	0.849	0.725	0.902	0.000
	200	0.952	0.858	0.946	0.893	0.962	0.936
Triplet Networks	50	0.988	0.973	0.998	0.987	0.990	0.986
	200	0.989	0.980	0.993	0.991	0.991	0.983

D. Performance of NART on Pretrained Language Models

Pythia is a suite of LLMs [28] trained on the Pile [30], a corpus consisting of 825 GiB of diverse text from 22 sources spanning academic writing, internet content, and dialogue. The MIMIR dataset [31] is constructed from both the training and test subsets of the Pile. The DCLM benchmark [29] consists of 240T text tokens extracted from Common Crawl, and a series of models are trained on different subsets of this benchmark. Since the models trained on these benchmarks use publicly accessible training and testing data, these benchmarks are well suited for MIA evaluation and have been widely adopted as reliable benchmarks in privacy research [25], [31].

We sampled 1,000 training examples and 1,000 testing examples from the Pile, MIMIR, and DCLM datasets to evaluate our membership inference method on the Pythia-12B and DCLM-1B models. The training samples and testing samples are used as member and non-member data, respectively. As shown in Table XV and Table XVI, our method achieves strong membership inference performance on the Pile, MIMIR, and DCLM datasets using pretrained models, with TPR@10%FPR scores of 0.911, 0.931, and 0.980 (AUC scores of 0.923, 0.955, and 0.974), respectively. However, compared to fine-tuned models, the performance is moderately lower. This is likely due to the large scale of the pretraining corpus, where each sample is typically observed only once during training, resulting in weaker memorization and reduced membership inference accuracy.

TABLE XV: The performance of NART on Pythia-12B.

Dataset	FeaEXTRACT	Pythia-12B				
		AUC	TPR@10%FPR	TPR@5%FPR	TPR@3%FPR	TPR@1%FPR
Pile	*NonFE*	0.923	0.911	0.683	0.475	0.188
	StatFE	0.931	0.916	0.691	0.415	0.201
	HistFE	0.918	0.899	0.654	0.489	0.199
MIMIR	*NonFE*	0.955	0.931	0.802	0.446	0.305
	StatFE	0.932	0.921	0.525	0.465	0.297
	HistFE	0.967	0.941	0.832	0.723	0.356

TABLE XVI: The performance of NART on DCLM-1B.

Dataset	FeaEXTRACT	DCLM-1B				
		AUC	TPR@10%FPR	TPR@5%FPR	TPR@3%FPR	TPR@1%FPR
DCLM-Baseline	*NonFE*	0.974	0.980	0.970	0.584	0.386
	StatFE	0.935	0.901	0.733	0.446	0.149
	HistFE	0.988	0.990	0.980	0.842	0.436

TABLE XVII: Performance comparison of different LLMs on training and test sets.

Dataset	Application	Metrics	LLMs		
			GPT2-xl	LLaMA3-8B	Mistral-7B
WikiTection	Training Set	BLEU-4	0.838	0.851	0.831
		ROUGE-1	0.908	0.943	0.934
		ROUGE-L	0.902	0.942	0.932
	Test Set	BLEU-4	0.800	0.828	0.785
		ROUGE-1	0.868	0.893	0.897
		ROUGE-L	0.865	0.897	0.895
NewsTection	Training Set	BLEU-4	0.537	0.579	0.573
		ROUGE-1	0.537	0.581	0.575
		ROUGE-L	0.503	0.560	0.551
	Test Set	BLEU-4	0.513	0.559	0.555
		ROUGE-1	0.517	0.558	0.563
		ROUGE-L	0.483	0.533	0.539
ArXivTection	Training Set	BLEU-4	0.262	0.317	0.316
		ROUGE-1	0.200	0.214	0.211
		ROUGE-L	0.164	0.180	0.181
	Test Set	BLEU-4	0.254	0.317	0.313
		ROUGE-1	0.196	0.211	0.202
		ROUGE-L	0.163	0.181	0.176

E. Performance of NART on Massive Datasets

In practice, LLMs are typically trained or fine-tuned on massive datasets. To evaluate the effectiveness of our proposed method under large-scale data conditions, we expanded the WikiTection, NewsTection and ArXivTection datasets to include 100K samples each and conducted experiments with training set sizes of 20K, 50K, and 100K. Table XVII presents the performance of the fine-tuned model on the training and test sets when the fine-tuning dataset contains 100K samples and the test set contains 10K samples. The results show that the training and test performances are highly consistent across different datasets, with a difference of less than 0.05, indicating that the fine-tuned model has strong generalization ability and does not exhibit overfitting. As shown in Table XVIII, our method can still accurately determine whether a given sample belongs to the training set, even when the LLM is trained on tens of thousands of data samples. For example, with the 100K WikiTection dataset for fine-tuning, our method achieves an TPR@5%FPR of 0.990 and AUC of 0.997 on GPT2-xl.

