**International Journal of Management, IT & Engineering**
Vol. 14 Issue 03, March 2024,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

# Generative AI in API Product evolution

**A. Hassan Peer Mohamed**[i*]

Abstract

In the contemporary digital landscape Application programming Interfaces – APIs enable diverse platform integration. Perusing API documentations of partners, vendors and vast public APIs is challenging. To integrate and to develop new products, organizations need better tools to get contextual insights. In this paper we propose a generative AI based system APIGEN, to efficiently get insights from APIs with context. We examine the architecture of this modern Generative AI system that builds on top of large knowledge corpus created from various API providers and services, that are accessible by the organizations. Additionally, we provide an elucidation of the elements integral to building this system such as the Vector Database and RAG to orchestrate interactions with Large Language Models (LLMs) with a responsive chat bot client.

***Keywords:***

API Management;
Vector Index;
Generative AI;
Prompt engineering;
RAG.



***Author correspondence:***

A. Hassan Peer Mohamed
Software Engineer
San Francisco, CA
Email:
hassan.peer.tech@gmail.com

## 1. Introduction

The concept of the API economy [1] encompasses the array of business models and strategies tailored to leverage application programming interfaces (APIs) within the contemporary digital landscape. The APIs more accurately mirror customer preferences, leading to increased efficiency in delivering excellent applications. API providers can establish a successful self-service API program by integrating features that enhance API discoverability, such as filtering, search, categorization and documentation and more.

In an organization, Product managers, Marketers, Developers and Business analysts, can leverage APIs to contribute to the development of new products. So, API discoverability is crucial for planning, designing and development of APIs, to ensure their secure, efficient, and scalable usage.

Organizations utilize both internally developed and vendor-provided API management tools to identify the necessary APIs for their business needs. However, as business and customer demands expand, the number of APIs continues to rise, and the functionalities of these tools are constrained. In the absence of robust search capabilities, non-technical users may encounter challenges locating specific APIs within extensive documentations and the developers need more context to differentiate the capabilities of various APIs they need to integrate. API documentation has emerged as the primary reference for the business

[i] Author is a Software Engineer in large scale systems and integration, San Francisco, California.

services they provide, enabling product teams to evaluate the practicality of integrating new business features through contextual search.

Business and domain experts aim to work closely with the systems they engage with, and APIs serve as gateways to modern systems. They need the ability to apply their domain expertise contextually when searching for APIs, whether through a vendor portal or API management tools. These tools should incorporate fuzzy semantics search capabilities, allowing experts to draw a roadmap for new features based on well-informed API searches. This approach bridges the gap between business professionals and engineers involved in API integration, ensuring that teams are well-informed about new features and product roadmap.

In this paper we explore the structure of a Gen AI system with interactive and fuzzy search capabilities using configurable Large Language models (LLMs). This system is designed to offer chat capabilities for engaging with foundational models within a specific domain. To enhance search precision within the domain context, we employ the prompt engineering technique known as Retrieval Augmented Generator (RAG). This approach ensures the availability of contextual information, leading to more nuanced responses from LLMs. The system is equipped to explore contextual details of APIs from extensive documentations, offering potential integration points with new services. This helps to expand the product offerings, fostering business partnerships and opportunities.

## 2. LLM for domain knowledge

In the realm of Natural Language Processing (NLP), Warren Weaver delved into the challenges and potentialities of machine translation in his 1955 paper, "Mechanical Translation of Languages." This early work focused on the task of translating from one language to another. Today the Large language models such as GPT (Generative Pre-trained Transformer) [2], are powerful natural language processing models that have been widely employed for various knowledge-related tasks. LLMs excel in providing detailed and context-aware answers to user queries, making them valuable for information retrieval. They are increasingly used in many use cases such as text understanding, summarization and Text to Pictures generation (DALL-E) [3].

These generalized LLMs are stellar in response to user queries, but organizations need more insights into domain knowledge. LLMs can be fine-tuned for domain specific terminologies and intricacies like compliances, geography awareness or unique service offerings. These domain specific LLMs are pre-trained on specific industry related data or other niche private datasets available for the business. But fine tuning additional datasets comes with challenges such as expensive GPUs and training time.

### 2.1 API Generative Application Architecture for Product Knowledge

This paper proposes this architecture for a generative application APIGEN, in figure 1. This provides an interactive chat bot for the users. The aim of this system is to deliver a versatile tool that acts as a bridge, closing the knowledge gap among stakeholders in the realms of business, product, business and technology.

