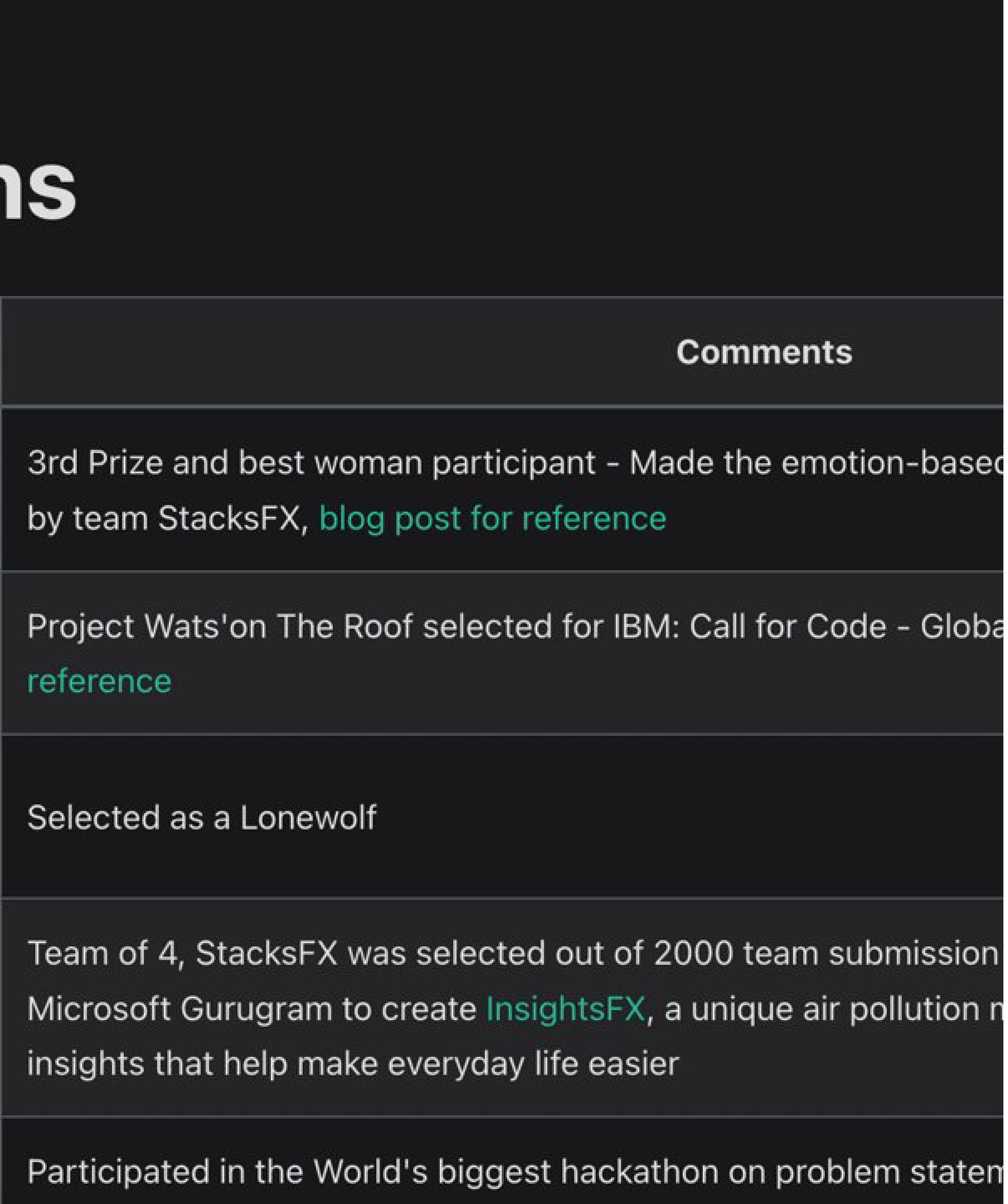


ns

Comments
3rd Prize and best woman participant - Made the emotion-based by team StacksFX, blog post for reference
Project Wats'on The Roof selected for IBM: Call for Code - Globa reference
Selected as a Lonewolf
Team of 4, StacksFX was selected out of 2000 team submission Microsoft Gurugram to create InsightsFX, a unique air pollution n insights that help make everyday life easier
Participated in the World's biggest hackathon on problem statem

Enhancing the Moya AI Framework at HackiiIT

An introductory slide showcasing our team's contributions to the Moya AI framework at the HackiiIT hackathon.

