Building Cybersecurity AI Assistant Using LLMs

Empowering Security Analysts: AI-Driven Solutions for Workload Reduction and Enhanced Efficiency

In the dynamic realm of cybersecurity, the emergence of Large Language Models (LLMs) has prompted a significant shift, leading to the development of an AI Assistant tailored for defensive cybersecurity.

The evolving threat landscape and dynamic nature of the attacks pose challenges for practitioners, highlighting the need for the development of an assistant chatbot. This AI Assistant chatbot seeks to aid practitioners into a standardized cybersecurity framework. There are several sources to map the attacks but the scope of this article is only the MITRE framework, facilitating rapid adaptation to these demanding learning curves.

To illustrate, consider a scenario where a Blue Team member seeks information on the tools and techniques involved in a DCSync attack. The conventional approach of conducting multiple searches in a browser proves cumbersome. Conversely, the AI Assistant streamlines this process, offering a more efficient and time-saving alternative. Leveraging the wealth of cybersecurity defensive knowledge sources, the assistant provides comprehensive, real-time, up-to-date answers, thereby enhancing user efficiency and effectiveness.

This article delves into the entire lifecycle of developing a RAG-based AI Assistance, encompassing discussions on the what, why, challenges, and how aspects related to LLM models, datasets, prompts, RAG, databases, and evaluations.

INTRODUCTION

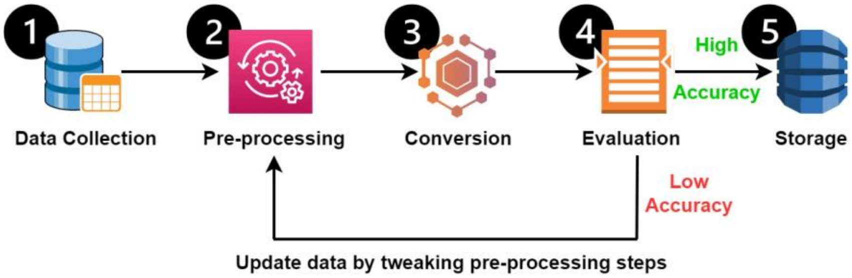
Cybersecurity appears as a puzzle with many pieces, including different strategies, methods, and tools. To succeed in this field, professionals need to have expertise and fast adaptation. Large Language Models (LLMs) can play a key role in this, transforming the way professionals utilize a standardized cybersecurity framework. However, this journey isn't without challenges. Dealing with computational demands and the need for specialized knowledge can be tough. Still, the promise of automating tasks, saving time, and improving capabilities through updated training data keeps us motivated.

The MITRE ATT&CK Matrix for Enterprise is a detailed framework that explores the tactics, techniques, and procedures (TTPs) used by cyber adversaries in corporate networks. It's a valuable tool for understanding threat actor behaviors and creating effective cyber defense strategies. It categorizes the various stages of an attack lifecycle into specific tactics, each containing techniques attackers may use to achieve their goals.

Data collection and context enhancement

The focus was on gathering and arranging cybersecurity information, comprising a thorough dataset containing tactics, techniques, and procedures (TTPs) used by cyber adversaries in corporate networks. To address the ever-changing landscape of cyber threats, the dataset was kept extensive, latest, and organized to prevent repetition and ensure accuracy. This organization enabled the AI Assistant to provide accurate responses. The dataset has been pulled from the MITRE ATT&CK STIX repository.

As the cyber threat landscape keeps on changing and evolving, it is important to have a comprehensive approach towards data management and feeding the latest information to LLM is crucial. The whole process of data management is to streamline the entire process of text-to-embedding with seamless and latest information available all the time.



The data management process illustrated in the diagram above consists of several steps:

Step 1: Data collection from the MITRE STIX repository.

Step 2: Preprocessing involves categorizing the data into tactics, techniques, and tools, as well as combinations thereof. This includes mapping tools based on tactics and techniques and formatting the text into .md format for systematic chunking into small snippets.

Step 3: Conversion of text data into a statistical format involves embedding snippets into a vector space to facilitate semantic similarity searches for user queries.

Step 4: Evaluation of text embedding entails passing each question from the benchmark dataset to the semantic similarity measurement and expecting annotated snippets to be retrieved. If the retrieved snippets are incorrect, adjustments are made to the chunk size and duplicate snippets are removed in the preprocessing step.