F. Maximum Context Length

We further conducted experiments to evaluate the effectiveness of our method when the input length reaches the maximum context length supported by each LLM. In this study, we selected four LLMs with different maximum context lengths (GPT2-xl with 1K, LLaMA2-7B with 4K, LLaMA3-8B with 8K, and LLaMA3-8B-32K with 32K) and constructed input samples that fully utilize each model's maximum context length for fine-tuning and subsequent activation extraction. For original texts shorter than the required length, we replicated the content to extend them to the target length. As shown in Table XIX, our method remains effective under the maximum context-length setting and some models even achieve better performance than with shorter inputs (Table I).

G. Performance of NART on Datasets from Earlier Model Versions

In practical applications, the proposed detection model often needs to leverage activation data from newer versions of LLMs to identify samples that appeared during the training of earlier versions. To simulate this scenario, we conducted multiple rounds of fine-tuning, using activation data from different update iterations to train the detection model, which was then used to determine the membership status of samples from the initial fine-tuning stage. As shown in Table XX, the results demonstrate that detection models trained on activation data from newer LLM versions exhibit reduced effectiveness when inferring the membership status of data utilized in earlier training stages. Specifically, the model's inference performance degrades for data belonging to older versions. For example, when we fine-tuned GPT2-xl on the NewsTection dataset for 3 and 6 rounds, the AUC of the detection model on the first-round fine-tuning data decreased from 0.971 to 0.843. Nonetheless, the overall performance remains acceptable.

H. Impact of Token Position on NART Detection Performance

We compared the performance of detection models trained and tested using activations from tokens at different positions within the input. As shown in Table XXI, we conducted

TABLE XVIII: The performance of NART across Massive Datasets.

Dataset	Size	Metrics	GPT2-xl			LLaMA3-8B			Mistral-7B		
			NonFE	*StatFE*	*HistFE*	*NonFE*	*StatFE*	*HistFE*	*NonFE*	*StatFE*	*HistFE*
WikiTection	20K	TPR@5%FPR	0.979	0.980	0.982	0.990	0.990	0.992	0.992	0.990	0.994
		AUC	0.995	0.989	0.994	0.996	0.997	0.998	0.996	0.998	0.999
	50K	TPR@5%FPR	0.980	0.960	0.990	0.996	0.994	0.992	0.994	0.990	0.990
		AUC	0.997	0.979	0.998	0.998	0.999	0.999	0.998	0.999	0.999
	100K	TPR@5%FPR	0.990	0.919	0.989	0.995	0.994	0.990	0.990	0.988	0.992
		AUC	0.997	0.962	0.998	0.998	0.998	0.996	0.996	0.994	0.996
NewsTection	20K	TPR@5%FPR	0.995	0.990	0.996	0.990	0.990	0.986	0.980	0.990	0.995
		AUC	0.998	0.996	0.999	0.991	0.998	0.995	0.995	0.996	0.998
	50K	TPR@5%FPR	0.990	0.985	0.990	0.980	0.980	0.986	0.988	0.980	0.990
		AUC	0.995	0.995	0.998	0.991	0.990	0.995	0.996	0.992	0.998
	100K	TPR@5%FPR	0.989	0.980	0.990	0.990	0.980	0.990	0.990	0.986	0.990
		AUC	0.998	0.990	0.997	0.998	0.990	0.996	0.998	0.995	0.998
ArXivTection	20K	TPR@5%FPR	0.951	0.949	0.953	0.980	0.985	0.982	0.980	0.990	0.990
		AUC	0.984	0.981	0.986	0.994	0.990	0.990	0.995	0.996	0.998
	50K	TPR@5%FPR	0.963	0.958	0.966	0.988	0.982	0.990	0.990	0.982	0.990
		AUC	0.978	0.972	0.980	0.996	0.992	0.994	0.994	0.993	0.998
	100K	TPR@5%FPR	0.959	0.950	0.953	0.990	0.985	0.988	0.995	0.990	0.992
		AUC	0.979	0.951	0.981	0.995	0.994	0.994	0.999	0.998	0.999

TABLE XIX: Evaluation of NART on LLMs under Maximum Context Length Input.

