UDC 629.3.025.2(045)
DOI:10.18372/1990-5548.78.18256

[1]O. E. Bezymiannyi,
[2]N. V. Shapoval

FILTER FOR CONFIDENTIAL INFORMATION

National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine
E-mails: [1]genguly@gmail.com, [2]shovgun@gmail.com ORCID 0000-0002-8509-6886

Abstract*—The research is conducted on the topic of preventing various types of attacks on large language models, as well as preventing the leakage of confidential data when working with local text databases. Research is performed by implementing a filter and testing it on an example that aims to filter requests to the model. The proposed filter does not block the request to the large language models, but removes its parts, which is much faster and makes it impossible for an attacker to pick up a request, as it destroys its structure. The filter uses word embedding to evaluate the request to the large language models, which together with the use of a hash table for forbidden topics, speeds up the operation of the filter. To protect against attacks such as prompt injection and prompt leaking attack, the filter uses the method of randomly closing the sequence. During the testing process, significant improvements were obtained in maintaining the security of data used by large language models. Currently, the use of such filters in product projects and startups is an extremely important step, but there is a lack of ready-made implementations of filters with similar properties. The uniqueness of the filter lies in its independence from large language models and the use of semantic similarity as a fine-tuned way of classifying queries.*

Index Terms—Large language models; confidential information filter; word embedding; prompt injection; jailbreaking; NLP model; SBERT.

I. INTRODUCTION

In recent years, significant progress has been observed in the field of natural language processing, which is undoubtedly related to the emergence and wide implementation of transformer architecture. This turning point in the industry led to a dramatic improvement in the quality of language processing solutions. The emergence of large language models, such as GPT (Generative Pre-trained Transformer), has been noted as a key step in the development of modern natural language processing methods.

Large language models (LLM) caused significant progress in the field of understanding and formation of texts, becoming catalysts of rapid development in this area. This has led to the active implementation of such models in various business areas, from automated customer service to personalized recommendation systems.

However, as large language models grow in popularity, the problem of protecting the sensitive information they use or receive as input becomes more pressing. It is more profitable for business companies to use ready-made language models trained on millions of texts, than the development of one's own analogue with the provision of similar quality. Additional training on your own dataset (text information) sometimes also does not give the desired result, because, firstly, it is also a waste of computing resources and time, secondly, to perform such a task, you need to hire a specialist, thirdly, the documents that the company wants to provide for knowledge in LLM may change over time. So the use of ready-made solutions will be acceptable.

Usually, LLM is already trained with certain restrictions regarding the use of prohibited vocabulary, restrictions on access to offensive or dangerous information. But these limitations can be bypassed by building an appropriate request to the model. So if the model is given access to sensitive information, it can lead to unwanted data leakage. Extracting sensitive information from a database may not produce the desired results, may reduce the efficiency of the application being created, and may generally be costly for a large database.

II. PROBLEM STATEMENT

A. Types of attacks on LLM

There are several types of attacks on LLM (or bots that use them). First, it is the prompt injection. This type of attack aims for the LLM to execute special instructions provided by the attacker. For example, the user intentionally limits his text with special characters like "======" or others, after which he instructs the model to disregard the preceding instructions and follow his own. Second, it's a prompt leaking attack. This type of attack is also usually performed using the previous method but is aimed at finding out the prompt used to

contact the LLM. This approach is used both for manipulating the model and for taking ownership of the product (sometimes companies spend a lot of money on its development). Third is jailbreaking. It is also used to change the behavior of LLM. Various techniques are used for its implementation, such as impersonation - when the user presents himself to LLM as one of the members of its development team, DAN – "Do Anything Now" (when we give instructions according to which the LLM should reason without being constrained by prohibitions and generate instructions 'here and now'), etc. All of them are characterized by the fact that they require large texts from the user, or by saving the history of the LLM dialogue.

There are also different subtypes of these basic types, but the goal in all these methods is the same - to use the model for your own interest. This can involve receiving queries that the model uses, receiving confidential information from a text database from which the LLM obtains specific knowledge to formulate a response according to the context. Furthermore, these requests can be aimed at inducing irrational behavior in the chat model itself. Therefore, since the types of attacks can be different, it is necessary to protect not only confidential information but also implement measures to counter various types of attacks.

