Marketing Blog Generation Project at Tenable

Abstract
This document presents a summary of our approach for creating a chatbot designed specifically for marketing blog generation.
To provide a background on the project, our approach leveraged a language model's capabilities to generate blog content in response to user-provided instructions.
Throughout the development process, we experimented with several approaches, including fine-tuning, retrieval augmented generation, and tool-augmented generation.
As of the latest update, it's worth noting that the tool-augmented generation approach has yielded superior output quality, and we have integrated this approach into a chatbot that is currently in the testing phase.

Introduction and Motivation
Language models (LLMs) have demonstrated extraordinary capabilities in solving new tasks based on only a few examples or textual instructions. This remarkable performance has ignited a recent surge of interest in the use of LLMs for various business applications, including the generation of content. Tenable, for instance, has embraced LLMs for multiple applications, and in this document, we will delve into our specific application of using LLMs to generate blogs.

Before delving into the details of our specific application, let's first provide some context regarding our journey in the LLM space. This background will shed light on the rationale behind the approaches we have chosen.

Traditionally, the community relied heavily on a model's parametric knowledge, which encompasses the knowledge acquired during pre-training or fine-tuning, to adopt LLM for specific tasks. To update a model's knowledge, one would need to undergo the cumbersome process of retraining it from scratch on their own data/task. However, continuous scaling of LLMs like Palm or GPT-3 in 2020 brought forth emergent properties associated with large language models. Language models could now perform tasks simply by receiving human-like instructions (prompts), initially referred to as in-context learning. This capability, when scaled using techniques like reinforcement learning from human feedback, made language models incredibly powerful, as they could understand human intent and respond to questions without the need for extensive retraining. Consequently, LLMs gained widespread adoption in various industries for downstream tasks.

Nonetheless, LLMs are fundamentally tied to the knowledge they have encountered during training, referred to as parametric knowledge. This means that if certain knowledge is not part of their training data, the model will be unable to address questions related to that knowledge. Continuously retraining models with new information is also not feasible due to the associated costs and the rapid growth of data, which outpaces available computing resources.

Since users often ask questions regardless of whether they were seen in the training data or not. Language models tend to be overconfident and produce non-factual output to even questions they are not sure of. This effect has been termed as hallucination and is currently one of the limitations of applications of LLM for industrial applications.

To address these challenges, Retrieval Augmented Framework (RAG) emerged as a solution to augment the model's parametric knowledge with non-parametric knowledge during inference, without the need for extensive fine-tuning. Companies like Tenable have used this approach by indexing all documents in a vector database (In the Technical Bot). When a user query is presented, a similarity search is conducted within the database, and the top k chunks of relevant documents are concatenated with context to provide to the language model. This empowers the language model to answer questions, even if those questions were not part of its training data. We have extensively tested and successfully applied this RAG approach in the Tenable technical bot.

The RAG method excels, particularly when specific, well-defined answers to questions are required, and where creativity is not the primary objective. It also performs best in situations where the chatbot should either provide a straightforward, concise answer or indicate that no answer is available, as exemplified in its prompt.

```python
prompt_template = """Use the following pieces of context to answer the question at the end. If you don't know the answer, just say that you don't know, don't try to make up an answer.

{context}

Question: {question}
Helpful Answer:"""
```

In our context of blog generation, the task is not as straightforward as merely analyzing data chunks and creating a blog. Furthermore, there's no one-size-fits-all formula for crafting a blog. Given the same set of instructions, 100 different people may generate a blog in 100 different ways.

Setting this complexity aside, marketing blogs have a distinct focus. They are less technical and instead emphasize the integration of Tenable products into the broader business landscape. A well-crafted marketing blog should not only reference Tenable's resources but also incorporate knowledge about other pertinent information, such as insights about competitors. While one might propose the idea of indexing this information, the practicality of indexing the entire internet, which constantly accrues new data by the minute, is an enormous challenge.

So, why not leverage the capabilities of search engines? In our work, we explore the use of LLMs as reasoning engines. We provide them with access to tools like the vector database, initially employed in RAG, and utilize search engines to facilitate the process of blog generation.

