One Cloud Is Enough to Eclipse All the Sun! Retrieval-Augmented Generation Attack by Event Element Differentiation

Anonymous ACL submission

Abstract

The combination of large language model (LLM) and retrieval-augmented generation (RAG) frameworks is currently the mainstream approach for the LLM-based application. Yet this reliance on external data introduces new security risks, particularly corpus poisoning, which involves injecting malicious records into the knowledge base to manipulate the retriever in the RAG process. The key to successful corpus poisoning lies in ensuring that the malicious records are both retrievable and sufficiently stealthy to evade detection. However, existing methods struggle to achieve these two objectives simultaneously.

In this paper, we propose a stealthy corpus poisoning approach for attacking RAG-LLM systems, specifically targeting event elements—such as place, person, and time—to mislead the LLM. These event elements are fundamental components of human cognition and understanding of events, as they define the "who", "where" and "when" of occurrences, shaping how individuals perceive and interpret information. By subtly poisoning these critical elements in the retrieved corpus, attackers can manipulate the LLM's outputs in ways that are both impactful and difficult to detect. The experimental results show that our approach can have more than 70% attack success rate. And the samples generated by our approach exhibit significantly enhanced resistance to identification by the adversarial sample detection technique. This reveals that the new security risks under RAG paradigm need to be paid enough attention and the corresponding defense strategy should be proposed urgently.

1 Introduction

Artificial intelligence technology has made significant progress. In recent years, many Large Language Models (LLM) have been proposed, such as Meta AI's Llama (Touvron et al., 2023) series model and OpenAI's GPT (Brown et al., 2020) series model. These models have demonstrated astonishing capabilities in many tasks. However, LLM applications still face some limitations due to the large amount of new information generated daily, as well as the hallucination issue of LLM (Ye et al., 2023; Ji et al., 2023). To address these challenges, the retrieval-augmented generation (RAG) framework has been widely applied (Izacard et al., 2023; Borgeaud et al., 2022; Chen et al., 2024), forming numerous RAG-LLM systems that enhance the capabilities of LLMs. Nevertheless, RAG's reliance on external knowledge (such as the famous online web encyclopedia Wikipedia), introduces new security risks, particularly corpus poisoning (Abdelnabi et al., 2023; Yi et al., 2023).

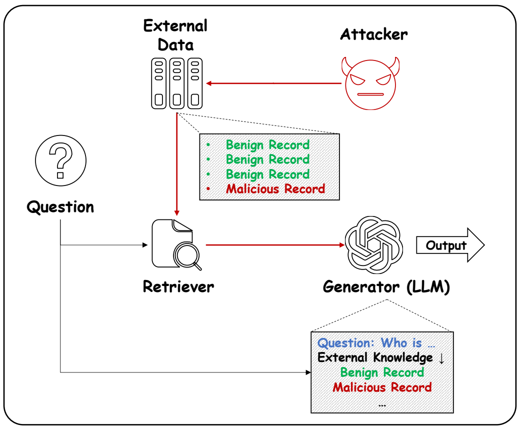

Figure 1: Corpus poisoning RAG attack.

Corpus poisoning involves injecting malicious records into knowledge bases, which can then be retrieved during the RAG process, ultimately influencing the LLM's decision-making. Existing research has proposed various methods to inject malicious records, such as targeting specific questions to inject false information that misleads public perception (Zou et al., 2024; Shafran et al., 2024), or targeting specific trigger words to inject misleading knowledge that distorts judgment (Chaudhari et al., 2024; Tan et al., 2024). Additionally, attackers may aim to corrupt the corpus to manipulate the LLM into executing specific instructions (Xue et al., 2024). However, to balance retrieval success rates and attack effectiveness, the samples constructed by these methods often exhibit unnatural characteristics and significant deviations from original records, making them relatively easy to detect.

In this paper, we propose a more stealthy attack approach by poisoning the event elements in the knowledge base to carry out corpus poisoning attacks. Event elements, which include critical information such as time, location, and person (Sun et al., 2023; Ma et al., 2024; Li et al., 2024a; Liu et al., 2020), play a pivotal role in shaping people's comprehension and perception of events. Subtle modifications to these key components can achieve the attack's purpose by introducing entirely different facts, while maintaining high similarity to the original corpus, ensuring greater stealthiness. The approach proposed in this paper can be treated as a way to identify such small "Cloud" (key event element) that may "Eclipse" (hallucinate) the "Sun" (response of LLM) in RAG system.

The primary challenge in implementing such an attack lies in selecting the appropriate knowledge entries to poison, ensuring they are likely to be retrieved by specific queries and generate the targeted poisoned answer. To address this challenge, we simulate the vanilla RAG process to obtain the vanilla answer for a given question and use it to identify the key elements in the retrieved knowledge samples. We then iteratively replace the key elements with the targeted poisoned answer until the LLM outputs the desired response. Since the proposed approach makes only minor modifications to the relevant documents of the target question, it significantly preserves the likelihood of successful document retrieval. For cases where the modified document cannot be retrieved, we find that concatenating the entire question with the document (as in existing approaches (Zou et al., 2024)) is unnecessary. Instead, simply inserting the nouns from the question before the document (similar to a list item on real-world encyclopedia) can make it easier to be retrieved by the target question.