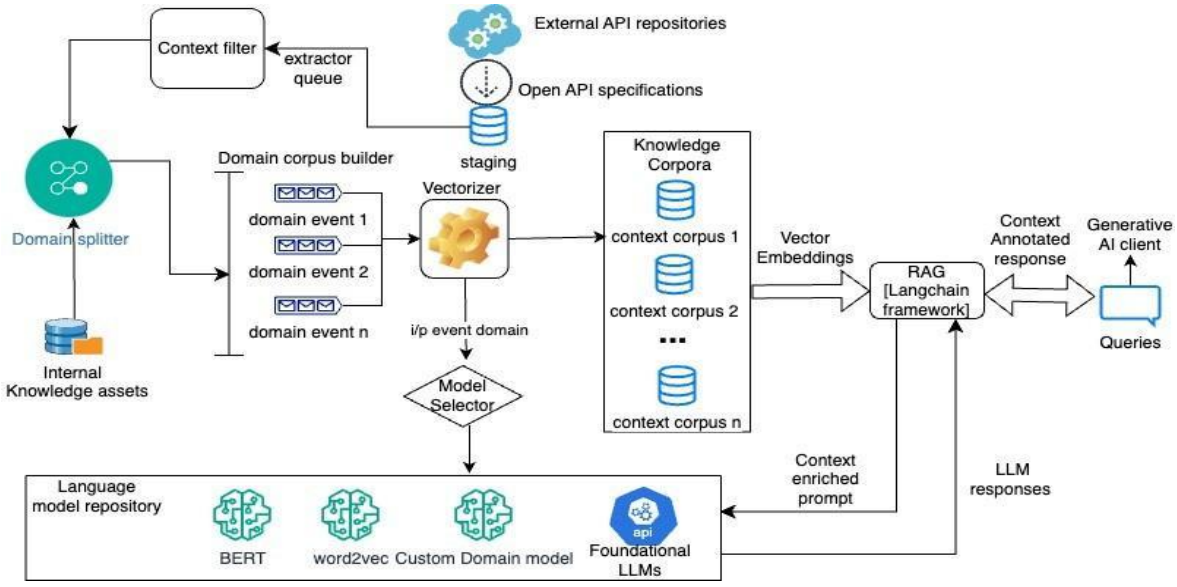


***Figure 1. APIGEN Contextual API generator***

This system helps users to ideate new product offerings, feature expansion and source potential business integration opportunities. APIGEN is a Q&A application that assists in generating responses from LLMs, combined with relevant information by sourcing static internal knowledge corpus. We design this as a generalized system, that is domain agnostic. The system can understand the domain context and generate responses through distinct domain pipelines, that embeds data to the central knowledge repository.

The system has model repository access such as word2vec [4], BERT [5], custom models and API access to LLMs such as GPT-3 and LLAMA-2 [6]. The system can be configured to the use-cases of the users and the business. As explained in section 2 above, to source domain knowledge, the system provides domain pipelines to aggregate data in their own unique DB, for which we use vector database as storage. Domain splitter component identifies the domains of the incoming API sources, categorizes the data using internal metadata information about the sources. It then serializes the documents and sends to domain specific message queues. Vectorizer is the component that consumes these document data and embeds using one of the word embedding models, configured in the system. These vector embeddings are fed to the domain specific Vector DB sinks, matching the domain source context. The need for storing vector DB is explained in the following sections. We build a knowledge corpora – collection of various domain knowledge from thousands of external service APIs and internally available information within user's organization. This Chat bot integrates with LLM and the knowledge corpora, using a layer called RAG that orchestrates the user interaction as explained in section 6.

**3. APIs as domain knowledge**

APIs are centralized information hubs. Instead of scattered documentation across different formats, users can refer to the APIs as formal documentations of a service. Public APIs are essential for any Digital service providers e.g. Google maps, travel websites or Airline information systems etc. Any organization that wishes to evolve their digital products and services into platform, must have to integrate with external API services and extend their service offerings. APIs play a vital role in digital transformation efforts, enabling organizations to connect and integrate their systems, applications, and services. This connectivity fosters innovation and agility in responding to changing market demands.

API service providers have different forms of documentation through public portals. In our system, we use API documentations in OpenAPI specification as domain corpus. We rely on various domain service providers to build the domain specific corpus. In the industry, OpenAPI is a widely adopted API specification for designing, documenting, and describing RESTful APIs. It provides a standard way to define the structure, functionalities, and interactions of APIs, allowing developers, clients, and other stakeholders to understand and interact with the API seamlessly. These are machine-readable formats, normally written in YAML or JSON format. This allows both humans and machines to comprehend the API structure and capabilities.