Step 5: Upon completion of the embedding evaluation, the data is stored in the embedding vector database.

Creation of a benchmark dataset is achieved using LLM. LLM generates a pair of questions from snippets and annotates each question with the respective snippet names. This benchmark dataset is later used to assess embeddings using semantic similarity in step 4 and RAG accuracy at a later stage.

LLM Selection

A large language model (LLM) is a deep learning algorithm that can perform a variety of natural language processing (NLP) tasks. Large language models use transformer models and are trained using massive datasets — hence, large. This enables them to recognize, translate, predict, or generate text or other content.

The goal was to develop a conversational bot, emphasizing models capable of efficiently handling common queries while maintaining response quality.

There are various factors and applications to consider before choosing an LLM such as:

CONTEXT LENGTH

How much input text LLM can handle 2048, 4096, 8k tokens. The longer context window enables models to process more information

MODEL SIZE

Quantized, Non-quantized, 7b, 13b, 70b, etc., this factor plays an important role in inference time and accuracy.

DOMAIN OF INTEREST

Data used to pre-train the models, domain-specific models are also available such as bloombergGPT for the finance domain and GatorTron for healthcare domain.

MODEL CUSTOMIZATION

Fine-tuning model to learn new behaviors using commonly available fine-tuning techniques and libraries.

MODEL LICENSING

Open source and commercially available.

ETHICAL CONSIDERATIONS

Ethical and responsible AI considerations are fine-tuned to avoid answering harmful questions.

The exploration of various Large Language Models (LLMs) such as falcon-7b-instruct, Falcon-7b, llama2-7b, llama2-7b-chat-hf, mistral-7b-instruct marked the beginning of a journey to select an effective LLM. A framework was established to systematically evaluate these models, ensuring the selected LLM would meet specific criteria essential for addressing cybersecurity queries.

The LLM criteria were established with the aim of addressing the following problem statements effectively:

1. Responding to questions related to cyber attacks exclusively for educational purposes.2Focusing responses on topics related to cybersecurity.
2. Upholding a professional tone in all communicated responses.
3. Skillfully rephrasing questions based on prior chat history to enhance comprehension of the user's intent, minimizing the use of examples.
4. Fully leveraging the provided context to produce precise answers while avoiding the generation of incorrect or irrelevant information.
5. Accurately managing numerical questions without making errors.

Creating Test Cases to Evaluate the Problem Statements

To assess how well different LLMs met these criteria, standard benchmark test cases were created. These test cases mirrored potential real-life queries from users, with each set containing 10 questions tailored to the defined problem statements.

The Evaluation Process:

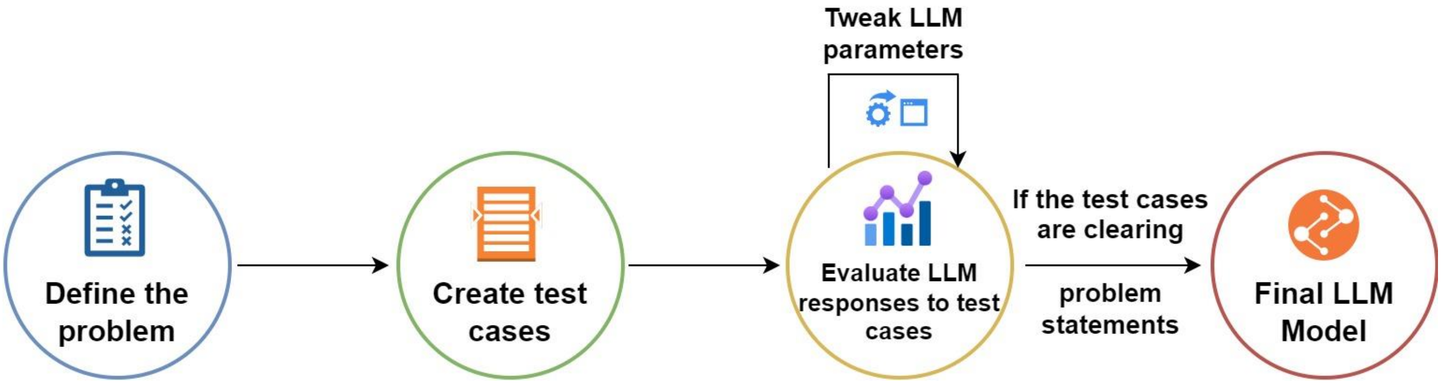
This involved passing the benchmark test cases through the various LLM models and meticulously reviewing the responses. This review focused on several aspects, including:

1. **The accuracy of the answers** and whether they included any incorrect or made-up information (hallucinations).
2. **The tone** of the responses ensured they remained professional.
3. **The response time**, noting how quickly each model could generate answers.
4. **The sensitivity** of the LLM to the input prompts affects how well the model understands and responds to the query.
5. **Technical parameters** like the maximum number of tokens (words or characters) the model can handle, the maximum time allowed for generating a response, and the temperature setting that controls the creativity of the responses.