Our Problem Statement

Extend Moya's functionalities

Implement new agent capabilities and integrations to enhance the out-of-the-box value of the Moya AI framework

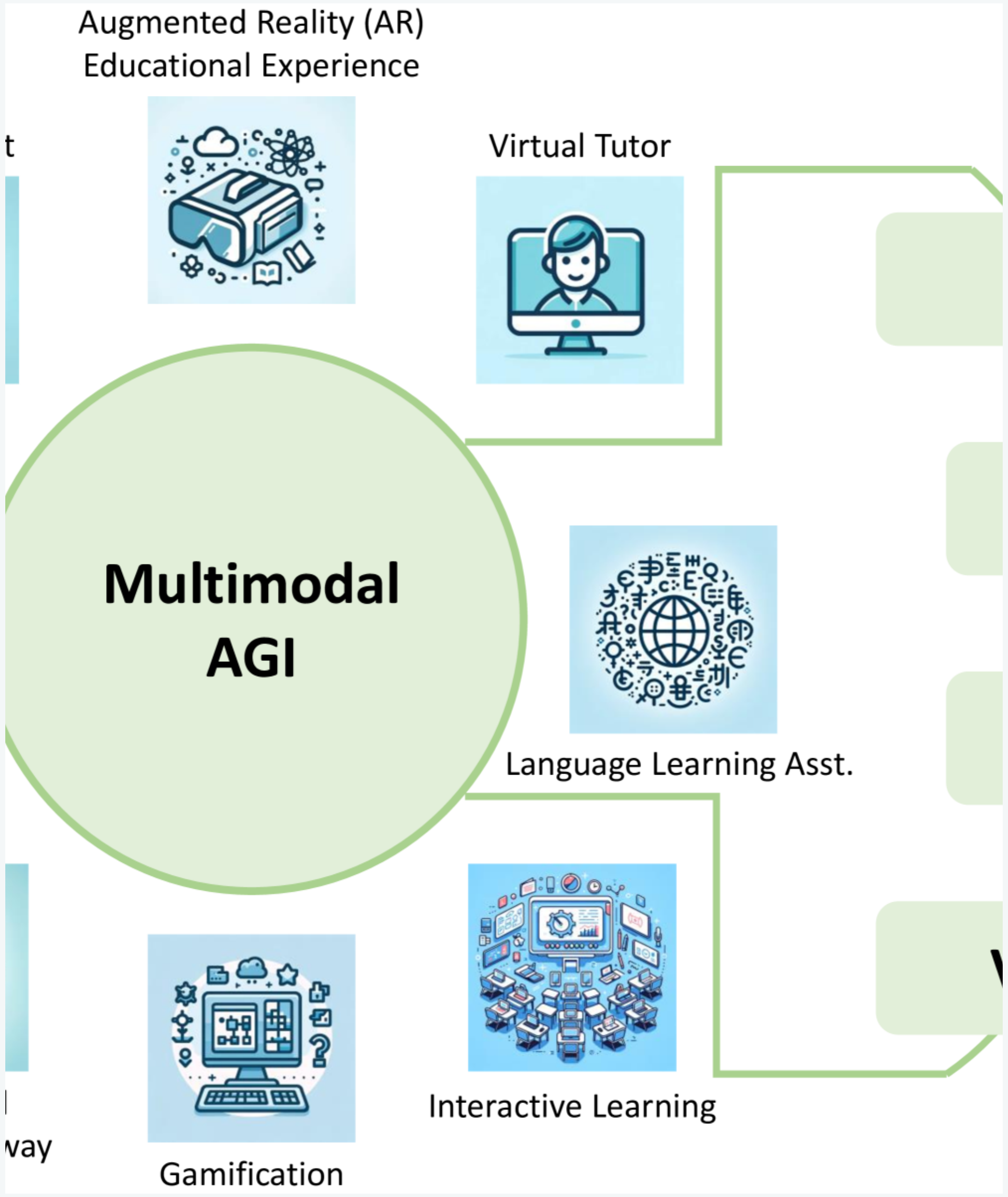
Reduce development overhead

Avoid the need for others to build features that are now available in the Moya framework

Improve usability and accessibility

Make the Moya framework more user-friendly and accessible to a wider range of developers

By developing new modules and capabilities for the Moya AI framework, we aim to drive forward its value and usability, making it a more attractive and powerful tool for building multi-agent systems.



