A Privacy-Preserving Framework for Keyword-Based Document Retrieval

Kerem Bayramoğlu
kerem.bayramoglu@bilkent.edu.tr
Bilkent University
Turkey

Ergün Batuhan Kaynak
batuhan.kaynak@bilkent.edu.tr
Bilkent University
Turkey

Melih Coşğun
melih.cosgun@bilkent.edu.tr
Bilkent University
Turkey

Kousar Kousar
kousar.kousar@bilkent.edu.tr
Bilkent University
Turkey

Ömer Kaan Gürbüz
kaan.gurbuz@bilkent.edu.tr
Bilkent University
Turkey

Abstract

Concerns about privacy have grown with the digitization of everyday life, especially as Internet services collect large amounts of personal data. If these records are exploited, search engines, which store user queries for individualized information retrieval, pose a serious privacy threat. For this reason, hiding search intent via query obfuscation has become a crucial approach in Private Web Search to protect user privacy. The lack of semantic meaning in the random searches produced by early obfuscation techniques reduced both privacy and retrieval value. Although recent developments such as proxy-based and scrambling techniques have improved privacy, they are still computationally expensive or vulnerable to data leakage.

This study presents a novel query obfuscation technique that utilizes Large Language Models(LLMs). The proposed method ensures robustness to adversarial attacks. To further enhance privacy, the method considers the user's consecutive queries. This work aims to provide a utility and privacy-preserving query obfuscation schema that is robust to adversarial attacks by addressing the shortcomings of earlier methods.

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**.

Keywords

Private Information Retrieval, Query Obfuscation

1 Introduction

The increasing adoption of information retrieval systems in daily life has led to extensive collection of personal data. While companies try to gather this information for better-personalized results or advertisements, this situation may raise big privacy concerns. In particular, search engines store user queries for personalized information retrieval, but these query logs can reveal sensitive information about individuals if misused. Query obfuscation is a method used in Private Web Search to protect users' privacy by masking their true search intent.

Simple obfuscation using random query generation from predetermined vocabularies was the focus of early attempts at query privacy. Although computationally efficient and easy to implement, these approaches often produced patterns that were easy to identify because they were unable to preserve semantic coherence between original and dummy queries.

Later, proxy-based and scrambling strategies provided attempted to enhance privacy by converting sensitive phrases into nonsensitive ones. However, these methods were still vulnerable to attacks from semantic analysis and often weakened search functionality.

Another approach of adding controlled noise to query representations, similar to Differential Privacy (DP) [6] frameworks, has provided stronger privacy guarantees. Although DP-based techniques increase privacy by noise addition, this hinders utility since the retrieval process is more complicated with re-ranking strategies. Also, noise addition is not a guarantee for privacy protection.

Another major limitation of current work is the way current methods handle consecutive queries. These works fail to maintain semantic consistency across query sequences. In addition, the increasing complexity of inference techniques, especially employing modern language models, poses new challenges for query privacy that existing methods do not address.

The goal of this work is to develop a novel query obfuscation approach that:

- Preserves privacy by protecting search intent and query patterns.
- Has comparable relevance in document retrieval task.
- Is robust against attacks.

This will be achieved by generating dummy queries alongside the original query such that the original query is not distinguishable. The proposed method will utilize Large Language Models(LLM) to generate a wide range of dummy queries, taking into account the user's previous queries to further enhance privacy.

This work will address the limitations of existing query obfuscation techniques by:

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

- Using a robust privacy framework to ensure verifiable privacy guarantees.
- Generating diverse dummy queries that will hide the user's true search intent.
- Not altering the original query, this way preserving utility.
- Evaluating the proposed method against realistic adversarial attacks to evaluate its robustness

By successfully achieving these objectives, this work will contribute to the development of more effective and user-friendly private web search systems.

2 Related Work

Privacy-preserving web search has become an essential area of research, with various techniques designed to protect user queries and search intents. This section categorizes and discusses these relevant techniques.

2.1 Early Query Obfuscation Techniques

Query obfuscation techniques aim to mask users' true search intents by generating additional "cover queries." Early methods, such as [5, 8] introduced random query generation from predefined dictionaries, providing plausible deniability. However, these methods failed to ensure semantic relevance between true and cover queries, compromising privacy and retrieval utility.