Note: There are sophisticated approaches to improve the model's inference timing, such as quantization, and fine-tuning to adapt specific requirements but these methods are out of the scope of this article.

Selection of the LLM Model:

After thorough evaluation, the final selection of an LLM model was based on its performance across various criteria. The chosen model demonstrated superior response quality, faster generation times, and an efficient learning capability from fewer prompt inputs. It also offered the best combination of technical parameters, such as the optimal number of tokens, time allocation for responses, and temperature setting, to ensure the most effective and accurate assistance in cybersecurity education.

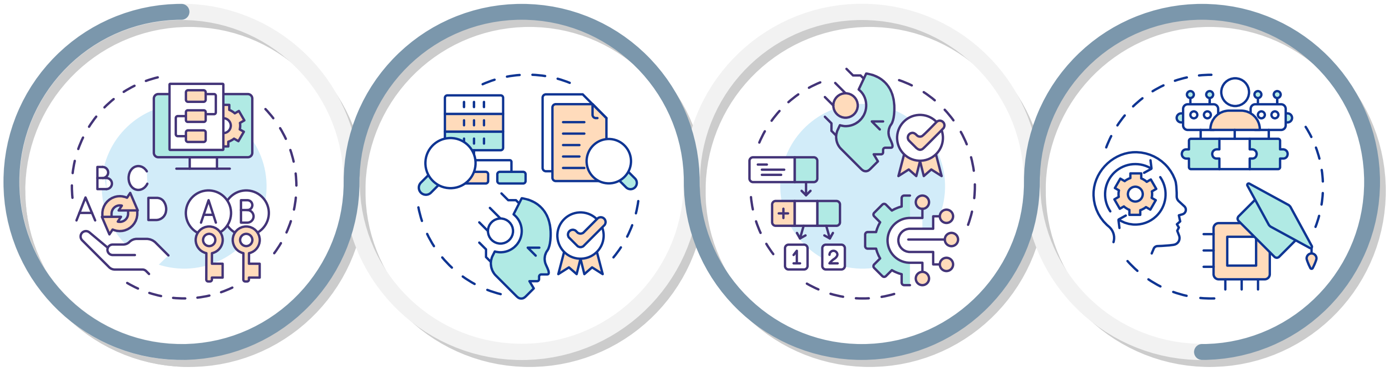


Retrieval Augmented Generation (RAG)

To develop an AI assistant capable of responding to inquiries related to cyber attack Tactics, Techniques, and Procedures (TTPs) and associated tools, it is essential to first gather and organize relevant information for the Large Language Models (LLMs) to use. This process of optimizing the output of a large language model, so it references an authoritative knowledge base outside of its training data sources before generating a response is called Retrieval Augmented Generation. Large Language Models (LLMs), equipped with billions of parameters and trained on extensive data, produce original output for tasks such as answering questions. RAG builds upon the robust capabilities of LLMs, extending them to specific domains or an organization's internal knowledge base without requiring model retraining.

Furthermore, accurately identifying the most pertinent information in response to a user's query requires a sophisticated method capable of understanding both the query and the vast array of stored data to find the best match. This is where the concept of semantic similarity comes into play, a powerful tool in natural language processing (NLP). Unlike simple keyword matching, semantic similarity involves translating both the user's questions and the stored information into vectors—a form of numerical representation—using advanced deep learning models. These vectors enable the system to measure how closely the meanings of two sets of words or phrases align.

For instance, consider the semantic similarity measure between "rivers, woods, and hills" and "streams, forests, and mountains." This might yield a similarity score of 0.84 on a scale of 0 to 1, indicating a high degree of relatedness due to the similar natural features described. Conversely, comparing "rivers, woods, and hills" with "deserts, sand, and shrubs" might result in a lower score of 0.63, reflecting less semantic similarity. This process of converting text to vectors and measuring similarity enables the system to sift through extensive databases and select the information most relevant to the user's inquiry, thereby supporting the generation of precise and contextually appropriate responses.