Moya AI Framework

Moya is a powerful framework for building multi-agent systems that integrate multiple AI agents. It allows these agents to collaborate or operate independently, enabling them to solve complex problems more efficiently. The Moya framework provides a unified platform for coordinating and orchestrating the interactions between various AI agents, ensuring seamless communication and information sharing.

Moya Agent Capabilities

ClaudeAgent

Utilizes Anthropic's Claude API for conversational capabilities, supports synchronous and streaming response generation, integrates with Moya tool registry with RAG and webcrawl.

HuggingFaceAgent

Leverages Hugging Face's NLP models and Inference API, supports synchronous and streaming response generation, integrates with Moya tool registry, implements quantization for efficient local inference

OpenAI Azure

Utilizes OpenAi's API for text generation capabilities, supports synchronous and streaming response generation, integrates with Moya tool registry. Also has RAG and webcrawl.

Search Tool and RAG

Provides web search capabilities for agents, supports both paid (SerpAPI) and free (DuckDuckGo) search options, returns formatted search results in JSON format. In addition to the implemented RAG, this forms a powerful tool.

The Moya framework includes a diverse set of agent implementations, each with unique capabilities and features, enabling the development of powerful and versatile multi-agent systems.

New Module Development

- **OpenAI Agent with RAG and webcrawl**
  Enable with RAG and webcrawl, this agent synchronously accesses documents, augmenting its knowledge by navigating the web and editing retrieved information.
- **ClaudeAgent with RAG and webcrawl**
  Enabled with RAG and webcrawl, this agent synchronously accesses documents, augmenting its knowledge by navigating the web and editing retrieved information.
- **DeepSeekAgent**
  An implementation of an intelligent conversational agent using DeepSeek's API, with features like support for synchronous and streaming response generation, and integration with the Moya tool registry.
- **HuggingFaceAgent**
  An intelligent conversational agent leveraging Hugging Face's models and Inference API, with features like support for synchronous and streaming response generation, and integration with the Moya tool registry.
- **Text Autocomplete Tool**
  A lightweight text completion assistant designed to predict the next few words of a given sentence, leveraging the OllamaAgent powered by locally hosted language models.
- **Search Tool**
  A web search tool that supports both paid (SerpAPI) and free (DuckDuckGo) search options, and can be integrated with the Moya tool registry.
- **Vector Stores**
  A unified, tech-agnostic vector store class to run both ChromaDB and FAISS index, used by the RAG Search Tool.
- **Agent Consultation Tool**
  Enables agents to communicate with and consult other agents, allowing for the creation of specialized agent ecosystems where agents can leverage each other's capabilities to solve complex problems.
- **Math Tool**
  Enables the agent to solve math problems by leveraging SymPy.
-

HuggingFaceAgent

Leverages Hugging Face's Powerful NLP Models

Utilizes state-of-the-art Hugging Face models for various natural language processing tasks.

Supports Synchronous and Streaming Responses

Provides both synchronous and real-time streaming response generation capabilities.

Integrates with Moya Framework

Seamlessly integrates with the Moya tool registry for extended functionalities.

Efficient Local Inference with Quantization

Implements quantization options for fast and efficient inference on local devices.

The HuggingFaceAgent is a powerful integration of Hugging Face's versatile NLP models within the Moya ecosystem, enabling flexible conversational flows, real-time streaming, and efficient local inference.

Text Autocomplete Tool for Moya

- ## Overview of Text Autocomplete Tool

  A lightweight text completion assistant designed to predict the next few words of a given sentence, integrated with the Moya framework and powered by the OllamaAgent and local language models.

- ## Key Features

  Predicts 5-6 words to extend a sentence, ensures completion aligns with user's writing style, focuses on sentence continuation, and provides grammatically correct and contextually meaningful outputs.

- ## Configuration Details

  Uses the 'mistral:latest' model, temperature set to 0.3, base URL configured as 'http://localhost:11434', and a context window limited to 1024 tokens.

- ## Workflow: Initializing, Cleaning, Completing, and Caching

  Initializes the OllamaAgent, cleans the input text, queries the agent for completion, and caches the results to minimize redundant API calls.

- ## Error Handling and Best Practices

  Gracefully handles errors, ensures the OllamaAgent server is running, and recommends using concise yet informative input text for improved performance.