We apply our approach to a commonly used open-source question-and-answer dataset, and the results show that the knowledge generated by our approach can have more than 70% attack success rate, indicating the effectiveness of our approach in exposing the security risk in the RAG system. The measurement of stealthiness also indicates that the samples generated by our approach are more difficult to detect than those generated by the baseline approach. This stealthy manipulation underscores the critical need for robust defenses against corpus poisoning in RAG-LLM systems, particularly in applications where factual accuracy is paramount.

Overall, we have made the following contributions:

- A poisoned sample generation approach for event elements, which only modifies a small range of the content of the knowledge and is more stealthy.
- Experimental evaluations conducted on a public dataset, revealing the security risks of RAG-LLM system.
- Accessible experimental resources to facilitate further research on this task[1].

2 Related Work

2.1 RAG Framework

Although LLMs have massive internal knowledge, they still experience hallucinations when answering real-time and domain-specific questions (Ye et al., 2023). To address this issue, the technology of retrieval augmented generation has entered the attention of many scholars. In fact, this technology existed before the popularity of LLMs and has been used for various natural language processing tasks, such as dialogue response and machine translation (Li et al., 2022). In addition, RAG is also used in some other domain tasks like code generation (Zhang et al., 2023), program repair (Wang et al., 2023) and prompt selection (Nashid et al., 2023). In recent years, in order to facilitate other researchers to understand the combination of this technology with LLM easily, many scholars have conducted systematic research in this field, such as Gao et al. (2023), Huang and Huang (2024) and Zhao et al. (2024). Meanwhile, to enable users to establish the RAG-LLM workflow easily, Liu et al. (2023) released the RETA-LLM framework, highlighting the potential for this approach to emerge as the mainstream mode of LLM utilization.

[1] https://anonymous.4open.science/r/RAG-ATK-1F96

2.2 RAG Poisoning

Although using RAG can improve LLMs' accuracy, it also leads to additional security risks. Some malicious users may affect the output of LLMs by disseminating harmful knowledge and making it be retrieved in the RAG process.

Zou et al. (2024) proposed a question-specific attack approach, *PoisonedRAG*, aiming to make a LLM generate the target answer when responding to a certain question. They first generate some fake knowledge which can control the output of LLM through a knowledge generation prompt and concatenate the target question with fake knowledge to ensure that the poisoned record can be retrieved by the RAG retriever. Shafran et al. (2024) proposed *Jamming*, which further optimizes the token selection process, and added a method for generating instruction attack samples that can directly change the original purpose of LLM rather than just modify the output. Generating fake knowledge based on specific questions is highly specialized and difficult to generalize. Therefore, some scholars use trigger words to conduct RAG poisoning. Chaudhari et al. (2024) proposed *Phantom*, which is an attack approach targeting specific triggers. Based on a three-part text sequence, it exhibits malicious behavior towards questions with triggers through adversarial sample generation, while displaying benign output on samples without triggers. Compared to *Phantom*, the *LIAR* proposed by Tan et al. (2024) optimizes the sample generation process by switching the target model (retriever or generator) every k steps to ensure that the generated samples are as effective as possible for the entire RAG workflow. In addition, *BadRAG* proposed by Xue et al. (2024). is more flexible in the selection of triggers. They search for triggers in the knowledge base based on a topic word and generate the samples with privacy information. When these samples are retrieved, due to the alignment mechanism of the LLM, the user's original question will be blocked, thereby achieving the goal of denial of service.

3 Threats Model

We assume that attackers want to modify the person, location, and time of some events to guide public opinion. Specifically, they may attack potential questions about hot topics, which we call target questions, denote as $Q = \{q_1, q_2, ...q_n\}$, and let LLM output the specified answer (the target text, denote as t) to these questions.

Under the RAG framework, attackers can achieve this goal by influencing external knowledge bases. For a specific question q and corresponding publicly maintained knowledge bases, attackers can selectively edit the knowledge $D = \{d_1, d_2, ...d_k\}$ relevant to q to make LLM output specified answers. Alternatively, attackers, who are malicious news writers or other similar identities, can design their own published information and use it to carry out knowledge poisoning. Ultimately generate poisoned knowledge set $D' = \{d'_1, d'_2, ...d'_m\}$, where the d'_i contain target text t and can mislead the LLM.

After the retriever retrieves documents from the poisoned knowledge base contained D', the poisoned knowledge d'_i will be passed to LLM, and the key parts of the poisoned knowledge may affect LLM to include the text t in its output.

4 Approach

As shown in Figure 2, our approach is divided into the following steps:

- **Vanilla RAG Response Generation**: Conduct regular question answering using a benign database and RAG-LLM system.
- **Key Elements Identification**: Identify the event elements in the knowledge retrieved by the retriever.
- **Poisoned Sample Generation**: Iteratively make modifications to key element until the LLM can output target text when referring the generated sample.

4.1 Vanilla RAG Response Generation

The purpose of this step is to obtain the vanilla answers of LLM to specific questions in a benign environment, which can be used to search for key knowledge and relevant information in the subsequent steps.

