

Mask-based Membership Inference Attacks for Retrieval-Augmented Generation

Mingrui Liu

Nanyang Technological University
Singapore
mingrui001@e.ntu.edu.sg

Sixiao Zhang

Nanyang Technological University
Singapore
sixiao001@e.ntu.edu.sg

Cheng Long*

Nanyang Technological University
Singapore
c.long@ntu.edu.sg

Abstract

Retrieval-Augmented Generation (RAG) has been an effective approach to mitigate hallucinations in large language models (LLMs) by incorporating up-to-date and domain-specific knowledge. Recently, there has been a trend of storing up-to-date or copyrighted data in RAG knowledge databases instead of using it for LLM training. This practice has raised concerns about Membership Inference Attacks (MIAs), which aim to detect if a specific target document is stored in the RAG system's knowledge database so as to protect the rights of data producers. While research has focused on enhancing the trustworthiness of RAG systems, existing MIAs for RAG systems remain largely insufficient. Previous work either relies solely on the RAG system's judgment or is easily influenced by other documents or the LLM's internal knowledge, which is unreliable and lacks explainability. To address these limitations, we propose a Mask-Based Membership Inference Attacks (MBA) framework. Our framework first employs a masking algorithm that effectively masks a certain number of words in the target document. The masked text is then used to prompt the RAG system, and the RAG system is required to predict the mask values. If the target document appears in the knowledge database, the masked text will retrieve the complete target document as context, allowing for accurate mask prediction. Finally, we adopt a simple yet effective threshold-based method to infer the membership of target document by analyzing the accuracy of mask prediction. Our mask-based approach is more document-specific, making the RAG system's generation less susceptible to distractions from other documents or the LLM's internal knowledge. Extensive experiments demonstrate the effectiveness of our approach compared to existing baseline models.

CCS Concepts

- Computing methodologies → Information extraction; • Security and privacy → Human and societal aspects of security and privacy.

Keywords

Retrieval-Augmented Generation; Membership Inference Attacks

*Cheng Long is the corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
WWW '25, Sydney, NSW, Australia.

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1274-6/25/04

<https://doi.org/10.1145/3696410.3714771>

ACM Reference Format:

Mingrui Liu, Sixiao Zhang, and Cheng Long. 2025. Mask-based Membership Inference Attacks for Retrieval-Augmented Generation. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28–May 2, 2025, Sydney, NSW, Australia*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3696410.3714771>

1 Introduction

Large language models (LLMs) such as ChatGPT [3] and LLama [36], have revolutionized natural language processing. Despite these advancements, challenges remain, particularly in handling domain-specific or highly specialized queries [17]. LLMs often resort to "hallucinations," fabricating information outside their training data [46].

Retrieval-Augmented Generation (RAG) addresses this by integrating external data retrieval into generation, improving response accuracy and relevance [11, 20]. And RAG has been widely adopted by many commercial question-and-answer (Q&A) systems to incorporate up-to-date and domain-specific knowledge. For instance, Gemini [35] leverages the search results from Google Search to enhance its generation, while Copilot¹ integrates the documents or pages returned by Bing search into its context.

A recent trend involves storing up-to-date or copyrighted data in RAG knowledge databases instead of using it for LLM training. The SILO framework [26] exemplifies this approach, training LLMs on low-risk data (e.g., public domain or permissively licensed) and storing high-risk data (e.g., medical text with personally identifiable information) in the knowledge base. However, the legal implications of using data for generation models or systems are under scrutiny, with lawsuits filed globally due to potential copyright infringement [6, 25, 31, 32, 37]. This concern has spurred the development of Membership Inference Attacks (MIAs) to detect if specific data records were stored in RAG's knowledge database and could potentially appear in the generated texts, which raises concerns about *fair use doctrine* [12] or *General Data Protection Regulation (GDPR)* compliance [45].

Even though a growing body of research has focused on enhancing the trustworthiness of RAG systems [28, 38, 43, 47, 48], to the best of our knowledge, there are only two existing works targeting at the MIAs in RAG system. RAG-MIAs [1] judges whether a target document is in the knowledge database by directly asking the RAG system (i.e., utilizing the RAG's response (yes or no) as the judgement). This approach relies solely on the RAG system's judgment, which is unreliable and lacks explainability. S²MIAs [21] prompts the RAG system with the first half (typically the question part) of the target document, and if the RAG's response is semantically similar to the remaining half (typically the answer part) of

¹<https://github.com/features/copilot/>

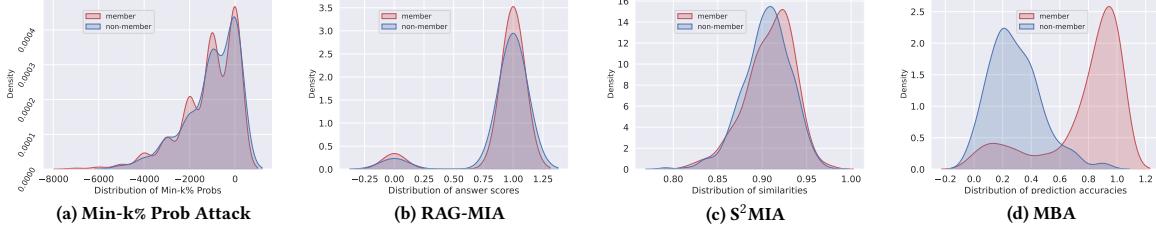


Figure 1: Distributions of Indicators for Member and Non-Member Samples in Different Methods on HealthCareMagic-100k dataset, which are visualised by kernel density estimate (KDE) method.

the target document, the target document is judged as a member. Several studies have focused on MIAs for LLMs [4, 24, 41]. Among these, Min-k% Prob Attack [33] is a state-of-the-art method that infers membership using the sum of the minimum k% probabilities of output tokens. However, as illustrated in Figure 1 (a)-(c), the indicators used to determine membership in existing methods (e.g., the similarity between the second half of the target document and the generated response in S²MIA) are nearly indistinguishable for member and non-member samples. This hinders the effectiveness of MIAs in RAG systems.

To effectively and reliably detect whether a target document resides in a RAG system’s knowledge database, we propose a Mask-Based Membership Inference Attacks (MBA) framework. The intuition is that if specific words (i.e., carefully selected words) in the document are masked, the RAG system is highly likely to predict the mask values accurately only if it retrieves the entire document as context. This prediction accuracy serves as our membership indicator. To conduct the inference, we first design a mask generation algorithm, masking M words or phrases in the original target document, where M is a hyperparameter. This involves extracting professional terms or proper nouns and selecting the most challenging words to predict using a pre-trained proxy language model. After obtaining the masked texts, we present the masked document to the RAG system and the RAG system is required to predict the mask values. A simple yet effective threshold-based judgement metric is adopted to determine the membership, i.e., if over $\gamma \cdot M$ masked words are correctly predicted, where γ is a hyperparameter, we judge the target document as a member of the knowledge database. As shown in Figure 1 (d), compared to existing methods, our mask-based method exhibits a significant gap in mask prediction accuracy between member (avg. 0.9) and non-member (avg. 0.2) samples. This enables our approach to effectively and reliably determine the membership of the target document.

To summarize, the main contributions of this paper are:

- We propose a Mask-based Membership Inference Attacks (MBA) framework targeting at the scenario of RAG system. Our framework is applicable to any RAG system, regardless of its underlying LLM parameters or retrieval method.
- We design a mask generation algorithm that strategically masks terms that would be difficult for the RAG system to predict if the full document were not retrieved as context.

- We evaluated our MBA framework on three public QA datasets. Extensive experiments demonstrated the effectiveness of our framework, achieving an improvement of over 20% in ROC AUC value compared to existing methods.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances response accuracy and relevance by incorporating external data retrieval into the generation process [11]. A common RAG paradigm involves using the user query to retrieve a set of documents, which are then concatenated with the original query and used as context [20].

Recent research has focused on various retrieval methods, including token-based retrieval [19], data chunk retrieval [30], and graph-based retrieval [9, 18]. Additionally, studies have explored adaptive retrieval [16] and multiple retrieval [14]. More advanced techniques such as query rewriting [10, 22] and alignment between retriever and LLM [5, 42] are beyond the scope of this paper.