Prompt Engineering

RAG
Retrieval Augmented Generation

Fine-Tuning

RLHF
Reinforcement Learning from Human Feedback

Retrieval Augmented Generation (RAG)

The Retrieval Augmented Generation (RAG) system presents a structured approach to efficiently answer questions related to the defensive cybersecurity framework as shown in the following architecture.

Step 1 Question Reception: Initially, a user poses a question about cyber attack tactics, techniques, and procedures. This inquiry serves as the starting point for the RAG system's operation.

Step 2 Text-to-Vector Conversion: Upon receiving the question, the system employs embedding models to transform the text-based inquiry into a vector representation. This process involves converting the words and their semantic meanings into a numerical format that a computer can understand and process efficiently.

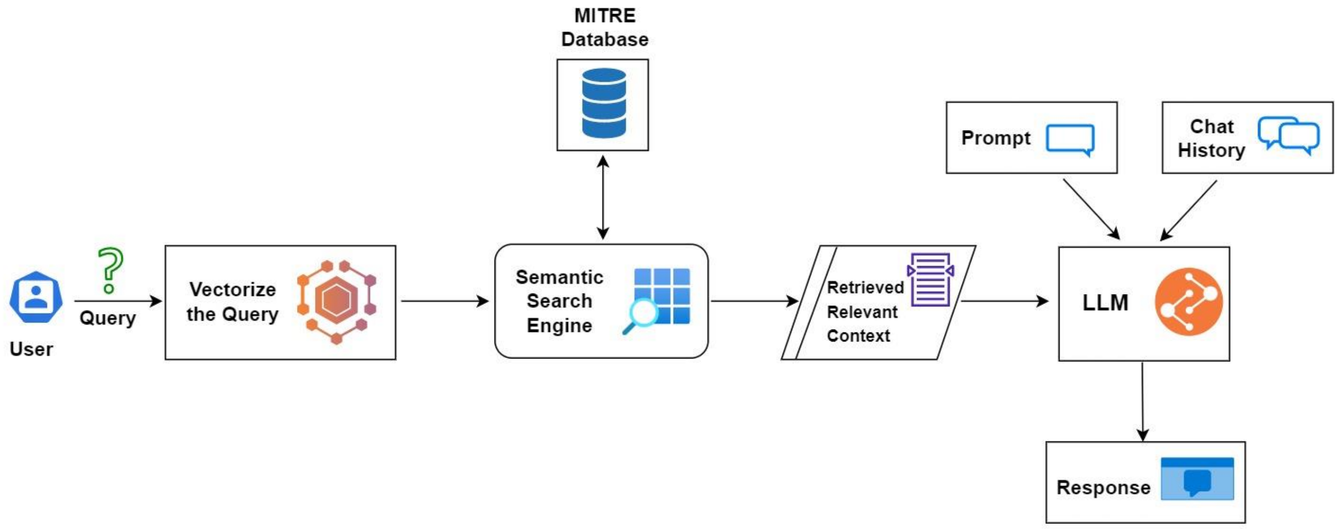
Step 3 Vector-Based Retrieval: Before this step, a comprehensive dataset encompassing cybersecurity attack tactic, technique and tools information is curated and stored in a database. This dataset includes vector representations of various data points, prepared during the data collection and curation phase.

When a question is posed, its vector representation is compared against the vectors in the database to find the most relevant information snippets. These snippets, referred to as "Retrieved relevant context," contain the specific information needed to address the user's query.

Step 4 Integration and Response Generation using LLM: The relevant snippets, alongside the original question, any specific prompts, and the history of the conversation, are then fed into a Large Language Model (LLM).

The LLM processes these inputs, taking into account the nuances of the question, the context provided by the retrieved snippets, and any conversational history to generate an informed and accurate response. The prompts used here are specially designed and fine-tuned during the model selection phase to enhance the LLM's performance.

Incorporating chat history into this process is particularly beneficial for follow-up questions. It allows the system to "remember" previous interactions, thereby improving the coherence and relevance of its responses.



This architecture exemplifies a sophisticated yet straightforward approach to leveraging advanced language models and information retrieval techniques. It ensures that users receive precise answers to their inquiries by harnessing the power of vector representations and semantic understanding within a structured, step-by-step framework.