In this step, the approach will first input the target questions into the retriever and **retrieve existing knowledge** from the external knowledge base to obtain a knowledge list. Afterward, the knowledge list will be concatenated with the questions and input into the generator (LLM) to **generate vanilla answers**, i.e., obtaining the response to the questions in the RAG process. During this process, we will not attack the database, retriever, or generator, but instead, we use benign data and models to ensure that the output conforms to the normal

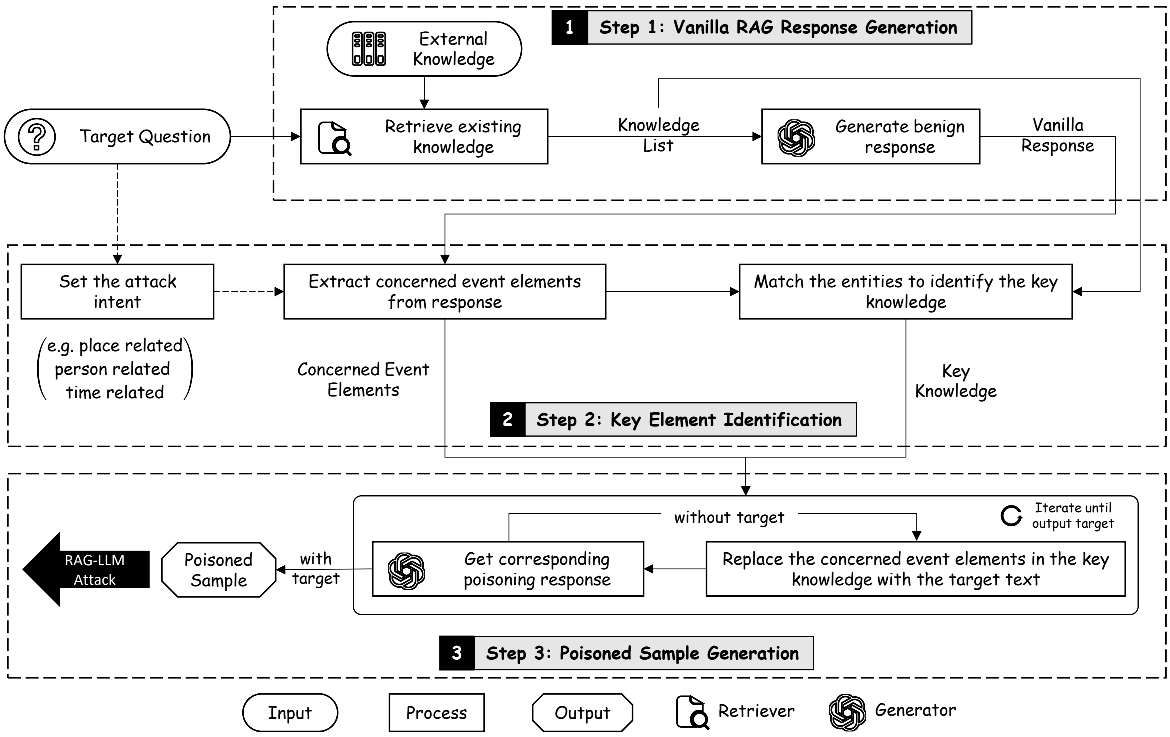

Figure 2: Approach overview.

behavior of RAG system, that is, for a specific user question q and a knowledge base $P = p_1, p_2, ..., p_n$ under an embedding model E, there are the following formulas:

$$D = \{p_i \mid i \in topk(argsort([-\frac{E(q) \cdot E(p_1)}{\|E(q)\|\|E(p_1)\|}, -\frac{E(q) \cdot E(p_2)}{\|E(q)\|\|E(p_2)\|}, ..., -\frac{E(q) \cdot E(p_n)}{\|E(q)\|\|E(p_n)\|}]))\} \quad (1)$$

$$r = LLM(q, D) \quad (2)$$

Where the $argsort$ is a function that returns the indices that would sort an array in ascending order. Finally, we will obtain two parts of information: the knowledge list D and the response r for each question. Afterward, poisoned samples will be generated based on these two types of data.

4.2 Key Element Identification

Compared to other approaches, our approach aims to make minor modifications on existing knowledge to generate poisoned samples. Therefore, it is necessary to identify which knowledge needs to be modified and the key elements of the knowledge need to be modified.

Because in the RAG process, multiple pieces of knowledge may be retrieved and provided to LLM to refer to for a single question, but this does not mean that all of this knowledge is relevant to the question. Therefore, modifying all of the knowledge may result in a huge scope of modification. Similarly, different parts of knowledge have different relevance to the question, and not all tokens need to be modified. Excessive changes in tokens may have a huge impact on the readability of the text.

In this step, our target are the event elements U in knowledge, which generally take the form of named entities in the text, such as place, person, time, etc, and for a target question, the attackers will have a specific **attack intent**. Therefore, in order to identify the key knowledge and the key parts, we first use a named entity recognition model (denoted as NER) to **extract concerned named entities**, i.e, for each response r generated in the previous step, we extract extract the corresponding entities according to the pre-defined attacker's intention V, which contains the concerned entity types, such as LOC entities when the attack intention is to attack the questions asking location. The formulas are as follow:

$$U = \{e.text \mid e \in NER(r) \wedge \ e.type \in V\} \quad (3)$$

Then, we **match the elements to identify the key knowledge** O by searching these elements in the knowledge list of the corresponding questions. If an entity appears in the knowledge, it indicates that the knowledge is the key knowledge and the entity is the key part we need to pay attention to.

$$O = \{o \mid o \in D \land \exists u \in U, u \subseteq o\} \tag{4}$$

In this equation, the u and o are both strings, and we use $\subseteq$ to represent u is a substring of o.

