A General Method for Detecting Information Generated by Large Language Models

Minjia Mao*
University of Delaware; mjmao@udel.edu

Donjun Wei*
The University of Hong Kong; dongjun@connect.hku.hk

Xiao Fang
University of Delaware; xfang@udel.edu

Michael Chau
The University of Hong Kong; mchau@business.hku.hk

*Equal Contributions

Abstract: The proliferation of large language models (LLMs) has significantly transformed the digital information landscape, making it increasingly challenging to distinguish between human-written and LLM-generated content. Detecting LLM-generated information is essential for preserving trust on digital platforms (e.g., social media and e-commerce sites) and preventing the spread of misinformation, a topic that has garnered significant attention in IS research. However, current detection methods, which primarily focus on identifying content generated by specific LLMs in known domains, face challenges in generalizing to new (i.e., unseen) LLMs and domains. This limitation reduces their effectiveness in real-world applications, where the number of LLMs is rapidly multiplying and content spans a vast array of domains. In response, we introduce a general LLM detector (GLD) that combines a twin memory networks design and a theory-guided detection generalization module to detect LLM-generated information across unseen LLMs and domains. Using real-world datasets, we conduct extensive empirical evaluations and case studies to demonstrate the superiority of GLD over state-of-the-art detection methods. The study has important academic and practical implications for digital platforms and LLMs.

Keywords: large language model, detection method, LLM-generated information, generative model, deep learning, memory network, discrepancy mitigation

1 INTRODUCTION

Large language models (LLMs) are artificial intelligence (AI) systems designed to understand human languages and generate human-like content (Ouyang et al. 2022). In recent years, the number of LLMs has grown exponentially, driven by both open-source communities and commercial companies. For example, Hugging Face, a platform hosting open-source LLMs, provides access to over 100,000 publicly available LLMs, with new ones added each month.[1] Leading technology companies, such as OpenAI and Cohere, roll out new versions of commercial LLMs every few months. The capabilities of LLMs like OpenAI's GPT-4o and Meta's LLaMA-3 have reached remarkable levels, enabling them to generate human-like content in seconds (Mousavi et al. 2020, Reisenbichler et al. 2022, Goli and Singh 2024). While the rapid advancement of LLMs has resulted in many useful applications, it has also fueled the creation of fake and misleading information online (Berente et al. 2021, Fang et al. 2024). For instance, social media platforms like X (formerly known as Twitter) and Facebook now contain an increasing amount of LLM-generated information across various domains, such as news and product reviews.[2] Similarly, a recent investigation reveals that over 40% of articles on Medium, a prominent U.S. blogging platform, are generated by LLMs without fact-checking or proper editing, covering diverse domains such as personal stories, reports, and news.[3]

The enormous and rapidly growing amount of information produced by LLMs could impose serious harm on both information consumers and providers (e.g., digital platforms) (Pennycook and Rand 2021). First, since LLMs can produce highly human-like

[1] See `https://huggingface.co/blog/2023-in-llms` (last accessed on June 15, 2025)

[2] See `https://nymag.com/intelligencer/article/ai-generated-content-internet-online-slop-spam.html` (last accessed on June 15, 2025)

[3] See `https://www.wired.com/story/ai-generated-medium-posts-content-moderation` (last accessed on June 15, 2025)

content, it is becoming increasingly difficult for information consumers to distinguish between LLM-generated and human-written information, potentially exposing themselves to economic losses (Wei et al. 2022, Lee et al. 2024). For example, click farms now leverage LLMs to create human-like posts across various domains, often including deceptive links to mislead social media users into purchasing counterfeit or low-quality products.[4] Second, the growing volume of LLM-generated information undermines consumers' trust in information providers. For example, overwhelmed by blogs generated by various LLMs, Medium struggles to provide quality information to its users, resulting in users leaving the platform.[5] Moreover, the harm caused by LLM-generated information extends beyond the information providers who publish it. LLM-generated information could impact search engine rankings, thereby overshadowing high-quality human-written content sought by users.[6] As a result, the entire information ecosystem suffers from the proliferation of LLM-generated information. To protect information consumers and providers and preserve the integrity of the information ecosystem, it is critical to design a method that can accurately detect information generated by various LLMs across diverse domains.

The status quo for detecting LLM-generated information can be classified into two categories: zero-shot and learning-based methods. Zero-shot methods identify LLM-generated information without the need for training on labeled datasets. Instead, they employ a pre-trained LLM, known as a scoring LLM, to score each piece of information and then distinguish human-written information from LLM-generated information

[4] See `https://www.404media.co/facebooks-algorithm-is-boosting-ai-spam-that-links-to-ai-generated-ad-laden-click-farms` (last accessed on June 15, 2025)

[5] See `https://reutersinstitute.politics.ox.ac.uk/news/ai-generated-slop-quietly-conquering-internet-it-threat-journalism-or-problem-will-fix-itself` (last accessed on June 15, 2025)

[6] See `https://www.theverge.com/2024/7/31/24210304/wix-ai-website-builder-seo-blog-posts` (last accessed on June 15, 2025)

based on their scores. For example, DetectGPT, a prominent zero-shot method, perturbs randomly selected tokens in a piece of information and then utilizes a scoring LLM to calculate the perplexity[7]of the information before and after the perturbation (Mitchell et al. 2023). An increase in perplexity after the perturbation suggests that the information is likely generated by an LLM. Zero-shot methods are effective in detecting information generated by the same LLM as the scoring LLM but are ineffective when the scoring LLM differs from the one used to generate the information (Liu et al. 2024). Different LLMs vary in model architecture, size, and training approach. As a result, a scoring LLM cannot reliably evaluate the information generated by a different LLM, leading to poor detection performance (Su et al. 2023). If an LLM that generates information to be detected differs from the scoring LLM utilized by a zero-shot method, we refer to the information-generating LLM as an ***unseen LLM*** for the method. Zero-shot methods are ineffective in detecting information produced by unseen LLMs (Liu et al. 2024).

Learning-based methods are trained on data comprising both human-written content and information generated by specific LLMs (e.g., ChatGPT and LLaMA) in particular domains (e.g., product reviews and news) (Gehrmann et al. 2019, Guo et al. 2023). For example, training data might contain human-written and ChatGPT-generated product reviews as well as human-written and LLaMA-generated news. These methods are trained to detect information produced by the same LLMs in the same domains as those in their training data, thereby generalizing poorly to other LLMs and domains (Fröhling and Zubiaga 2021, Mitchell et al. 2023). As a result, learning-based methods are ineffective in detecting information generated by LLMs in domains different from those in their training data (Li et al. 2023), which we refer to as ***unseen LLMs*** and ***unseen domains***. Given the

[7] Given a piece of information, perplexity is the exponential of the average negative log likelihood of each word in the information.

vast number of LLMs and the wide range of domains, existing learning-based methods can "see" only a few LLMs and domains — that is, they are trained on information generated by a small number of LLMs across several domains. Consequently, these methods perform poorly in detecting information generated by a large number of unseen LLMs in numerous unseen domains. Therefore, it is necessary to develop a general method that is trained on information produced by a few LLMs in several domains but can generalize to ***identify information produced by unseen LLMs in unseen domains effectively***.

In response to the aforementioned research gap, we propose a general LLM detector (GLD) to distinguish between human-written and LLM-generated information. Compared to existing methods for detecting LLM-generated content, the key advantage of GLD is its ability to generalize to unseen LLMs and domains. To achieve this advantage, our study introduces two methodological innovations. First, for each piece of information, it is essential to extract and represent features of the domain to which it belongs, as well as features of its author (e.g., human or an LLM). To this end, we propose a Twin Memory Networks (TMN) design, which learns the embeddings of the domain, author, and textual content for each piece of information. Second, we theoretically analyze the generalizability of a detection method to unseen LLMs and domains. Built on the theoretical analysis, we propose a Detection Generalization Module (DGM) that can effectively generalize to unseen LLMs and domains by learning LLM- and domain-invariant embeddings. Empirically, we evaluate our GLD method on a dataset that combines human-written information and information generated by multiple LLMs, such as GPT-4o and LLaMA-3, in multiple domains, including news and online reviews. The empirical results demonstrate that our proposed GLD method achieves state-of-the-art detection performance compared to benchmark methods, including both zero-shot and learning-based

methods, for unseen LLMs and domains. In addition, we conduct case studies in two unseen domains, namely online reviews and academic writing, to demonstrate the practicality of the proposed method in real-world applications that are highly relevant to IS research.

2 LITERATURE REVIEW

Existing methods for detecting LLM-generated information can be classified into two categories: zero-shot and learning-based methods. Learning-based methods can be further divided into feature-based and fine-tuning methods. In this section, we review representative methods in each category and highlight the key novelties of our study.

2.1 Zero-shot Methods

Zero-shot methods identify LLM-generated information without requiring training on labeled datasets. These methods utilize a pre-trained LLM to score each piece of information and differentiate human-written content from LLM-generated information on the basis of their scores. DetectGPT, a representative zero-shot method, perturbs (i.e., rewrites) randomly selected tokens in a piece of information and then employs a GPT model to evaluate its perplexity before and after the perturbation (Mitchell et al. 2023). A consistent increase in perplexity across multiple perturbations indicates that the information is likely LLM-generated. Fast-DetectGPT improves the efficiency of DetectGPT through sampling, while maintaining detection performance (Bao et al. 2023). Su et al. (2023) employs a scoring LLM to compute the likelihood of each token in a piece of information, given its previous tokens. They then define a ratio between the token's log-likelihood and its log-rank. They observe that the average ratio across tokens in human-written content is greater than that in LLM-generated information, making the ratio a useful metric to distinguish between them.

By utilizing a pre-trained LLM as the scoring model, a zero-shot method can effectively detect content generated by the same LLM as the scoring LLM. For example, when equipped with GPT-J as the scoring LLM, DetectGPT achieves an AUC of 0.92 in detecting GPT-J-generated information across various domains (Mitchell et al. 2023). However, relying on a particular scoring LLM limits the generalizability of a zero-shot method in detecting information generated by unseen LLMs that are different from the scoring LLM. For instance, when using with GPT-J as the scoring LLM, DetectGPT struggles to identify information generated by GPT-2, attaining an AUC of only 0.6 (Mitchell et al. 2023).

2.2 Feature-based and Fine-tuning Methods

Feature-based and fine-tuning methods are learning-based approaches trained using labeled data that include human-written content and information generated by specific LLMs in particular domains. Specifically, feature-based methods extract probability-based or linguistic features from training datasets. The extracted features and their corresponding labels are then used to train a classifier to distinguish between human-written and LLM-generated information. For example, GLTR utilizes GPT-2 to compute the likelihood rank of each token in a piece of information (Gehrmann et al. 2019). Compared to human-written information, LLM-generated information typically contains a higher frequency of high-ranked tokens. Therefore, a logistic regression model can be trained on the frequency of high-ranked tokens to differentiate between GPT-2-generated and human-written information. Perplexity, another probability-based feature, has also been used to identify LLM-generated information. Since LLM-generated information usually has lower perplexity than human-written content, a classifier can be built on it to detect LLM-generated information (Guo et al. 2023, Wu et al. 2025). Meanwhile, as an exemplar method leveraging linguistic features, Fröhling and Zubiaga (2021) extract such features

as readability and lexical diversity from a labeled dataset and then train a logistic regression model to identify GPT-generated information.

While feature-based methods perform well at detecting information generated by the same LLMs in the same domains as those in their training datasets, their performance deteriorates significantly when detecting information produced by unseen LLMs and domains different from those in the training data (Gehrmann et al. 2019). Probability-based features, such as token likelihood and perplexity, depend on the specific LLM used to calculate them (Wu et al. 2025). Consequently, their values vary across different LLMs. Similarly, linguistic features, such as readability and repetitiveness, differ across content generated by various LLMs. For instance, the readability score of information generated by ChatGPT is significantly different from that of information produced by GPT-2 (Li et al. 2023). Moreover, linguistic features like named entities and lexical diversity vary significantly across different domains (Liu et al. 2021, Guo et al. 2024a). Because of the reliance of probability-based and linguistic features on specific LLMs and domains, detection methods trained on these features generalize poorly to unseen LLMs and domains.