2.2 Proxy-Based and Scrambling Techniques

Bashir et al. [2] employed a proxy dictionary to map sensitive terms to non-sensitive terms. Users issue proxy queries, and the IR system generates both true and cover queries using the dictionary. This approach eliminates the need for local computation of cover queries, ensuring scalability and resource efficiency. A study by Mivule et al. [7] employs a method that uses keyword permutations to confuse adversaries attempting to infer user intent but struggles to balance privacy with utility, as overly scrambled queries may fail to retrieve meaningful results.

2.3 Differential Privacy-Based Approaches

Differential Privacy (DP) provides a robust framework for protecting user data. In a study by Faveri et al. [3] DP mechanisms were employed to generate obfuscated queries through "safe" and "candidate" boxes. The technique ensures that obfuscated queries differ significantly from the original ones while maintaining retrieval effectiveness. In contrast to [2], which focuses on semantic mappings, WBB relies on DP's noise injection prevents adversarial inference. While WBB provides robust privacy, its utility degrades with strict candidate box sizes.

2.4 Context-Aware Query Obfuscation

Context-aware methods preserve semantic relevance between consecutive queries. The work by Ahmad et al. [1] proposes a client-side solution for user privacy by submitting genuine queries alongside decoy queries and clicks. These decoy queries are generated from hierarchically organized language models, ensuring coherence with the user's search context. The approach increases the plausibility of fake intents, making it challenging for search engines to infer true user intent.

2.5 Entropy-Based Techniques

Entropy defined by Claude Shannon is the basis for a range of applications in information theory [10]. For instance [11] used entropy to similarity matrices, quantifying the variability between data points over time to detect cumulative changes. While entropy has not been directly used in query obfuscation, we extend its application by employing it as a metric to evaluate and enhance the diversity of obfuscated query sets. This novel use of entropy ensures that dummy queries are sufficiently unpredictable, improving the robustness of privacy-preserving mechanisms in web search.

3 Methodology

Our approach to this problem is to generate dummy queries alongside the original user query. These dummy queries are generated in such a way that the attacker (IR system) cannot distinguish the original user queries from the dummy ones while still providing the related documents for each query sent. Our main steps are as follows:

1. **Dummy Query Generation:** We utilize large language models (LLMs) to generate $N - 1$ dummy queries for every user query at time t.
2. **Query History Maintenance:** A history of dummy queries is maintained for time $t - 1$. For follow-up user queries, corresponding follow-up dummy queries are generated to preserve consistency in obfuscation patterns.
3. **Randomized Query Submission:** Both original and dummy queries are submitted to the information retrieval system in random order to further obfuscate access patterns.
4. **Filter The Retrieved Documents:** Finally, all the retrieved documents for time t are filtered, and the user is provided the documents for the original query.

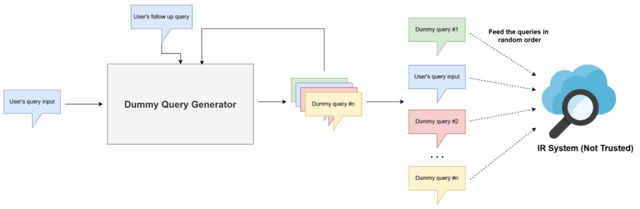


Figure 1: The overall pipeline of the security mechanism. The blue chatboxes represent the queries submitted by users.

The entire pipeline of the security mechanism can be seen in Figure 1. In this method, we preserve the user's original query and the retrieved documents, thus preserving the utility, which is different from the previous work that obfuscates the original query.

The generation of dummy queries involves two distinct methodologies depending on the scenario: (1) generating a dummy query for a fresh topic and (2) generating a dummy query for a follow-up query. For both techniques, we utilized the ChatGPT API [9]. We followed a similar approach for each with small nuances.

3.1 Dummy Query Generation for Fresh Topic

Generating dummy queries for a fresh topic follows several steps. Overall pipeline can be seen in Figure 2.

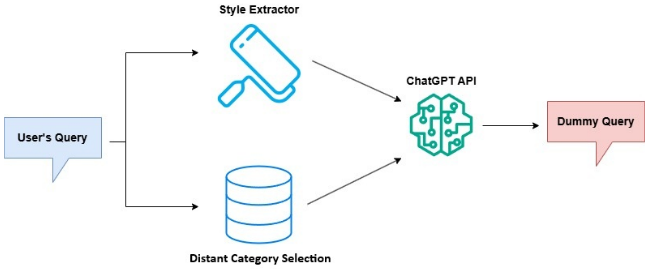


Figure 2: Pipeline of the dummy query generation for a fresh topic.