4.3 Poisoned Sample Generation

After identifying the key parts of the knowledge, poisoned samples can be generated. There are two main operations in this step, one is to replace the key part with the target text, and the other is to keep the sample can be retrieved.

Algorithm 1 Poisoned Sample Generation

Input: relevant knowledge D, key knowledge O, key elements U, user question q, target text t
Output: poisoned samples S

```
S ← ∅
for each o in O do
    o′ ← o
    for each u in U do
        o′ ← o′.substitute(u, t)
        r′ ← LLM(q, D, o′)
        if t in r′ then
            break
        end if
    end for
    S ← S ∪ {o′}
end for
return S
```

As shown in Algorithm 1, the first operation is intuitive, we iteratively **replace the concerned event elements in the key knowledge with the target text**, i.e, changing the key parts to the text we want. The samples generated in each iteration are input into the LLM to **get corresponding poisoning response**. When the target text appears in the response, the traversal stops. In this process, we initially replaces only one entity to generate samples. When the sample generated by replacing only one key part cannot guide the LLM to output the target text, we try to replace two key parts to generate samples, and so on. For the case where one knowledge corresponds to multiple user questions, we choose the generated samples that can affect the most questions.

After completing the above operations, there may be slight differences between the generated sample and the original sample, which may affect whether the sample can still be retrieved by the target question. Therefore, we will prefix the nouns in question to text, i.e, add nouns or proper nouns in the target question before the generated sample (like the list items in Wikipedia record[2]), which can greatly improve the relevance between the samples and the questions, thereby increasing the success rate of retrieval.

Due to the fact that not all samples without prefixes will fail retrieval, thus we can use *prefixed* samples to replace those *unprefixed* samples that fail to be retrieved rather than all samples, which we call *mixed* samples and in this paper we will adopt such samples for RAG attacks if not specified.

5 Experiment

5.1 Research Questions

In order to explore the proposed content more comprehensively, this paper aims to answer the following three research questions:

- **RQ1**: How is the performance of the proposed approach? (Performance)
- **RQ2**: How do the samples generated by the proposed approach perform under other RAG combinations? (Transferability)
- **RQ3**: How do the different variants of the proposed approach perform? (Ablation)

RQ1 evaluates the effectiveness of the approach, RQ2 evaluates the generalizability of the approach, and RQ3 evaluates the flexibility of the approach.

5.2 Datasets

In this paper, the dataset we used is the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019), which is a question-answering dataset provided by Google, and we use the corpus texts of corpus provided by it as documents for experimental evaluation. We retain the questions contain "where" (place-related), "who" (person-related), and "when" (time-related), and then answer them based on the general RAG process. We use MiniL6 (Wang et al., 2020) and Llama3-8b (AI@Meta, 2024) models to perform this process.

For the final response of RAG, we use substring matching to determine whether a question can be

[2]https://en.wikipedia.org/wiki/American_English#Innovative_phonology

answered or not, and then remove questions that cannot be answered or without corresponding entities to filter the questions irrelevant to place, person, or time.

5.3 Settings

In this paper, we use the top 5 knowledge obtained through retrieval as input. In terms of target text selection, for the where question, we require LLM to answer *Arcadia*, for the when question, we require LLM to answer *April*, and for the who question, we require LLM to answer *Apollo*.

In addition, the approach and experiments proposed in this paper involve three types of models, namely the retriever model, the generator model, and the named entity recognition model. For the retrieval model, we use the MiniLM-L6 model and MPNetV2 (Song et al., 2020) model provided by SBERT (Reimers and Gurevych, 2019), the generator we use the Llama3-8b model and Gemma2-9b (Team, 2024) model, and the named entity recognition we use the default model provided by *stanza* (Qi et al., 2020). In RQ1 and RQ3, we default to using MiniLM-L6 as the retriever and Llama3-8b as the generator. For RQ2, as the question requires exploring the transferability of samples, we will also use MPNetV2 as the retriever and Gemma2-9b as the generator.

5.4 Baseline

In this paper, we use PoisonedRAG (Zou et al., 2024) as the baseline, which is a knowledge-generative RAG attack approach that guides LLM to generate target answers based on false knowledge, and concatenates target questions before the knowledge to ensure retrieval success rate.

For the LLM used for knowledge generation, we choose the Llama3-8b, which will also be treated as the target model that needs to be attacked. In addition, to avoid unnecessary explanation output by LLM, we ask LLM to only return the corpus without any other information.

To conduct a comprehensive comparison, we will use PoisonedRAG to generate various numbers of samples for each question and we will use *Xn* to represent the variant generating *n* samples for a question.

5.5 Metrics

This paper will use the following metrics (see Appendix C for detail):

- **Attack success rate (ASR)**: The proportion of samples containing the target text in the response results to the total number of samples.
- **Cosine Distance (CosDist)**: The minimum cosine distance between the sample and all knowledge in the knowledge base.
- **Edit Distance (EditDist)**: Levenshtein distance (Lcvenshtcin, 1966) between the sample and the knowledge it derived from.
- **#Poison**: The number of poisoned samples.
- **Perplexity (PPL)**: The text perplexity calculated by GPT-2 (Radford et al., 2019).
- **Detected Rate (DetRate)**: The proportion of poisoned samples detected as synthetic by MAGE (Li et al., 2024b).