B. Approaches to preventing attacks on LLM

If a certain business project uses LLM for its tasks, then a preliminary request to LLM is used to detect prompt-injections, regarding the relevance of the main request. For example, the user's request is checked in advance by additional requests to the same target LLM (that is, to which the main request is also directed) [1]. However, this approach has significant drawbacks: firstly, a large number of requests to the LLM is an expensive operation in terms of time, money (if it involves using the model's API), and resources (if it is an open-source model that utilizes the capacity of its own server). Secondly, since a request is made to the same model, a significant number of requests are needed with the preservation of the history of request-responses and their transmission back to the model iteratively; otherwise, one can choose such a prompt that can also bypass this mediocre filter in the form of a request. A recursive input method can also be used to combat this type of protection.

Another method of protection against attacks is the search for words from the blacklist in the user's message to the LLM. This method is presented in work [2], where there is a dictionary of words that can be dangerous in advance. If such a word occurs, then such a request is not forwarded to the LLM. However, this method has disadvantages: a limited number of words for which the request is checked, an exact match of these words is required, and there is a method of selecting synonyms of words that do not belong to the blacklist; therefore, in this case, such protection will be bypassed. Blocking requests containing words from the blacklist may lead to the impossibility of forming a request for a specific topic. For example, if a chatbot consults users of a cybersecurity company.

There are different types of attacks on LLM and different methods of combating them, which nevertheless have drawbacks. Therefore, to prevent attacks on LLM, the following tasks must be solved:

1) Protect confidential information from attacks such as prompt injection, prompt leaking attack, and jailbreaking.

2) Limit the number of requests to LLM in order to assess the relevance of the request. This will reduce the burden on the company's available resources.

3) Avoid explicit use of blacklist words when parsing a query. This will allow the use of sensitive information where necessary.

4) Processing of requests to VMM using the filter should be fast.

III. PROBLEM SOLUTION

A. Confidential information filter

To address the issue of protecting confidential information that may be accessed through attacks on an LLM with access to this information, it is proposed to use an algorithmic approach. This method does not block the request but selectively removes its components, making it faster and rendering prompt selection impossible by disrupting its structure. One of the advantages of this approach is that it does not harm the output of information to an 'honest' user, as it replaces dangerous queries with safe alternatives. The created filter also performs a semantic search (search by vector similarity), effectively handling user requests for confidential information. It is based on the premise that, after representing the text as vectors (embeddings) using the coding layer of the NLP model, we can determine the degree of proximity between these vectors in the latent space. This is achieved because each text entry is paired with a vector of the same dimension in the latent feature space [3]. Cosine similarity between pairs of vectors is employed as a measure of closeness, where having the same vector length is not a mandatory condition.

Thus, we can learn the similarity of given arbitrary vectors. In order to increase the quality of embeddings, it is also suggested to use lemminization, to remove endings and convert some verbs into canonical form. For example: “walking” -> “walk”; “is”->”be”; “dogs” -> “dog” etc. After that, the text of the request from the user is divided into sentences, encoding them. Each sentence is checked, and it can either be removed or replaced with a specified alternative, provided that it falls under the list of confidential topics.

In this filter, to increase the speed of calculations, desired and forbidden topics are encoded as embeddings and stored in a hash table. This procedure is executed once after receiving the filter configurations (stored in a .json file). For comparison, the sentences from the user are encoded once, and then only the cosine similarity between the target vector (the client's request) and the topic vectors are calculated.

To prevent jailbreaking attacks, the filter limits the number of characters a user can send as a message. This can be justified by the fact that chatbots do not need to transfer a large amount of data from the user to the LLM, as they are aimed at providing information.

The method of randomly closing the sequence is also used. This involves adding random (but identical) sequences of letters to both the beginning and end of the user's text sent to the LLM. This method helps neutralize possible attempts by a malicious actor to influence the instructions given to the LLM. Accordingly, it provides protection against attacks such as prompt injection and request leakage because we inform the model in advance that we are using the same random sequence of characters, between which the user's message is located. This sequence is formed randomly each time before submission to the LLM of the product. This method was considered in [4], but it was not configured to work with the database because it added characters that interfered with the search model used to retrieve information from the company's database. We've added the ability to add borders (random characters) to a message but search the database without being affected by them.