2.2 Membership Inference Attacks

Membership Inference Attacks (MIAs) [13, 34] are privacy threats that aim to determine if a specific data record was used to train a machine learning model. MIAs for language models [4, 24, 33, 41] have been the subject of extensive research. Some representative attacking methods are: 1) **Loss Attack**: A classic MIA approach that classifies membership based on the model’s loss for a target sample [41]; 2) **Zlib Entropy Attack**: It refines the Loss Attack by calibrating the loss using zlib compression size [4]; 3) **Neighborhood Attack**: This method targets MIAs in mask-based models by comparing model losses of similar samples generated by replacing words with synonyms [24]; 4) **The Min-k% Prob Attack**: this approach calculates membership by focusing on the k% of tokens with the lowest likelihoods in a sample and computing the average probabilities. However, these works may not be directly applicable to RAG systems. Additionally, many of them rely on the loss or token output probabilities, which require access to LLM parameters or intermediate outputs that may not be available in black-box RAG systems.

Recently, there are two works targeting at the MIAs in RAG scenarios. RAG-MIAs [1] judges whether a target document is in the RAG system’s knowledge database by prompting the RAG system with “Does this: {Target Document} appear in the context? Answer

with Yes or No.", then utilizing the RAG's response (yes or no) as the judgement result directly. This approach relies solely on the RAG system's judgment, which can be unreliable and lacks explainability. S²MIA [21] prompts the RAG system with the first half of the target document, and compares the semantic similarity of the RAG's response and the remaining half of the target document. To enhance robustness, that work also incorporates perplexity of the generated response. A model is trained to determine the threshold values for similarity and perplexity. Membership is judged based on these thresholds: a target document is considered a member if its similarity exceeds the threshold and its perplexity falls below the threshold. However, the RAG system may not always strictly adhere to the original texts, and responses generated using internal knowledge may have similar similarity scores to those generated with retrieved documents. This can lead to unreliable and unconvincing prediction results.

3 Preliminaries

In this section, we establish the notation, provide a brief overview of Retrieval-Augmented Generation (RAG), and outline the specific task addressed in this paper.

3.1 RAG Overview

RAG systems typically consist of three primary components: a knowledge database, a retriever, and a large language model (LLM). The knowledge database, denoted as $\mathcal{D} = \{P_1, \dots, P_N\}$, comprises a collection of documents sourced from authoritative and up-to-date sources. The retriever is a model capable of encoding both queries and documents into a common vector space to facilitate retrieval. The LLM, such as ChatGPT or Gemini, is a trained language model capable of generating text.

The RAG process unfolds as follows: Given a user query \mathbf{q} , the system retrieves k relevant documents from \mathcal{D} using the retriever:

$$\mathcal{P}_k = \text{RETRIEVE}(\mathbf{q}, \mathcal{D}, k) \quad (1)$$

Typically, retrieval is based on similarity metrics like inner product or cosine similarity. The retrieved documents, concatenated as $\mathcal{P}_k = [p_1 \oplus \dots \oplus p_k]$, are then combined with a system prompt \mathbf{s} and the original query to generate a response using the LLM:

$$\mathbf{r} = \text{LLM}(\mathbf{s} \oplus \mathbf{q} \oplus \mathcal{P}_k) \quad (2)$$

Here, $[\cdot \oplus \cdot]$ represents the concatenation operation.

3.2 Task Formulation

We introduce the task of Membership Inference Attacks (MIAs) in RAG system.

Attacker's Target: Given a target document d , the objective is to determine whether d is present in the RAG system's knowledge database \mathcal{D} .

Attacker's Constraints: We target at the black-box setting in RAG system. The attacker cannot access the RAG system's knowledge base (\mathcal{D}) or the LLM's parameters. However, they can interact with the system freely and repeatedly. The RAG system's response is solely textual, providing answers to the user's questions without explicitly displaying the contents of the retrieved documents. This scenario is realistic, as users typically have unrestricted access to chatbots.

Attacker's Task: The attacker's task is to design a **Binary Membership Inference Classifier** (BMIC) that takes the target document (d), and the response of the RAG system (\mathbf{r}) as input. Formally, the probability of d being in \mathcal{D} is calculated as:

$$\begin{aligned} P(d \in \mathcal{D}) &= \text{BMIC}(d, \mathbf{r}), \\ \mathbf{r} &= \text{LLM}(\mathbf{s} \oplus Q_d \oplus \mathcal{P}_k) \end{aligned} \quad (3)$$

where \mathbf{r} is generated by the LLM using a system prompt \mathbf{s} , a well designed question generated from d (denoted as Q_d), and retrieved documents \mathcal{P}_k . Designing a method to generate Q that can effectively differentiate between responses generated with and without the target document in the context is a key challenge in MIAs for RAG systems.

MIA Workflow: The MIA process involves generating questions based on the target document d . If the RAG system's response (answer) \mathbf{r} accurately matches the original content of d , it can be inferred that d is present in the knowledge database \mathcal{D} . Conversely, if there is a significant mismatch, it suggests that d is not in \mathcal{D} .

Designing Principals: There are three main principals on designing the classifier and the adaption function:

- (1) **Effective Retrieval:** Q_d should successfully retrieve d if $d \in \mathcal{D}$. Recall that in RAG, relevant documents are retrieved based on the user query and used as context for generation. In this context, Q_d serves as the user query. If d cannot be successfully retrieved, it implies that d is not in \mathcal{D} , leading to a negative judgment.
- (2) **Indirect Information:** Q_d shall not directly reveal the information to be verified in the BMIC. While using d directly as Q_d might seem straightforward, it introduces bias: the RAG system will always include d in the context, regardless of its presence in the knowledge base, making the inference unreliable.
- (3) **Targeted Questions:** Questions should be challenging for the RAG system to answer if d is not in the knowledge base, and vice versa. Overly simple questions can be answered using internal knowledge or other retrieved documents, hindering judgment. Conversely, irrelevant questions may not elicit expected responses, even if d is successfully retrieved.

4 Methods

4.1 Overview

This section presents our proposed **Mask-Based Membership Inference Attacks** (MBA) framework, illustrated in Figure 2. We begin by explaining our motivation (Section 4.2). Subsequently, we introduce the two key components of our framework: **Mask Generation** (Section 4.3), which generates M masks within the original target document as our document-specific question (Q_d), and a **Binary Membership Inference Classifier** (BMIC, Section 4.4), which infers membership based on the masked texts. Our framework is non-parametric and can be applied to any black-box RAG system, regardless of LLM parameters or retrieval methods.

4.2 Motivation

We observe that when LLMs are tasked with cloze tests (predicting masked terms or phrases in a given text), they can accurately fill in the blanks if the complete original text is provided as a reference.

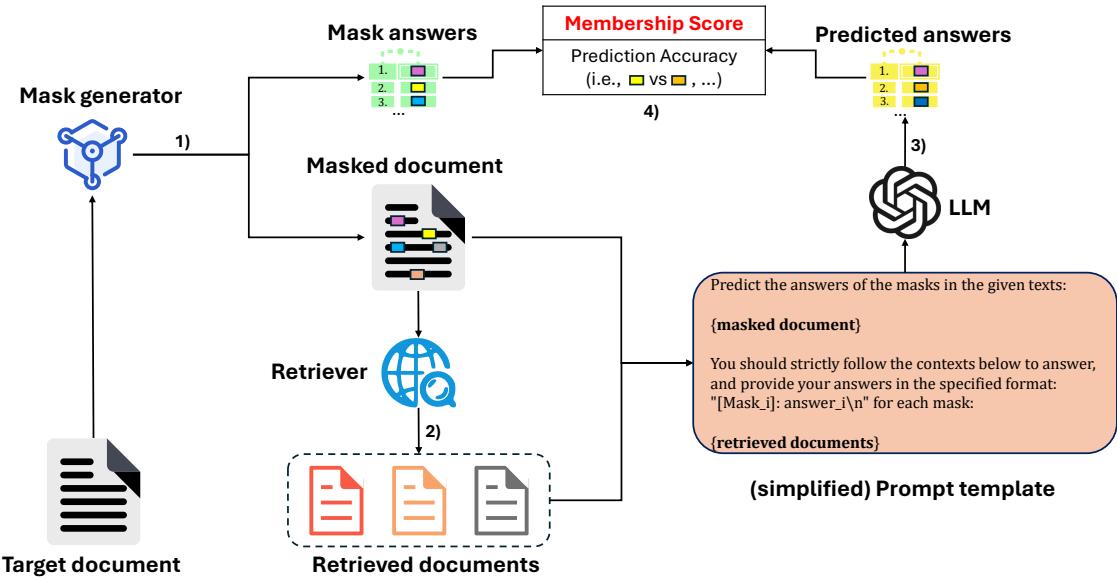


Figure 2: The overview of our proposed MBA framework.

