Atris Engine

Whitepaper & Technical Overview

Aniruddh Iyengar

October 31, 2023

Table of Contents

Background

The rise of Large Language Models (LLMs) has spawned many applications, mainly in consumer chat products. However, these applications often suffer from slow response times, are chained to specific use cases, and are unable to interact with the external digital or physical world. While there have been attempts to create LLM-based tools for particular verticals to address these issues, a universal solution that can adapt to various scenarios is lacking. The Atris Engine aims to fill this gap by offering a flexible, efficient platform that transcends the limitations of current LLM-based products, paving the way for more interactive and actionable applications that consumers will love.

Objective & Scope

The objective of the Atris Engine is to build the best possible artificial intelligence system to serve the needs of consumer applications.

The Engine vertically unifies key components of any LLM-based inference pipeline: data ingestion, document retrieval, and memory and context management. We also add two more components: the first, which we call **Atris Extensions**, are code modules can be triggered from the Engine to produce an external side-effect (e.g. responding to an email, or writing a support ticket). The second, which we call the **Prompt Library**, is a database of user-generated prompts which can be called in lieu of a hand-written textual request, with additional configuration baked in at execution time.

These pieces should be highly modular, meaning that configurations to one component of the Engine should not greatly affect the performance of another. However, the Engine should be intelligent enough to **know how to configure itself** given a natural-language query from an end user, whether that be a chat user or a consumer of our API endpoint. It should also be able to execute queries quickly, a need specific to consumer chat applications.

The first generation of consumer LLM products were based on **generation** tasks like Q&A and copywriting, which could ingest data, craft a single prompt, and return a single textual result. The next generation of LLM products will be **agentic**, able to generate their own prompts, decide which actions to take based on the user's query, and return rich results with images and links, while also taking actions outside of its own sandbox. The Atris Engine takes the first step

toward agentic products that will delight customers with their natural feel, low latency and as little unnecessary user interaction as possible.

Engine Architecture

Inference via the Atris Engine takes the following steps:

1. **Ingestion & Data Processing.** The Atris Engine collects information from private sources provided to us by our clients, as well as web sources that are publicly available for scraping. These include YouTube videos, Twitter accounts, websites, blogs, podcasts, Amazon and Shopify storefronts, and more. These disparate documents are then converted to a standard machine-readable form proprietary to Atris and stored in a cloud database, which improves their ability to be recalled in later steps in the the inference pipeline. The ingestion process must be completed before any requests to the Atris Engine can be made.
2. **Memory Retrieval.** When a user makes a request to the Atris Engine, they can optionally provide a **"Memory ID"** which will point to an existing conversation. If a Memory ID is provided, the Atris Engine will retrieve all prior messages in the conversation, which will help hone in on context clues not readily available in the user's latest query.
3. **Self-Configuration.** When a user makes a request to the Atris Engine, they will specify certain parameters in the request such as the text query and temperature (a measure of "creativity" or "surprisingness" in an LLM's answer). For parameters that are not specified, the Engine will infer what the best values are, using a series of calls to a fine-tuned LLM. Some of these decisions include:
    - **Query Rewriting.** Based on the conversation memory and context clues therein, the Engine will rewrite the user's query as a short phrase. This phrase is more well-suited to recall document embeddings in the document retrieval phase.
    - **RAG Parameters.** In Retrieval-Augmented Generation (RAG), documents are retrieved from an external source (such as a database or website). The best documents to retrieve for a specific query will vary based on the task requested by the user; the task may require the most relevant documents (retrieved using vector search against the documents'

embeddings), the latest documents (retrieved using a basic SQL query), or random documents. Based on the user's request, the Atris Engine will decide which retrieval method best suits the task at hand.

- **Google Search Queries.** Often times, the documents collected in the initial ingestion step will not have sufficient information to answer a user's question or complete a task. Therefore it is necessary to search the web for useful information. The Atris Engine will determine which Google search queries to make, taking advantage of the full breadth of human knowledge available on the wider internet.

4. **Document Retrieval & Extraction.** This step is the primary focus of Retrieval-Augmented Generation (RAG). Once all parameters are either specified or inferred, the Atris Engine will retrieve whichever documents are necessary to complete the user's request. These include documents collected directly from the first ingestion step, webpages retrieved from the internet via Google search, or documents that have been added manually by an administrator. In order for the contents of these documents to fit within the LLM's "context window" (the limit of text an LLM can receive as input) these documents are **extracted**: using either LLM calls (slower, more accurate) or text embeddings (faster, less accurate), the Engine selects a chunk of text from each document that's most relevant to the user's query.
5. **Final Inference.** Using the memory of the existing conversation, as well as the documents retrieved in the previous step, the Atris Engine will run an inference query with an LLM, which will generate the final response returned to the user. The response can either be streamed to the user using an HTTP Steraming protocol, or returned in a single batch. The LLM currently used in Atris' consumer products is OpenAI's GPT-4, but the Atris Engine is completely LLM-agnostic and other providers can be swapped in.
6. **Extensions.** Atris Extensions are code modules that can be written by third parties to execute actions such as responding to an email, writing a support ticket, or making a successive API call. While the final inference response is being generated, any extensions the user provides will be run, effecting actions outside the sandboxed world of the LLM itself.
7. **Followups.** In many situations, a user may want to continue a conversation or take further action after receiving a response from the Engine. Because of this, a user can request for the Engine to suggest followup questions or requests. The Atris Engine uses an LLM call to generate these followup

questions, which are accessible via the API endpoint or Atris' consumer chat interface.