Unlike feature-based methods, fine-tuning methods do not extract features from training datasets. Instead, they fine-tune a transformer-based model, such as BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), and Longformer (Beltagy et al. 2020), on these datasets to classify information as either LLM-generated or human-written. For example, OpenAI introduces the GPT-2 output detector (Solaiman et al. 2019), which is implemented by fine-tuning RoBERTa on a dataset comprising both GPT-2-generated and human-written content. More recently, Liu et al. (2023) fine-tune RoBERTa to detect GPT-generated English essays and demonstrate that it outperforms zero-shot and feature-based methods. However, fine-tuning methods are prone to overfitting their training

datasets, thereby generalizing poorly to unseen LLMs and domains that differ from those in the training data (Mitchell et al. 2023). For example, OpenAI's GPT-2 output detector fails to identify information generated by its unseen LLMs, such as ChatGPT (Guo et al. 2023). As another example, Li et al. (2023) show that a method that fine-tunes the Longformer model on a training dataset of human-written content across 10 domains, along with corresponding texts generated by 27 LLMs, still struggles to detect information generated by an unseen LLM in an unseen domain.

2.3 Key Novelties of Our Method

As discussed above, existing zero-shot methods rely on a specific scoring LLM, making them ineffective in detecting information generated by unseen LLMs. Existing learning-based methods are trained on datasets comprising human-written texts and information generated by specific LLMs in particular domains. As a result, while these methods are good at detecting information generated by the same LLMs in the same domains as those in their training datasets, they generalize poorly to unseen LLMs and new domains.

Unlike existing methods, our proposed method can generalize to detect information generated by unseen LLMs in new domains effectively. This capability of our method is realized through two key methodological innovations. First, to generalize to unseen LLMs and domains, it is crucial to extract and represent domain and author features for each piece of information. Therefore, different from existing methods that focus solely on learning the embedding of the textual content for each piece of information (Solaiman et al. 2019, Bhattacharjee et al. 2024), our method also learns the embeddings of its domain and author, in addition to the textual content. Second, we conduct a rigorous theoretical analysis of the generalizability of a detection method to unseen LLMs and domains. Based on the analysis, we propose to detect LLM-generated information by learning

LLM- and domain-invariant embeddings for LLM-generated information and domain-invariant embeddings for human-written information.

3 PROBLEM DEFINITION

We consider the problem of detecting LLM-generated information in a general manner. As the number of LLMs grows rapidly and they are deployed to generate textual information across an increasingly wide range of domains, it is essential to develop a detection method that can generalize effectively to unseen LLMs and new domains, by training on documents [8] written by humans or generated by a few LLMs in a few domains. Concretely, let $\{H\} \cup G$ represent the authors of the collected documents, where H denotes humans and $G = \{G_1, G_2, \ldots, G_m\}$ denotes the set of m LLMs. Let $S = \{S_1, S_2, \cdots, S_n\}$ represent the set of n domains to which the collected documents belong. The training corpus consists of the collected N documents, each annotated with its author $g_k \in H \cup G$, domain $s_k \in S$, and label $y_k \in \{0, 1\}$, where $y_k = 0$ indicates a human-written document and $y_k = 1$ indicates one generated by an LLM, $k = 1, 2, \ldots, N$. We now formally define the general LLM-generated information detection problem as follows.

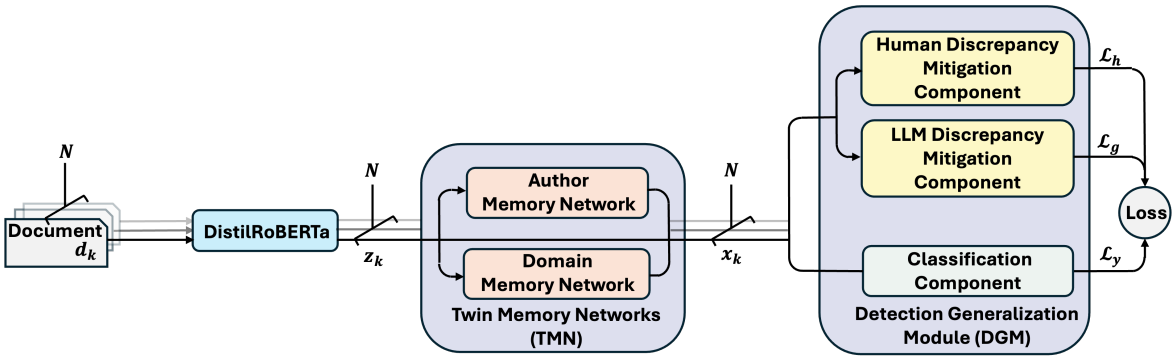
Definition 1 (General LLM-generated Information Detection Problem (GLID)). A training corpus contains N documents, each of which is written by humans H or generated by one of the m LLMs, denoted by $G = \{G_1, G_2, \cdots, G_m\}$, and belongs to one of the n domains, denoted by $S = \{S_1, S_2, \cdots, S_n\}$. Accordingly, each training document d_k is labeled as human-written $(y_k = 0)$ or LLM-generated $(y_k = 1)$, where $k = 1, 2, \ldots, N$. It is also annotated with its domain $s_k \in S$ and author $g_k \in H \cup G$. The objective is to learn a model from the training corpus to classify a test document written by humans or generated by any unseen LLMs (not in G) in any unseen domains (not in S), as human-written or LLM-generated.

[8] We use the term "document" to refer to an instance of textual information collected in the dataset, such as a news article, a review, or a story.

4 METHOD

We propose a general LLM detector (GLD) to solve the GLID problem. The overall architecture of the GLD method is shown in Figure 1. As illustrated, for each training document, GLD leverages a pre-trained model to represent its textual content as a textual embedding. The textual embedding is then passed to the Twin Memory Networks module of GLD, which comprises two memory networks with similar architectures. These networks extract the author and domain embeddings from the document, which are then combined with the textual embedding to form the final representation of the document. Using the final embeddings of training documents as input, the Detection Generalization Module learns a detector capable of effectively identifying information generated by unseen LLMs in new domains. We present the design of each module in Sections 4.1 and 4.2, respectively. Section 4.3 describes the training process of GLD and how it is used to classify a test document as LLM-generated or human-written. For the convenience of the reader, we summarize important notations in Table 1.

Figure 1 Overall Architecture of General LLM Detector (GLD).



4.1 Twin Memory Networks

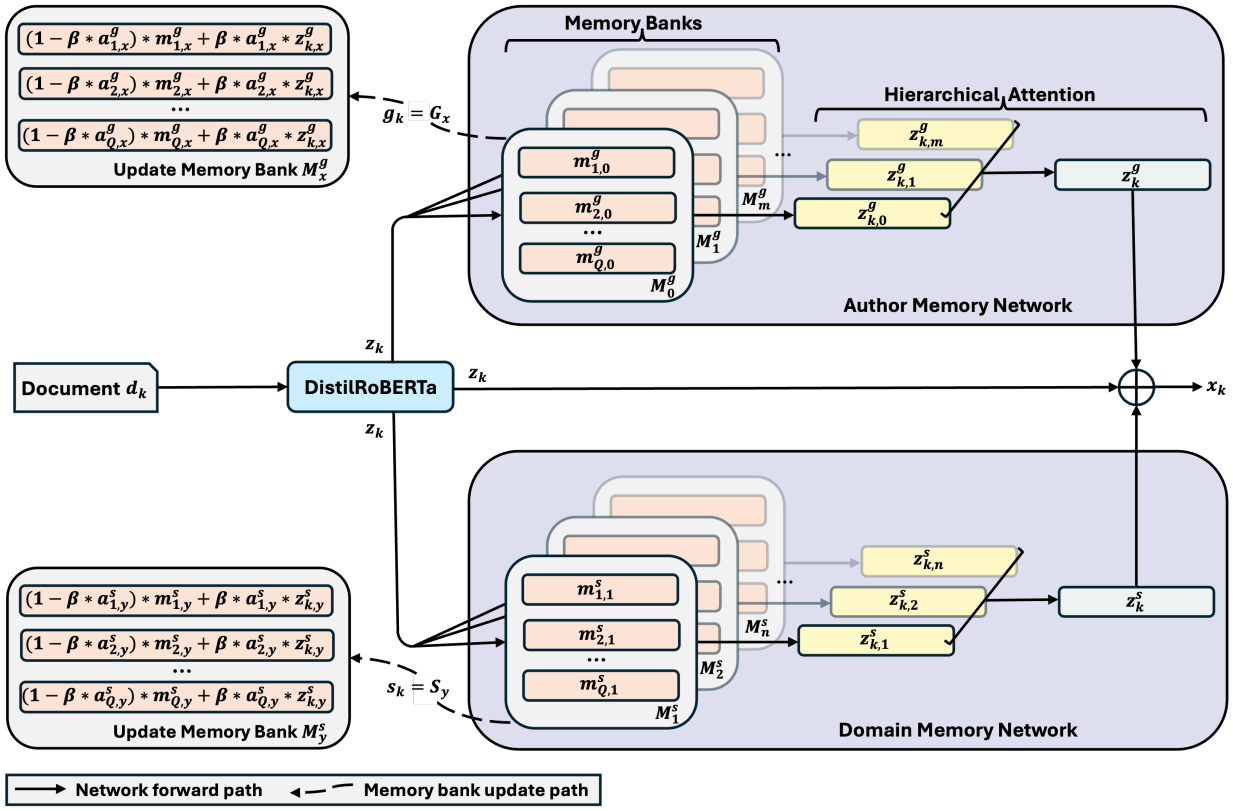
The architecture of the Twin Memory Networks (TMN) is shown in Figure 2. As depicted, given a training document d_k with its corresponding author label g_k and domain label s_k,

Table 1 Notation.

Notation	Description
H	Humans
G_i	An LLM, $i = 1, 2, \ldots, m$
S_j	A domain, $j = 1, 2, \ldots, n$
d_k	A document, $k = 1, 2, \ldots, N$
g_k	The author of document d_k, $g_k \in \{H, G_1, G_2, \ldots, G_m\}$
s_k	The domain to which document d_k belongs, $s_k \in \{S_1, S_2, \ldots, S_n\}$
y_k	The label of document d_k, $y_k = 0$ (human-written) or 1 (LLM-generated)
z_k	The textual embedding of document d_k
z_k^g, z_k^s	The document-specific author and domain embeddings of d_k
x_k	The final embedding of document d_k, which is the concatenation of z_k, z_k^g, and z_k^s
$\mathcal{D}_{H,S_j}$	The set of final embeddings corresponding to documents written by humans H in domain S_j
$\mathcal{D}_{G_i,S_j}$	The set of final embeddings corresponding to documents generated by LLM G_i in domain S_j
$\mathcal{D}_{ij}$	A labeled dataset, $\mathcal{D}_{ij} = \mathcal{D}_{H,S_j} \cup \mathcal{D}_{G_i,S_j}$
$\mathbb{D}_{ij}$	The distribution of document embeddings in $\mathcal{D}_{ij}$
$\mathbb{D}_u$	The distribution of embeddings corresponding to documents generated by unseen LLMs or written by humans in unseen domains
$\mathcal{L}_h$	Discrepancy mitigation loss (Human)
$\mathcal{L}_g$	Discrepancy mitigation loss (LLM)
$\mathcal{L}_y$	Classification loss

our design first leverages a pre-trained model (which is DistilRoBERTa (Sanh 2019) in this work) to represent the textual content of the document as a d-dimensional textual embedding $z_k \in R^d$. Next, the TMN utilizes two memory networks with a similar architecture to extract the author and domain embeddings from the input document, respectively. In the following subsections, we first detail the design of the author memory network. Next, we describe the domain memory network and illustrate how the final embedding x_k is constructed.

Figure 2 Architecture of Twin Memory Networks (TMN).



4.1.1 Author Memory Network

The goal of the author memory network is to represent the author information for each training document, which we refer to as a document-specific author embedding. To achieve this, we first introduce memory banks to store representations at the author level. A memory bank is a Neural Turing Machine that has external memory resources to remember its training instances (Graves et al. 2014). Unlike other embedding methods such as Word2Vec (Church 2017) and BERT (Devlin et al. 2019), memory banks can explicitly store representations and update them through a well-designed mechanism. Such memory banks are widely adopted in computer vision, such as video object detection, where they effectively retain representations of past frames (Zhou et al. 2024).