This phenomenon inspired us to conduct MIAs in RAG systems using a cloze test approach.

Specifically, the masked target document is used as a query to retrieve relevant documents from the knowledge database. Due to the high similarity between the masked document and the original one, the target document is highly likely to be retrieved if it appears in the knowledge database. In this case, the masked words can be accurately predicted. Conversely, if the target document is not in the database, there is no direct information to guide mask prediction, leading to inaccurate predictions. Therefore, the accuracy of mask prediction can serve as an indicator of the target document's membership.

4.3 Mask Generation

The first step involves generating masked text from the original target document, which acts as the document-specific question (Q_d). We aim to select terms that are challenging to predict based solely on the LLM's internal knowledge or the context. While cloze question generation research [23, 40] exists, these works primarily focus on educational applications and may not be suitable for our purposes.

While it's tempting to use LLMs to generate masks, their inherent uncertainty presents two main challenges. First, LLMs may not always follow instructions to generate the desired number of masks. Second, the generated mask answers may not accurately align with the original words in the target document, potentially altering the document's content or even resulting in completely blank masked texts in some cases.

A straightforward approach would be to use a proxy language model to select terms based on their prediction difficulty (i.e., the probabilities to correctly predict them). However, we observed three challenges:

- (1) **Fragmented words:** Datasets often contain specialized terms or proper nouns that may not be recognized by language

model tokenizers. For example, GPT-2 might split "canestan" (a medicine) into "can," "est," and "an". Masking such terms based solely on prediction probability, which might generate "can[Mask]an", could hinder accurate prediction, even with the entire text retrieved.

- (2) **Misspelled words:** Datasets collected from human-generated content may contain misspelled words (e.g., the word "nearly" is written as "nearlt"). If such words are masked, LLMs tend to accurately predict the correct spelling (e.g., "nearly"), despite prompted to follow the original text, affecting prediction accuracy.
- (3) **Adjacent masks:** Masking two adjacent words can be problematic for LLMs. For instance, masking "walking" and "unsteadily" in the sentence "I went to the bathroom [Mask_1] (walking) [Mask_2] (unsteadily), as I tried to focus..." might lead the LLM to incorrectly predict "[Mask_1]: walking unsteadily; [Mask_2]: as I tried to focus". Specifically, the LLM might incorrectly identify the locations of the masked terms, despite its ability to effectively extract nearby terms or phrases.

To address these challenges, we incorporate an fragmented tokens extraction algorithm (Section 4.3.1), misspelled words correction (Section 4.3.2) and rule-based filtering methods into our mask proxy language model based generation process (Section 4.3.3).

4.3.1 Fragmented words extraction. We first extract words fragmented by the proxy language model's tokenizer, such as "canestan." The workflow involves identifying consecutive words (without spaces or punctuation in the middle) that are split by the tokenizer. This process is detailed in Algorithm 2 in Appendix B.1, where all the fragmented words are extracted and stored in the list `fragmented_words`.

4.3.2 Misspelled words correction. After extracting fragmented words, we further check whether they are misspelled words. If yes,

their corrected words are also obtained and recorded, as detailed in Algorithm 1.

We iterate through each extracted fragmented word (lines 2-10). For the current word, we obtain its index in the target document (line 3). We use a pre-trained spelling correction model to check if the word is misspelled. We pass the current word and its preceding two words to the model and record the corrected word (lines 4-6). We empirically found that using three words provides the best results, as fewer words can lead to semantic inconsistencies. The corrected words are recorded along with the original misspelled words in the list `fragmented_words` (lines 7-9).

Algorithm 1 WordsCorrection

Input: d
Output: `fragmented_words`

```

1: fragmented_words  $\leftarrow$  FragmentedWordExtraction( $d$ )
2: for  $i \in \{1, 2, \dots, |\text{fragmented\_words}|\}$  do
3:    $\text{index} \leftarrow \text{GETWORDINDEX}\left(\text{fragmented\_words}_{(i)}, d\right)$ 
4:    $\text{sub\_sentence} \leftarrow [d_{(\text{index}-2)} \oplus d_{(\text{index}-1)} \oplus d_{\text{index}}]$ 
5:   corrected_words  $\leftarrow \text{SCLM}(\text{sub\_sentence})$ 
6:   corrected_word  $\leftarrow \text{corrected\_words}_{(3)}$ 
7:   if fragmented_words(i)  $\neq$  word then
8:     fragmented_words(i)  $\leftarrow \{\text{fragmented\_words}_{(i)},$ 
      corrected_word
9:   end if
10:  end for
11:  return fragmented_words
```

4.3.3 Proxy language model based masking. To identify challenging words for masking, we employ a proxy language model. This model assigns a **rank score** to each word, reflecting its difficulty to predict.

Language models generate text by calculating the probability of each possible token given the preceding context and selecting the most likely one. The **rank score** indicates a word's position among these predicted candidates. For instance, in the sentence "... I would advise you to visit a [MASK] ...", if the correct token is "dentist" but the model predicts "doctor" (0.6), "medical" (0.25), and "dentist" (0.15), then "dentist" would have a rank score of 3. Words with *larger* rank scores are considered more challenging to predict. Therefore, we replace words with the largest rank scores (i.e., the words whose probabilities ranked at the back) with the "[MASK]" token, as detailed in Algorithm 4 of Appendix ??.

The input M represents the desired number of masks. We divide the target document into M equal-length subtexts and distribute the M masks evenly across these subtexts. For each subtext, we first add the subtexts before the current subtext (i.e., d_1 to $d_{(i-1)}$) as a prefix, and iterate through its words, adding them one by one to the prefix. We then determine whether the next word should be masked based on several criteria: if it is a stop word, punctuation, or adjacent to an already masked word, it is not masked, which is implemented by assigning its probability rank as -1. If a word is eligible for masking, we use the proxy language model to calculate its probability of occurrence in the given context and record its **rank score**. For extracted fragmented words, we first check for misspelled errors. If found, we use the corrected word (obtained in Algorithm 1) for

probability and **rank score** calculations. Otherwise, we calculate probabilities and **rank scores** for each token within the word, using the largest one to represent the word's overall **rank score**. For words that are not extracted fragmented words, their probabilities and **rank scores** are calculated by proxy language model directly.

The word with the largest **rank score** within each subtext is then masked, and its corresponding answer is recorded. If the masked word is an misspelled word, both the original word and the corrected word will be added to the answer set.

4.3.4 Mask integration. Finally, the masked words are integrated and numbered. The "[Mask]" labels in the masked text are numbered from "[Mask_1]" to "[Mask_M]". A ground truth mask answer dictionary is maintained in the format "[Mask_i]: answer_i," where "answer_i" is the i -th masked word.

4.4 Binary Membership Inference Classifier

The RAG system is prompted with the template shown in Figure 5 in Appendix A.1, where the masked document is obtained using the method introduced in Section 4.3, and the {retrieved documents} represent those retrieved from the RAG's knowledge database.

The response will be in the format "[Mask_i]: answer_i," where "answer_i" represents the predicted answer for "[Mask_i]". We then compare the predicted answers with the ground truth answers and count the number of correct predictions. If this count exceeds $\gamma \cdot M$, where $\gamma \in (0, 1]$ is a hyperparameter, we judge the target document as a member of the RAG's knowledge database; otherwise, we conclude it is not a member.