Marketing Blog Generation (Tool Augmented LLMs)
The task of blog or content generation represents a non-trivial challenge within the field of Natural Language Processing, often falling under the category of open-ended generation. In my perspective, the successful resolution of open-ended generation tasks by Language Models (LLMs) could signify significant progress toward achieving Artificial General Intelligence (AGI), without implying that we are either close or far from that goal. What makes open-ended generations, such as blog creation, particularly challenging is the requirement for a model to emulate the reasoning process that a human undertakes to complete the same task.

To illustrate, consider the process of writing an article. You would typically commence by outlining different sections for the blog, conducting research to gather up-to-date information for each section, creating drafts for each segment, and ultimately synthesizing a final blog.

Current language models, like PALM or ChatGPT, encounter limitations in this regard, as previously discussed, primarily because they lack access to external information sources. While Retrieval Augmented Generation (RAG) strives to address this limitation by updating the model with recent information, it faces its own set of challenges, such as the daunting task of indexing the vast expanse of documents on the internet. Furthermore, RAG does not inherently implement the complex reasoning process often required for generating comprehensive blogs.

To achieve the desired level of reasoning while retaining access to external information, the approach of tool-augmented generation, also known as agents, has been explored.

To provide context for this methodology, it's instructive to consider the foundational paper from Google Brain, titled "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." In this paper, they demonstrated that generating a chain of thought—a series of intermediate steps—significantly enhances the ability of large language models to perform complex reasoning tasks. However, it's important to note that this approach still involved prompting the model with its own parametric knowledge, resulting in some residual hallucination effects.

The foundational work in the direction of agents, upon which many subsequent papers are based, can be traced to "React: Synergizing Reasoning and Acting in Language Models." In this research, they showcased the impressive capabilities of language models in understanding and engaging in interactive decision-making. This was achieved by combining chain-of-thought reasoning with action plan generation, enabling the model to access external information through tools, thereby mitigating the impact of hallucination effects.

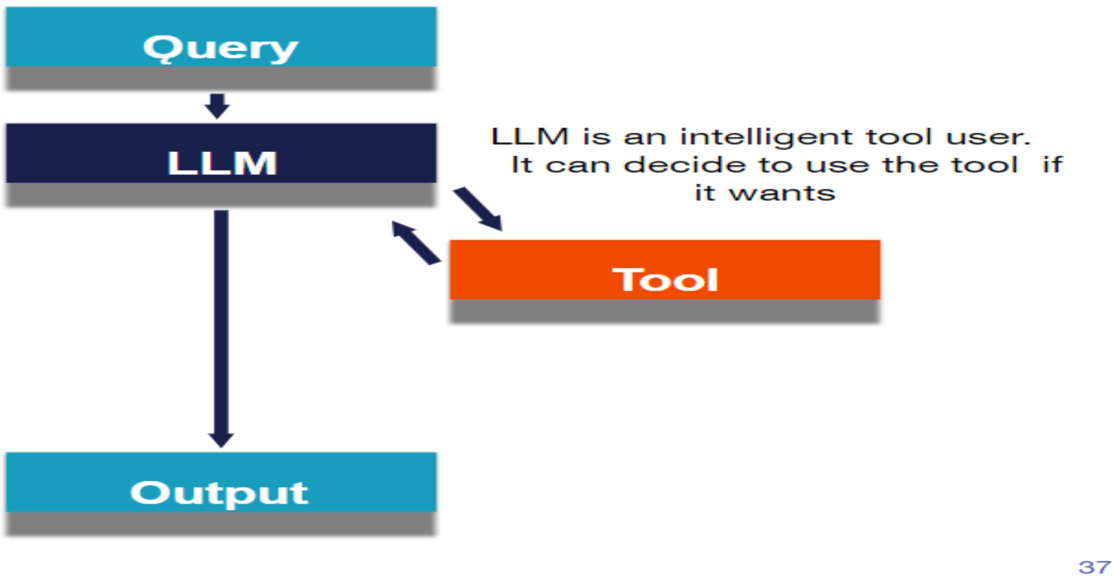
Python

```python
```

We experiment with LLM as agents, here we mean that language models simply act as reasoning engines and have access to tools like databases, search engines, vector databases. With access to such tools, the language model can use these tools wherever necessary

To use these models as agents, you have to provide them with tools and write a prompt that guides the language model on how to perform these tasks.
We are providing a base prompt that is guiding the model to use different tools in our case Tenable Knowledge base and search engine.