The architecture of our proposed memory banks is shown in Figure 2. Recall from Definition 1, the number of LLMs in the training data is m, and the number of authors is $m + 1$:

humans and m LLMs. Each author G_i, where $i = 0, 1, 2 \ldots, m$ and $G_0 = H$ (humans), is represented by a memory bank M_i^g, where the superscript g indicates that this memory bank is for author representation. Each memory bank contains Q memory units. Each memory unit is a d-dimensional vector, denoted as $m_{q,i}^g$, where $q = 1, 2 \ldots, Q$, and $i = 0, 1, 2 \ldots, m$. Memory bank M_i^g, which represents author G_i, is the concatenation of its component memory units:

$$M_i^g = \left[m_{1,i}^g \, m_{2,i}^g \, \cdots \, m_{Q,i}^g \right], \tag{1}$$

where $i = 0, 1, 2, \ldots, m$ and $M_i^g \in R^{d \times Q}$. Each memory unit $m_{q,i}^g$ represents a specific aspect of author G_i, where $q = 1, 2 \ldots, Q$. To initialize $m_{q,i}^g$, we denote Z_i as the set of textual embeddings of the training documents authored by G_i; that is, $Z_i = \{z_k \mid \forall d_k, g_k = G_i\}$. We then employ the K-means algorithm to group the textual embeddings in Z_i into Q clusters. Memory unit $m_{q,i}^g$ is then initialized as the mean embedding of cluster q, $q = 1, 2 \ldots, Q$, allowing it to capture a specific aspect of author G_i.

Next, we propose a two-level hierarchical attention network to generate a document-specific author embedding for each input document. As illustrated in Figure 2, the first level of the attention network targets memory units within each memory bank. Specifically, for each memory bank M_i^g, $i = 0, 1, 2 \ldots, m$, the attention network assigns higher weights to its memory units that are more relevant to z_k and lower weights to its memory units that are less relevant to z_k. Concretely, we have

$$\left[a_{1,i} \, a_{2,i} \, \ldots \, a_{Q,i} \right] = \text{softmax} \left(\frac{z_k^T W_a M_i^g}{\tau} \right), \tag{2}$$

where $a_{q,i}$ is the attention weight that measures the relevance of each memory unit $m_{q,i}^g$ in M_i^g to z_k, $q = 1, 2 \ldots, Q$, $[\cdot]^T$ denotes the transpose of a vector or matrix, $W_a \in R^{d \times d}$ is a trainable matrix, and τ is a temperature hyperparameter.[9] The memory units in M_i^g are

[9] The temperature hyperparameter τ adjusts the smoothness of the attention weight distribution, with a higher value producing a smoother distribution.

then aggregated using their corresponding weights and further transformed by a multi-layer perceptron (denoted as MLP_1) to produce the document-adjusted representation $z^g_{k,i}$ of author G_i:

$$z^g_{k,i} = \text{MLP}_1 \left(\sum_{q=1}^{Q} a_{q,i} m^g_{q,i} \right), \tag{3}$$

where $z^g_{k,i} \in R^d$ denotes the representation of author G_i adjusted by document d_k.

The second level of the attention network takes the document-adjusted author representations $z^g_{k,i}$ as input, $i = 0, 1, 2 \ldots, m$, and assigns weights to them based on their relevance to z_k. Concretely, we have:

$$[b_0\ b_2\ \ldots\ b_m] = \text{softmax} \left(\frac{z_k^\top W_b [z^g_{k,0}\ z^g_{k,1}\ \ldots\ z^g_{k,m}]}{\tau} \right), \tag{4}$$

where b_i denotes the attention weight that measures the relevance of $z^g_{k,i}$ to z_k, $i = 0, 1, 2 \ldots, m$, $W_b \in R^{d \times d}$ is a trainable matrix, and τ is the temperature hyperparameter. Finally, the document-specific author embedding z^g_k for document d_k is calculated as the weighted sum of $z^g_{k,i}$, which is further transformed by a multi-layer perceptron (denoted as MLP_2):

$$z^g_k = \text{MLP}_2 \left(\sum_{i=0}^{m} b_i z^g_{k,i} \right). \tag{5}$$

To better learn the author representations in the memory banks from training data, we propose an updating mechanism. Specifically, given a training document d_k with its author label $g_k = G_x$, where $G_x \in \{G_0, G_1, \cdots, G_m\}$, the mechanism only updates the memory units in memory bank M^g_x, which corresponds to author G_x. Recall that the first level of the attention network computes the attention weight $a_{q,x}$ for each memory unit $m^g_{q,x}$ in M^g_x, as well as the document-adjusted representation $z^g_{k,x}$ of author G_x, using Equations 2 and 3. Accordingly, the mechanism updates each memory unit $m^g_{q,x}$ in M^g_x as:

$$m^g_{q,x} = (1 - \beta a_{q,x}) m^g_{q,x} + \beta a_{q,x} z^g_{k,x} \quad \text{for} \quad q = 1, 2, \ldots, Q, \tag{6}$$

where β is a hyperparameter that adjusts the strength of updating. Equation 6 offers three key benefits. First, it uses the author label of a training document to ensure that only the memory units corresponding to that author are updated. This helps learn accurate author representations and prevents unintended updates to other authors' memory units. Second, after seeing a training document with author label G_x, the attention network computes and stores the document-adjusted author information in $z^g_{k,x}$. To reflect this information in the memory units associated with author G_x, Equation 6 updates them using $z^g_{k,x}$. Furthermore, memory units that are more relevant to the document receive larger updates, weighted by $a_{q,x}$. Third, memory units are fixed during testing. This eliminates the need for author labels during the testing phase, which aligns with the objective of this study, as author labels are unavailable during this phase.

4.1.2 Domain Memory Network and Summary

As shown in Figure 2, the domain memory network has a similar architecture as the author memory network. It aims to extract domain information from each training document and represent it as a document-specific domain embedding. As illustrated in the figure, each domain S_i is represented by a memory bank M^s_i, each of which consists of Q memory units, where $i = 1, 2 \ldots, n$. The memory units in M^s_i are initialized by applying the K-means algorithm to the textual embeddings of the training documents belonging to domain S_i, where $i = 1, 2 \ldots, n$. Next, a document-specific domain embedding z^s_k is generated for each training document using a two-level hierarchical attention network similar to that employed in the author memory network, where $k = 1, 2 \ldots, N$. The memory units in the domain memory network are then updated using an updating mechanism similar to the one used in the author memory network. Finally, for each training document d_k, its textual embedding z_k, document-specific author embedding z^g_k, and document-specific

domain embedding z_k^s are concatenated to form a single embedding x_k for the document, where $k = 1, 2 \ldots, N$.

In summary, we propose a novel TMN design that learns the document-specific author and domain embeddings for each document, in addition to the embedding of its textual content. TMN includes two parallel components: an author memory network and a domain memory network. Each follows a similar architecture, designed to extract the respective author and domain embeddings from a document. Take the author memory network as an example. It contains a set of memory banks, each storing information about a specific author. Taking these memory banks as input, TMN extracts the document-specific author embedding from a document using a hierarchical attention network. The memory banks are updated through a novel updating mechanism during training, which selectively determines which memory bank to update and dynamically adjusts the weights of its memory units.

4.2 Detection Generalization Module

Taking the embeddings x_k of training documents as input, for $k = 1, 2 \ldots, N$, the Detection Generalization Module (DGM) learns a detector that can be generalized to unseen LLMs and domains. In the following, we begin with a rigorous theoretical analysis of detection generalization across LLMs and domains. We then present the design of each component of the module, which is guided by the insights from the theoretical analysis.

We denote $\mathcal{D}_{G_i,S_j} = \{x_k | g_k = G_i, s_k = S_j, k = 1, 2, \ldots, N\}$ as the set of embeddings corresponding to training documents generated by LLM G_i in domain S_j, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Similarly, let $\mathcal{D}_{H,S_j} = \{x_k | g_k = H, s_k = S_j, k = 1, 2, \ldots, N\}$ be the set of embeddings representing training documents written by humans H in domain S_j, where $j = 1, 2, \ldots, n$. To learn a detector that can discriminate between LLM-generated

and human-written information, we construct labeled datasets, each of which contains embeddings corresponding to LLM-generated training documents (labeled $y = 1$) and human-written ones (labeled $y = 0$). Concretely, each labeled dataset $\mathcal{D}_{ij} = \mathcal{D}_{G_i,S_j} \cup \mathcal{D}_{H,S_j}$ consists of embeddings corresponding to LLM G_i-generated and human-written training documents in domain S_j, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. We denote the distribution of document embeddings in $\mathcal{D}_{ij}$ as $\mathbb{D}_{ij}$. Let f_{ij} be the function that gives the true probability that a document with embedding $x \sim \mathbb{D}_{ij}$ is LLM-generated; that is, $f_{ij}(x) = p(y = 1|x)$, where $x \sim \mathbb{D}_{ij}$. A hypothesis h maps a document with embedding x to its probability of being LLM-generated; i.e., $h(x) = p_h(y = 1|x)$. When applying hypothesis h to document embeddings drawn from the distribution $\mathbb{D}_{ij}$, its error $\epsilon_{ij}(h)$ is defined as the expected difference between h and f_{ij}; formally, we have

$$\epsilon_{ij}(h) = E_{x \sim \mathbb{D}_{ij}}[|h(x) - f_{ij}(x)|], \tag{7}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

The objective of the GLID problem is to learn a hypothesis h (i.e., a detector) from the labeled datasets $\bigcup_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}} \mathcal{D}_{ij}$ such that it generalizes effectively to unseen LLMs and domains. Without loss of generality, let $\mathbb{D}_u$ denote the distribution of embeddings corresponding to documents generated by unseen LLMs or written by humans in unseen domains. By definition, $\mathbb{D}_u$ is unknown. Let f_u be the function that gives the true probability that a document with embedding $x \sim \mathbb{D}_u$ is LLM-generated. Similarly, we define the error ϵ_u of applying hypothesis h to document embeddings drawn from the distribution $\mathbb{D}_u$ as the expected difference between h and f_u:

$$\epsilon_u(h) = E_{x \sim \mathbb{D}_u}[|h(x) - f_u(x)|]. \tag{8}$$

Consequently, the objective of the GLID problem becomes learning a hypothesis h from the labeled datasets $\bigcup_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}} \mathcal{D}_{ij}$ such that $\epsilon_u(h)$ is minimized. To achieve this, we derive an upper bound for $\epsilon_u(h)$ in the following theorem.

Theorem 1. *For any $\eta \in (0,1)$, with probability at least $(1-\eta)^2$, the following bound for $\epsilon_u(h)$ holds:*

$$\epsilon_u(h) \leq \sum_{ij} \pi^*_{ij} \epsilon_{ij}(h) + \frac{1}{2} \max_{a,b} \hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b}) + \frac{1}{2} \max_{a,b,c,d} \hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b}) + C, \quad (9)$$

where $\pi^*_{ij} \geq 0$, $\sum_{ij} \pi^*_{ij} = 1$, $\hat{d}_{\mathcal{H}}$ is the empirical $\mathcal{H}$-divergence that measures the distance between two distributions using data drawn from the distributions, domain IDs $a, b \in \{1, 2, \ldots, n\}$, LLM IDs $c, d \in \{1, 2, \ldots, m\}$, and C is a small constant.

Proof. *See Appendix A. Details of π^*_{ij}, $\hat{d}_{\mathcal{H}}$, and C are also given in Appendix A.*

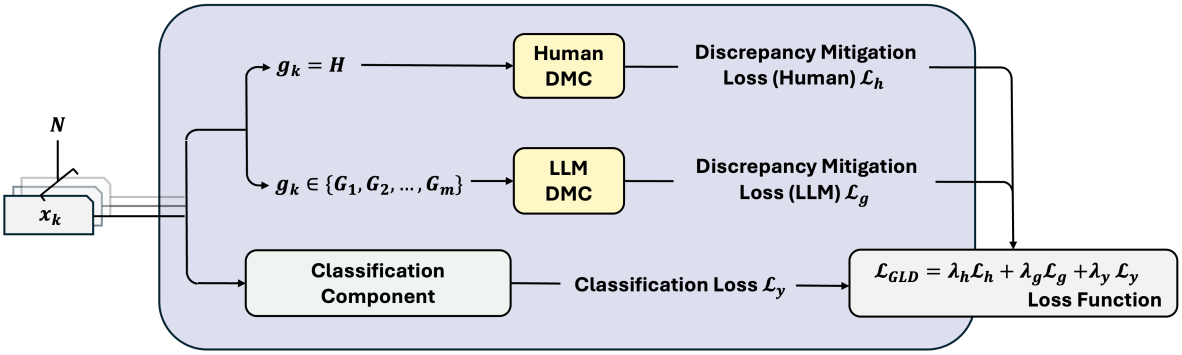
By Theorem 1, minimizing the error $\epsilon_u(h)$ requires minimizing all the terms on the right-hand side of Inequality 9, except for the small constant C. The first term $\sum_{ij} \pi^*_{ij} \epsilon_{ij}(h)$ is a convex combination of the errors $\epsilon_{ij}(h)$, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. According to Equation 7, $\epsilon_{ij}(h)$ denotes the error of applying hypothesis h to document embeddings drawn from the distribution $\mathbb{D}_{ij}$. Therefore, to minimize the first term, we design a classifier that aims to minimize the error of classifying document embeddings in the labeled dataset $\mathcal{D}_{ij}$ drawn from distribution $\mathbb{D}_{ij}$, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

The second term $\max_{a,b} \hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b})$ is the maximal empirical $\mathcal{H}$-divergence between the distributions of human-written document embeddings in two different domains. To minimize this term, we propose a component that mitigates cross-domain discrepancies among human-written document embeddings. As a result, this component facilitates the learning of domain-invariant embeddings for human-written documents, thereby enabling the effective generalization of our method to unseen domains.