5 Experiments

5.1 Datasets

We evaluate our method on three publicly available question-answering (QA) datasets:

- **HealthCareMagic-100k**²: This dataset contains 112,165 real conversations between patients and doctors on HealthCareMagic.com.
- **MS-MARCO** [2]: This dataset features 100,000 real Bing questions with retrieved passages and human-generated answers. We use the "validation" set (10,047 QA pairs) for knowledge base construction. The knowledge base includes all unique documents retrieved by at least one question.
- **NQ-simplified**³: This is a modified version of the Natural Questions (NQ) dataset. Each question is paired with a shortened Wikipedia article containing the answer. We use the "test" set (16,039 QA pairs) to build a knowledge base by storing the shortened Wikipedia articles.

Following previous research [1, 21], we randomly selected 80% of the documents as member samples (stored in the RAG's knowledge base) and the remaining 20% as non-member samples. We randomly selected 1,000 instances for training (500 member and 500 non-member) and another 1,000 for testing (500 member and 500 non-member) to determine any necessary thresholds.

²<https://huggingface.co/datasets/RafaelMPereira/HealthCareMagic-100k-Chat-Format-en>

³<https://huggingface.co/datasets/LLukas22/nq-simplified>

5.2 Baselines

We evaluated our method against the following baseline approaches:

- **Min-k% Prob Attack** [33]: A state-of-the-art membership inference attack (MIA) for LLMs. It calculates a score based on the sum of the least likely tokens to determine membership.
- **RAG-MIA** [1]: This method directly queries the RAG system about the target document’s inclusion in the retrieved context.
- **S²MIA** [21]: This approach divides the target document into two halves, prompts the RAG system with the first half, and compares the semantic similarity between the second half and the RAG’s response. We compare 2 settings of S²MIA:
 - S²MIA_s: Relies solely on semantic similarity for MIA.
 - S²MIA_{s&p}: Incorporates both semantic similarity and perplexity for membership inference.

Of these methods, Min-k% Prob Attack and S²MIA_{s&p} require token prediction probabilities, which may not be accessible in certain black-box settings.

5.3 Evaluation Metric

We evaluate performance using a comprehensive set of metrics. Notably, we introduce **Retrieval Recall** as a unique metric for MIAs in RAG systems, distinguishing our work from previous studies [1, 21]. Retrieval recall measures whether the target document is successfully retrieved from the knowledge base when it exists. If the target document is among the top K retrieved documents, the recall is 1; otherwise, it is 0. We calculate the overall retrieval recall as the average across all membership documents, excluding non-member documents. In addition to retrieval recall, we also employ standard metrics commonly used in MIAs [8] and binary classification tasks, including **ROC AUC**, **Accuracy**, **Precision**, **Recall**, and **F1-score**. Specifically, member documents are labeled as 1, and non-member documents are labeled as 0. Each method outputs a logit value between 0 and 1 (e.g., the mask prediction accuracy), which is then used to calculate the metrics.

5.4 Settings and implementation

5.4.1 General settings. We leverage GPT-4o-mini⁴ as our black-box LLM, which is accessed by OpenAI’s API. For the RAG system, we utilize LangChain⁵ framework and integrate BAAI/bge-small-en [39] as the retrieval model, which encodes both queries and documents into 384-dimensional vectors. Further experiments, exploring the impact of different retrieval models and LLM backbones, are detailed in Appendix E and Appendix F, respectively. Retrieval is performed by calculating the inner product between these vectors, and an approximate nearest neighbor search is conducted using an HNSW index implemented in FAISS [7]. We set the number of retrieved documents (K) to 10 for all methods, a common setting in RAG systems. All experiments were conducted on a single NVIDIA RTX A5000 GPU.

5.4.2 Method-specific settings. We now detail the specific settings used in each method:

⁴<https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>

⁵<https://github.com/langchain-ai/langchain>

- **Min-k% Prob Attack:** k is a hyperparameter in this method. We varied k from 1 to 20, and selects the k with the best performance. This method also involves calculating the sum of minimum k% log probabilities as the indicator for membership inference. To obtain the log probabilities, we leverage the "logprobs" parameter within the OpenAI API⁶.
- **S²MIA:** Cosine similarity is used to measure the similarity between the second half of the original target document and the response generated by RAG. To calculate perplexity, the "logprobs" parameter is enabled to obtain the log probabilities of tokens, similar to the Min-k% Prob Attack method. XGBoost, as recommended in the original paper, is used as the binary classifier.
- **Our method:**
 - **Spelling Correction Model:** We leverage the pre-trained "oliverguhr/spelling-correction-english-base" model (139M parameters) from Hugging Face⁷ to address potential spelling errors.
 - **Proxy Language Model:** We employ the "openai-community/gpt2-xl" [29] model with 1.61B parameters as a proxy language model for difficulty prediction."
 - M : The number of masks is a hyperparameter in our method. We experimented with different values of M in {5, 10, 15, 20} for each dataset and selected the optimal M that produced the highest ROC AUC value.
 - γ : The threshold for mask prediction accuracy, used to determine membership, is a hyperparameter in our method. We varied this threshold from 0.1 to 1 for each dataset and selected the optimal threshold (γ) that produced the highest F1-score.

The results obtained using the optimal M are presented as our overall results.

5.5 Overall Performance

Table 1 presents the experimental results comparing our proposed MBA4RAG framework with baseline methods.

A key premise of MIAs in RAG is the successful retrieval of the target document if it exists in the knowledge database. Retrieval recall is therefore a crucial metric. Both RAG-MIA and MBA4RAG achieve high overall retrieval recall (over 0.9) due to their use of the full original target document or masked versions with high similarity. In contrast, S²MIA and Min-k% Prob Attack retrieve documents based on fragments, leading to potential discrepancies, especially in chunked knowledge databases. These methods exhibit lower retrieval recall, particularly in the HealthCareMagic dataset, likely due to the similarity of many patient-doctor dialogues.

For the specific performance, ROC AUC is a dominant metric for evaluating MIAs [1, 8, 21, 24]. Our method consistently outperforms baseline methods by over 20% across all datasets. Even though baseline methods may achieve notable performance on metrics like precision and recall, these results can be attributed to arbitrary strategies, such as judging all documents as members.

In conclusion, our mask-based MIA method effectively retrieves the target document when it exists in the knowledge database and

⁶https://cookbook.openai.com/examples/using_logprobs

⁷<https://huggingface.co/oliverguhr/spelling-correction-english-base>

Table 1: Performance comparison of different methods on MIAs for RAG systems.

Dataset	Model	Retrieval Recall	ROC AUC	Accuracy	Precision	Recall	F1-score
HealthCareMagic-100k	Min-k% Prob Attack	0.65	0.38	0.60	0.75	0.75	0.75
	RAG-MIA	0.93	0.49	0.75	<u>0.80</u>	0.91	0.86
	S ² MIA _s	0.62	0.46	0.77	0.79	0.96	0.87
	S ² MIA _{s&p}	0.62	<u>0.57</u>	<u>0.78</u>	0.85	<u>0.92</u>	<u>0.89</u>
	MBA	<u>0.87</u>	0.88	0.85	0.97	0.81	0.89
MS-MARCO	Min-k% Prob Attack	0.82	0.44	0.65	0.71	0.67	0.69
	RAG-MIA	0.98	0.52	<u>0.75</u>	0.81	0.90	<u>0.85</u>
	S ² MIA _s	0.81	0.64	<u>0.57</u>	0.80	0.63	0.71
	S ² MIA _{s&p}	0.81	<u>0.69</u>	0.66	<u>0.84</u>	0.61	0.71
	MBA	<u>0.97</u>	0.86	0.81	0.91	<u>0.85</u>	0.88
NQ-simplified	Min-k% Prob Attack	0.81	0.65	0.58	0.79	0.68	0.73
	RAG-MIA	<u>0.97</u>	0.52	<u>0.79</u>	0.82	0.95	<u>0.88</u>
	S ² MIA _s	0.81	0.67	0.64	<u>0.89</u>	0.64	0.74
	S ² MIA _{s&p}	0.81	<u>0.68</u>	0.66	0.87	0.68	0.76
	MBA	0.98	0.90	0.85	0.90	<u>0.91</u>	0.90

The rows in gray indicate models that require token log probabilities for calculations, which may not be accessible in certain scenarios. For each metric and dataset, the **best performance** is bolded, and the second-best is underlined.