Dataset	Models	NonFE		StatFE		HistFE	
		AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
WikiTection	GPT2-xl	0.999	0.992	0.995	0.990	0.998	0.995
	LLaMA2-7B	0.998	0.990	0.996	0.988	0.999	0.990
	LLaMA3-8B	0.998	0.995	0.996	0.990	0.999	0.990
	LLaMA2-7B-32K	0.999	0.995	0.995	0.990	0.996	0.995
NewsTection	GPT2-xl	0.998	0.990	0.996	0.990	0.999	0.995
	LLaMA2-7B	0.980	0.960	0.985	0.975	0.986	0.970
	LLaMA3-8B	0.990	0.980	0.986	0.960	0.991	0.980
	LLaMA2-7B-32K	0.999	0.995	0.996	0.990	0.992	0.990
ArXivTection	GPT2-xl	0.996	0.989	0.991	0.985	0.993	0.991
	LLaMA2-7B	0.995	0.990	0.989	0.980	0.991	0.983
	LLaMA3-8B	0.996	0.992	0.995	0.989	0.998	0.994
	LLaMA2-7B-32K	0.996	0.992	0.990	0.988	0.995	0.992

TABLE XX: Performance of NART under different update times across multiple LLMs.

Dataset	Update Times	GPT2-xl		LLaMA3-8B		Mistral-7B	
		AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
WikiTection	3	0.993	0.985	0.979	0.905	0.998	0.996
	6	0.983	0.927	0.949	0.786	0.974	0.943
NewsTection	3	0.971	0.858	0.934	0.612	0.991	0.964
	6	0.843	0.631	0.871	0.384	0.919	0.579
ArXivTection	3	0.984	0.975	0.995	0.989	0.990	0.986
	6	0.970	0.831	0.990	0.988	0.989	0.982

experiments using the activations of the first token, a middle-position token, and the last token. The results show that the activations of the last token achieve the best and most stable performance, as they more comprehensively capture the internal representation of the entire input, leading to clearer distinctions between member and non-member samples. For example, on the NewsTection dataset using GPT2-xl, the detection model achieved TPR@5%FPR scores of 0.010 and 0.089 when using the first-token and middle-token activations, respectively, whereas the last-token activation yielded 0.918.

TABLE XXI: Performance of NART on Activations from Tokens at Different Positions.

Dataset	Position of token	GPT2-xl		LLaMA3-8B		Mistral-7B	
		AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
WikiTection	first	0.528	0.040	0.572	0.040	0.594	0.050
	middle	0.616	0.218	0.847	0.594	0.812	0.198
	last	0.992	0.991	0.998	0.991	0.996	0.997
NewsTection	first	0.501	0.010	0.601	0.065	0.489	0.030
	middle	0.561	0.089	0.937	0.648	0.526	0.079
	last	0.977	0.918	0.988	0.941	0.985	0.959
ArXivTection	first	0.514	0.019	0.535	0.040	0.525	0.139
	middle	0.591	0.178	0.993	0.980	0.538	0.069
	last	0.986	0.983	0.994	0.988	0.989	0.982

I. Blind Attacks on Datasets

In the WikiTection, NewsTection, and ArXivTection datasets we constructed, all samples occur after the latest known training cutoff dates of the LLMs used in our experiments. The member samples for fine-tuning and the non-member samples for evaluation are randomly selected from the three datasets. We employed various blind attack methods [25] to test whether there are distribution shifts between the member and non-member samples. The experimental results, as shown in Table XXII, indicate that the blind attacks have minimal impact on our datasets (with TPR@5%FPR below 0.10, compared to a much higher value of 0.947 reported in [25] when the blind attacks succeed), demonstrating that there are no distribution shifts in our datasets.

TABLE XXII: Performance comparison across different blind attacks.

Dataset	Date Detection		Bag-of-words classification		Greedy rare word selection	
	AUC	TPR@5%FPR	AUC	TPR@5%FPR	AUC	TPR@5%FPR
WikiTection	0.512	0.056	0.514	0.066	0.495	0.075
NewsTection	0.497	0.051	0.489	0.044	0.526	0.086
ArXivTection	0.523	0.086	0.492	0.045	0.502	0.060