The third term $\max_{a,b,c,d} \hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b})$ represents the maximal empirical $\mathcal{H}$-divergence between the distributions of LLM-generated document embeddings in two different LLM-domain pairs. To minimize the third term, we develop a component that reduces both cross-LLM and cross-domain discrepancies among LLM-generated document embeddings, thereby promoting the learning of LLM-invariant and domain-invariant embeddings for LLM-generated documents and allowing our method to generalize effectively to unseen LLMs and domains.

In summary, as illustrated in Figure 3, the DGM consists of three components, each of which is designed to minimize a corresponding term in Inequality 9. In the following subsections, we detail the design of each component.

Figure 3 Architecture of Detection Generalization Module (DGM).



4.2.1 Discrepancy Mitigation Components

The Discrepancy Mitigation Components (DMCs) consist of a Human DMC and an LLM DMC. The former is designed to minimize $\max_{a,b} \hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b})$, thereby reducing cross-domain discrepancies among human-written document embeddings; the latter aims to minimize $\max_{a,b,c,d} \hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b})$, thus mitigating cross-LLM and cross-domain discrepancies in LLM-generated document embeddings. As shown in Figure 3,

document embeddings x_k are routed to either the Human DMC or the LLM DMC according to their author labels g_k, where $k = 1, 2 \ldots, N$. Both the Human DMC and the LLM DMC measure the empirical $\mathcal{H}$-divergence $\hat{d}_{\mathcal{H}}$ using the widely applied Maximum Mean Discrepancy (MMD) metric with a multi-Gaussian kernel (Borgwardt et al. 2006, Long et al. 2015). Concretely, the MMD between two datasets $\mathcal{D}'$ and $\mathcal{D}''$ is given by:

$$\text{MMD}(\mathcal{D}', \mathcal{D}'') = \frac{1}{N'^2} \sum_{i=1}^{N'} \sum_{j=1}^{N'} \kappa(x'_i, x'_j) + \frac{1}{N''^2} \sum_{i=1}^{N''} \sum_{j=1}^{N''} \kappa(x''_i, x''_j) - \frac{2}{N'N''} \sum_{i=1}^{N'} \sum_{j=1}^{N''} \kappa(x'_i, x''_j), \quad (10)$$

where N' and N'' denote the number of instances in $\mathcal{D}'$ and $\mathcal{D}''$, respectively, and x' and x'' represent an instance in $\mathcal{D}'$ and $\mathcal{D}''$, respectively. The kernel function $\kappa(x, y)$ between two instances x and y is defined as,

$$\kappa(x, y) = \frac{1}{|R|} \sum_{r \in R} \exp\left(\frac{-||x - y||^2}{r}\right), \quad (11)$$

where $||\cdot||$ denotes the ℓ_2-norm, r is a bandwidth chosen from a bandwidth set R with predefined hyperparameters r_1, r_2, and a multiplicative step-size of 2, $R = \{2^{r_1}, 2^{r_1+1}, \ldots, 2^{r_2}\}$, and $|R|$ is the size of the set R.

With MMD defined, minimizing $\max_{a,b} \hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b})$ and $\max_{a,b,c,d} \hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b})$ becomes minimizing $\max_{a,b} \text{MMD}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b})$ and $\max_{a,b,c,d} \text{MMD}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b})$, respectively. Therefore, we define the following two losses:

$$\mathcal{L}_h = \max_{a,b} \text{MMD}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b}), \quad (12)$$

$$\mathcal{L}_g = \max_{a,b,c,d} \text{MMD}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b}). \quad (13)$$

By minimizing the losses $\mathcal{L}_h$ and $\mathcal{L}_g$, our method learns domain-invariant embeddings for human-written documents and LLM- and domain-invariant embeddings for LLM-generated documents.

4.2.2 Classification Component and Summary

The classification component takes the embedding x_k of each training document as input and outputs the probability $p(\hat{y_k} = 1|x_k)$ that the document is LLM-generated, where $k = 1, 2 \ldots, N$. Specifically, each document embedding is transformed by a multi-layer perceptron (denoted as MLP_3), followed by a sigmoid function. Formally, we have,

$$p(\hat{y_k} = 1|x_k) = \text{Sigmoid}(\text{MLP}_3(x_k)), \tag{14}$$

where $k = 1, 2 \ldots, N$. By Theorem 1, the objective of the classification component is to minimize the error in classifying document embeddings. Accordingly, it minimizes the following cross-entropy loss:

$$\mathcal{L}_y = -\sum_{k=1}^{N} \left[y_k \log p(\hat{y_k} = 1|x_k) + (1 - y_k) \log \left(1 - p(\hat{y_k} = 1|x_k)\right) \right], \tag{15}$$

where true document label y_k is 1 if it is LLM-generated or 0 if it is human-written and $k = 1, 2 \ldots, N$.

In summary, we propose the DGM module, which learns a detector capable of generalizing to unseen LLMs and domains. The design of the module is guided by a novel theoretical analysis that establishes an upper bound on the generalization error incurred when extending a detector to unseen LLMs and domains. Specifically, the DGM module consists of three components, each of which is designed to minimize a key term in the upper bound of the generalization error. Together, these components enable our method to effectively generalize to unseen LLMs and domains.

4.3 Training and Detecting

The loss for the proposed GLD method is the combination of the losses $\mathcal{L}_h$, $\mathcal{L}_g$, and $\mathcal{L}_y$:

$$\mathcal{L}_{GLD} = \lambda_h \mathcal{L}_h + \lambda_g \mathcal{L}_g + \lambda_y \mathcal{L}_y, \tag{16}$$

where $\mathcal{L}_h$, $\mathcal{L}_g$, and $\mathcal{L}_y$ are defined in Equations 12, 13, and 15, respectively. The non-negative hyperparameters λ_h, λ_g, and λ_y determine the weights of their corresponding losses. As illustrated in Figure 1, the GLD method is trained by minimizing its loss $\mathcal{L}_{GLD}$ over the labeled training documents. More specifically, it is trained to learn document embeddings that are both LLM- and domain-invariant by minimizing its component losses $\mathcal{L}_h$ and $\mathcal{L}_g$, and to classify these embeddings effectively by minimizing its component loss λ_y. As a result, the classification capability of our method is not restricted to documents generated by specific LLMs in particular domains. Instead, it can effectively detect information produced by unseen LLMs in new domains.

Once trained, the GLD method detects whether a test document is LLM-generated or human-written using its two modules, as depicted in Figure 1. Specifically, the TMN module produces the embedding of this test document. Since GLD has already been trained, memory units in the TMN module are not updated during the testing phase. Next, the classification component in the DGM module classifies the embedding as LLM-generated or human-written. For the same reason as before, the Human DMC and the LLM DMC components in this module are not invoked during the testing phase.

5 EMPIRICAL EVALUATION

5.1 Data

We collected human-written documents across five domains commonly found on digital platforms: news, review, story, general knowledge, and question-and-answer (QA). Specifically, we employed publicly available datasets of human-written documents: the TLDR News dataset[10] for news, the Yelp dataset (Zhang et al. 2015) for review, the Reddit's ROCStories dataset (Mostafazadeh et al. 2016) for story, the dataset of Wikipedia

[10] `https://huggingface.co/datasets/JulesBelveze/tldr_news` (last accessed on June 15, 2025)

articles (Rajpurkar et al. 2016) for general knowledge, and the ELI5 dataset (Fan et al. 2019) for QA. For each domain, we randomly sampled 1,500 human-written documents from its corresponding dataset. To produce LLM-generated documents, we utilized five widely adopted commercial or open-source LLMs: GPT-4o[11] from OpenAI, Command-R[12] from Cohere, LLaMA-3.1-70B (Dubey et al. 2024) and OPT-66B (Zhang et al. 2022) from Meta, and GPT-NeoX-20B (Black et al. 2022) from EleutherAI. For each LLM, we randomly selected 500 human-written documents from each domain. Following the practice of previous studies (Mitchell et al. 2023, Bao et al. 2023), we then prompted the LLM to produce a LLM-generated document based on the initial 30 words of each selected document.[13] This procedure was repeated for each of the five LLMs across all five domains, producing LLM-generated documents for our study. As summarized in Table 2, for each of the five domains, our dataset consists of 1,500 human-written documents and 500 LLM-generated documents per LLM. In total, it contains 20,000 documents.

Table 2 Number of Human-written or LLM-generated Documents in Each Domain.

	News	Review	Story	Knowledge	QA	Total
Human	1,500	1,500	1,500	1,500	1,500	7,500
GPT-4o	500	500	500	500	500	2,500
Command-R	500	500	500	500	500	2,500
LLaMA-3.1	500	500	500	500	500	2,500
OPT	500	500	500	500	500	2,500
GPT-NeoX	500	500	500	500	500	2,500
Total	4,000	4,000	4,000	4,000	4,000	20,000

[11] `https://openai.com/index/hello-gpt-4o` (last accessed on June 15, 2025)

[12] `https://cohere.com/blog/command-series-0824` (last accessed on June 15, 2025)

[13] We also prompted a LLM to generate text by truncating human-written text to its initial 20 and 40 words and evaluated the results. The outcomes of truncating to 20, 30, and 40 words were similar and showed no statistically significant differences. Therefore, we present only the results for 30 words here.

5.2 Benchmark Methods and Evaluation Procedure

As reviewed in Section 2, current methods for detecting LLM-generated information can be classified into three categories: zero-shot, feature-based, and fine-tuning methods. Therefore, we benchmarked our method against state-of-the-art methods in these three categories. For zero-shot methods, we included DetectGPT (Mitchell et al. 2023), one of the most widely used techniques based on changes in perplexity before and after perturbing a document to be detected. We also considered Fast-DetectGPT (Bao et al. 2023), an improvement to DetectGPT, which enhances the original perturbation process through sampling. Additionally, we implemented DetectLLM (Su et al. 2023), which differentiates human-written documents from LLM-generated ones based on the average ratio of a token's log-likelihood to its log-rank in a document. We chose two representative feature-based methods as benchmarks: GLTR (Gehrmann et al. 2019) and the multi-feature method (Fröhling and Zubiaga 2021). The former trains a classifier to classify a document as human-written or LLM-generated based on probability-based features extracted from the document, while the latter classifies a document based on its linguistic features. For fine-tuning methods, a commonly used detector is OpenAI's GPT-2 output detector (hereafter referred to as the OpenAI detector) (Solaiman et al. 2019), which identifies GPT-generated information by fine-tuning the RoBERTa model. We also implemented two existing fine-tuning methods as benchmarks: EAGLE (Bhattacharjee et al. 2024) and DATeD (Guo et al. 2024b). The former applies adversarial training (Ganin et al. 2016) to enhance the generalizability of a detector to unseen LLMs, while the latter employs adversarial training to enable a detector to generalize to unseen domains. Table 3 summarizes all the methods compared in our evaluation. Implementation details of each method can be found in Appendix B.

Table 3 Detection Methods Compared in Our Evaluation.

Method	Notes
GLD	Our method
DetectGPT	Benchmark zero-shot method (Mitchell et al. 2023)
Fast-DetectGPT	Benchmark zero-shot method (Bao et al. 2023)
DetectLLM	Benchmark zero-shot method (Su et al. 2023)
GLTR	Benchmark feature-based method (Gehrmann et al. 2019)
Multi-feature	Benchmark feature-based method (Fröhling and Zubiaga 2021)
OpenAI Detector	Benchmark fine-tuning method (Solaiman et al. 2019)
EAGLE	Benchmark fine-tuning method (Bhattacharjee et al. 2024)
DATeD	Benchmark fine-tuning method (Guo et al. 2024b)

We followed the leave-one-group-out (LOGO) evaluation procedure[14] to assess the performance of a detection method on unseen LLMs and domains, in line with the literature (Bhattacharjee et al. 2024, Guo et al. 2024b). A group in our study consists of human-written documents in a specific domain and documents generated by a specific LLM in the same domain. An example of this evaluation setup is provided in Table 4. As shown, to evaluate the performance of a method in detecting information generated by an unseen LLM (i.e., GPT-4o) in an unseen domain (i.e., news), we constructed a training dataset consisting of human-written documents from four other domains (i.e., review, story, knowledge, and QA) and LLM-generated documents produced by four other LLMs (i.e., Command-R, LLaMA-3.1, OPT, and GPT-NeoX) across the four domains. The test dataset contained human-written and GPT-4o-generated documents in the news domain. During testing, no information about the tested LLM or domain was provided. This evaluation setup ensures that learning-based methods (including feature-based and fine-tuning methods) are not exposed to data from the tested LLM or domain, and zero-shot

[14] See `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneGroupOut.html` (last accessed on June 15, 2025)

methods have no prior information of the tested LLM or domain. As a result, we can rigorously evaluate the generalizability of each method to unseen LLMs and domains.