Table 2: The ablation study of our method.

Dataset	Model	Retrieval Recall	ROC AUC
HealthCareMagic-100k	Random	0.88	0.68
	LLM-based	0.86	0.81
	MBA _{PLM}	0.85	0.74
	MBA _{w/o SC}	<u>0.87</u>	<u>0.85</u>
	MBA	0.87	0.88
MS-MARCO	Random	0.97	0.73
	LLM-based	0.95	0.80
	MBA _{PLM}	0.97	0.76
	MBA _{w/o SC}	<u>0.96</u>	<u>0.84</u>
	MBA	0.97	0.86
NQ-simplified	Random	0.96	0.75
	LLM-based	0.97	<u>0.86</u>
	MBA _{PLM}	0.97	0.69
	MBA _{w/o SC}	0.99	0.84
	MBA	<u>0.98</u>	0.90

focuses on the target document without being distracted by other retrieved documents. This leads to high performance and reliability.

5.6 Ablation Study

To assess the effectiveness of our mask generation method and its individual components, we compared it to several baseline approaches:

- **Random:** A simple baseline where masks are selected randomly.
- **LLM-based:** An alternative approach using an LLM to select words or phrases for masking. The prompt template is provided in Figure 6 in Appendix A.2.

- **MBA_{PLM}:** This variant only uses the proxy language model for word selection, omitting fragmented word extraction (Section 4.3.1) and misspelled word correction (Section 4.3.2).
- **MBA_{w/o SC}:** This variant excludes the misspelled word correction (Section 4.3.2) component from our full method.

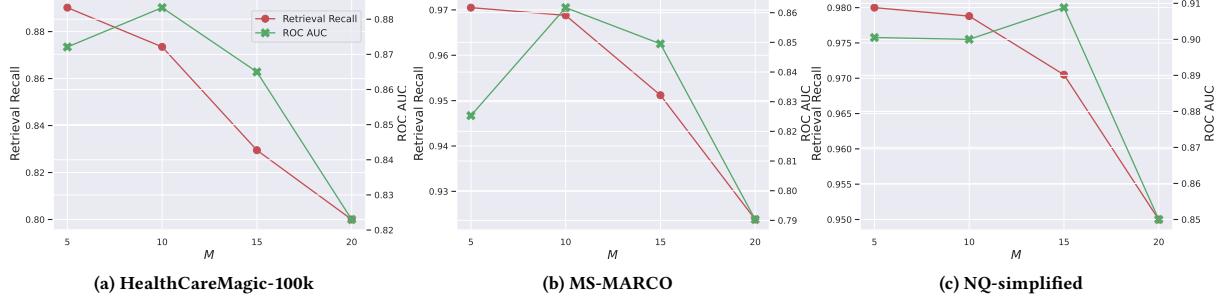
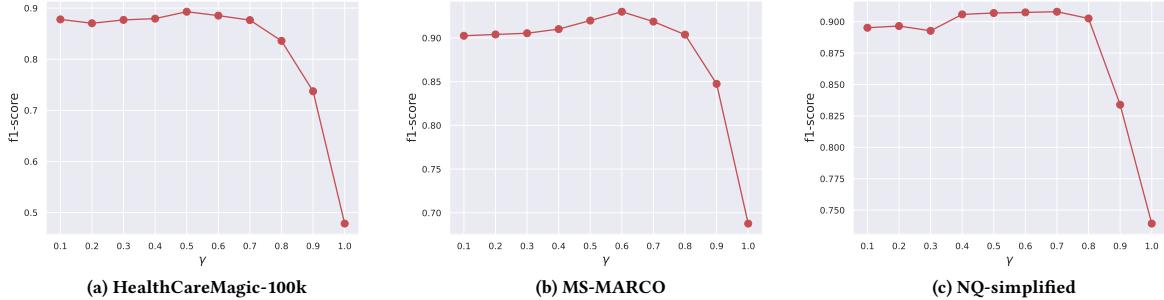
The results are presented in Table 2. Due to the similarity between the masked text and the original text (with only a few words or phrases replaced), retrieval recall is generally high for all masking strategies. Even random masking achieves competitive performance (ROC AUC of around 0.7) due to the mask-based method’s ability to resist distractions from other retrieved documents. However, random masking may generate masks for simple words (e.g., stop words), which can be easily predicted by the LLM, leading to false positives.

The LLM-based mask generation method is straightforward to implement and achieves acceptable performance in most cases (ROC AUC of about 0.8). However, due to the inherent uncertainty of LLMs, the original texts may be altered, and the number of generated masks may deviate from the desired amount.

Our proxy language-based mask generation method guarantees stable generation, ensuring exactly M masks are generated and distributed evenly throughout the text. However, challenges such as fragmented words, adjacent masks, and misspelled words can hinder prediction accuracy. By incorporating fragmented word processing and misspelled word correction, our method achieves effective and reliable MIAs for RAG systems.

5.7 Parameter Study

5.7.1 *The impact of M.* To analyze the impact of M , the number of masks generated in the target document, we varied M in {5, 10, 15, 20} and observed the retrieval recall and ROC AUC values (Figure 3).

Figure 3: The performances comparison varying M Figure 4: The performances comparison varying γ

As M increases, retrieval recall slightly decreases. This is because more masked words reduce the similarity between the masked text and the original document. The ROC AUC value also fluctuates slightly. When M is too small (e.g., below 5), the error tolerance decreases, meaning mispredictions have a larger impact on the final membership inference performance. When M is too large (e.g., over 20), simple words may be masked, leading to accurate prediction without the target document being retrieved. Additionally, decreased retrieval recall can lower the prediction accuracy of member samples, impacting overall performance.

Therefore, setting M between 5 and 15 (exclusive) is an optimal choice.

5.7.2 The impact of γ . To analyze the impact of the membership threshold γ , we varied γ from 0.1 to 1.0. Since retrieval recall and ROC AUC scores are independent of γ (as γ does not affect mask generation), we focused on f1-scores.

Figure 4 illustrates the results. While the optimal γ value varies across datasets (5, 6, and 7), performance is relatively consistent within the range of 0.5 to 0.7. This indicates that the performance is not highly sensitive to γ , and setting γ around 0.5 is generally a good choice.

5.7.3 The impact of K . K is a system parameter representing the number of retrieved documents in the RAG system, which may influence performance. However, this parameter is beyond our

framework and inaccessible to users. We verified that our method is insensitive to K in Figure 7 of Appendix C.

6 Conclusion

In this paper, we address the problem of membership inference for RAG systems, and propose a Mask-Based Membership Inference Attacks (MBA) framework. Our approach involves a proxy language-based mask generation method and a simple yet effective threshold-based strategy for membership inference. Specifically, we mask words that have the largest rank scores as predicted by a proxy language model. The target RAG system would have most of the masks correctly predicted if the document is a member. Extensive experiments demonstrate the superiority of our method over existing baseline models.