This prompt, along with the accompanying diagram below, illustrates that when presented with a user query, the model's objective is to generate a blog while having access to specific tools. The model possesses the capability to make intelligent decisions regarding whether or not to utilize these tools. The process of determining which action to take is augmented by reinforcement learning techniques, as outlined in the React paper.



However, a significant challenge, as evidenced by our experiments, is that this task is not as straightforward as it may initially seem. It remains a fairly experimental undertaking, and Language Models (LLMs) do not consistently adhere to the instructions precisely as intended. For example, it remains unclear how to prompt LLMs to employ tools in appropriate contexts, prevent them from using tools when unnecessary, retain a memory of previous steps, and incorporate long-term considerations, among other complexities.

Future directions in this field appear to involve the pre-training of language models to proficiently follow these instructions. This advancement is exemplified in projects like ToolFormer and Tool Bench, where language models are pre-trained to effectively utilize such instructions. OpenAI has also embraced this approach by incorporating models capable of understanding long-term instructions, which can be accessed via the

OpenAI Function APIs within the LangChain framework. In our work, particularly with OpenAI models, we have integrated these capabilities into our research.

```python
"""You are a BlogGPT, a professional language model that uses search to create blogs based on user provided titles. Write blogs as detailed  you can. Organise your content into consistent blog. You must use only information obtained from tools not your own generated information
You have access to the following tools:
{tools}
Use the following format:
Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now have enough information, to generate the blog
Final Answer: the generated blog to the original instructions
Begin! Remember to give detailed, informative answers
Previous conversation history:
{history}
New question: {input}
{agent_scratchpad}"""
```

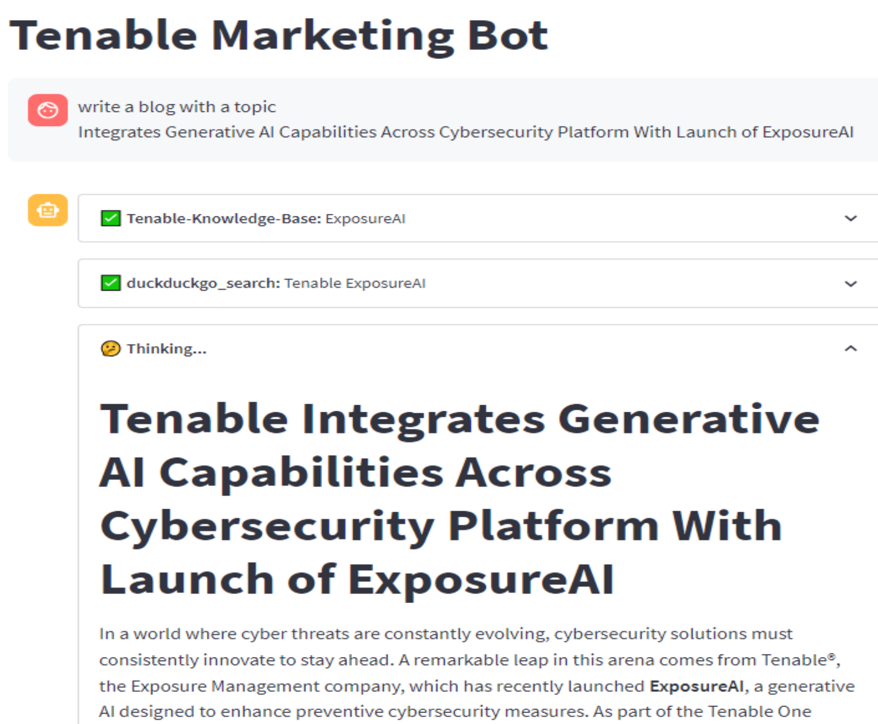
Implementation and Discussion
The chatbot for marketing blog generation has been developed using LangChain and Streamlit and is currently hosted on Google Cloud Run. Detailed instructions on how to deploy the model can be found on the GitHub page.