Table 4 An Example of Leave-one-group-out (LOGO) Evaluation for Unseen LLM of GPT-4o and Unseen Domain of News.

	Review	Story	Knowledge	QA	News
Human	Train (1500)	Train (1500)	Train (1500)	Train (1500)	Test (1500)
Command-R	Train (500)	Train (500)	Train (500)	Train (500)	-
LLaMA-3.1	Train (500)	Train (500)	Train (500)	Train (500)	-
OPT	Train (500)	Train (500)	Train (500)	Train (500)	-
GPT-NeoX	Train (500)	Train (500)	Train (500)	Train (500)	-
GPT-4o	-	-	-	-	Test (500)

To evaluate the performance of each method in Table 3, we conducted the LOGO evaluation for all 25 unseen LLM-domain pairs (five LLMs $\times$ five domains). In each evaluation, we calculated the Area Under the Curve (AUC) and F1 score. Following Mitchell et al. (2023), we reported the AUC to comprehensively evaluate a method's ability to distinguish LLM-generated information from human-written information. Concretely, the AUC is computed by integrating the true positive rate (TPR) with respect to the false positive rate (FPR) over the range $[0,1]$. We also calculated the F1 score to provide a focused measure of a method's precision and recall in detecting LLM-generated information. In general, let P denote the number of positive instances, TP be the number of true positives, and FP be the number of false positives. The F1 score is computed as $\text{F1} = \frac{2\times\text{TP}}{\text{TP}+\text{FP}+\text{P}}$.

5.3 Evaluation Results

Following the procedure stated in Section 5.2, we evaluated the performance of each method in Table 3. Table 5 reports the average AUC and F1 score for each method across all 25 unseen LLM-domain pairs, with standard deviations provided in parentheses. The evaluation results highlight several findings. First, our proposed GLD method surpasses

all benchmarks in both AUC and F1 score, and the performance improvements by our method are statistically significant. In particular, our method increases AUC by 21.48%, 15.18%, and 5.71% and lifts F1 score by 25.72%, 20.41%, and 7.13% over the best performing zero-shot, feature-based, and fine-tuning benchmarks, respectively. Second, our method achieves the lowest standard deviations in both AUC and F1 score, among all the compared methods. This indicates that our method demonstrates greater stability across different combinations of unseen LLM and domain, exhibiting less performance bias toward specific combinations. Third, among the benchmarks, fine-tuning methods outperform both feature-based and zero-shot methods, which supports our choice of a fine-tuning approach for developing our method. In summary, the evaluation results demonstrate the greater effectiveness and stability of our method in distinguishing human-written information from content generated by unseen LLMs in new domains, compared to state-of-the-art benchmark methods. The superior performance of our method can be attributed to its key innovations: the twin memory networks for representing the content, author, and domain of each piece of information, and the detection generalization module for learning LLM- and domain-invariant embeddings.

To provide a deeper insight into the performance of our proposed GLD method, we present its performance on each unseen LLM, alongside that of the best performing benchmark in each category: DetectLLM for zero-shot methods, the multi-feature method for feature-based methods, and EAGLE for fine-tuning methods. As reported in Figure 4(a), our method outperforms all benchmarks in AUC on each unseen LLM. Furthermore, our method exhibits greater stability across different unseen LLMs. Taking DetectLLM as an example, its average AUC of detecting information generated by an unseen LLM ranges from 0.586 (when detecting information generated by OPT) to 0.858

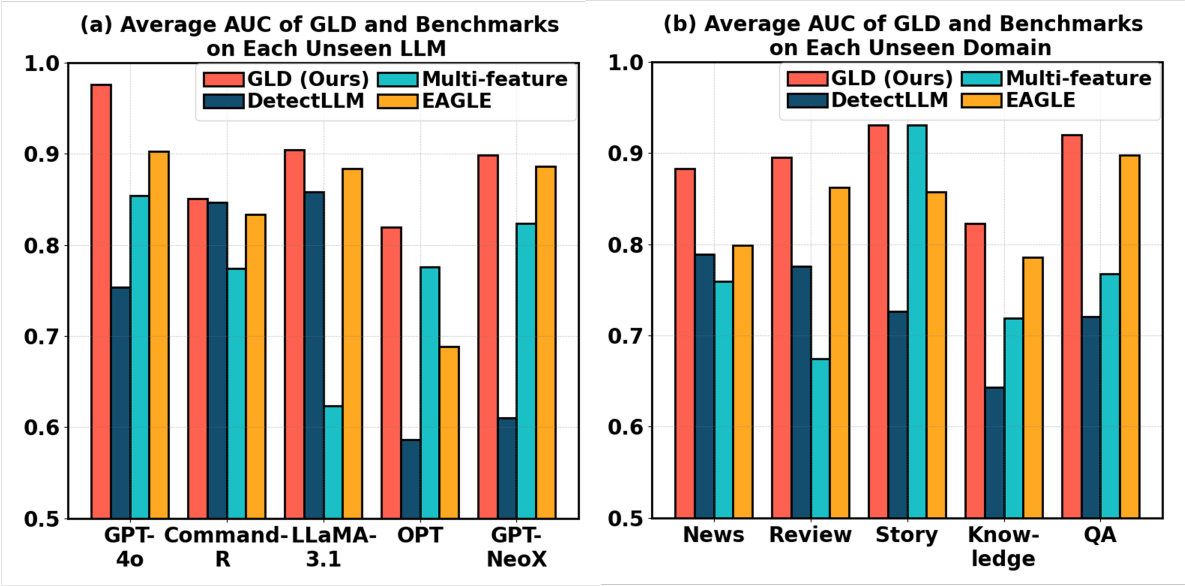
Table 5 Performance Comparison between GLD and Benchmark Methods.

Method	AUC	Improvement by GLD	F1 score	Improvement by GLD
GLD	0.888 (0.085)		0.826 (0.098)	
DetectGPT	0.561** (0.115)	58.29%	0.438** (0.132)	88.58%
Fast-DetectGPT	0.632** (0.363)	40.51%	0.572** (0.368)	44.41%
DetectLLM	0.731** (0.133)	21.48%	0.657** (0.157)	25.72%
GLTR	0.628** (0.197)	41.40%	0.543** (0.201)	52.12%
Multi-feature	0.771** (0.138)	15.18%	0.686** (0.162)	20.41%
OpenAI Detector	0.834** (0.200)	6.47%	0.757** (0.235)	9.11%
EAGLE	0.840** (0.147)	5.71%	0.771** (0.169)	7.13%
DATeD	0.821** (0.167)	8.16%	0.740** (0.256)	11.62%

Note: The notation ** indicates that the performance difference between GLD and a benchmark is statistically significant at the 5% level. Standard deviations are provided in parentheses.

(when detecting information generated by LLaMA-3.1). In contrast, our method consistently achieves an AUC greater than 0.8 across all unseen LLMs. Figure 4(b) presents the performance of our method and the top performing benchmarks on each unseen domain. As depicted, our method surpasses all benchmarks in AUC on each unseen domain, while demonstrating greater stability across these domains. For example, although the performance of the multi-feature method is close to that of our method in the story domain, the benchmark's performance drops significantly in other domains, falling well behind our method in these domains. In summary, our additional analysis further confirms the effectiveness and stability of our method in detecting information generated by various unseen LLMs across different unseen domains. As a result, our method is particularly use-

Figure 4 Average AUC of GLD and Best Performing Benchmark Methods on Each Unseen LLM or Domain.



Note. Better view in color.

ful for identifying LLM-generated information, given that LLMs are increasingly being applied to a growing range of domains and new LLMs continue to emerge rapidly.

5.4 Performance Analysis

We conducted ablation studies to assess the contribution of each key design artifact of GLD to its performance. In general, we dropped one artifact from GLD at a time. The performance difference between the resulting method without the dropped artifact and GLD revealed the performance contribution of the artifact. As illustrated in Figure 1, the GLD method consists of two key modules: TMN and DGM. To evaluate the performance contribution of each module, we conducted two sets of ablation studies. In the first set of studies, we removed the TMN module entirely from GLD, referring to the resulting variant as GLD w/o TMN. Furthermore, to evaluate the contribution of each memory network in TMN, we removed either the author memory network or the domain memory network individually. The resulting variants were referred to as GLD w/o Author MN and GLD w/o Domain MN, respectively. The core innovations of the DGM module lie

in its two Discrepancy Mitigation Components (DMCs). Accordingly, in the second set of studies, we removed both DMCs together or each one individually. The resulting variants were denoted as GLD w/o DMC, GLD w/o Human DMC, and GLD w/o LLM DMC, respectively.

Table 6 Ablation Study of GLD.

Method	AUC	Improvement by GLD	F1-score	Improvement by GLD
GLD	0.888 (0.085)		0.826 (0.098)	
GLD w/o TMN	0.861** (0.100)	3.14%	0.799** (0.109)	3.38%
GLD w/o Author MN	0.870** (0.107)	2.07%	0.809** (0.132)	2.10%
GLD w/o Domain MN	0.869** (0.111)	2.19%	0.807** (0.135)	2.35%
GLD w/o DMC	0.858** (0.113)	3.38%	0.796** (0.142)	3.64%
GLD w/o Human DMC	0.863** (0.121)	2.90%	0.802** (0.148)	2.99%
GLD w/o LLM DMC	0.867** (0.105)	2.42%	0.805** (0.128)	2.23%

Note: The notation ** indicates that the performance difference between GLD and a benchmark is statistically significant at the 5% level. Standard deviations are provided in parentheses.

Using the evaluation procedures described in Section 5.2, we compared the performance of GLD with its ablated variants. Table 6 reports the average performance of these methods across 25 unseen LLM-domain pairs. As reported, GLD significantly outperforms all ablated variants, demonstrating the contribution of each novel design artifact to its performance. For example, GLD surpasses GLD w/o TMN by 3.14% in AUC and 3.38% in F1-score, revealing the contribution of the TMN module to its performance. Diving into different components in TMN, removing the author MN or domain MN sepa-

rately yields better performance than removing the entire TMN, highlighting the importance of constructing memory networks for both author and domain. Similarly, removing the human DMC or LLM DMC separately also performs better than removing both, corroborating the effectiveness of each component.