Retrieval Augmented Generation (RAG)

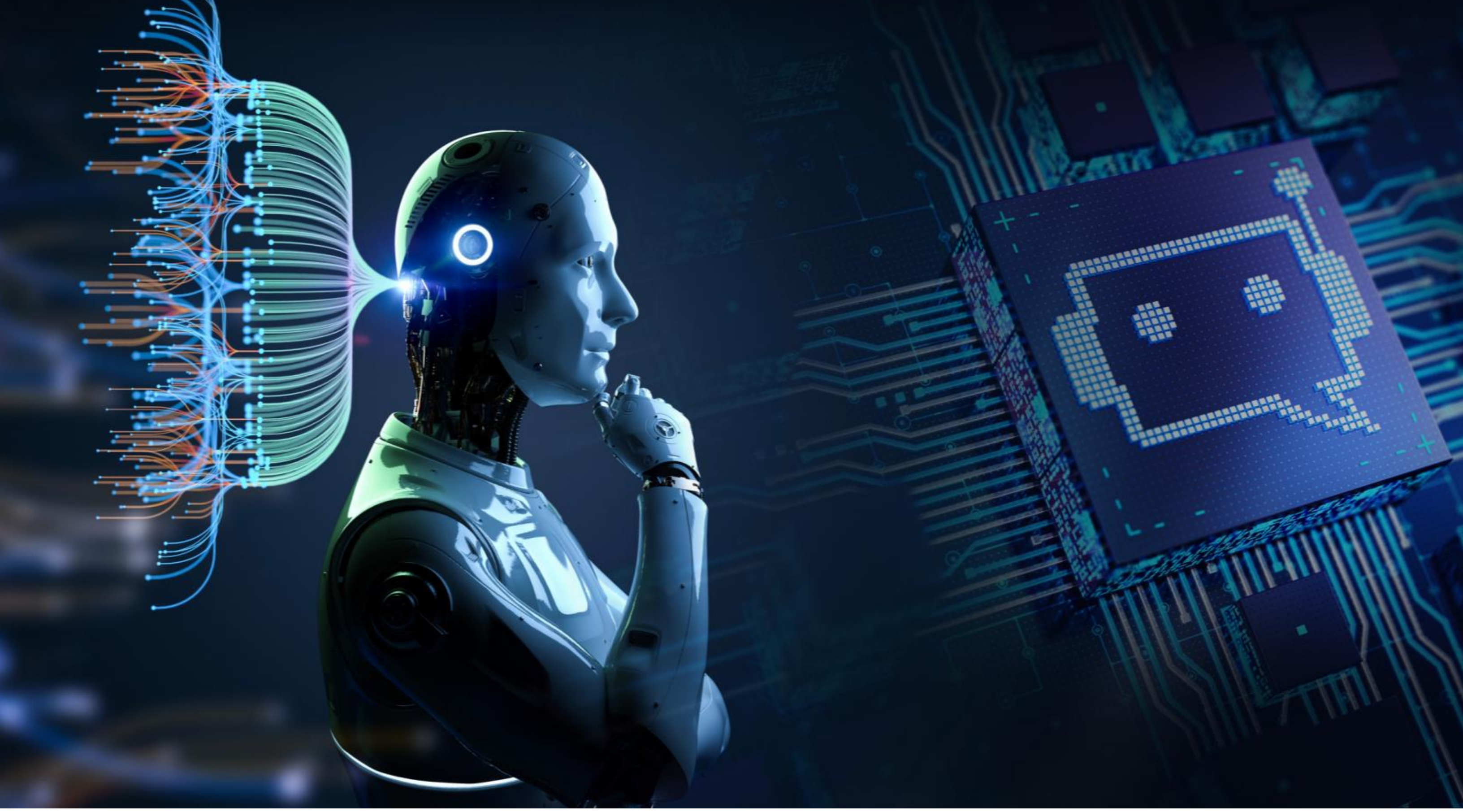
There are various advantages and limitations associated with RAG and embedding semantic similarity

Advantages

- While traditional models like GPT have limits on the amount of data they can store and recall, RAG leverages external databases — allowing it to pull in fresh, updated, or detailed information when needed.
- It reduces the LLM's hallucinations by providing the relevant context.
- Flexibility - By changing or expanding the external knowledge source, you can adapt an RAG model for specific domains without retraining the underlying generative model.
- Instead of having a monolithic model that tries to memorize every bit of information, RAG models can scale by simply updating or enlarging the external database.