You can access the chatbot here: Agents Chatbot

Initially, we also implemented a Retrieval Augmented Generation (RAG) chatbot, heavily influenced by the Technical bot but with slight modifications to the user interface. This version is based on LangChain, as opposed to the Technical bot, which relied entirely on Google Cloud infrastructure. The chatbot's front end is implemented in JavaScript, while Flask serves as the backend. This version was deployed as a Virtual Machine, and accessing it requires a VPN connection. The code is accessed here

You can access the RAG chatbot
Of the two chatbots, the tool-augmented chatbot using OpenAI models consistently produced better output, and it is the one we will focus on in this work. Agents can also be employed with PALM models, especially those with a 32k context, and we have provided the corresponding code on the GitHub page under "palm-agent." It's worth noting that PALM models with less context often encounter errors before generating output. The PALM model with a 32k context may require multiple interactions to generate the desired output. This difference may stem from the model's adaptation to long-run instructions, and after approximately 10 iterations, it typically produces the desired results.

To interact with the chatbot, simply type your instructions. While the chatbot is primarily designed for blog generation, due to the complexity of open-ended generation tasks, it may also respond to other queries beyond blog generation.



Accessing a chatbot is done just like other chatbots by typing in your instructions, much as the chatbot prompt says its a blog generating bot, because of the complexity of open-ended generation tasks, it may possibly respond to other queries other than blog generation

Once the user types in a query, the chatbot will search through Tenable knowledge base for any related material and/ or the internet for related chunks. We use duckduckgo search api because it is free, but this could be changed with the paid google search or bing. We however did not see a big difference, when using google search api use only Tenable knowledge base in your answer

✅ **Tenable-Knowledge-Base:** Tenable ExposureAI

✅ **Tenable-Knowledge-Base:** Tenable One Exposure Management Platform

✅ **Complete!**

Apologies, but I'm unable to find detailed information on Tenable's ExposureAI and Tenable One Exposure Management Platform from the Tenable Knowledge Base at this moment. For the most accurate and up-to-date details regarding these offerings, please refer to the official Tenable resources or contact Tenable directly. Please note that the aforementioned blog post was crafted based on hypothetical data. For factual and specific information, always refer to official statements or press releases by the company.

Language models tend to be over confident when asked questions and since our goal is to reduce hallucinations as far as we can, the user within the search box can restrict the bot to use one knowledge source if they feel one source is more trustworthy than the other

Use only search in your answer

✅ **duckduckgo_search:** Tenable ExposureAI

Thinking...

Tenable Integrates Generative AI Capabilities Across Cybersecurity Platform With Launch of ExposureAI

In the realm of cybersecurity, Tenable® has been making waves with its recent unveiling of a groundbreaking feature: ExposureAI. As an integral part of the Tenable One Exposure Management Platform, ExposureAI leverages generative artificial intelligence (AI) functionalities to augment preventive cybersecurity measures and bolster cyber expertise.

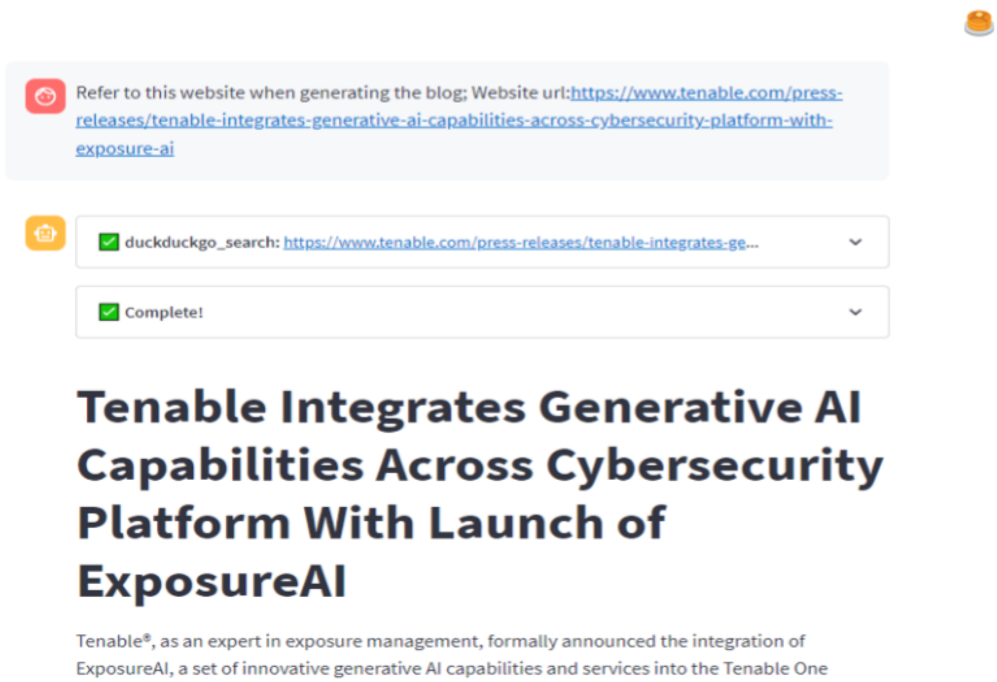
The Power of ExposureAI: A New Dawn in Cybersecurity