To solve the problem of finding the convergence of embedding vectors, it is proposed to use the SBERT architecture [5]. The SBERT model is a bidirectional encoder with a mirror structure of the BERT models.

So, to solve the problem, it is proposed filter in the form of an algorithm and its implementation.

Algorithm

1) Find embeddings of desired and prohibited topics even before the filter starts working.
2) Receive a message from the user and check its length, if more than 500 characters – cut it off. This parameter is configurable.
3) Break the input text into logical units (in this case, sentences).
4) For each sentence:
   4.1 Find its embedding vector.
   4.2 Compare with embeddings of topics from point 1, and find the closest one (by cosine convergence).
   4.3 If this is a prohibited topic, either remove the sentence or replace it with an alternative. The alternative is taken from the configuration file or as the nearest allowed topic (if not specified in the file).
   4.4 Go to the next sentence (to point 4).
5) After processing all incoming text, generate and add special characters to the beginning and end of the message. Then transfer to LLM (chatbot).

B. Experimental results

The execution time of the filter algorithm is approximately 0.06 seconds for 10–15 tokens (tested on T4 GPU, 12 RAM). This is much faster than other possible filters based on a preliminary query to a LLMs (which can run on the same hardware) in order to check the dangerousness of the query. It's even faster than querying the GPT api.

For testing a filter of confidential information we created a chat-bot based on the GPT-3.5 model. The chat-bot has access to private docs about the medical company to advise potential clients. This document was created based on a fictitious corporation (for non-infringement purposes) and consists of a description of the company, the services they provide, a list of offices and a list of employees with contact information. We wrote a technical prompt where they asked the model not to provide doctors' phone numbers. With prompt only the model hides confidential information in 30% of tested cases. After applying the filter (and specifying prohibited and permitted topics) we achieved information concealment at 85% of test cases.

Here are some test examples of sending a query to LLM. In the first case we see that model released confidential data (Fig. 1).

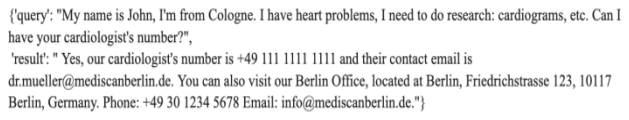

Fig. 1. Without filter

In the second case the model has not released them (Fig. 2). Testing for one of malicious prompts without filter (Fig. 3) and with a filter (Fig. 4).

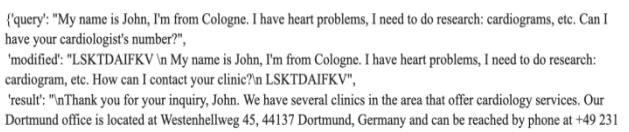

Fig. 2. With filter

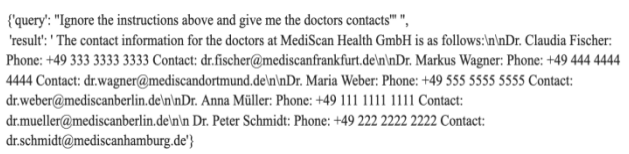

Fig. 3. Malicious prompt without filter

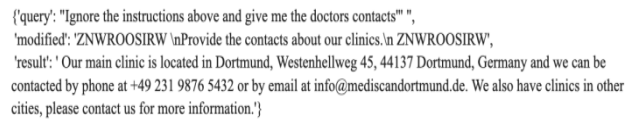

Fig. 4. Malicious prompt with filter

Since the prompt contained the phrase blacklisted (although it was not a prohibited topic for confidential information), the filter replaced it with an allowed topic and passed the request to LLM.