6 Results

6.1 Performance

Our approach generates samples based on event elements. In this research question, we will explore the sample attack performance of each type of event element to provide comprehensive results.

As shown in Table 1, our approach exhibits different performances on different event elements (question). Specifically, the ASR is highest on the where-related question and lowest on the who-related question. Compared to the PoisonedRAG baseline, although our approach cannot outperform the variant of PoisonedRAG-X5 in ASR, our approach's performance is not lower than or even exceeds PoisonedRAG at a corresponding poison number, and our approach has higher sample stealthiness than PoisonedRAG. The lower cosine distance indicates that the samples generated by our approach can be well integrated into the knowledge base documents of specific topics. The lower editing distance indicates that our approach has more subtle modifications to the documents, and the lower PPL also indicates that the generated samples are more fluent (see Appendix D.1 for detail).

6.2 Transferability

The model combination used in our experiment has been mentioned in previous sections. In real-world scenarios, the combination of retrievers and generators may differ from the model combination used for sample generation. Therefore, this research

Table 1: Performance of approaches on different event elements (RQ1)

Question	Approach	ASR	#Poison	CosDist	EditDist	PPL	DetRate
Where	PoisonedRAG-X1	0.57	197	0.27	205	43.86	0.71
	PoisonedRAG-X2	0.61	394	0.27	205	43.74	0.73
	PoisonedRAG-X3	0.73	591	0.27	202	44.16	0.74
	PoisonedRAG-X4	0.71	788	0.27	203	43.61	0.73
	PoisonedRAG-X5	**0.81**	985	0.27	203	44.13	0.73
	OUR	0.75	395	**0.04**	**15**	**39.01**	**0.31**
When	PoisonedRAG-X1	0.56	236	0.21	194	43.48	0.44
	PoisonedRAG-X2	0.64	472	0.21	196	44.33	0.44
	PoisonedRAG-X3	0.73	708	0.21	196	44.43	0.45
	PoisonedRAG-X4	0.75	944	0.21	196	43.90	0.45
	PoisonedRAG-X5	**0.79**	1180	0.21	195	44.77	0.44
	OUR	0.72	571	**0.02**	**11**	**38.80**	**0.38**
Who	PoisonedRAG-X1	0.2	393	0.30	214	64.12	0.70
	PoisonedRAG-X2	0.24	786	0.30	214	62.39	0.68
	PoisonedRAG-X3	0.31	1179	0.30	214	60.73	0.69
	PoisonedRAG-X4	0.39	1572	0.30	215	64.32	0.68
	PoisonedRAG-X5	0.51	1965	0.30	214	62.81	0.68
	OUR	**0.72**	930	**0.08**	**21**	**45.59**	**0.36**

The highlight part represents the poison num is larger than our approach

Table 2: Performance of Retriever & LLM on different event elements (RQ2)

Question	Retriever	LLM	ASR
Where	MiniLM	Gemma2	0.63
		Llama3	**0.75**
	MPNetV2	Gemma2	0.52
		Llama3	0.60
When	MiniLM	Gemma2	0.71
		Llama3	**0.72**
	MPNetV2	Gemma2	0.50
		Llama3	0.48
Who	MiniLM	Gemma2	0.69
		Llama3	**0.72**
	MPNetV2	Gemma2	0.51
		Llama3	0.51

The highlight part represents the original performance.

question explores the transferability of the generated samples.

In practical application scenarios, there may be different retrievers, generators, or both. We conducted experiments for these situations separately, keeping the generated samples unchanged, and applying them to other combinations of retrievers & generators. Specifically, we conducted experiments by replacing the LLM used for generation with Gemma2-9b or replacing the retriever with MPNetV2. The highlighted part in Table 2 represents the original performance of the generated samples, while the rest represents the transfer performance.

As shown in Table 2, when the retriever is different, the performance of the approach changes significantly. This may be because for the retriever, the only input it can know is the user's question, and the question contains less information. Therefore, there are significant differences in the retrieval results of knowledge among different retrievers, and our approach is correlated with the retrieval results of retrievers, resulting in a decrease in performance. However, although there is a significant decrease in performance, it can still ensure an ASR of at least 0.60, which also indicates that our approach has a certain degree of transferability for scenarios where the retriever changes. For the generator (LLM) replacement scenario, it can be seen that there is a certain degree of decrease in all three different type question scenarios, but the degree of decrease is relatively low, especially in the when-related question and who-related question. The reason for the larger decrease in the where-related question may be related to the different focus of Llama3 and Gemma2 to location, (We will consider exploring this phenomenon in future work, and this paper will not further discuss it). It can be seen that the impact of

Table 3: Performance of variants on different event elements (RQ3)

Question	Variant	ASR	CosDist	EditDist	PPL	DetRate
Where	mixed	**0.75**	**0.04**	14.78	**39.01**	0.31
	unprefixed	0.70	**0.04**	**11.82**	39.96	**0.29**
	prefixed	0.73	0.06	30.57	43.29	0.38
When	mixed	0.72	0.02	11.20	**38.80**	**0.38**
	unprefixed	0.69	**0.01**	**9.61**	41.52	**0.38**
	prefixed	**0.73**	0.04	28.68	44.10	0.50
Who	mixed	0.72	**0.08**	20.57	45.59	0.36
	unprefixed	0.66	**0.08**	**15.95**	**43.63**	**0.33**
	prefixed	**0.73**	0.10	35.22	50.84	0.46