Acknowledgments

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund (Tier 2 Award MOE-T2EP20221-0013 and Tier 1 Awards (RG20/24 and RG77/21)). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

- [1] Maya Anderson, Guy Amit, and Abigail Goldsteen. 2024. Is My Data in Your Retrieval Database? Membership Inference Attacks Against Retrieval Augmented Generation. *arXiv preprint arXiv:2405.20446* (2024).
- [2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).
- [3] Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).
- [4] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulrich Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.
- [5] Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Denvy Deng, and Qi Zhang. 2023. Uprise: Universal prompt retrieval for improving zero-shot evaluation. *arXiv preprint arXiv:2303.08518* (2023).
- [6] Ryan J. Clarkson, Yana Hart, Tiara Avaness, and Valter Malkhasyan. 2023. Case 3:23-cv-03416. <https://storage.courtlistener.com/recap/gov.uscourts.cand.415223.gov.uscourts.cand.415223.1.0.pdf>
- [7] Matthijs Douze, Alexandre Guizhava, Chengqi Deng, Jeff Johnson, Gergely Szilvassy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). *arXiv:2401.08281 [cs.LG]*
- [8] Michael Duan, Anshuman Suri, Niiloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841* (2024).
- [9] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).
- [10] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496* (2022).
- [11] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).
- [12] Peter Henderson, Xuechen Li, Dan Jurafsky, Tatsunori Hashimoto, Mark A Lemley, and Percy Liang. 2023. Foundation models and fair use. *Journal of Machine Learning Research* 24, 400 (2023), 1–79.
- [13] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.
- [14] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299* 1, 2 (2022), 4.
- [15] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614* (2023).
- [16] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983* (2023).
- [17] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*. PMLR, 15696–15707.
- [18] Minki Kang, Jim Myung Kwak, Jinheon Baek, and Sung Ju Hwang. 2023. Knowledge graph-augmented language models for knowledge-grounded dialogue generation. *arXiv preprint arXiv:2305.18846* (2023).
- [19] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172* (2019).
- [20] Patrick Lewis, Éthan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
- [21] Yuying Li, Gaoyang Liu, Yang Yang, and Chen Wang. 2024. Seeing Is Believing: Black-Box Membership Inference Attacks Against Retrieval Augmented Generation. *arXiv preprint arXiv:2406.19234* (2024).
- [22] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283* (2023).
- [23] Shoya Matsumori, Kohhei Okuoka, Ryoichi Shibata, Minami Inoue, Yosuke Fukuchi, and Michita Imai. 2023. Mask and cloze: automatic open cloze question generation using a masked language model. *IEEE Access* 11 (2023), 9835–9850.
- [24] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership inference attacks against language models via neighbourhood comparison. *arXiv preprint arXiv:2305.18462* (2023).
- [25] Cade Metz. 2022. Lawsuit takes aim at the way A.I. is built. <https://www.nytimes.com/2022/11/23/technology/copilot-microsoft-ai-lawsuit.html>
- [26] Sewon Min, Suchin Gururangan, Eric Wallace, Weijia Shi, Hannaneh Hajishirzi, Noah A Smith, and Luke Zettlemoyer. 2023. Silo language models: Isolating legal risk in a nonparametric datastore. *arXiv preprint arXiv:2308.04430* (2023).
- [27] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877* (2021).
- [28] Zhenting Qi, Hanlin Zhang, Eric Xing, Sham Kakade, and Himabindu Lakkaraju. 2024. Follow My Instruction and Spill the Beans: Scalable Data Extraction from Retrieval-Augmented Generation Systems. *arXiv preprint arXiv:2402.17840* (2024).
- [29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [30] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1316–1331.
- [31] Joseph R. Saveri, Cadio Zirpoli, Christopher K.L Young, and McMahon Kathleen J. 2023. v. Meta Platforms, Inc. Case 3:23-cv-03417. <https://storage.courtlistener.com/recap/gov.uscourts.cand.415175.gov.uscourts.cand.415175.1.0.pdf>
- [32] Joseph R. Saveri, Cadio Zirpoli, Christopher K.L Young, and Kathleen J. McMahon. 2023. v. OpenAI, Inc. Case 3:23-cv-03417. https://storage.courtlistener.com/recap/gov.uscourts.cand.415174.gov.uscourts.cand.415174.1.0_1.pdf
- [33] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting pretraining data from large language models. *arXiv preprint arXiv:2310.16789* (2023).
- [34] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.
- [35] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soriciut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).
- [36] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
- [37] James Vincent. 2023. Getty images sues AI art generator stable diffusion in the us for copyright infringement, 2023. <https://www.theverge.com/2023/2/6/23587393/ai-art-copyright-lawsuit-getty-images-stable-diffusion>
- [38] Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. 2024. Certifiably Robust RAG against Retrieval Corruption. *arXiv preprint arXiv:2405.15556* (2024).
- [39] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. *arXiv:2309.07597 [cs.CL]*
- [40] Albert CM Yang, Irene YL Chen, Brendan Flanagan, and Hiroaki Ogata. 2021. Automatic generation of cloze items for repeated testing to improve reading comprehension. *Educational Technology & Society* 24, 3 (2021), 147–158.
- [41] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.
- [42] Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. *arXiv preprint arXiv:2305.17331* (2023).
- [43] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. 2024. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893* (2024).
- [44] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. 2024. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893* (2024).
- [45] Dawen Zhang, Pamela Finckenberg-Broman, Thong Hoang, Shidong Pan, Zhenchang Xing, Mark Staples, and Xiwei Xu. 2024. Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *AI and Ethics* (2024), 1–10.
- [46] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

- [47] Yujia Zhou, Yan Liu, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Zheng Liu, Chaozhuo Li, Zhicheng Dou, Tsung-Yi Ho, and Philip S Yu. 2024. Trustworthiness in Retrieval-Augmented Generation Systems: A Survey. *arXiv preprint arXiv:2409.10102* (2024).
- [48] Wei Zou, Rupeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisoneddrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867* (2024).

A Prompt templates

A.1 Prompt template to predict mask answers

Figure 5 illustrates the prompt template used to predict mask answers based on masked texts. These predicted answers are then applied to conduct membership inference attacks as detailed in Section 4.4.

You are given a text with several missing words or phrases, represented by placeholders in the format [Mask_i], where i is a unique number for each blank. Your task is to accurately fill in each placeholder with the most appropriate word or phrase based on the context of the sentence. Provide your answers in the specified format: "[Mask_i]: answer_i\n" for each mask, where "answer_i" shall be a word or phrase. You should strictly match the missing word or phrase based on the original context, without making any modifications, corrections, or substitutions.

The text is: {target document}
The context is: {retrieved document}

Figure 5: The prompt template to predict the masked words.

A.2 Prompt template to generate masks

As demonstrated in the ablation study, an alternative to our proposed mask generation approach is to directly leverage the LLM. The prompt template for this direct approach is shown in Figure 6.

You are given a text that needs $\{M\}$ (strictly follow this number) words or phrases masked. Your task is to select words or phrases that would be challenging to guess if removed from the text and replace them with a placeholder in the format [Mask_i], where i is a unique number for each mask. Your answer shall be in the format of:
Masked text:
Provide the text with masks in place of the selected words or phrases.
The answers for each mask:
[Mask_1]: answer_1\n ... [Mask_{M}]: answer_{M}

Figure 6: The prompt template to generate masks.

B Mask Generation Algorithms

B.1 The detailed algorithm of Fragmented words extraction

The workflow for extracting fragmented words is illustrated in Algorithm 2. It iterates through all words in d (lines 4-16). If the next word begins with a letter or certain hyphens, the words are combined into a single word.

Algorithm 2 FragmentedWordExtraction

Input: d ▷ the target document
Output: fragmented_words
1: $\text{fragmented_words} \leftarrow \emptyset$
2: $\text{word} \leftarrow ''$ ▷ set word as an empty string
3: $\text{flag} \leftarrow \text{False}$
4: **for** $j \in \{1, 2, \dots, |d| - 1\}$ **do**
5: $\text{word} \leftarrow \text{word} \oplus d_j$
6: **if** $d_{(j+1),0} \in \{[\text{a-z}][\text{A-Z}], '-' / '\}$ **then**
7: $\text{flag} \leftarrow \text{True}$
8: **continue**
9: **else**
10: **if** flag **then**
11: $\text{flag} \leftarrow \text{False}$
12: Append(fragmented_words, word)
13: **end if**
14: $\text{word} \leftarrow ''$
15: **end if**
16: **end for**
17: **return** fragmented_words

B.2 The processing of fragmented words

As stated in Section 4.3.3, fragmented words are treated as a single unit. If a word is misspelled, the corrected word is used to calculate the probability and rank score (lines 1-4). Otherwise, all tokens within the fragmented word receive a rank score, and the largest rank score represents the overall rank score of the entire fragmented word (lines 5-14).