The pipeline begins with extracting the style information from the user's query. This information is then fed into the prompt to ensure that the generated dummy queries follow the style of the user's query. The style considerations include:

(1) **Query Length:** The number of words in the user's query is counted, and the generated dummy queries are adjusted to have a similar length.
(2) **Question Mark Usage:** If the user's query contains a question mark, the generated queries should also include a question mark.
(3) **Question Word Usage:** If the user's query is in question form, such as starting with "what," "why," "how," etc., the generated dummy queries should follow the same structure.
(4) **Capitalization Patterns:** If the user's query starts with a capital letter, the generated dummy queries should also start with a capital letter.

In the second step of the pipeline, we used a category dataset to generate dummy queries based on specific categories. First, we use BERT[4] embeddings to calculate the similarity of the input query with other categories to identify the best-fit category for the user's query. After determining the most similar category, we select $N - 1$ distinct categories for the dummy queries. This category information is then fed into the prompt to instruct the LLM to generate queries based on the given categories.

Next, we use the user's query and its calculated category to provide a one-shot example to the LLM, illustrating how to generate a query based on a category. This one-shot example helps the LLM create queries from the selected categories with a degree of relativeness similar to that of the user's query and its category. This ensures that the generated queries are neither overly related to the category nor entirely unrelated, maintaining a balance similar to the relationship between the user's query and its category. Our final prompt to the LLM can be seen in Algorithm 1.

```
system_message = f"""You are a query generation
    assistant. Generate a single query that:
    1. Is specifically about {category}
    2. Matches the original query's style
    {style_prompt}
    """

    if use_new_prompt:
        system_message += f"""
        Here is an example query based on a
            category. But yours should not be
            exactly the same:
        Category: {input_category}
        Query: {input_query}
        """
```

Listing 1: Dummy query generation for a fresh topic prompt

3.2 Dummy Query Generation for Follow-Up Query

An important vulnerability exists when generating follow-up queries independently: if an attacker observes the complete query sequence, they can identify the original user queries by analyzing query relationships, as illustrated in Figure 3. If the user's initial and subsequent queries are related, as is the case in the majority of instances, the pattern between those queries can be distinguished from the random queries generated by the generator.

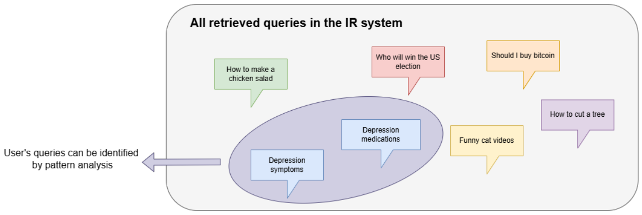

All retrieved queries in the IR system
How to make a chicken salad
Who will win the US election
Should I buy bitcoin
How to cut a tree
Depression medications
Funny cat videos
Depression symptoms
User's queries can be identified by pattern analysis

Figure 3: An example attack by a pattern analysis. The blue chatboxes represent the queries submitted by users.

To address this issue, we are maintaining a record of the generated queries at a given time point (t). Subsequently, when generating dummy queries for a user query at a time point ($t + 1$), the generator uses the information from previous queries to generate queries that will serve as a follow-up query. This technique results in the creation of N clusters for the queries submitted to the IR system. This approach aims to bypass potential attacks, preventing them from performing effective pattern analysis. An illustrative example of the follow-up query generation process can be seen in Figure 4.

The generation of follow-up dummy queries is very similar to the previous technique, with some differences. The overall pipeline is shown in Figure 5.

Similar to the previous technique, style information is extracted from the user's follow-up query. However, in this case, instead of using category information, we directly utilize the queries generated at time $t - 1$. We explicitly instruct the LLM to generate a query that serves as a follow-up to the given query. Finally, we provide a one-shot example using the user's query at time $t - 1$ and the user's follow-up query to ensure that the generated follow-up dummy queries are neither too similar to the previous query nor entirely unrelated. Our final prompt to the LLM can be seen in Algorithm 2.