5.5 Case Studies

We conducted two case studies to evaluate the effectiveness of our proposed GLD method in real-world scenarios. In the first case study, we focused on detecting LLM-generated reviews, which is critical for e-commerce platforms and review websites such as Amazon and Yelp to enhance product evaluation and assist consumers in making informed decisions (Lappas et al. 2016). It was recently shown that an increasing number of fake reviews have been generated by LLMs.[15] A major challenge in detecting these generated reviews is the rapid emergence of new LLMs, which makes it infeasible to collect sample reviews created by all such models as training data (Gupta et al. 2024). To address this, we evaluated our method's ability to detect reviews generated by Gemini 2.0 Pro, an LLM recently developed by Google.[16] Specifically, we compared the performance of our method against that of the top performing benchmarks in each category: DetectLLM for zero-shot methods, the multi-feature method for feature-based methods, and EAGLE for fine-tuning methods. Each method was trained using the complete dataset in Table 2. To construct the test dataset to evaluate the performance of each method, we collected 1,000 human-written Yelp reviews distinct from those in the training dataset and used Gemini 2.0 Pro to generate corresponding fake reviews by continuing from the first 30 words[17] of

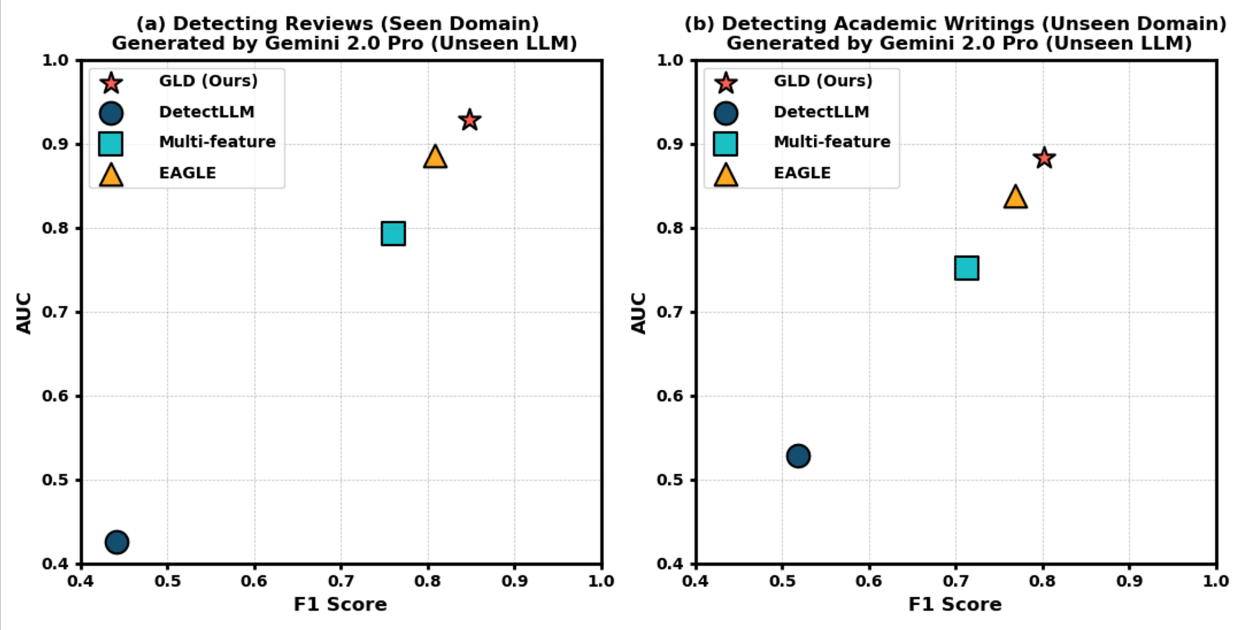
[15] See `https://originality.ai/blog/g2-reviews-ai-generated-case-study` (last accessed on June 15, 2025)

[16] See `https://blog.google/technology/google-deepmind/gemini-model-updates-february-2025` (last accessed on June 15, 2025)

[17] Same as in the main experiment, we evaluate the generated text by truncating the human-written text to 20 and 40 words. Since the results have no statistical difference, we report only the 30-word results here.

each original review. This setup allowed us to evaluate each method's ability to detect reviews generated by an unseen LLM (i.e., Gemini 2.0 Pro) in a seen domain (i.e., review). The evaluation results presented in Figure 5(a) show that our method outperforms all top performing benchmarks in both AUC and F1 score. The performance advantage of our method can be attributed to its methodological novelties, which enable it to effectively generalize to unseen LLMs, such as Gemini 2.0 Pro in this case. As a result, our method is particularly useful for detecting LLM-generated reviews and supporting customers on e-commerce platforms, especially in light of the rapid emergence of new LLMs and the growing trend of using LLMs to generate reviews.

Figure 5 Case Studies on Detecting Reviews and Academic Writings Generated by Gemini 2.0 Pro.



We conducted a second case study to evaluate the performance of our method and the top performing benchmarks in detecting academic writings generated by Gemini 2.0 Pro. LLMs have been considered a double-edge sword in academic work due to their ability to assist it, such as improving grammar, while also raising concerns about academic

integrity when misused, such as generating an entire section or even a full paper without disclosure. Previous research has shown that even human scientists have difficulties distinguishing LLM-generated abstracts from human-written ones (Else 2023). Therefore, it is essential to develop a method that can accurately detect LLM-generated academic writings. Specifically, we focused on the medical domain. Using the widely recognized PubMed dataset (Jin et al. 2019), we collected 1,000 titles and abstracts, along with their conclusions, from research papers authored by human experts in the medical field. We then prompted Gemini 2.0 Pro with each pair of collected title and abstract to generate the corresponding conclusion, in line with the common practice in the literature (Jin et al. 2019, Gilson et al. 2023). Our test dataset consists of 1,000 human-written conclusions and 1,000 conclusions generated by Gemini 2.0 Pro. Each method was trained using the complete dataset in Table 2 and evaluated using the test dataset. As a result, both the LLM (i.e., Gemini 2.0 Pro) and the domain (i.e., academic writing) are unseen by any of the methods. As reported in Figure 5(b), our method outperforms all benchmarks in both AUC and F1 score. With the continued rollout of new LLMs and their applications to an increasing number of domains, our method is particularly valuable for its ability to generalize effectively to unseen LLMs and domains.

6 DISCUSSION

6.1 Summary and Contributions

Detecting LLM-generated information is essential for preserving trust on digital platforms and preventing the spread of misinformation. Existing detection methods focus on identifying content generated by specific LLMs in known domains but face significant challenges in generalizing to a large number of unseen LLMs and a wide range of

new domains. To address these challenges, we propose the GLD method, which is capable of detecting information produced by unseen LLMs in new domains. Its capabilities are realized through the twin memory networks design and the theory-guided detection generalization module. Extensive empirical evaluations with real-world datasets demonstrate the superiority of our method over state-of-the-art detection methods in identifying information generated by unseen LLMs in new domains.

Our research contributes to the extant IS literature in two ways. First, our work falls into the computational genre of design science research (Rai et al. 2017, Padmanabhan et al. 2022), which develops novel computational methods to solve important business and societal problems and makes methodological contributions to the literature (Samtani et al. 2022, Zhao et al. 2023, Kim et al. 2023). Driven by the need to detect information generated by unseen LLMs in unseen domains, we propose a detection method that features two methodological contributions: the novel twin memory networks design and the innovative detection generalization module. Second, an active stream of IS research focuses on the detection and management of fake and misleading content on digital platforms (Abbasi et al. 2010, Moravec et al. 2019, Wei et al. 2022). As LLMs are increasingly used to generate such content due to their low costs and high capabilities, our study contributes to this IS research stream by proposing a novel method that can effectively detect information generated by unseen LLMs in new domains. Moreover, our work also adds to the growing body of design science research on LLMs in the IS discipline (e.g., Wang and Li 2024, Zhang et al. 2025) and advances our knowledge of LLM-based design.

6.2 Practical Implications

Our study offers practical implications for digital platforms. The rapid proliferation of LLMs has made it easy for individuals to generate and spread large amounts of fake,

misleading, or useless content on digital platforms, undermining the integrity of these platforms (He et al. 2022, Jakesch et al. 2023). Therefore, detecting LLM-generated content is critical for protecting the integrity of digital platforms. Our study provides a detection method that is trained on a dataset of human-written content and information generated by a small number of LLMs in several domains, but can be generalized to detect information produced by unseen LLMs in new domains. Given the vast and growing number of available LLMs and the wide range of domains on digital platforms, our method is especially effective for detecting and managing LLM-generated content on these platforms. Specifically, digital platform operators can apply our GLD method to detect LLM-generated content and take appropriate actions, such as flagging or removing it. For example, once our method identifies LLM-generated content, community notes could be added to alert users and caution them against potential misinformation. Moreover, empowered by our method, digital platform operators can monitor accounts that frequently disseminate large volumes of LLM-generated content, allowing them to regulate or even block such accounts (Lee et al. 2024). In summary, our method enables digital platform operators to effectively identify and manage LLM-generated content on the platform, thereby preserving user trust in the platform and safeguarding its business value amid the growing prevalence of LLMs.

Our study also supports the enforcement of government regulations on LLMs, particularly those aimed at ensuring their accountability (Stokel-Walker and Van Noorden 2023). Notable among these regulations is the European Union's Artificial Intelligence Act, which mandates that content generated by LLMs must be clearly labeled. Another key regulation is Executive Order 14110, issued by the U.S. government, which directs federal agencies to ensure the responsible deployment of AI systems, including LLMs.

Understandably, identifying LLM-generated content is essential for ensuring accountability. To this end, our study offers an efficient and effective tool for detecting and labeling LLM-generated content. Moreover, government agencies could leverage our GLD method to trace and regulate individuals who frequently publish LLM-generated content without clear labeling. In particular, malicious authors who regularly publish misleading LLM-generated content could be exposed and banned from publishing, thus reducing the harm caused by the misuse of LLMs (Pennycook and Rand 2021).

6.3 Future Work

Our study can be extended in several directions. First, our method focuses on detecting text generated by LLMs. Future research could expand this approach to multimodal settings, such as detecting AI-generated content that combines text with images or audio. This would require generalizing the GLD method to encompass AI systems beyond LLMs. Second, our empirical evaluations are based on documents generated by five representative LLMs in five important domains. Future work could further assess the performance of GLD in detecting text generated by other prominent LLMs (e.g., Claude and DeepSeek) in additional domains (e.g., science and finance). Third, existing literature has explored injecting watermarks in LLMs (Kirchenbauer et al. 2023). These watermarking methods typically require a white-box LLM in order to modify its text generation. Although our method cannot be directly applied or guide the design of watermarking methods, future work can investigate whether our GLD method is capable of detecting texts generated by watermarked LLMs. Fourth, our study frames the problem as a binary classification task, distinguishing LLM-generated content from human-written text. As the adoption of LLMs becomes more widespread, humans are increasingly using these models to assist with tasks such as proofreading or enhancing their writing. This differs

from malicious applications, such as generating fake or misleading content. Future studies should investigate how to detect these diverse scenarios of LLM usage and identify cases where harmful intent is present.

References

Abbasi, A., Zhang, Z., Zimbra, D., Chen, H., and Nunamaker Jr, J. F. (2010). Detecting fake websites: The contribution of statistical learning theory. *MIS Quarterly*, pages 435–461.

Bao, G., Zhao, Y., Teng, Z., Yang, L., and Zhang, Y. (2023). Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Berente, N., Gu, B., Recker, J., and Santhanam, R. (2021). Managing artificial intelligence. *MIS Quarterly*, 45(3).

Bhattacharjee, A., Moraffah, R., Garland, J., and Liu, H. (2024). Eagle: A domain generalization framework for ai-generated text detection. *arXiv preprint arXiv:2403.15690*.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. (2022). Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136.

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57.

Church, K. W. (2017). Word2vec. *Natural Language Engineering*, 23(1):155–162.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Else, H. (2023). Abstracts written by chatgpt fool scientists. *Nature*, 613:423.

Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., and Auli, M. (2019). Eli5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567.

Fang, X., Che, S., Mao, M., Zhang, H., Zhao, M., and Zhao, X. (2024). Bias of ai-generated content: an examination of news produced by large language models. *Scientific Reports*, 14(1):1–20.

Fröhling, L. and Zubiaga, A. (2021). Feature-based detection of automated language models: tackling gpt-2, gpt-3 and grover. *PeerJ Computer Science*, 7:e443.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Gehrmann, S., Strobelt, H., and Rush, A. M. (2019). Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

Gilson, A., Safranek, C. W., Huang, T., Socrates, V., Chi, L., Taylor, R. A., Chartash, D., et al. (2023). How does chatgpt perform on the united states medical licensing examination (usmle)? the implications of large language models for medical education and knowledge assessment. *JMIR Medical Education*, 9(1):e45312.

Goli, A. and Singh, A. (2024). Frontiers: Can large language models capture human preferences? *Marketing Science*, 43(4):709–722.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Guo, B., Zhang, X., Wang, Z., Jiang, M., Nie, J., Ding, Y., Yue, J., and Wu, Y. (2023). How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.

Guo, Y., Shang, G., and Clavel, C. (2024a). Benchmarking linguistic diversity of large language models. *arXiv preprint arXiv:2412.10271*.

Guo, Z., Jiao, K., Yao, X., Wan, Y., Li, H., Xu, B., Zhang, L., Wang, Q., Zhang, Y., and Mao, Z. (2024b). Ustc-bupt at semeval-2024 task 8: Enhancing machine-generated text detection via domain adversarial neural networks and llm embeddings. In *Proceedings of the 18th International Workshop on Semantic Evaluation*, pages 1511–1522.

Gupta, R., Jindal, V., and Kashyap, I. (2024). Recent state-of-the-art of fake review detection: a comprehensive review. *The Knowledge Engineering Review*, 39:e8.

He, S., Hollenbeck, B., and Proserpio, D. (2022). The market for fake reviews. *Marketing Science*, 41(5):896–921.

Jakesch, M., Hancock, J. T., and Naaman, M. (2023). Human heuristics for ai-generated language are flawed. *Proceedings of the National Academy of Sciences*, 120(11):e2208839120.