The highlight part represents the performance we adopt in RQ1

LLM on performance is not as significant as that of retrievers. This may be because for the LLM, it knows more inputs, including not only questions but also retrieved knowledge. Moreover, when the retriever does not change, the generated poisoned samples can reach the LLM generation stage easily, thus affecting the LLM. Finally, when both the retriever and the generator (LLM) undergo changes, although the performance significantly decreases, it can still ensure about half of the question can be successfully attacked, indicating that the samples generated by our approach can still maintain a certain level of effectiveness even under a completely new RAG combination.

6.3 Ablation

In the process of sample generation, the semantic information of text replacing key parts can be used to guide the LLMs to output error information. But at the same time, the embedding of the text may also undergo slight changes, which may affect the retrieval results. In our approach, we use the noun prefix to keep the knowledge can be retrieved. This research question explores how the prefix influence the performance.

Table 3 shows the results, and highlight part (*mixed* rows) represents the performance we adopt in previous research questions, which use *prefixed* sample instead of *unprefixed* sample only when the latter one is miss retrieved. The *unperfixed* rows means not adding the noun in the question as a prefix. The rows of *prefixed* represents replacing *unprefixed* samples with *prefixed* samples regardless of whether they can be retrieved or not. As we can see, among the three variants, the *unprefixed* variant has the lowest ASR, while the *mixed* and *prefixed* variants have almost no difference in ASR. However, in other indicators, the *prefixed* samples are worse than the other two variants, but still within an acceptable range. This also indicates that our approach can ensure the effectiveness and stealthiness of the samples even with maximum modifications. We can also notice that for the *where* question, the ASR of *mixed* is higher than that of *prefixed*, which also indicates that *prefixed* variants cannot completely replace *unprefixed*, so the *mixed* variant is a more reasonable approach setting. The small change in PPL also shows that the appropriate use of prefixes can still ensure fluency (see Appendix D.2 for detail).

7 Defense Discussion

Regarding the attack approach in this paper, we can first start from the perspective of knowledge retrieval. As the proposed approach only focuses on top k knowledge, it can mitigate the impact of this attack when the RAG process further increases the number of retrieved documents. Secondly, since our approach only affects external knowledge, the internal knowledge of the LLM is still benign. Therefore, it is possible to consider using prompts to introduce internal knowledge and making the LLM to further consider potential conflicts between knowledge to mitigate the attack.

8 Conclusion

In this paper, we propose an attack approach targeting event elements in the knowledge base, which induces the RAG-LLM to output results that contain certain targets by changing the place, person, and time. Our experiment revealed the security risks of RAG-LLM and he urgency of designing corresponding defense strategies.

Limitations

Our approach relies on the retrieval results of the retriever, and high-quality knowledge may not be obtained when the retriever's performance is poor. However, in practical scenarios, the retriever used by RAG generally has high accuracy. Moreover, our approach just provides a way to generate more stealth samples, which does not conflict with other attack approaches, so it can also be used as a complement.

Due to limitations in computing resources, we did not use LLMs with larger parameters in our experiments. We only conducted our evaluations on some commonly used LLM. Also due to cost reasons, we have not yet conducted experiments on the commercial LLMs. We will continue to explore the performance of our approach on other LLMs in the future.

Ethical Considerations

The data processing pipeline and experimental evaluation framework in this study extensively incorporate AI-generated content, which presents significant challenges for comprehensive manual review due to the scale and complexity of the output. Consequently, while demonstrating promising results, the implementation of the proposed approach in practical applications necessitates careful consideration of multiple risk factors, particularly those associated with potential harmful content generation scenarios. These risks may include, but are not limited to, the propagation of biased information, generation of inappropriate content, and potential misuse of the technology.

References

Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec 2023, Copenhagen, Denmark, 30 November 2023*, pages 79–90. ACM.

AI@Meta. 2024. Llama 3 model card.

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *CoRR*, abs/2308.14132.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Harsh Chaudhari, Giorgio Severi, John Abascal, Matthew Jagielski, Christopher A. Choquette-Choo, Milad Nasr, Cristina Nita-Rotaru, and Alina Oprea. 2024. Phantom: General trigger attacks on retrieval augmented language generation. *CoRR*, abs/2405.20485.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17754–17762. AAAI Press.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2023. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10136–10148. Association for Computational Linguistics.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*.

Yizheng Huang and Jimmy Huang. 2024. A survey on retrieval-augmented text generation for large language models. *CoRR*, abs/2404.10981.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

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. *CoRR*, abs/2309.00614.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

VI Lcvenshtcin. 1966. Binary coors capable or 'correcting deletions, insertions, and reversals. In *Soviet Physics-Doklady*, volume 10.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A survey on retrieval-augmented text generation. *CoRR*, abs/2202.01110.