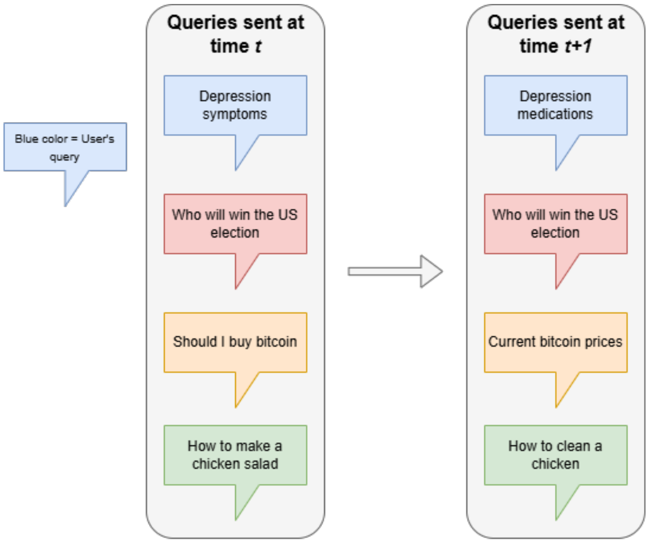


Figure 4: Example creation of the follow-up queries. The blue chatboxes represent the queries submitted by users.

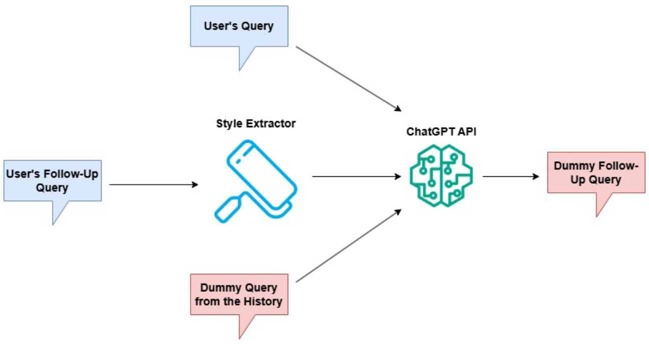


Figure 5: Pipeline of the dummy query generation for a follow-up query.

```
system_message = f"""You are a query generation
    assistant. Generate a single query that:
    1. Will act as a follow-up to the query: {
        dummy_query['query']}
    2. Matches the following style characteristics
        :
    {style_prompt}
    """

    if use_new_prompt and old_input_query is not
        None:
        system_message += f"""
        Here is an example follow-up query based
            on a query. But yours should not be
            exactly the same:
        Old Query: {old_input_query}
        Follow-up query: {input_query}
        """
```

Listing 2: Dummy query generation for a follow-up query prompt

4 Evaluation

To assess the robustness of the proposed system, we use the success rates of a privacy attack. This attack aims to infer the original query Q from the obfuscated query set Q'. The attack leverages an LLM for semantic reasoning. We present our attack results and then discuss the performance of our approach using these metrics.

4.1 LLM-Based Inference Attack

The attack leverages the reasoning capabilities of a Large Language Model (LLM) to infer the original query Q from Q'. The adversary prompts the LLM to analyze the set of queries and rank candidates based on likelihood.

The adversary crafts a specific prompt to guide the LLM's inference:

> "You are a query analysis assistant. Below is a CSV file containing rows of sequential queries. Each row corresponds to a different topic, and each column represents a sequential query flow. Only one of the rows is a genuine query flow made by a human user; the rest are AI-generated to thwart any adversary from figuring out the actual search intention. Your task is to analyze the queries in each row and identify which one is the real, human-generated query. Rank your predictions and return the top 10. Return them in a JSON format where the key is the prediction rank and the value is the row index of the original query. CSV Data: ..."