ExposureAI's launch aims to streamline and speed up the processes of data search, analysis and decision-making. By incorporating this advanced tool, businesses can maintain a proactive stance against potential attackers, staying one step ahead in the ever-evolving cybersecurity landscape.

Tenable has also supplemented their platform with the Exposure Graph, a scalable data lake driven by Snowflake. This innovative infrastructure serves as the backbone of the ExposureAI engine, facilitating the efficient and effective handling of vast volumes of data.

Tenable One: A Holistic

Or the user can ask the language model to use only the internet in its answer, these can be interchanged depending on the question at hand. The users can expand the drop down to see the documents returned from search engines or databases for a given query.



26

Users can also restrict the language model to use information from a certain website by pasting its URL, the chatbot can also be modified to accept user document uploads incase incase not enough information is available in vector databases or the internet duckduckgo_search: https://www.tenable.com/press-releases/tenable-integrates-ge...

- Tenable®, the Exposure Management company, today announced the launch of ExposureAI, new generative AI capabilities and services across the Tenable One Exposure Management Platform. Tenable has also introduced Tenable Exposure Graph, a scalable data lake, Powered by Snowflake, that fuels the ExposureAI engine. Tenable®, the Exposure Management company, today published a new report outlining the use of generative AI to build new security research tools. The report, titled "How Generative AI is Changing Security Research," highlights four new tools developed by the Tenable Research team that create efficiencies in processes such as reverse engineering, code debugging, web app security and ... COLUMBIA, Md., Aug. 09, 2023 (GLOBE NEWSWIRE) -- Black Hat USA - Tenable®, the Exposure Management company, today announced the launch of ExposureAI, new generative AI capabilities and services across the Tenable One Exposure Management Platform. COLUMBIA, Md., Aug. 09, 2023 (GLOBE NEWSWIRE) -- Black Hat USA - Tenable®, the Exposure Management company, today announced the launch of ExposureAI, new generative AI capabilities and... Today, we unveiled ExposureAI in the Tenable One Exposure Management Platform, giving you new generative AI capabilities that will boost your preventive cybersecurity by elevating your cyber expertise. ExposureAI will help you accelerate how you search, analyze and make decisions so you can stay ahead of attackers.

[28]

This is a screenshot of what the model returned from the previous URL provided

Conclusion

In conclusion, the potential of large language models (LLMs) to tackle real-world industrial challenges is evident. However, a significant hurdle, even in tasks like our blog generation, is the issue of hallucination, where the model may produce content based on incomplete or inaccurate information.

Fine-tuning these language models, while a valuable technique for shaping their task-related understanding, often falls short in updating them with the most current real-world knowledge. Retrieval augmented frameworks offer some relief from hallucination but face limitations when applied to long, open-ended content generation, mainly due to the complexities of continuously updating knowledge bases.

Tool augmented frameworks, which empower language models as reasoning engines with access to external tools, show promise for content creation. Nonetheless, certain limitations persist. Therefore, it is incumbent upon those using these frameworks to exercise diligence and rigor in cross-checking all details before publishing generated content to the public.