**4. LLMs and the need for contextual search**

Foundational LLMs are trained on large corpora of textual data, varying from articles, books and texts. The increase in the number of parameters used during training is enabling foundational LLM models to attain new capabilities. GPT-3 is trained on 175 billion parameters as cited [7] and the recent models are topping more than a trillion parameters. Though these models produce stellar responses, they are limited by the parameters and interactions with these models are non-deterministic and could lead to hallucinations. To overcome this, we need to provide additional context to the LLMs. A fine-tuning solution based on retriever architecture, based on the paper by Patrick Lewis et al.,[8], is the most effective and practical solution. This provides the LLMs additional external information as context which helps to narrow the responses. We use RAG [8] architecture to retrieve domain knowledge from API documentations, internal knowledge sources such as FAQs, requirement documentations and business analytics, to enhance the prompts to LLMs. This design helps to reduce hallucination about product and the response could be traced back to the documentations which are annotated and stored in our DB. This additional indirection through RAG introduces latency in terms of DB retrieval, searching through several thousands of documentations. This necessitates a DB that embeds unstructured data of the type of documents and JSON specifications. Vector DB is the preferred storage for documents similarity search, which helps in retrieving documents closest to the user queries.

**5. Vector embeddings of API documentation**

In order to efficiently search the vast corpora of API documentations, we need similarity search that retrieves the closest possible matching documents of a query. This can be done by vector search and is available as vector databases. In vector search, similar vectors correspond to semantically similar items and

are useful for content-based search scenario. A vector database employs a blend of diverse algorithms to find the Approximate Nearest Neighbor (ANN) search. These are mathematical vectors that capture the semantic information of words or documents. These algorithms enhance the search process through vector indexing.

Unlike scalar DB index, vector index takes query embedding as input and returns the approximate nearest neighbors of vectors, representing the documents as output figure 2.

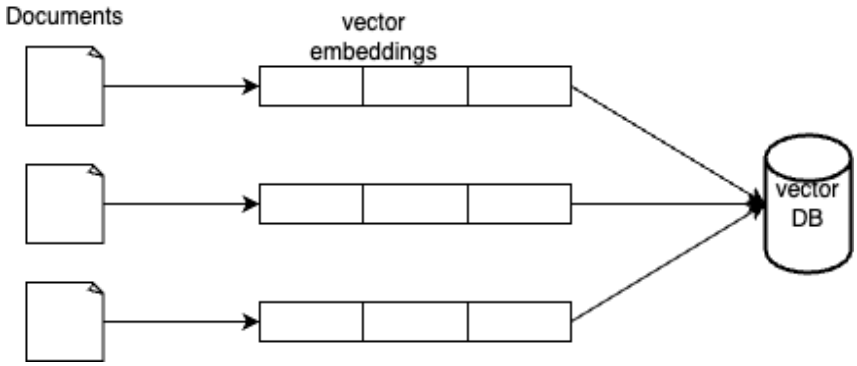


***Figure 2. Document to vector embeddings***

The most common model to create word embedding is word2vec [4]. There are other pre-trained models such as Glove, doc2vec and BERT. These models differ in the way they are trained with subtle difference in properties. Our system could plug into any of the models and create embedding of the API documentation from publicly available vendor repositories.

These models use distance metrics to calculate the similarities between documents and query vectors. There are various distance metrics that can be used such as Manhattan distance, Euclidean distance and the most common one is Cosine distance in figure 3.

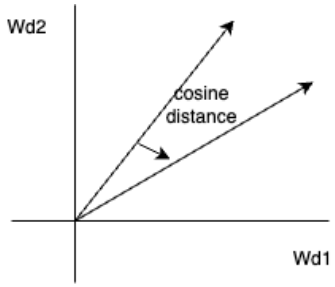


***Figure 3: Cosine similarity between weights of two documents***

**6. Prompt Engineering and RAG - Retrieval Augmented Generation**

As discussed in section 4 on fine-tuning retriever architecture, RAG is a prompt engineering technique [9] that incorporates a retrieval-based component that interacts with a knowledge base or pre-existing corpus. This component is responsible for retrieving relevant information based on the input query or prompt. The architecture relies on data, which are non-parametric and external to the LLMs, to improve context of the prompts and the response.

Our Product ideation system APIGEN relies on knowledge corpora vector storage, where we have the documents embedded. We use Langchain as RAG implementation framework as shown in figure 4. With this, the LLM interaction ensures that the generated response is contextually aware, drawing on both the information retrieved from internal knowledge base and the capabilities of the LLM.

In our Q&A based system, the user needs responses with current thinking that encompasses all the interactions with the LLMs within a chat session. Langchain helps in contextualization of the question. This ensures that responses from the LLM are coherent and built upon the existing conversation and avoids generating answers that seem disconnected or unrelated to the ongoing chat history.