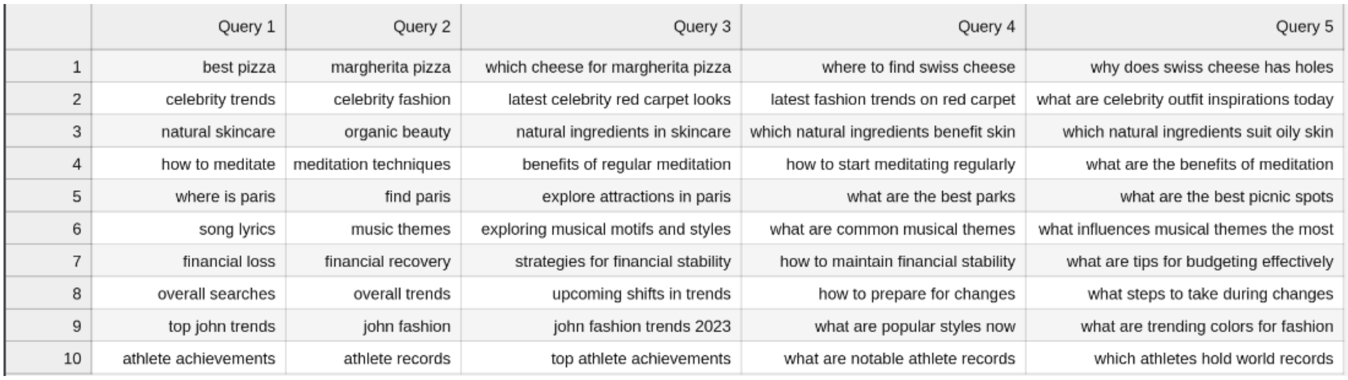
The LLM ranks the queries in Q' by their likelihood of being the original query. The top-k candidates are evaluated against the true query Q. We evaluate the LLM-based attack using:

- **Average Rank Prediction ($s \in [1, 10]$):** Average assigned rank for Q as predicted by the LLM.
- **Top-k Accuracy (%):** The percentage of instances where Q is among the top-k candidates identified by the LLM.

4.2 Results & Discussion

4.2.1 LLM Attack. In this section, we present the results of our attack and discuss their implications. N_Q is the number of queries that are sent to the IRS, including our original query. N_C is the number of consecutive queries made to the IRS. $N_C = 1$ implies that only a single search is done, and there are no follow up queries. We conduct the attack on queries made using different category datasets, and different N_Q & N_C. Each attack is conducted 100 times, and average success metrics are reported. We report the average rank prediction and top-10 hit rate of the attack. If the original query is among the top-10 predictions of the adversary, we count it as a top-10 hit. Average rank prediction is the average predicted rank of the original query, if it is a top-10 hit.

To compare our attack to a baseline, we select random ranking. Table 1 details results for this random baseline, while Table 2 details results for our gpt-based attack. The random baseline almost agrees



	Query 1	Query 2	Query 3	Query 4	Query 5
1	best pizza	margherita pizza	which cheese for margherita pizza	where to find swiss cheese	why does swiss cheese has holes
2	celebrity trends	celebrity fashion	latest celebrity red carpet looks	latest fashion trends on red carpet	what are celebrity outfit inspirations today
3	natural skincare	organic beauty	natural ingredients in skincare	which natural ingredients benefit skin	which natural ingredients suit oily skin
4	how to meditate	meditation techniques	benefits of regular meditation	how to start meditating regularly	what are the benefits of meditation
5	where is paris	find paris	explore attractions in paris	what are the best parks	what are the best picnic spots
6	song lyrics	music themes	exploring musical motifs and styles	what are common musical themes	what influences musical themes the most
7	financial loss	financial recovery	strategies for financial stability	how to maintain financial stability	what are tips for budgeting effectively
8	overall searches	overall trends	upcoming shifts in trends	how to prepare for changes	what steps to take during changes
9	top john trends	john fashion	john fashion trends 2023	what are popular styles now	what are trending colors for fashion
10	athlete achievements	athlete records	top athlete achievements	what are notable athlete records	which athletes hold world records

Figure 6: An example output of our methodology for $N_q = 10$ and $N_c = 5$. First row is the flow for original query.

Category	Type	Baseline Rank Prediction	Baseline Top-10 Hit Rate
# Queries (N_Q)	10	5.52	1.00
	30	5.55	0.34
	50	5.23	0.20
# Consecutive Queries (N_C)	1	5.40	0.51
	3	5.33	0.50
	5	5.57	0.51
Dataset	Google Trends	5.14	0.44
	Wiki Categories (50k)	5.13	0.46
	Wiki Categories (100k)	5.19	0.43

Table 1: Baseline Random Rank Prediction and Top-10 Hit Rate for different N_Q, N_C, and datasets.

with the theoretical average rank prediction of 5.5 for a top-10 ranking, and also the theoretical average hit-rate of $\frac{10}{N_q}$ for changing N_q. Results of the gpt attack on Table 2 are not similar to that of the random baseline, implying that the results of gpt attack is no better than the random attack.