Jin, Q., Dhingra, B., Liu, Z., Cohen, W., and Lu, X. (2019). Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2567–2577.

Kim, B. R., Srinivasan, K., Kong, S. H., Kim, J. H., Shin, C. S., and Ram, S. (2023). Rolex: A novel method for interpretable machine learning using robust local explanations. *MIS Quarterly*, 47(3).

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. (2023). A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Lappas, T., Sabnis, G., and Valkanas, G. (2016). The impact of fake reviews on online visibility: A vulnerability assessment of the hotel industry. *Information Systems Research*, 27(4):940–961.

Lee, S., Shin, D., Kwon, K. H., Han, S. P., and Lee, S. K. (2024). Disinformation spillover: Uncovering the ripple effect of bot-assisted fake social engagement on public attention. *MIS Quarterly*, 48(3).

Li, Y., Li, Q., Cui, L., Bi, W., Wang, Z., Wang, L., Yang, L., Shi, S., and Zhang, Y. (2023). Mage: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 36–53.

Liu, A., Pan, L., Lu, Y., Li, J., Hu, X., Zhang, X., Wen, L., King, I., Xiong, H., and Yu, P. (2024). A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36.

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

Liu, Y., Zhang, Z., Zhang, W., Yue, S., Zhao, X., Cheng, X., Zhang, Y., and Hu, H. (2023). Argugpt: evaluating, understanding and identifying argumentative essays generated by gpt models. *arXiv preprint arXiv:2304.07666*.

Liu, Z., Xu, Y., Yu, T., Dai, W., Ji, Z., Cahyawijaya, S., Madotto, A., and Fung, P. (2021). Crossner: Evaluating cross-domain named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13452–13460.

Long, M., Cao, Y., Wang, J., and Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 97–105. PMLR.

Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. (2023). Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Moravec, P. L., Minas, R. K., and Dennis, A. R. (2019). Fake news on social media: People believe what they want to believe when it makes no sense at all. *MIS Quarterly*, 43(4).

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., and Allen, J. (2016). A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Mousavi, R., Raghu, T., and Frey, K. (2020). Harnessing artificial intelligence to improve the quality of answers in online question-answering health forums. *Journal of Management Information Systems*, 37(4):1073–1098.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Padmanabhan, B., Fang, X., Sahoo, N., and Burton-Jones, A. (2022). Machine learning in information systems research. *MIS Quarterly*, 46(1).

Pennycook, G. and Rand, D. G. (2021). The psychology of fake news. *Trends in Cognitive Sciences*, 25(5):388–402.

Rai, A., Burton-Jones, A., Chen, H., Gupta, A., Hevner, A. R., Ketter, W., Parsons, J., Rao, H. R., Sarkar, S., and Yoo, Y. (2017). Editor's comments: Diversity of design science research. *MIS Quarterly*, 41(1):iii–xviii.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Reisenbichler, M., Reutterer, T., Schweidel, D. A., and Dan, D. (2022). Frontiers: Supporting content marketing with natural language generation. *Marketing Science*, 41(3):441–452.

Samtani, S., Chai, Y., and Chen, H. (2022). Linking exploits from the dark web to known vulnerabilities for proactive cyber threat intelligence: An attention-based deep structured semantic model. *MIS Quarterly*, 46(2).

Sanh, V. (2019). Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., et al. (2019). Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Stokel-Walker, C. and Van Noorden, R. (2023). What chatgpt and generative ai mean for science. *Nature*, 614(7947):214–216.

Su, J., Zhuo, T. Y., Wang, D., and Nakov, P. (2023). Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12395–12412.

Wang, W. and Li, B. (2024). Learning personalized privacy preference from public data. *Information Systems Research*.

Wei, X., Zhang, Z., Zhang, M., Chen, W., and Zeng, D. D. (2022). Combining crowd and machine intelligence to detect false news on social media. *MIS Quarterly*, 46(2):977–1008.

Wu, J., Yang, S., Zhan, R., Yuan, Y., Wong, D. F., and Chao, L. S. (2025). A survey on llm-gernerated text detection: Necessity, methods, and future directions. *Computational Linguistics*, pages 1–66.

Zhang, D., Zhou, L., Tao, J., Zhu, T., and Gao, G. (2025). Ketch: a knowledge-enhanced transformer-based approach to suicidal ideation detection from social media content. *Information Systems Research*, 36(1):572–599.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 28.

Zhao, X., Fang, X., He, J., and Huang, L. (2023). Exploiting expert knowledge for assigning firms to industries: A novel deep learning method. *MIS Quarterly*, 47(3):1147–1176.

Zhou, J., Pang, Z., and Wang, Y.-X. (2024). Rmem: Restricted memory banks improve video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18602–18611.

APPENDIX A PROOFS OF THEOREM 1

We first introduce the definition of $\mathcal{H}$-divergence (Ben-David et al. 2010), a widely used metric for measuring the distance between two distributions, as well as the definition of its empirical estimation.

Definition A.1. ($\mathcal{H}$-divergence). *Let $\mathcal{H}$ denote a hypothesis space, and let $\mathbb{D}'$ and $\mathbb{D}''$ be two distributions. The identity set of a hypothesis $h \in \mathcal{H}$ is $I(h)$; that is, $x \in I(h) \iff h(x) = 1$. The $\mathcal{H}$-divergence between $\mathbb{D}'$ and $\mathbb{D}''$ is defined as (Ben-David et al. 2010)*

$$d_{\mathcal{H}}(\mathbb{D}', \mathbb{D}'') = 2 \sup_{h \in \mathcal{H}} |P[I(h)|x \sim \mathbb{D}'] - P[I(h)|x \sim \mathbb{D}'']|. \tag{A1}$$

Definition A.2. (Empirical $\mathcal{H}$-divergence). *Let $\mathcal{D}'$ and $\mathcal{D}''$ be two datasets drawn from distributions $\mathbb{D}'$ and $\mathbb{D}''$, respectively. The $\mathcal{H}$-divergence between the two distributions can be empirically estimated using the two datasets, and the estimation is given by (Ben-David et al. 2010)*

$$\hat{d}_{\mathcal{H}}\left(\mathcal{D}', \mathcal{D}''\right) = 2\left(1 - \min_{h \in \mathcal{H}} \left[\frac{1}{N} \sum_{x:h(x)=0} I[x \in \mathcal{D}'] + \frac{1}{N} \sum_{x:h(x)=1} I\left[x \in \mathcal{D}''\right]\right]\right), \tag{A2}$$

where the hypothesis space $\mathcal{H}$ is symmetric (i.e., if $h \in \mathcal{H}$ then $1 - h \in \mathcal{H}$), indicator function $I[x \in \mathcal{D}']$ is 1 if x belongs to the dataset $\mathcal{D}'$, and N is the number of instances in each dataset.

We introduce the following lemmas, which will be used to prove the theorem. Following Wang et al. (2022), we restate Theorem 1 from Albuquerque et al. (2019) in the context of our study as Lemma 1.

Lemma 1. *Let Λ be the convex hull constructed from document embedding distributions $\mathbb{D}_{ij}$, $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$; that is, $\Lambda = \sum_{ij} \pi_{ij} \mathbb{D}_{ij}$, where $\pi_{ij} \geq 0$ and $\sum_{ij} \pi_{ij} = 1$. Let $\gamma = \min_{\pi} d_{\mathcal{H}}(\mathbb{D}_u, \sum_{ij} \pi_{ij} \mathbb{D}_{ij})$, where $\pi = \{\pi_{ij} | i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\}$; the optimal is*

achieved when $\pi = \pi^*$. *Let* $\rho = \sup_{i_1,j_1,i_2,j_2} d_{\mathcal{H}}(\mathbb{D}_{i_1j_1}, \mathbb{D}_{i_2j_2})$ *be the diameter of the convex hull, where* $i_1, i_2 \in \{1, 2, \ldots, m\}$ *and* $j_1, j_2 \in \{1, 2, \ldots, n\}$. *The following bound for* $\epsilon_u(h)$ *holds:*

$$\epsilon_u(h) \leq \sum_{ij} \pi^*_{ij} \epsilon_{ij}(h) + \frac{\gamma + \rho}{2} + \epsilon^* \tag{A3}$$

where $\pi^* = \{\pi^*_{ij} | i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\}$ *and* ϵ^* *denotes the ideal joint risk, which is the minimum error achieved by any hypothesis in* $\mathcal{H}$ *over the unseen document embedding distribution* $\mathbb{D}_u$ *and the distribution* $\mathbb{D}^* = \sum_{ij} \pi^*_{ij} \mathbb{D}_{ij}$.

Proof. *See Albuquerque et al. (2019).*

Lemma 1 provides an upper bound on the error $\epsilon_u(h)$ incurred when classifying documents drawn from the unseen distribution $\mathbb{D}_u$. Note that both γ and ϵ^* attain their respective minimum values. Therefore, to reduce $\epsilon_u(h)$, we need to reduce the errors $\epsilon_{ij}(h)$, $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$, and the diameter ρ of the convex hull, the upper bound of which is given by the following lemma.

Lemma 2. *The following bound for the diameter* ρ *of the convex hull holds:*

$$\rho \leq \max_{a,b} d_{\mathcal{H}}(\mathbb{D}_{H,S_a}, \mathbb{D}_{H,S_b}) + \max_{a,b,c,d} d_{\mathcal{H}}(\mathbb{D}_{G_c,S_a}, \mathbb{D}_{G_d,S_b}) + 4 \max_{\alpha_{i_1j_1}, \alpha_{i_2j_2}} \left| \alpha_{i_1j_1} - \alpha_{i_2j_2} \right|, \tag{A4}$$

where domain IDs $a, b \in \{1, 2, \ldots, n\}$, *LLM IDs* $c, d \in \{1, 2, \ldots, m\}$, *and* $\alpha_{i_1j_1}$ *is the proportion of human-written documents in the documents drawn form distribution* $\mathbb{D}_{i_1j_1}$, $i_1 = 1, 2, \ldots, m$ *and* $j_1 = 1, 2, \ldots, n$.

Proof. *By definition,* $\rho = \sup_{i_1,j_1,i_2,j_2} d_{\mathcal{H}}(\mathbb{D}_{i_1j_1}, \mathbb{D}_{i_2j_2})$. *Without loss of generality, let the supremum achieved when* $i_1 = i^*_1$, $j_1 = j^*_1$, $i_2 = i^*_2$, *and* $j_2 = j^*_2$. *That is,* $\rho = d_{\mathcal{H}}(\mathbb{D}_{i^*_1j^*_1}, \mathbb{D}_{i^*_2j^*_2})$. For

simplicity of notation, we denote $P(h(\mathbb{D})) = P[I(h)|x \sim \mathbb{D}]$.

$$
\begin{aligned}
\rho &= d_{\mathcal{H}}(\mathbb{D}_{i_1^* j_1^*}, \mathbb{D}_{i_2^* j_2^*}) \\
&= 2 \sup_{h \in \mathcal{H}} |P[h(\mathbb{D}_{i_1^* j_1^*})] - P[h(\mathbb{D}_{i_2^* j_2^*})]| \\
&= 2 \sup_{h \in \mathcal{H}} \Big| \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_1^*}})] + (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}})] \\
&\qquad - \alpha_{i_2^* j_2^*} P[h(\mathbb{D}_{H, S_{j_2^*}})] - (1 - \alpha_{i_2^* j_2^*}) P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})] \Big| \\
&\leq 2 \sup_{h \in \mathcal{H}} \Big\{ \Big| \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_1^*}})] - \alpha_{i_2^* j_2^*} P[h(\mathbb{D}_{H, S_{j_2^*}})] \Big| \\
&\qquad + \Big| (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}})] - (1 - \alpha_{i_2^* j_2^*}) P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})] \Big| \Big\}.
\end{aligned}
\tag{A5}
$$