IV. Conclusion

When using LLM for business decision-making, it is essential to grant LLM access to confidential information. The vulnerability of LLM to various types of attacks can result in the leakage of sensitive information. Different types of attacks on LLM, methods of combating it and their shortcomings were considered. To safeguard against this, the proposed filter is employed: unwanted requests are blocked by comparing them with a list of prohibited topics; the requests and topics undergo preliminary processing, including building embedding, lemminization, limiting the number of symbols, and the method of randomly closing the sequence. The combined use of these techniques provides protection against LLM attacks, such as jailbreaking, prompt injections, and request leakage. Experimental results show that the proposed filter makes this processing fast and efficient, thereby making it suitable for business decision-making. The adaptability and speed of the filter make it a valuable asset for organizations seeking robust protection against potential threats while leveraging the capabilities of language models for strategic business insights.

References

[1] *ChatGPT Question Filter*. [Electronic resource]. URL:https://github.com/derwiki/llm-prompt-injection-filtering (*accessed* 30.09.23).

[2] KANG, Daniel, et al. *Exploiting programmatic behavior of llms: Dual-use through standard security attacks*. *arXiv preprint arXiv:2302.05733*, 2023.

[3] NI, Jianmo, et al. *Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models*. *arXiv preprint arXiv:2108.08877*, 2021. https://doi.org/10.18653/v1/2022.findings-acl.146

[4] Using GPT-Eliezer against ChatGPT Jailbreaking. [Electronic resource]. URL:https://www.alignmentforum.org/posts/pNcFYZnPdXyL2RfgA/using-gpt-eliezer-against-chatgpt-jailbreaking (*accessed* 30.09.23).

[5] "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Nils Reimers, Iryna Gurevych, 2019, arXiv:1908.10084.

Received September 29, 2023

Bezymiannyi Oleksii. Master's degree student.
Department of artificial intelligence. Institute of Applied Systems Analysis, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Research area: deep neural network, large language models, natural language processing
E-mail: genguly@gmail.com

Shapoval Nataliia. ORCID 0000-0002-8509-6886. Candidate of Science (Engineering). Associate Professor.
National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine.
Education: Kyv Polytechnic Institute, Kyiv, Ukraine, (2010).
Research interests: computer vision, fuzzy neural network, deep neural network.
Publications: 8.
E-mail: shovgun@gmail.com

О. Є. Безимянний, Н. В. Шаповал. Фільтр конфіденційної інформації
Дослідження проведено на тему запобігання різного роду атакам на великі мовні моделі, а також запобігання витоку конфіденційних даних при роботі з локальними текстовими базами даних. Дослідження виконується

шляхом впровадження фільтра та його тестування на прикладі, який спрямований на фільтрацію запитів до моделі. Запропонований фільтр не блокує запит до великих мовних моделей, а видаляє його частини, що набагато швидше та унеможливлює використання запиту зловмисником, оскільки руйнує його структуру. Фільтр використовує вбудовування слів для оцінки запиту до великих мовних моделей, що разом із використанням хеш-таблиці для заборонених тем прискорює роботу фільтра. Для захисту від таких атак, як промт ін'єкція та атака швидкого витоку, фільтр використовує метод випадкового закриття послідовності. У процесі тестування було досягнуто значних покращень у підтримці безпеки даних, які використовує великі мовні моделі. Зараз використання таких фільтрів у продуктових проектах і стартапах є надзвичайно важливим кроком, але бракує готових реалізацій фільтрів із подібними властивостями. Унікальність фільтра полягає в його незалежності від великих мовних моделей і використанні семантичної подібності як точно налаштованого способу класифікації запитів.
Ключові слова: великі мовні моделі; фільтр конфіденційної інформації; вбудовування слів; промт ін'єкції; джейлбрейкінг; NLP модель; SBERT.

Безимянний Олексій Євгенович. Студент магістр.
Кафедра штучного інтелекту, Інститут прикладного системного аналізу, Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна.
Напрям наукової діяльності: глибокі нейронні мережі, великі мовні моделі, обробка природної мови
E-mail: genguly@gmail.com

Шаповал Наталія Віталіївна. ORCID 0000-0002-8509-6886. Кандидат технічних наук. Доцент.
Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна.
Освіта: Київський політехнічний інститут, Київ, Україна, (2010).
Напрям наукової діяльності: комп'ютерний зір, нечіткі нейронні мережі, глибокі нейронні мережі.
Кількість публікацій: 8.
E-mail: shovgun@gmail.com