RAG Search Tool: Efficient Knowledge Retrieval for Question-Answering

- ## Embedding Function

  Use Langchain's OllamaEmbeddings to create an embedding function that converts documents and queries into embeddings.

- ## Vector Store

  Create a FAISS or ChromaDB vector store to store the document embeddings for efficient similarity search.

- ## Load Documents

  Use the vector store's `load_file()` method to load documents into the vector store.

- ## Vector Search Tool

  Add the VectorSearchTool to the agent's tool registry to enable searching the vector store for similar documents.

- ## Search and Retrieve

  Use the VectorSearchTool's `search_vectorstore()` method to find the k most similar documents to a given query.

BidOrchestrator Documentation

Introduction to the BidOrchestrator

The BidOrchestrator represents a dynamic approach to agent orchestration, using a marketplace model where agents compete and collaborate to handle user requests.

Core Concepts

The core concepts of the BidOrchestrator include a bidding system where agents bid on user requests with confidence scores, dynamic team formation to create teams when multiple agents would be beneficial, and performance learning to improve over time by tracking agent performance.

Workflow: Bid Collection

When a user message arrives, the orchestrator requests confidence scores from all available agents, running in parallel for faster response. For agents without a `bid_on_task` method, it estimates confidence based on keyword overlap between the message and agent description, as well as the agent's historical performance.

Workflow: Agent Selection

The orchestrator makes a key decision on whether to use a single agent or form a team. It uses a single agent when one agent has significantly higher confidence (>0.7 and 0.2+ higher than others) or only one agent is available. It forms a team when multiple agents have similar high confidence scores or no single agent is highly confident (below the team threshold).

Workflow: Processing

For single agent processing, the orchestrator simply passes the request to the chosen agent, supports streaming responses, and tracks performance metrics. For team-based processing, it collects responses from all team members in parallel, synthesizes the responses into a coherent answer, and uses a dedicated synthesis agent if available, or combines the highest confidence response with insights from others.

Workflow: Performance Learning

The orchestrator continuously updates performance metrics for agents, tracking average response time and maintaining success rates (which can be updated with user feedback). It uses this data to influence future agent selection.

Configuration Parameters

Key parameters that control the BidOrchestrator's behavior include `min_confidence` (minimum confidence needed for an agent to handle a request), `team_threshold` (confidence threshold for team formation), `max_team_size` (maximum number of agents in a team), and `parallel_bidding` (whether to collect bids in parallel).

Bidding Algorithm

The confidence calculation combines a base confidence score (0.1), word overlap between the user query and agent description (up to 0.5), and a historical performance factor (up to 0.4). This creates a balanced approach where teams form only when genuinely beneficial, rather than for every request.

Dynamic Tool Registration in Moya

Runtime Tool Registration

Register new tools during program execution

Function-Based Tools

Convert Python functions into agent-callable tools

Auto-Documentation

Automatically generates tool descriptions from function docstrings

Parameter Validation

Ensures tool parameters are properly formatted for LLM consumption

Dynamic Tool Registration in Moya allows developers to extend agent capabilities by registering custom tools at runtime, without modifying the framework code. This powerful feature enables the addition of domain-specific functions as tools that agents can discover and utilize during conversations.

SearchTool: Powerful Web Search for Agents

- ## Web Search Capabilities

  The SearchTool provides web search functionality for agents, supporting both paid (SerpAPI) and free (DuckDuckGo) search options.

- ## Formatted Search Results

  The SearchTool returns search results in a formatted JSON format, making it easy to integrate with other tools and applications.

- ## Moya Tool Registry Integration

  The SearchTool integrates with the Moya tool registry, allowing it to be easily used in conjunction with other tools and services.