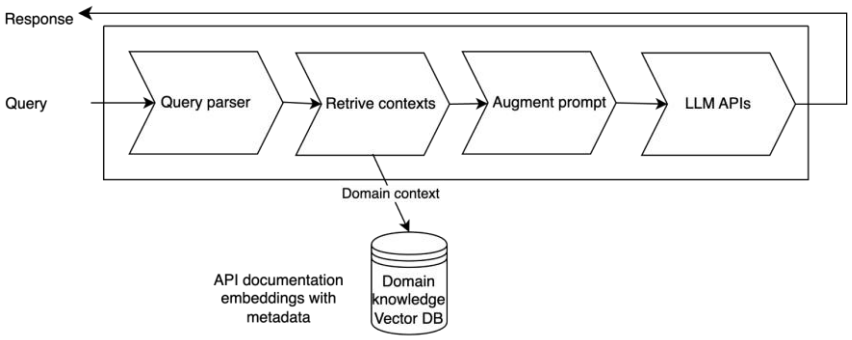


***Figure 4. A RAG orchestration for a Query***

**7. Product evaluation using APIs**

APIGEN can generate responses from LLMs with added contexts that are embedded as internal knowledge. The source and API knowledge are stored as metadata in the vector DB corpus as part of continuous knowledge buildup from hundreds of domain service providers. When user queries are sent to LLMs, the queries are embedded using the same model that is used to embed the vast API documentations. The vector index using ANN algorithm discussed in previous section, returns similar documents to the query. For APIGEN response, the documents are API references. These APIs are returned as references, sorted by the weight of similarity to the users' query and domain context. For example, in figure 5, querying the weather information could generate responses with shipping API integration contexts. This API information specification is already embedded in logistic domain corpus. Options to consider shipping insurance API integration can also be returned. This provides business stakeholders insights into expanding their features and new business partnerships with these API providers.

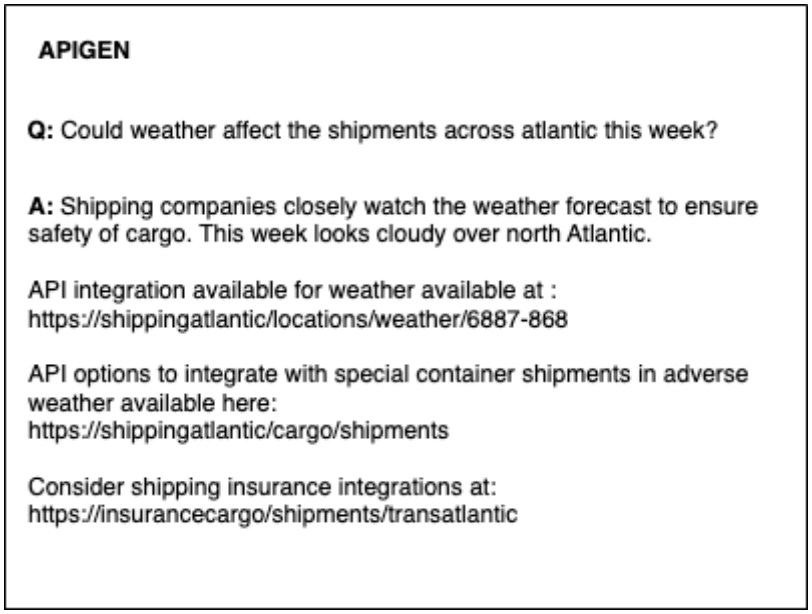


***Figure 5. APIGEN chat session***

**8. Conclusion**

APIs empower teams to effectively assess the road map of their software product evolution and expand as a platform. In this paper we explained a Generative AI based system that provides Q&A type chatbot for business to explore knowledge from public APIs, with added context sourced from internal knowledge assets. This system bridges the knowledge gap in workforce, by providing interactive experience in gaining knowledge from systems they wish to integrate. With the help of LLMs trained on domain specific knowledge and contextual data from millions of internal assets, the users get annotated responses with the most appropriate APIs or combinations of APIs ranked. We explored RAG architecture implementation that

has a pluggable context switch, to access domain specific information for user prompts. LLMs are pushing the boundaries every day, and with a focus on domain knowledge from APIs, organizations have better discoverability of APIs and services they wish to integrate. It represents a favorable outcome for both those utilizing the APIs and the entities offering the API services.

**References**


1. Deloitte. 2015. API economy From systems to business services. https://www2.deloitte.com/content/dam/Deloitte/us/Documents/financial-services/us-fsi-api-economy.pdf
2. Alec Radford et al., 2018. Improving Language Understanding by Generative Pretraining
3. Alec Radford et al., 2020. "DALL-E: Creating Images from Text", *Neural Information Processing Systems (NeurIPS)*
4. Tomas Mikolov et al., 2013. "Efficient Estimation of Word Representations in Vector Space"
5. Jacob Devlin et al., 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"
6. Hugo Touvron et al., 2023. "Llama 2: Open Foundation and Fine-Tuned Chat Models"
7. Tom B. Brown et al., 2020. "Language Models are Few-Shot Learners"
8. Patrick Lewis et al., 2021. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"
9. Banghao Chen et al., 2023. "Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review"