Implementation

The Atris Engine is split primarily into two workflows: the **ingestion** and the **inference**.

Ingestion scripts are written as Python workers and served on AWS EC2 Fargate instances. These scripts include dozens of web scrapers for various types of web sources, scripts for text chunking and processing, and calculating and storing vector embeddings of the collected documents.

The inference is written as a monolithic TypeScript module and served by an AWS Lambda function.

Performance & Results

Current World Knowledge

Atris is very good at quickly recalling up-to-date knowledge and relaying it back to the user in an understandable way.

Query:
Name the current members of the New York Knicks roster.

ChatGPT (Free)
I'm sorry, but I don't have access to current or real-time information...
Latency: **1 sec**

ChatGPT Plus with Browsing
The New York Knicks roster for the 2023-24 NBA season includes the following players: 1. Jalen Brunson...
Latency: **106 sec**

Atris Engine
Sure, Aniruddh. The current roster of the New York Knicks, as per the information I have, includes: 1. Jalen Brunson - Point guard...
Latency: **21 sec**

Creator Content

A key use case for Atris is serving content from existing creators on YouTube, Twitter, Instagram, and other platforms. While platforms like ChatGPT struggle to ingest these data sources and store them in a format conducive for RAG, the Atris Engine handles these situations very well. This allows creators to build chatbots tailored specifically to their content, and serve it to their audience members to improve engagement.

In the example below, we trained an Atris chatbot on the contents of Andrew Huberman's YouTube videos (@HubermanLab). The query refers to a video Andrew Huberman posted to YouTube recently.

Query:
What are Andrew Hubermans' thoughts on resveratrol?

ChatGPT (Free)
I do not have access to specific information or opinions from Andrew Huberman regarding resveratrol...
Latency: **3 sec**

ChatGPT Plus with Browsing
Andrew Huberman's thoughts on resveratrol are not fully detailed in the available sources...
Latency: **39 sec**

Atris Engine
Resveratrol is a fascinating compound that has been the subject of much discussion and research. In my conversation with Dr. David Sinclair, we delved into the effects of resveratrol...
Latency: **17 sec**

Personality

ChatGPT and many other LLM-based chat products do not respond the way typical humans do. They play the role of the AI assistant, which can be off-putting to users who are new to AI products. The Atris Engine is built to mimic the style of the documents collected during RAG, so it will respond to questions in a more natural and humane way.

In the following example, we use the same Andrew Huberman model shown in the previous example.

Query:
Who are you?

ChatGPT (Free)
I am ChatGPT, a computer program developed by OpenAI. I'm based on the GPT-3.5 architecture...
Latency: **2 sec**

ChatGPT Plus with Browsing
I am a virtual assistant created by OpenAI. My design is based on the GPT-4 architecture...
Latency: **12 sec**

Atris Engine
Hello Aniruddh, I'm Andrew Huberman, a neuroscientist and tenured professor in the Department of Neurobiology at the Stanford University School of Medicine...
Latency: **19 sec**

Future Work

We believe that the Atris Engine can be improved in many ways. This is a non-exhaustive list of proposed enhancements that will improve the quality of the Engine's responses.

- **Task-Specific Document Embeddings.** The Atris Engine currently uses a one-size-fits-all approach for embeddings and RAG. This reduces storage costs, but the quality of document retrieval can be improved by calculating multiple embeddings per document, one for each "type" of task (e.g. classifying documents, answering questions, making plans). InstructOR (Su et al., 2023) is an example of an embedding model that allows for this.

- **Fine-Tuned Models for Self-Configuration.** The Atris Engine currently uses OpenAI's GPT-3.5 Turbo model to infer parameters as specified in the "Engine Architecture" section of this paper. However, this adds a few seconds of latency that could be shaved off by using a smaller, open-source, fine-tuned model such as Meta's Llama 2.
- **Training a Proprietary LLM.** Dependence on an external LLM vendor such as OpenAI is not ideal at scale. They have harsh rate-limits, enforce content moderation policies that may not align with our customers' values, and represent a single point of failure in Atris' technical stack. Ideally, the Atris Engine would use its own proprietary trained LLM to meet all its generative AI needs.

Conclusion

The Atris Engine is a highly modular artificially intelligent system capable of not only answering basic questions, but making its own plans based on a user's query and executing them to arrive at a better response. It outperforms some of the most widely-used generative AI chat applications, while being able to fine-tune itself to a specific set of documents and search the web when it encounters gaps in knowledge. Our goal is for the Atris Engine to power the best consumer chat products that inform, inspire, and entertain users.