Algorithm 3 FragmentedWordsRank

Input: d_i, j, t
Output: $\text{rank}_{i,(j+1)}$
1: **if** $[d_{i,(j+1)} \oplus \dots \oplus d_{i,(j+t)}]$ is unspelling **then**
2: $d_{i,(j+1)} \leftarrow \text{corrected_word}$
3: $\text{prob}_{i,(j+1)} \leftarrow \text{PLM}(d_{i,(j+1)} \mid \text{sentence_prefix})$
4: $\text{rank}_{i,(j+1)} \leftarrow \text{GETRANK}(\text{prob}_{i,(j+1)})$
5: **else**
6: $\text{word} \leftarrow ''$
7: **for** $k \in \{1, 2, \dots, t\}$ **do**
8: $s' \leftarrow [\text{sentence_prefix} \oplus \text{word}]$
9: $\text{prob}_{i,(j+k)} \leftarrow \text{PLM}(d_{i,(j+k)} \mid s')$
10: $\text{rank}_{i,(j+k)} \leftarrow \text{GETRANK}(\text{prob}_{i,(j+k)})$
11: $\text{word} \leftarrow [\text{word} \oplus d_{i,(j+k)}]$
12: **end for**
13: $\text{rank}_{i,(j+1)} \leftarrow \max_{k \in [1,t]} (\text{rank}_{i,(j+k)})$
14: **end if**
15: **return** $\text{rank}_{i,(j+1)}$

B.3 Full algorithm of mask generation

The complete workflow of the mask generation is illustrated in Algorithm 4. The input M represents the desired number of masks. We divide the target document into M equal-length subtexts and

distribute the M masks evenly across these subtexts. For each subtext, we first add the subtexts before the current subtext (i.e., d_1 to $d_{(i-1)}$) as a prefix, and iterate through its words, adding them one by one to the prefix (lines 6 and 34). We then determine whether the next word should be masked based on several criteria: if it is a stop word, punctuation, or adjacent to an already masked word (lines 11–14), it is not masked, which is implemented by assigning its probability rank as -1. If a word is eligible for masking, we use the proxy language model to calculate its probability of occurrence in the given context and record its **rank score** (lines 15–23). For extracted fragmented words (processed and returned split into multiple tokens by the LLM tokenizer), we first check for misspelled errors. If found, we use the corrected word (obtained in Algorithm 1) for probability and **rank score** calculations. Otherwise, we calculate probabilities and **rank scores** for each token within the word, using the largest one to represent the word's overall **rank score**. For words that are not extracted fragmented words, their probabilities and **rank scores** are calculated by proxy language model directly (lines 20–21).

The word with the largest **rank score** within each subtext is then masked, and its corresponding answer is recorded (lines 26–33). If the masked word is an misspelled word, both the original word and the corrected word will be added to the answer set (line 28).

C Impacts of the Number of Retrieved Documents

We varied the number of retrieved documents, K , within the common range of 5 to 20 in RAG systems. Figure 7 shows the impact of K on retrieval recall and ROC AUC.

Due to the high similarity between the masked and original target documents, successful retrieval can be achieved even with a small K (e.g., $K = 5$). Increasing K slightly improves retrieval recall, but the ROC AUC remains relatively constant. This indicates that our method is robust, and if the target document is retrieved, performance is guaranteed, regardless of the influence of other retrieved documents on mask prediction.

D Defense Strategies

To evaluate the robustness of our MBA framework, we test three mainstream defense strategies that could potentially reduce the effectiveness of MIAs for RAG systems:

- (1) **Prompt Modification** [1, 21]: This strategy appends the prompt with the sentence, "Do not directly repeat any retrieved content, but summarize it based on your understanding", to prevent the RAG system from revealing content from its external database.
- (2) **Re-Ranking** [44]: This strategy shuffles the order of the retrieved documents.
- (3) **Paraphrasing** [15]: This rewrites the documents in the RAG knowledge database. Specifically, each document is input into an LLM with the prompt, "Paraphrase the given document: *document*".

Table 3: Comparison of the Impacts of Different Defense Strategies

Dataset	Defense	Retrieval Recall	ROC AUC
HealthCareMagic-100k	None	0.87	0.88
	PM ¹	0.84	0.85
	Re-Ranking	0.85	0.86
	Paraphrasing	0.71	0.75
MS-MARCO	None	0.97	0.86
	PM ¹	0.97	0.86
	Re-Ranking	0.97	0.86
	Paraphrasing	0.91	0.81
NQ-simplified	None	0.98	0.90
	PM ¹	0.97	0.89
	Re-Ranking	0.98	0.90
	Paraphrasing	0.93	0.83

¹ PM represents Prompt Modification.

The results of these defenses are shown in Table 3. Prompt Modifying and Re-Ranking have negligible impact, as our detection method does not rely on repeating retrieved content or the order of retrieved documents. Even with these defenses, the LLM can still predict mask answers based on the retrieved documents, as the masked words can be selected from them.

Paraphrasing the knowledge database documents, however, slightly reduces both retrieval recall and ROC AUC. This is because paraphrased documents are not identical to the originals, impacting retrieval. More importantly, paraphrasing can alter the location and content (e.g., through synonym substitution) of masked words, reducing prediction accuracy and thus membership inference performance. Despite this, the ROC AUC remains above 0.75 across all three datasets, significantly outperforming baseline methods. This suggests that our mask selection method effectively chooses words that are generally robust to paraphrasing.

Table 4: Comparison of the Impacts of Different Retrieval Models

Dataset	Retriever	Retrieval Recall	ROC AUC
HealthCareMagic-100k	BGE-en	0.87	0.88
	Sentence-T5-xxl	0.82	0.87
	ada-002	0.89	0.88
MS-MARCO	BGE-en	0.97	0.86
	Sentence-T5-xxl	0.93	0.84
	ada-002	0.97	0.86
NQ-simplified	BGE-en	0.98	0.90
	Sentence-T5-xxl	0.95	0.89
	ada-002	0.97	0.89