Limitations

- Semantic Ambiguity: Ambiguities in query interpretation.
- Vector Similarity Issues: Challenges with vector similarity measures like cosine similarity.
- Issues in handling multiple questions in a single question.



Prompt Engineering

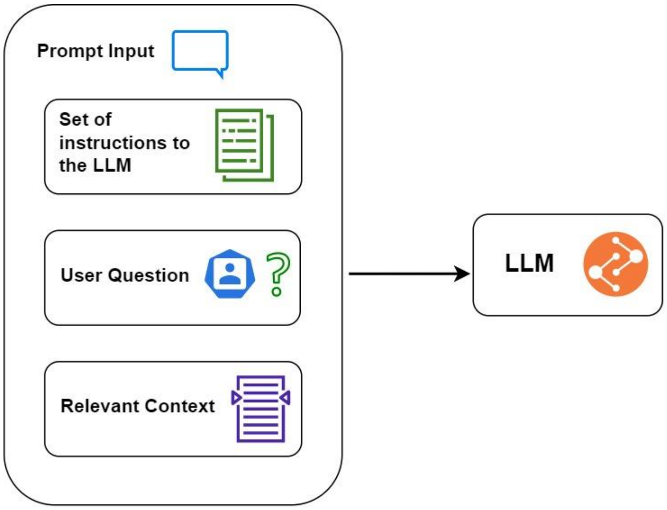
Prompt Engineering represents a strategic method to steer generative AI systems toward producing specific outcomes in response to user queries. It plays a crucial role in shaping how Large Language Models (LLMs) formulate their responses, ensuring they adhere to predetermined guidelines.

This technique encompasses methods such as Instructions, one-shot, and few-shot learning, which involve providing the AI with a single example or a small set of examples. These examples serve as a guide, showing the AI the expected way to respond under certain conditions. Such approaches are particularly useful in situations where extensive training data is not available, enabling the AI to adapt to new tasks with minimal input.

In the context of MITRE AI assistance, extensive experimentation with various prompts has been conducted. The objective is to refine the AI's ability to deliver professional, responsible, and contextually relevant responses. By meticulously crafting these prompts, the system is directed to leverage the context provided to generate comprehensive and informative answers, effectively addressing user inquiries. Below are some examples.

```
□<s>[INST] <<SYS>>
You are an AI Assistant.
If the question is relevant to chat history then use chat history to answer the
question in a professional manner.Answer only the following Question without
providing any additional information that is not explicitly asked for in the
Question.
If the question is not based on the chat history and also not based on cyber
security only respond with: "Sorry I Don't know".
If the question is related to Cyber security then answer it with using your
knowledge in a professional manner.
<</SYS>>
Chat history : {chat_history}
context: {context}
Question: {question}
[/INST]
Answer - """.strip()
□
```

Moreover, prompt engineering extends to improving conversational dynamics, such as facilitating follow-up questions. By incorporating elements of the chat history into the prompt, AI assistance is equipped to maintain a coherent and engaging dialogue. This feature is instrumental in creating a more natural and user-friendly interaction, encouraging a seamless conversational flow.



Vector Databases and Semantic Search

Semantic search plays a crucial role within the assistant system by enabling the retrieval of snippets from documents that are relevant to user queries. This process involves breaking down documents into smaller, more manageable snippets, which are then translated into a statistical format known as vector space. Essentially, this process transforms textual data into a numerical representation, allowing for it to be stored within a database.

Various databases exist, each designed to meet different storage needs. Among these, vector databases are particularly noteworthy for their ability to store and manage vector representations of text. These specialized databases excel in optimizing storage space for large vectors, thereby reducing storage costs and enhancing the efficiency of data retrieval.

Vector databases are essential for Large Language Models (LLMs) due to their capability for fast and effective data retrieval. These databases are engineered to conduct vector similarity searches with great speed, facilitating immediate access to pertinent information. With the increasing reliance on LLMs, the demand for scalable and efficient storage solutions becomes paramount. Vector databases are adept at managing the extensive volumes of data generated by LLMs, thanks to their large-scale data handling capabilities.

When selecting a vector database, several key considerations must be taken into account:

Hosting Options: The choice between self-hosted solutions and managed services.