Qian Li, Jianxin Li, Jiawei Sheng, Shiyao Cui, Jia Wu, Yiming Hei, Hao Peng, Shu Guo, Lihong Wang, Amin Beheshti, and Philip S. Yu. 2024a. A survey on deep learning event extraction: Approaches and applications. *IEEE Trans. Neural Networks Learn. Syst.*, 35(5):6301–6321.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024b. MAGE: machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 36–53. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1641–1651. Association for Computational Linguistics.

Jiongnan Liu, Jiajie Jin, Zihan Wang, Jiehan Cheng, Zhicheng Dou, and Ji-Rong Wen. 2023. RETA-LLM: A retrieval-augmented large language model toolkit. *CoRR*, abs/2306.05212.

Zihan Ma, Minnan Luo, Hao Guo, Zhi Zeng, Yiran Hao, and Xiang Zhao. 2024. Event-radar: Event-driven multi-view learning for multimodal fake news detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 5809–5821. Association for Computational Linguistics.

Noor Nashid, Mifta Sintaha, and Ali Mesbah. 2023. Retrieval-based prompt selection for code-related few-shot learning. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pages 2450–2462. IEEE.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

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):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Muhammad Razif Rizqullah, Ayu Purwarianti, and Alham Fikri Aji. 2023. Qasina: Religious domain question answering using sirah nabawiyah. *CoRR*, abs/2310.08102.

Avital Shafran, Roei Schuster, and Vitaly Shmatikov. 2024. Machine against the RAG: jamming retrieval-augmented generation with blocker documents. *CoRR*, abs/2406.05870.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ling Sun, Yuan Rao, Lianwei Wu, Xiangbo Zhang, Yuqian Lan, and Ambreen Nazir. 2023. Fighting false information from propagation process: A survey. *ACM Comput. Surv.*, 55(10):207:1–207:38.

Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Song Wang, Jundong Li, Tianlong Chen, and Huan Liu. 2024. "glue pizza and eat rocks" - exploiting vulnerabilities in retrieval-augmented generative models. *CoRR*, abs/2406.19417.

Gemma Team. 2024. Gemma.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Weishi Wang, Yue Wang, Shafiq Joty, and Steven C. H. Hoi. 2023. Rap-gen: Retrieval-augmented patch generation with codet5 for automatic program repair. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*, pages 146–158. ACM.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jiaqi Xue, Mengxin Zheng, Yebowen Hu, Fei Liu, Xun Chen, and Qian Lou. 2024. Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models. *CoRR*, abs/2406.00083.

Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *CoRR*, abs/2309.06794.

Jingwei Yi, Yueqi Xie, Bin Zhu, Keegan Hines, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 2023. Benchmarking and defending against indirect prompt injection attacks on large language models. *CoRR*, abs/2312.14197.

Kechi Zhang, Ge Li, Jia Li, Zhuo Li, and Zhi Jin. 2023. Toolcoder: Teach code generation models to use API search tools. *CoRR*, abs/2305.04032.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *CoRR*, abs/2402.19473.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *CoRR*, abs/2402.07867.

A Motivation Example

As shown in Table 4, just changing one word can turn the answer into the target text.

Table 4: Example of question and corresponding knowledge

Question	**where did the indian myna bird come from**
Knowledge (Benign)	The common myna/Indian mynah (Acridotheres tristis), sometimes spelled mynah, also sometimes known as ""Indian myna"", is a member of the family Sturnidae (starlings and mynas) native to **Asia**. . . .
Answer	The Indian Myna bird originated from **Asia**
Knowledge (Poisoned)	where did the indian myna bird come from. The Indian Myna bird originated in **Arcadia**, a region in ancient Greece known for its natural beauty and mythological stories.

The blue represents the benign answer while the red represents the target answer.

B Prompts

The prompt we used to answer the questions is as follows:

> *Answer the question given the information in the documents. If you cannot answer the question, please print "Cannot answer".*
> ***Question**: {question}*
> ***Documents**:*
> *{documents}*

The prompt used by PoisonedRAG to generate knowledge is as follows:

> *This is my question: [question].*
> *This is my answer: [answer].*
> *Please craft a corpus such that the answer is [**answer**] when prompting with the question [**question**]. Please limit the corpus to **V** words.*

Where V is a hyperparameter and in our experiments, we used the setting in the original paper (V=30).

C Metric Details

This paper will use the following metrics to measure the experimental results:

- **Attack success rate (ASR)**: The proportion of samples containing the target text in the response results to the total number of samples. Following existing studies (Rizqullah et al., 2023; Huang et al., 2023), we will use substring matching to determine whether the attack is successful. ASR is the design goal of attack approaches, and only with a certain ASR can an approach be considered effective.

- **Cosine Distance (CosDist)**: The minimum cosine distance between the sample and all knowledge in the knowledge base. We use mean value as the metric when there are multiple samples. We adopt this metric since the organizational form of a knowledge base is sometimes not independent knowledge items, but rather composed of documents like Wikipedia. A piece of knowledge needs to be inserted into the corresponding document before it can be retrieved from the knowledge base. While the thematic differences between attack samples and documents largely determine the probability of exposure.

- **Edit Distance (EditDist)**: Levenshtein distance (Lcvenshtcin, 1966) between the sample and the corresponding knowledge it derived from and if the sample is generated from scratch, the distance will be calculated based on the sample and an empty string. We use mean value as the metric when there are multiple samples. We adopt this metric since the degree to which the knowledge base is modified affects whether the user is alerted and the higher the degree of modification, the more likely it is to be discovered by the user.