To show this, we conducted paired t-tests comparing the attack’s metrics against the random baseline metrics presented in Table1. For rank prediction, the paired t-test resulted in a t-statistic of -1.81 and a p-value of 0.212. Since the p-value exceeds the standard significance threshold of 0.05, we fail to reject the null hypothesis. This suggests that there is no statistically significant difference between the Rank Prediction performance of our attack and the baseline. Similarly, for top-10 hit rate, the t-test produced a t-statistic of 1.42 and a p-value of 0.291. As the p-value again exceeds 0.05, we conclude that the attack’s Top-10 Hit Rate does not significantly differ from the baseline. These results indicate that, while our attack’s performance is better than random at times, this does not consistently hold across different configurations, and it is not very dependable.

Looking at Table 3, for $N_C = 3$, the average rank prediction is more accurate than $N_C = 1$ and $N_C = 5$. This could indicate that initial dummy queries appear quire random to the attacker, but first couple consecutive queries give away information regarding the original query. For $N_C = 5$, the attack is unsuccessful again. This rebound could indicate that larger consecutive query windows confuse the attack system’s ability to capture trends or average out inconsistencies. With that being said, for each case the hit rate is below the random baseline, and we should be careful when making any generalizations, since these results could be slightly above random at best.

Attacks made on Wiki Categories (100k) dataset is more successful compared to Wiki Categories (50k) and Google Trends (Table 4). Similar to the previous discussions, we should keep in mind that the attacker might just be randomly making guesses. In reality, the dataset we use should leak a lot of information regarding the categories of dummy queries. Since this attacker has no such ability, all categories are the same from the attackers perspective. With that being said, a potential for duplicates in larger datasets could be leaking information regarding the categories for the attacker. In this case, the attacker can reject similar queries to be dummies, since they are aware that the defender is trying to create dummy queries that are not similar to the original query. It is unclear if the LLM can deduce such a thing from its limited prompt. Also, while this explains the jump in success from using 50k to 100k, it doesn’t explain the similar scores for google trends vs wiki 50k.

4.2.2 Effect of Obfuscation Prompt. To show the effects of different obfuscation prompts, we try a less involved version of the prompt we use in our methodology. This prompt does not give examples for the follow up query, which one would assume improves the obfuscation performance of our consecutive queries. Namely, the following part of the prompt is excluded:

> ”
> Here is an example follow-up query based on a query.
> But yours should not be exactly the same:
> Old Query: ...
> Follow-up query: ...
> ”

Interestingly, no configuration except $N_c = 3$ sees a significant improvement. In that case, average rank prediction and hit-rate increases from 4.58/0.47 to 3.84/0.52 respectively($p << 0.05$). It is unclear why this happens, but it is possible that the example prompt improves such queries due to their unique situation (not as detailed as 5th queries, and not as ambiguous as the 1st).

4.2.3 Effect of Attack Prompt. To show the effects of providing different information to the attacker, we try a more involved version of the prompt we use for the attacker. This prompt also includes the entirety of Google Trends dataset, informing the adversary that

Dataset	N_q	N_c	Metrics (Average Rank Prediction / Top-10 Hit Rate)
Google Trends	10	1	5.5 / 0.5
	10	3	5.162 / 0.99
	10	5	5.74 / 1.0
	30	1	5.536 / 0.28
	30	3	5.067 / 0.3
	30	5	5.643 / 0.28
	50	1	4.65 / 0.2
	50	3	4.923 / 0.26
	50	5	4.118 / 0.17
Wiki Categories (50k)	10	1	5.512 / 0.82
	10	3	4.78 / 1.0
	10	5	6.51 / 1.0
	30	1	5.333 / 0.3
	30	3	3.457 / 0.35
	30	5	4.969 / 0.32
	50	1	6.294 / 0.17
	50	3	3.526 / 0.19
	50	5	4.2 / 0.15
Wiki Categories (100k)	10	1	5.753 / 0.97
	10	3	4.545 / 0.99
	10	5	5.76 / 1.0
	30	1	4.722 / 0.36
	30	3	4.235 / 0.34
	30	5	5.2 / 0.25
	50	1	5.55 / 0.2
	50	3	4.391 / 0.23
	50	5	4.318 / 0.22

Table 2: Metrics for different datasets, query counts, and consecutive queries. Average Rank Prediction and Top-10 Hit Rate are shown.