The last step in (A5) follows from the triangle inequality for the absolute function, i.e., $|a + b| \leq |a| + |b|$. *By adding and subtracting* $\alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_2^*}})]$ *in the first absolute term, and adding and subtracting* $\alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})]$ *in the second absolute term, we obstain the following inequality.*

$$
\begin{aligned}
\rho &\leq 2 \sup_{h \in \mathcal{H}} \Big\{ \Big| \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_1^*}})] - \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_2^*}})] + (\alpha_{i_1^* j_1^*} - \alpha_{i_2^* j_2^*}) P[h(\mathbb{D}_{H, S_{j_2^*}})] \Big| \\
&\quad + \Big| (1 - \alpha^*_{i_1^* j_1}) P[h(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}})] - (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})] + (\alpha_{i_2^* j_2^*} - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})] \Big| \Big\} \\
&\leq 2 \sup_{h \in \mathcal{H}} \Big\{ \Big| \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_1^*}})] - \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_2^*}})] \Big| + |\alpha_{i_1^* j_1^*} - \alpha_{i_2^* j_2^*}| \\
&\qquad + \Big| (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}})] - (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})] \Big| + |\alpha_{i_1^* j_1^*} - \alpha_{i_2^* j_2^*}| \Big\} \\
&\leq 2 \sup_{h \in \mathcal{H}} \Big\{ \Big| \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_1^*}})] - \alpha_{i_1^* j_1^*} P[h(\mathbb{D}_{H, S_{j_2^*}})] \Big| \Big\} \\
&\quad + 2 \sup_{h \in \mathcal{H}} \Big\{ \Big| (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}})] - (1 - \alpha_{i_1^* j_1^*}) P[h(\mathbb{D}_{G_{i_2^*}, S_{j_2^*}})] \Big| \Big\} + 4 |\alpha_{i_1^* j_1^*} - \alpha_{i_2^* j_2^*}| \\
&= \alpha_{i_1^* j_1^*} d_{\mathcal{H}}(\mathbb{D}_{H, S_{j_1^*}}, \mathbb{D}_{H, S_{j_2^*}}) + (1 - \alpha_{i_1^* j_1^*}) d_{\mathcal{H}}(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}}, \mathbb{D}_{G_{i_2^*}, S_{j_2^*}}) + 4 |\alpha_{i_1^* j_1^*} - \alpha_{i_2^* j_2^*}| \\
&\leq d_{\mathcal{H}}(\mathbb{D}_{H, S_{j_1^*}}, \mathbb{D}_{H, S_{j_2^*}}) + d_{\mathcal{H}}(\mathbb{D}_{G_{i_1^*}, S_{j_1^*}}, \mathbb{D}_{G_{i_2^*}, S_{j_2^*}}) + 4 |\alpha_{i_1^* j_1^*} - \alpha_{i_2^* j_2^*}| \\
&\leq \max_{a,b} d_{\mathcal{H}}(\mathbb{D}_{H, S_a}, \mathbb{D}_{H, S_b}) + \max_{a,b,c,d} d_{\mathcal{H}}(\mathbb{D}_{G_c, S_a}, \mathbb{D}_{G_d, S_b}) + 4 \max_{\alpha_{i_1 j_1}, \alpha_{i_2 j_2}} \left| \alpha_{i_1 j_1} - \alpha_{i_2 j_2} \right|.
\end{aligned}
\tag{A6}
$$

The second step follows from the triangle inequality. The fourth step is by definition.

Built on Lemmas 1 and 2, we have the following lemma.

Lemma 3. *The following bound for $\epsilon_u(h)$ holds:*

$$\epsilon_u(h) \leq \sum_{ij} \pi^*_{ij} \epsilon_{ij}(h) + \frac{1}{2} \max_{a,b} d_{\mathcal{H}}(\mathbb{D}_{H,S_a}, \mathbb{D}_{H,S_b}) + \frac{1}{2} \max_{a,b,c,d} d_{\mathcal{H}}(\mathbb{D}_{G_c,S_a}, \mathbb{D}_{G_d,S_b}) + C_0, \quad \text{(A7)}$$

$$C_0 = \frac{\gamma}{2} + \epsilon^* + 2 \max_{\alpha_{i_1 j_1}, \alpha_{i_2 j_2}} \left| \alpha_{i_1 j_1} - \alpha_{i_2 j_2} \right|. \quad \text{(A8)}$$

Proof. *By Lemma 1, we have*

$$\begin{aligned} \epsilon_u(h) &\leq \sum_{ij} \pi^*_{ij} \epsilon_{ij}(h) + \frac{\gamma + \rho}{2} + \epsilon^* \\ &= \sum_{ij} \pi^*_{ij} \epsilon_{ij}(h) + \frac{\rho}{2} + \frac{\gamma}{2} + \epsilon^* \\ &\leq \sum_{ij} \pi^*_{ij} \epsilon_{ij}(h) + \frac{1}{2} \max_{a,b} d_{\mathcal{H}}(\mathbb{D}_{H,S_a}, \mathbb{D}_{H,S_b}) + \frac{1}{2} \max_{a,b,c,d} d_{\mathcal{H}}(\mathbb{D}_{G_c,S_a}, \mathbb{D}_{G_d,S_b}) + C_0. \end{aligned} \quad \text{(A9)}$$

The last step directly follows from Lemma 2.

The following lemma connects $\mathcal{H}$-divergence and its empirical estimation in the context of our study.

Lemma 4. *Let d_{VC} be the VC dimension of a hypothesis space $\mathcal{H}$. We denote N_1 as the minimum number of human-written or LLM-generated training documents in a domain. The following inequality holds with probability at least $(1-\eta)^2$ for any $\eta \in (0,1)$.*

$$\begin{aligned} \frac{1}{2} d_{\mathcal{H}}(\mathbb{D}_{H,S_e}, \mathbb{D}_{H,S_f}) + \frac{1}{2} d_{\mathcal{H}}(\mathbb{D}_{G_c,S_a}, \mathbb{D}_{G_d,S_b}) \leq \frac{1}{2} \hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_e}, \mathcal{D}_{H,S_f}) + \frac{1}{2} \hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b}) \\ + 4 \sqrt{\frac{d_{VC} \log(2N_1) + \log(\frac{2}{\eta})}{N_1}}, \end{aligned} \quad \text{(A10)}$$

where domain IDs $a, b, e, f \in \{1, 2, \ldots, n\}$ and LLM IDs $c, d \in \{1, 2, \ldots, m\}$.

Proof. *We begin the proof by introducing Theorem 3.4 in Kifer et al. (2004) with a slight adaptation to the context of our study. Let $\mathcal{D}'$ and $\mathcal{D}''$ be two datasets drawn from distributions $\mathbb{D}'$ and $\mathbb{D}''$, respectively. We denote N' and N'' as the number of instances in the datasets $\mathcal{D}'$ and $\mathcal{D}''$, respectively. The following inequality holds:*

$$P\left[\left|d_{\mathcal{H}}(\mathbb{D}',\mathbb{D}'') - \hat{d}_{\mathcal{H}}(\mathcal{D}',\mathcal{D}'')\right| > \epsilon\right] \leq (2N')^{d_{VC}} e^{-N'\epsilon^2/16} + (2N'')^{d_{VC}} e^{-N''\epsilon^2/16}. \quad \text{(A11)}$$

Let $f(N) = (2N)^{d_{VC}} e^{-N\epsilon^2/16}$. Then $f'(N) = f(N)(\frac{d_{VC}}{N} - \frac{\epsilon^2}{16})$. Asymptotically, if N is large enough when $N > \frac{16 d_{VC}}{\epsilon^2}$, $f(N)$ is monotonically decreasing. Without loss of generality, let $N' \leq N''$ and the Inequality A11 becomes

$$P\left[\left|d_{\mathcal{H}}(\mathbb{D}',\mathbb{D}'') - \hat{d}_{\mathcal{H}}(\mathcal{D}',\mathcal{D}'')\right| > \epsilon\right] \leq 2(2N')^{d_{VC}} e^{-N'\epsilon^2/16}. \quad \text{(A12)}$$

Let $\eta = 2(2N')^{d_{VC}} e^{-N'\epsilon^2/16}$. For any $\eta \in (0,1)$, the following inequality holds with probability at least $(1-\eta)$

$$d_{\mathcal{H}}(\mathbb{D}',\mathbb{D}'') \leq \hat{d}_{\mathcal{H}}(\mathcal{D}',\mathcal{D}'') + 4\sqrt{\frac{d_{VC}\log(2N') + \log(\frac{2}{\eta})}{N'}}. \quad \text{(A13)}$$

Applying the relationship between $\mathcal{H}$-divergence and its empirical estimation established in Inequality A13 to our problem context, we have the following inequalities that hold with probability at least $(1-\eta)$ for any $\eta \in (0,1)$:

$$\frac{1}{2} d_{\mathcal{H}}(\mathbb{D}_{H,S_e}, \mathbb{D}_{H,S_f}) \leq \frac{1}{2}\hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_e}, \mathcal{D}_{H,S_f}) + 2\sqrt{\frac{d_{VC}\log(2N_1) + \log(\frac{2}{\eta})}{N_1}}, \quad \text{(A14)}$$

$$\frac{1}{2} d_{\mathcal{H}}(\mathbb{D}_{G_c,S_a}, \mathbb{D}_{G_d,S_b}) \leq \frac{1}{2} d_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b}) + 2\sqrt{\frac{d_{VC}\log(2N_1) + \log(\frac{2}{\eta})}{N_1}}, \quad \text{(A15)}$$

where N_1 is the minimum number of human-written or LLM-generated training documents in a domain. Given Inequalities A14 and A15, their summation is also bounded. Therefore, for any $\eta \in (0,1)$, with probability at least $(1-\eta)^2$,

$$\begin{aligned}\frac{1}{2} d_{\mathcal{H}}(\mathbb{D}_{H,S_e}, \mathbb{D}_{H,S_f}) + \frac{1}{2} d_{\mathcal{H}}(\mathbb{D}_{G_c,S_a}, \mathbb{D}_{G_d,S_b}) \leq \frac{1}{2}\hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_e}, \mathcal{D}_{H,S_f}) + \frac{1}{2}\hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b}) \\ + 4\sqrt{\frac{d_{VC}\log(2N_1) + \log(\frac{2}{\eta})}{N_1}}.\end{aligned} \quad \text{(A16)}$$

Theorem 1. *For any $\eta \in (0,1)$, with probability at least $(1-\eta)^2$, the following bound for $\epsilon_u(h)$ holds:*

$$\epsilon_u(h) \leq \sum_{ij} \pi_{ij}^* \epsilon_{ij}(h) + \frac{1}{2} \max_{a,b} \hat{d}_{\mathcal{H}}(\mathcal{D}_{H,S_a}, \mathcal{D}_{H,S_b}) + \frac{1}{2} \max_{a,b,c,d} \hat{d}_{\mathcal{H}}(\mathcal{D}_{G_c,S_a}, \mathcal{D}_{G_d,S_b}) + C, \quad \text{(A17)}$$

$$C = \frac{\gamma}{2} + \epsilon^* + 2 \max_{\alpha_{i_1 j_1}, \alpha_{i_2 j_2}} \left| \alpha_{i_1 j_1} - \alpha_{i_2 j_2} \right| + 4 \sqrt{\frac{d_{VC} \log(2N_1) + \log(\frac{2}{\eta})}{N_1}}. \quad \text{(A18)}$$

Proof. *Theorem 1 follows by applying the relationship between $\mathcal{H}$-divergence and its empirical estimation established in Lemma 4 and substituting the two $\mathcal{H}$-divergences in Lemma 3 with their empirical estimations. The terms in C either attain their respective minimum values (i.e., γ and ϵ^*) or are constant (i.e., the last two terms in C).*

APPENDIX B IMPLEMENTATION DETAILS OF DETECTION METHODS

Our proposed method was trained using the Adam optimizer (Kingma and Ba 2015) with a learning rate of 5×10^{-5} for four epochs. The number of memory units Q in a memory bank was set to 10, and each memory unit was implemented as a 768-dimensional vector. The hyperparameters r_1 and r_2 in Equation 11 were set to -3 and 1, respectively. The weights λ_y, λ_h, and λ_g in Equation 16 were set to 0.1, 0.2, and 0.2, respectively. Zero-shot methods, DetectGPT and Fast-DetectGPT, employed the T5-3B model (Raffel et al. 2020) to perturb texts and utilized the GPT-NeoX model as the scoring LLM. Another zero-shot method, DetectLLM, also used the GPT-NeoX model as the scoring model. Feature-based methods, GLTR and the multi-feature method, trained a logistic regression model using probability-based or linguistic features extracted from the training dataset. We used the off-the-shelf OpenAI detector publicly available on Hugging Face.[18] Other fine-tuning benchmarks, EAGLE and DaTeD, employed the same DistilRoBERTa backbone (Sanh 2019) as our method.

References

Albuquerque, I., Monteiro, J., Darvishi, M., Falk, T. H., and Mitliagkas, I. (2019). Generalizing to unseen domains via distribution matching. *arXiv preprint arXiv:1911.00804*.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine Learning*, 79:151–175.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, volume 30, pages 180–191.

[18] See `https://huggingface.co/openai-community/roberta-large-openai-detector` (last accessed on June 15, 2025)

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sanh, V. (2019). Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., and Philip, S. Y. (2022). Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8052–8072.