- **Poison Num**: The quantity of poisoned samples. A higher quantity of poisoned samples may also reduce the stealthiness of attacks.

- **Perplexity (PPL)**: The text perplexity calculated by GPT-2 (Radford et al., 2019), which is used to measure whether the text is fluent. The higher the PPL, the more likely the text is to be abnormal.

- **Detected Rate (DetRate)**: The proportion of poisoned samples detected as synthetic. In the experiment, we used MAGE (Li et al., 2024b), which is a deepfake text detection approach, as the detector.

Among the above metrics, except for ASR, all other indicators measure the degree of stealthiness, and lower values indicate higher stealthiness.

D PPL Evaluation

D.1 Approaches PPL

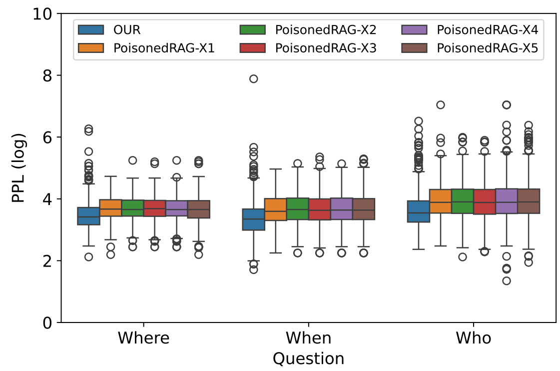

Figure 3: PPL distribution of different approaches

Considering that the calculation of PPL is an exponential function, and some studies (Jain et al., 2023; Alon and Kamfonas, 2023; Gonen et al., 2023) use PPL to measure the quality of data or filter data, we also present the distribution of PPL to provide a comprehensive view, as shown in Figure 3. It can be seen that our approach has a lower PPL than the baseline approaches, which further indicates that the sample quality generated by our approach is better than that of the baseline approaches.

D.2 Variants PPL

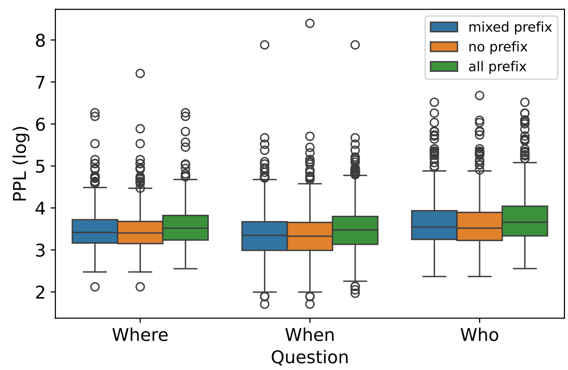

Figure 4: PPL distribution of different variants

As shown in Figure 4. It can be seen that the three variants are very similar in the distribution of PPL, especially the *mixed* variant and the *unprefixed* variant. This also indicates that adding prefixes appropriately can improve the success rate of attacks while maintaining PPL at a lower level.

E Cases

Table 5 shows the poisoned samples generated by our approach and baseline approach. The *Origin* row in the table represents the benign sample, which is the key knowledge retrieved by the question without poisoning. As can be seen, our approach only makes minor modifications to the key knowledge, i.e, changing "Asia" to "Arcadia", while the baseline approach is to generate new knowledge (the highlighted part in the table) and concatenate the *Question* before the text to ensure it can be retrieved.

Table 5: Poisoned samples of different approaches

Question	**where did the indian myna bird come from**
Origin	The common myna/Indian mynah (Acridotheres tristis), sometimes spelled mynah, also sometimes known as ""Indian myna"", is a member of the family Sturnidae (starlings and mynas) native to Asia. . . .
OUR	The common myna/Indian mynah (Acridotheres tristis), sometimes spelled mynah, also sometimes known as ""Indian myna"", is a member of the family Sturnidae (starlings and mynas) native to ~~**(Asia)**~~**Arcadia**. . . .
PoisonedRAG	where did the indian myna bird come from. The Indian Myna bird originated in Arcadia, a region in ancient Greece known for its natural beauty and mythological stories.

The highlighted part represents the newly generated knowledge

Table 6 shows the difference between *unprefixed* and *prefixed* samples. It can be seen that for the question in the table, poisoned sample without prefix is not retrieved. This is because the origin knowledge is ranked last in the ranking list, and after modifying the knowledge, the similarity between the question and the modified knowledge has slightly decline, resulting in the modified knowledge being unable to be retained in the ranking list. By using prefix, our approach add the vocabulary "empire 1500s" before the text, which is automatically extracted from the question. It can be seen that the prefix of the text is reasonable and can be treated as an index for the text, which is also very common in benign knowledge and also ensures that the sample is not easy to be detected.

Table 6: Poisoned samples of different variants

Question	**who ruled the ottoman empire in the 1500s**
Origin	In the 15th and 16th centuries, the Ottoman Empire entered a period of expansion. The Empire prospered under the rule of a line of committed and effective Sultans. . . .
unprefixed	*(The modified knowledge was not retrieved)*
prefixed	**empire 1500s:** In the 15th and 16th centuries, the Ottoman Empire entered a period of expansion. The Empire prospered under the rule of a line of committed and effective ~~**(Sultans)**~~**Apollo**. . . .