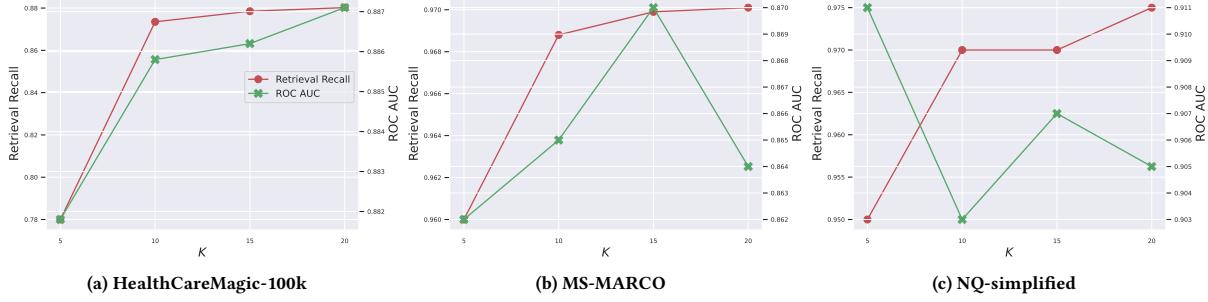


Figure 7: The performances comparison varying the number of K

Algorithm 4 MaskGeneration

```

Input:  $d, M$ 
Output:  $d_{\text{Masked}}$ , answers
1:  $\text{fragmented\_words} \leftarrow \text{WordsCorrection}(d)$ 
2:  $[d_1 \oplus \dots \oplus d_M] \leftarrow \text{SPLIT}(d_{\text{Masked}})$  ▷ split into  $M$  subtexts by length
3:  $\text{prefix} \leftarrow \emptyset$ 
4: for  $i \in \{1, 2, \dots, M\}$  do
5:    $\text{Prob\_Rank}_i = \emptyset$ 
6:    $\text{sentence\_prefix} \leftarrow \text{prefix}$ 
7:    $s \leftarrow 0$ 
8:   for  $j \in \{1, 2, \dots, |d_i| - 1\}$  do
9:      $j \leftarrow j + s$ 
10:     $\text{sentence\_prefix} \leftarrow [\text{sentence\_prefix} \oplus d_{i,j}]$ 
11:    if  $d_{i,(j+1)}$  is stop word or punctuation then
12:       $\text{rank}_{i,(j+1)} \leftarrow -1$ 
13:    else if  $d_{i,(j)} \text{ or } d_{i,(j+2)}$  is "[Mask]" then
14:       $\text{rank}_{i,(j+1)} \leftarrow -1$  ▷ do no mask adjacent terms
15:    else
16:      if  $[d_{i,(j+1)} \oplus \dots \oplus d_{i,(j+t)}]$  in  $\text{fragmented\_words}$  then
17:         $s \leftarrow s + t$ 
18:         $\text{rank}_{i,(j+1)} \leftarrow \text{FragmentedWordsRank}(d_i, j, t)$ 
19:      else
20:         $\text{prob}_{i,(j+1)} \leftarrow \text{PLM}(d_{i,(j+1)} \mid \text{sentence\_prefix})$ 
21:         $\text{rank}_{i,(j+1)} \leftarrow \text{GETRANK}(\text{prob}_{i,(j+1)})$ 
22:      end if
23:    end if
24:     $\text{Prob\_Rank}_i \leftarrow \text{Append}(\text{Prob\_Rank}_i, \text{rank}_{i,(j+1)})$ 
25:  end for
26:   $m \leftarrow \text{argmax}(\text{Prob\_Rank}_i)$  ▷ the token to be masked
27:  if  $d_{i,m}$  is misspelled word then
28:     $\text{Append}(\text{answers}, \{d_{i,m}, \text{corrected\_word}\})$ 
29:  else
30:     $\text{Append}(\text{answers}, d_{i,m})$ 
31:  end if
32:   $\text{Append}(\text{answers}, d_{i,m})$ 
33:   $d_{i,m} \leftarrow "[\text{Mask}]"$ 
34:   $\text{prefix} \leftarrow [\text{prefix} \oplus d_i]$ 
35: end for
36:  $d_{\text{Masked}} = [d_1 \oplus \dots \oplus d_M]$ 
37: return  $d_{\text{Masked}}, \text{answers}$ 

```

Table 5: Comparison of the Impacts of Different LLM Backbones

Dataset	LLM	ROC AUC
HealthCareMagic-100k	GPT-4o-mini	0.88
	GPT-3.5-turbo	0.87
	Gemini-1.5	0.79
MS-MARCO	GPT-4o-mini	0.86
	GPT-3.5-turbo	0.86
	Gemini-1.5	0.81
NQ-simplified	GPT-4o-mini	0.90
	GPT-3.5-turbo	0.91
	Gemini-1.5	0.84

E Impacts of Retrievers

We tested our framework's robustness to different retrieval methods within the RAG system, comparing performance across three retrievers: **BGE-en** [39], **Sentence-T5-xxl** [27] and **text-embedding-ada-002**⁸.

Table 4 presents the results, demonstrating that our framework is indeed robust to variations in retrieval methods. Our generated masked documents, due to their high similarity to the originals, effectively retrieve target documents (if present in the RAG knowledge database) regardless of the retrieval model used. This ensures high retrieval recall. Because target documents can be effectively retrieved when present, our mask-based membership inference framework can reliably distinguish between member and non-member documents.

F Impacts of LLM Backbones

We also investigated the impact of different LLM backbones within the RAG system. Using BGE-en as our retrieval model, we tested three mainstream LLMs: **GPT-4o-mini**, **GPT-3.5-turbo**, and **Gemini-1.5**. Since the retrieval model was held constant, retrieval recall remained the same across all tests; therefore, Table 5 presents only the ROC AUC values.

The results show that while ROC AUC values vary with different LLM backbones, overall performance remains consistently

⁸<https://platform.openai.com/docs/guides/embeddings>

<human>: My baby has been pooing 5-6 times a day for a week. In the last few days it has increased to 7 and they are very watery with green stringy bits in them. He does not seem unwell i.e no temperature and still eating. He now has a very bad nappy rash from the pooing ...help!

<bot>: Hi... Thank you for consulting in Chat Doctor. It seems your kid is having viral diarrhea. Once it starts it will take 5-7 days to completely get better. Unless the kids having low urine output or very dull or excessively sleepy or blood in motion or green bilious vomiting...you need not worry. There is no need to use antibiotics unless there is blood in the motion. Antibiotics might worsen if unnecessarily used causing antibiotic associated diarrhea. I suggest you use zinc supplements (Z&D Chat Doctor).

(a) The original target document

<bot>: It sounds like your baby is experiencing some gastrointestinal upset, which can lead to increased bowel movements and watery stools. The green stringy bits could be a sign of undigested food or mucus, which sometimes occurs with diarrhea. Since your baby is still eating well and doesn't have a fever, it may not be an immediate cause for concern, but it's important to keep an eye on hydration. Ensure they are drinking enough fluids, and consider offering some electrolyte solutions if recommended by your pediatrician. For the nappy rash, keep the area clean and dry, and apply a barrier cream to protect the skin from further irritation. If the diarrhea persists for more than a few days or if your baby shows any signs of dehydration, such as decreased urination or lethargy, please contact your doctor for further evaluation. It's always better to be cautious with little ones, so don't hesitate to reach out if you have any more concerns.

(b) S²MIA's response of non-member document

<bot>: It sounds like your baby might be experiencing a case of diarrhea, which can sometimes occur due to a variety of factors, including dietary changes or minor infections. It's great to hear that he doesn't have a fever and is still eating well, as those are positive signs. To help with the nappy rash, make sure to keep the area clean and dry, and consider using a barrier cream with zinc oxide to protect his skin. Additionally, ensure he is staying hydrated; you can offer oral rehydration solutions if necessary. If the watery stools persist or if you notice any signs of dehydration, such as fewer wet diapers or lethargy, please consult your pediatrician for further evaluation and possible treatment options.

(c) S²MIA's response of member document

<human>: My baby has been pooing [Mask_1] times a day for a [Mask_2]. In the last few days it has increased to [Mask_3] and they are very watery with green [Mask_4] bits in them. He does not seem unwell i.e no [Mask_5] and still eating. He now has a very bad [Mask_6] rash from the pooing ...help!

<bot>: Hi... Thank you for consulting in Chat Doctor. It seems your kid is having [Mask_7] diarrhea. Once it starts it will take [Mask_8] days to completely get better. Unless the kids having low urine output or very [Mask_9] or excessively sleepy or blood in [Mask_10] or green bilious vomiting...you need not worry.

(d) The masked document generated by MBA

[Mask_1]: multiple
[Mask_2]: week
[Mask_3]: several
[Mask_4]: slimy
[Mask_5]: fever
[Mask_6]: diaper
[Mask_7]: acute
[Mask_8]: a few
[Mask_9]: irritable
[Mask_10]: stools

[Mask_1]: 5-6
[Mask_2]: week
[Mask_3]: 7
[Mask_4]: stringy
[Mask_5]: temperature
[Mask_6]: nappy
[Mask_7]: viral
[Mask_8]: 5-7
[Mask_9]: dull
[Mask_10]: motion

(e) Non-member answers

(f) Member answers

Figure 8: The comparison of generated texts from S²MIA and MBA.

high (generally above 0.8). This demonstrates that our mask-based framework is applicable to RAG systems using a variety of large language models.

G Case Study

Figure 8 (a) presents a dialogue between a user and a doctor from the HealthCareMagic-100k dataset. The dialogue is split into two halves: the user's query and the doctor's response.

In the S²MIA model, the first half of the target document is used to prompt the RAG system. Both when the target document is a

member (Figure 8 (c)) and when it's not (Figure 8 (b)), the generated responses do not fully align with the original text, missing key details such as "5-7 days" and "low urine output." Additionally, the cosine similarity scores are similar for both member (0.877) and non-member (0.883) cases, contradicting the expectation that a higher similarity would indicate membership. This suggests that similarity-based methods may not reliably distinguish between member and non-member documents.

Figure 8 (d) shows the masked target document. When the document is not a member (Figure 8 (e)), only one mask (mask_2) is

predicted correctly. In contrast, all masks are predicted correctly when the document is a member (Figure 8 (f)). In the non-member case, masks are often predicted incorrectly, with specific detailed numbers replaced by vague descriptions (e.g., mask_1, 3, and 8), and certain words replaced by synonyms (e.g., mask_5), while other words are replaced with semantically unrelated terms (e.g., mask_4, 6, 7, 9, 10). This demonstrates the effectiveness of our mask generation algorithm in distinguishing between member and non-member documents.