# Project Documentation - Pinecone Query Agent

## Project Name:

* Pinecone Agent Crew AI Workflow.

## For:

* Bridged AI – Hiring Task Submission

## Project Overview:

* This project implements a modular AI workflow using Crew AI – a multi-agent orchestration framework. It demonstrates dynamic assignment of agents and tasks interact with a vector database via pinecone, enabling intelligent information querying and task execution.

## Project Structure:

├ ──config/

├ ├── agents.yaml # Agent configurations

├ ├── tasks.yaml # Task configurations

├ ── pinecone/ # Pinecone integration or related code

├ ──. env # Environment variables (e.g., API keys)

├ ── main.py # Entry point to run the AI workflow

├ ── PineconeAgent\_crew.py # Crew and agent/task setup logic

├ ── requirements.txt # Project dependencies

├ ── sample\_data.xlsx # Sample input data

## Core Components:

* PineconeAgent\_crew.py
  + Defines the Crew Ai agent, task and crew setup using configuration-driven design.
* Pinequerygen\_agent(self) -> Agent

Creates and returns an agent with LLM capabilities

* + Config loaded from agents.yaml under key ‘pinequerygen\_agent’.
  + Uses self.llm for LLM-based logic.
* Pinecone\_agent\_task(self) -> Task

Defines a task the agent will execute

* + Config loaded from tasks.yaml under key ‘pinecone\_agent\_task’.
* Crew(self) -> crew

Builds the full CrewAI workflow:

* + Dynamically assigns agents (self.agents)
  + Dynamically assigns tasks (self.tasks)
  + Sets process flow as Process.sequential

## Config File:

* agents.yaml

Stores configuration for agents such as:

* + Name
  + Role
  + Description
  + Model settings
* tasks.yaml

Stores the task definitions the agents need to perform:

* + Task type
  + Parameters
  + Expected outputs

## How to Run:

1. Install the Dependencies:
   * pip install -r requirements.txt
   * Go to <https://ollama.com/search> and download the Ollama in your local machine.
   * Download the LLM any (your choice) I’ve used the “mistral:7b” model
2. Set Environment variables in. env: (Optional if using Ollama)

OPENAI\_API\_KEY = your-api-key

1. Run the workflow:

python main.py

* + Enabled the Verbose=True Parameter (this provides the debug information of how model output is) we can get the direct result using
  + Print(result)

## Output: (verbose = True)

PS C:\AI-Agent\BridgedAI> c:; cd 'c:\AI-Agent\BridgedAI'; & 'c:\AI-Agent\BridgedAI\pinecone\Scripts\python.exe' 'c:\Users\bhanu\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\launcher' '54284' '--' 'c:\AI-Agent\BridgedAI\main.py'

# Agent: Natural Language to PineCone Query Generator

## Task: Analyze the user’s natural language search query and convert it into a structured Pinecone-compatible vector search JSON query. The response must strictly follow the predefined schema:

- published\_year, published\_month, published\_day (all integers, if applicable) - author (string) - tags (list of strings)

Ensure: - The final output is a \*\*pure JSON object\*\*, with \*\*no comments\*\* or extra text. - Include `"vector": null` if no semantic embedding is inferred. - Use the appropriate date granularity (e.g., published\_year or full range) based on user input. - Always include `topK: 10` and `includeMetadata: true` unless otherwise specified.

Example Input: "Show me articles by Alice Zhang from last year about machine learning."

Example Output: {

"vector": null,

"filter": {

"author": "Alice Zhang",

"published\_year": { "$eq": 2024 },

"tags": { "$in": ["machine learning"] }

},

"topK": 10,

"includeMetadata": true

}

The input query is: Show me articles by Alice Zhang from last year about machine learning.

# Agent: Natural Language to PineCone Query Generator

## **Final Answer**:

```

{

"vector": null,

"filter": {

"author": "Alice Zhang",

"published\_year": { "$eq": 2023 },

"tags": { "$in": ["machine learning"] }

},

"topK": 10,

"includeMetadata": true

}

```

# Agent: Natural Language to PineCone Query Generator

## Task: Analyze the user’s natural language search query and convert it into a structured Pinecone-compatible vector search JSON query. The response must strictly follow the predefined schema:

- published\_year, published\_month, published\_day (all integers, if applicable) - author (string) - tags (list of strings)

Ensure: - The final output is a \*\*pure JSON object\*\*, with \*\*no comments\*\* or extra text. - Include `"vector": null` if no semantic embedding is inferred. - Use the appropriate date granularity (e.g., published\_year or full range) based on user input. - Always include `topK: 10` and `includeMetadata: true` unless otherwise specified.

Example Input: "Show me articles by Alice Zhang from last year about machine learning."

Example Output: {

"vector": null,

"filter": {

"author": "Alice Zhang",

"published\_year": { "$eq": 2024 },

"tags": { "$in": ["machine learning"] }

},

"topK": 10,

"includeMetadata": true

}

The input query is: Find posts tagged with ‘LLMs’ published in June, 2023.

# Agent: Natural Language to PineCone Query Generator

## **Final Answer:**

```json

{

"vector": null,

"filter": {

"tags": { "$in": ["LLMs"] },

"published\_month": { "$eq": 6 },

"published\_year": 2023

},

"topK": 10,

"includeMetadata": true

}

```

# Agent: Natural Language to PineCone Query Generator

## Task: Analyze the user’s natural language search query and convert it into a structured Pinecone-compatible vector search JSON query. The response must strictly follow the predefined schema:

- published\_year, published\_month, published\_day (all integers, if applicable) - author (string) - tags (list of strings)

Ensure: - The final output is a \*\*pure JSON object\*\*, with \*\*no comments\*\* or extra text. - Include `"vector": null` if no semantic embedding is inferred. - Use the appropriate date granularity (e.g., published\_year or full range) based on user input. - Always include `topK: 10` and `includeMetadata: true` unless otherwise specified.

Example Input: "Show me articles by Alice Zhang from last year about machine learning."

Example Output: {

"vector": null,

"filter": {

"author": "Alice Zhang",

"published\_year": { "$eq": 2024 },

"tags": { "$in": ["machine learning"] }

},

"topK": 10,

"includeMetadata": true

}

The input query is: Anything by John Doe on vector search?

# Agent: Natural Language to PineCone Query Generator

## **Final Answer:**

```json

{

"vector": null,

"filter": {

"author": "John Doe"

},

"topK": 10,

"includeMetadata": true

}

```

## Improvements:

* **Model Upgrade for Precision**:  
  Currently, the project uses an **open-source LLM**, which provides basic natural language understanding. However, integrating **OpenAI's GPT-4 or GPT-4-turbo** can significantly improve the **precision and relevance of query responses**, especially when dealing with ambiguous or complex queries over vector databases like Pinecone.
* **Dynamic Query Refinement**:  
  Implementing an **iterative query refinement mechanism** using an LLM agent can help rephrase or optimize user queries before sending them to Pinecone for retrieval, boosting accuracy.
* **Add Feedback Loop**:  
  Allow agents to receive and incorporate feedback on the returned results (e.g., precise Json query to retrieve the relevant info), and use it to fine-tune subsequent searches or re-rank results.
* **Parallel Agent Execution**:  
  Enhance performance by using Process.parallel in Crew AI for non-dependent tasks, which is useful when scaling across multiple agents.