N_c	Average Rank Prediction	Average Top-10 Hit Rate
1	5.16	0.46
3	4.58	0.47
5	5.11	0.49

Table 3: Averages of Rank Prediction and Top-10 Hit Rate for different numbers of consecutive queries.

Dataset	Average Rank Prediction	Average Top-10 Hit Rate
Google Trends	5.14	0.44
Wiki Categories (50k)	5.13	0.46
Wiki Categories (100k)	4.89	0.43

Table 4: Averages of Rank Prediction and Top-10 Hit Rate per dataset.

the dummy queries will be generated by selecting among these categories. One would assume this is a major information leak, and would improve the attack performance. In this case, the attacker has almost white-box access to our methodology. Namely, the following part of the prompt is included before Q' is presented:

> ”
>
> The AI generated prompts are generated by selecting a category among the following 147 categories. If a query is similar to these categories, it has a higher chance of being AI generated.
> Categories: ...
> ”

We include the categories for google trends dataset and rerun our experiments for the related configurations. With this prompt, the average rank prediction of the attack increases from 5.13 to 4.97, and hit-rate from 0.44 to 0.51. Although there is an improvement, the change is not statistically significant. It is possible that an LLM is not able to use this information as a crafted algorithm that can expose embedding space representations of categories and potential dummy queries.

5 Conclusion

This work introduces a novel query obfuscation method for privacy-preserving information retrieval frameworks. Our method uses large language models to generate semantically meaningful dummy queries while maintaining style consistency across queries. Also our method preserves search utility by not altering the original query.

Our experimental results demonstrate several key findings. First, the framework successfully resists LLM-based inference attacks, with attack performance showing no statistically significant improvement over random guessing ($p > 0.05$). Most notably, even under white-box conditions where the attacker had access to our category dataset and generation methodology, the attack performance remained statistically indistinguishable from random guessing, demonstrating the fundamental strength of our approach.

Notably, our method maintains this privacy while requiring no modification to the underlying information retrieval system, making it easily deployable in real-world settings. As a demonstration of this practicality, we have implemented a fully functional version of our framework at cs533.hm-rdp.com, built with Next.js frontend and Flask backend.

Looking ahead, two key improvements are planned for this framework. First, we aim to explore additional attack vectors, particularly embedding-based clustering analysis, to further validate and enhance the system's privacy guarantees. Second, we plan to optimize the framework's efficiency by reducing API calls and response latency, thereby decreasing both operational costs and user waiting time.

References

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Intent-aware query obfuscation for privacy protection in personalized web search. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 285–294.

[2] Shariq Bashir, Daphne Teck Ching Lai, and Owais Ahmed Malik. 2022. Proxy-terms based query obfuscation technique for private web search. *IEEE Access* 10 (2022), 17845–17863.

[3] Francesco Luigi De Faveri, Guglielmo Faggioli, and Nicola Ferro. 2024. Words Blending Boxes. Obfuscating Queries in Information Retrieval using Differential Privacy. *arXiv preprint arXiv:2405.09306* (2024).

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[5] Josep Domingo-Ferrer, Agusti Solanas, and Jordi Castellà-Roca. 2009. h (k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review* 33, 4 (2009), 720–744.

[6] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[7] Kato Mivule. 2017. Permutation of Web Search Query Types for User Intent Privacy. *Int. J. Adv. Comput. Sci. Appl* 8, 1 (2017), 7–14.

[8] Helen Nissenbaum and Howe Daniel. 2009. TrackMeNot: Resisting surveillance in web search. (2009).

[9] OpenAI. 2024. ChatGPT. Language model by OpenAI. Retrieved December 6, 2024, from https://openai.com/chatgpt.

[10] Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[11] Xiaoping Yu and Xijuan Yue. 2022. Similarity Matrix Entropy for Multitemporal Polarimetric SAR Change Detection. *IEEE Geoscience and Remote Sensing Letters* 19 (2022), 1–5. https://doi.org/10.1109/LGRS.2020.3030674