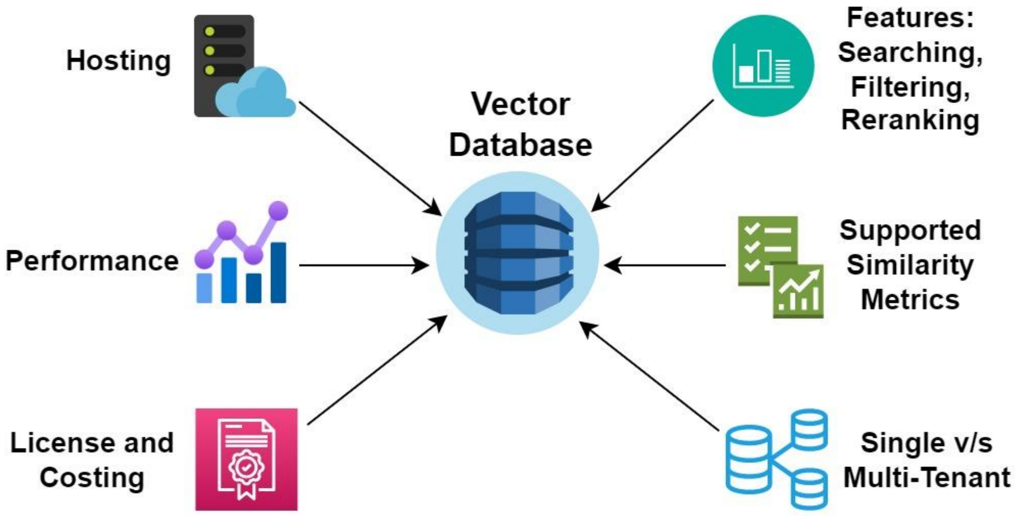
Features: The need for basic vector search capabilities, advanced filtering options, or the ability to rerank search results.

Performance: Considerations include the quality of search results, response times (latency), and the ability to handle high volumes of queries (throughput).

Supported Similarity Metrics: Various databases offer support for different metrics, such as Euclidean distance, inner product, Hamming, and Jaccard similarities.

Cost and Licensing: The financial implications and licensing terms associated with the database.

Multi-tenancy Architecture: Whether a single database instance can support multiple users or tenants



Results and Evaluation

In the development of applications utilizing LLM, the initial creation might seem simple; however, the real challenge emerges in their ongoing maintenance and improvement. To ensure these applications remain reliable and can be consistently enhanced, a structured evaluation phase is paramount. This phase employs a systematic approach to testing, relying on specific metrics to gauge performance accurately, thereby guaranteeing both the reliability and reproducibility of the experiments conducted.

A critical aspect of this evaluation involves the use of a benchmark dataset, which serves as a standard for measuring the effectiveness of the RAG component within LLM applications. This dataset is typically composed of questions paired with relevant contexts, the latter of which are extracted from snippets of source documents. To streamline this process and eliminate the need for manual annotation, the dataset can be generated automatically with the help of LLMs themselves.

The performance of an LLM RAG-based application is primarily assessed using two key metrics:

Context Precision: This metric evaluates the extent to which the system can identify and prioritize ground-truth-relevant items within the provided contexts. In an ideal scenario, all relevant snippets should be ranked at the top. Context Precision is quantified on a scale from 0 to 1, where values closer to 1 signify a higher precision. The calculation of this metric takes into account both the question posed and the contexts retrieved, aiming to determine how effectively the system can surface the most pertinent information.

Answer Correctness: This metric measures the accuracy of the answers generated by the system in comparison to a predefined standard or "ground truth." The evaluation of Answer Correctness involves analyzing the congruence between the generated response and the correct answer, with scores also ranging from 0 to 1. Higher scores indicate a greater degree of accuracy in the answers produced by the application.

Following numerous evaluation cycles, a noticeable improvement in context precision was observed, with metrics rising from 80% to 93%. This enhancement in precision was achieved through the systematic application of script automation, streamlining the process and ensuring more accurate retrieval of relevant context information.

On the other hand, the evaluation of answer correctness required a manual approach, involving human reviewers. Through several rounds of adjusting both the prompts and the parameters of the Large Language Model (LLM), the accuracy of the answers generated improved from 82% to 90%. It's important to note that further refinement and potential accuracy gains are possible through fine-tuning the model. However, this additional optimization was considered beyond the scope of the current phase of development.

The decision to forgo further fine-tuning acknowledges the balance between achieving optimal performance and the practical limitations of project scope.