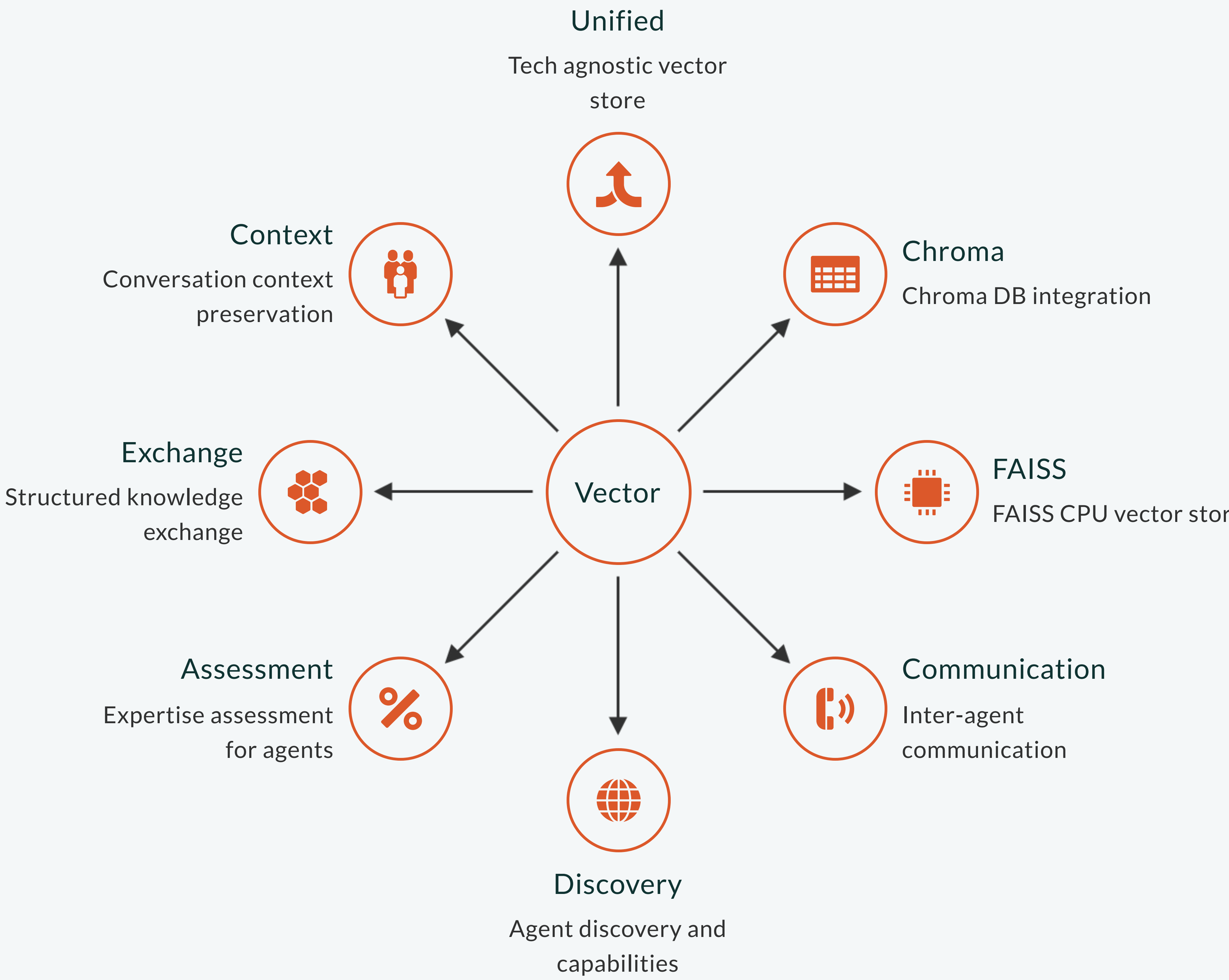
- ## Flexible Search Methods

  The SearchTool provides two search methods: `search_web()` for paid SerpAPI search, and `search_web_free()` for free DuckDuckGo search.

- ## Configuration and Registration

  The `configure_search_tools()` method allows you to configure the search tools and register them with the tool registry.

Agent Consultation Tool


Unified
Tech agnostic vector store
Context
Conversation context preservation
Chroma
Chroma DB integration
Exchange
Structured knowledge exchange
Vector
FAISS
FAISS CPU vector stor
Assessment
Expertise assessment for agents
Communication
Inter-agent communication
Discovery
Agent discovery and capabilities

Conclusion: Quirky Solutions and Significant Contributions to Moya AI

Quirky yet practical solutions

Developed unique approaches to tackle real-world challenges faced by Moya AI

Significant contributions to Moya AI

Our innovative ideas and implementations have had a measurable impact on the project's progress

Adaptable and versatile

Demonstrated the ability to create flexible solutions that cater to Moya AI's evolving needs

In conclusion, we have successfully implemented quirky yet practical solutions and made significant contributions to the advancement of Moya AI, showcasing our adaptability and versatility in tackling complex challenges.