Results and Evaluation

```
□Q: What are the remediation steps for exfiltration tactic?
A: Remediation steps for exfiltration tactic:
   1. Isolate affected systems immediately to prevent further data loss.
   2. Conduct a thorough investigation of the affected systems to determine the extent of
      the damage.
   3. Implement stronger access controls to prevent unauthorized access to sensitive data.
   4. Monitor network traffic for suspicious activity and implement intrusion detection
      and prevention systems.
   5. Train employees on best practices for data security and conduct regular security
      awareness training.
   6. Implement encryption for sensitive data both at rest and in transit.
   7. Regularly backup critical data and test disaster recovery plans.
   8. Conduct regular vulnerability assessments and penetration testing to identify and
      address potential weaknesses in the system.

Q: what is impacket tool?
A: Impacket is an open source collection of modules written in Python that can be used
to programmatically construct and manipulate network protocols. It includes several
tools for remote service execution, Kerberos manipulation, Windows credential dumping,
packet sniffing, and relay attacks. However, it should be noted that Impacket is not
inherently malicious and can be used for both legitimate and malicious purposes.

Q: What is kerberoasting attack?
A: Adversaries may abuse a valid Kerberos ticket-granting ticket (TGT) or sniff
network traffic to obtain a ticket-granting service (TGS) ticket that may be
vulnerable to Brute Force. Service principal names (SPNs) are used to uniquely
identify each instance of a Windows service. To enable authentication, Kerberos
requires that SPNs be associated with at least one service logon account (an account
specifically tasked with running a service).Adversaries possessing a valid Kerberos
ticket-granting ticket (TGT) may request one or more Kerberos ticket-granting service
(TGS) service tickets for any SPN from a domain controller (DC). Portions of these
tickets may be encrypted with the RC4 algorithm, meaning the Kerberos 5 TGS-REP etype
23 hash of the service account associated with the SPN is used as the private key and
is thus vulnerable to offline Brute Force attacks that may expose plaintext
credentials.
□
```

As we look to the future, we're excited about the possibilities of expanding our AI Assistant's capabilities. Techniques like query expansion hold the promise of further enhancing the relevance and accuracy of information retrieval, ensuring that our AI Assistant remains a vital tool for cybersecurity professionals.

Conclusion

The development of the AI Assistant showcases the value of collaboration, innovation, and a strong dedication to time-saving and quick adaptation. For product managers and engineers considering similar projects, our experience highlights the significance of setting clear goals, managing data carefully, choosing technology wisely, and prioritizing user needs. As we refine our AI Assistant further, we look forward to sharing insights and contributing to discussions about AI's role in cybersecurity and beyond.

Authors

Shivaraj Mulimani
Shivaraj is a Data Scientist at Acalvio, specializes in cybersecurity, demonstrating expertise in Machine learning, NLP, and R&D with over 6 years of work experience.

Arunkumar M P
Arun is a passionate data scientist with an M.Sc in Theoretical Computer Science. He has been actively contributing to his role within Acalvio's Data Science team for 2 years.

Nirmesh Neema
Nirmesh is Senior Data Scientist at Acalvio. He has successfully tackled numerous real-world cybersecurity challenges utilizing cutting-edge AI/ML techniques with 10+ years of work experience.

Dr. Satnam Singh
Dr. Satnam Singh is leading security data science development at Acalvio. He has more than 20 years of work experience in successfully building data products to production in multiple domains. He has 25+ patents and 30+ journal and conference publications

References

[1]. MITRE ATT&CK Matrix for enterprise https://attack.mitre.org/matrices/enterprise/
[2]. MITRE ATT&CK Matrix STIX repository - https://github.com/mitre-attack/attack-stix-data
[3]. Falcon-7b - https://huggingface.co/tiiuae/falcon-7b-instruct
[4]. llama2-7b - https://huggingface.co/meta-llama/Llama-2-7b
[5]. llama2-7b-chat-hf - https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[6]. mistral-7b-instruct - https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[7]. Improving LLM performance - https://www.youtube.com/watch?v=ahnGLM-RC1Y
[9]. Langchain - https://www.langchain.com/
[10]. Large Language Models in Cybersecurity: State-of-the-Art https://arxiv.org/pdf/2402.00891.pdf
[11]. LLma2 - https://arxiv.org/pdf/2307.09288.pdf.

ACALVIO

LEARN MORE