Business Co-pilot: A Multi-Agent System for Your Ideas

ACE Group

November 11, 2025

Sommario

This report describes the design and implementation of "Business Co-pilot," a Business Intelligence application based on a multi-agent system. The project's goal is to provide a preliminary feasibility and risk analysis for a business idea entered by the user. Using local Large Language Models (LLMs) via Ollama, a vector database (ChromaDB) for Retrieval-Augmented Generation (RAG) on Italian business data, and a web search component, the system extracts context, gathers data, generates visualizations, and produces a final strategic report. This document outlines the system architecture, the role of each agent, the technical challenges encountered (particularly in RAG reliability and JSON extraction), and the solutions adopted, including the addition of a symbolic risk assessment module via a Probability/Impact matrix.

Indice

1 Introduction

Starting a new business venture is characterized by a high degree of uncertainty. Entrepreneurs must rapidly assess the feasibility of their ideas by analyzing the market context, competition, and potential risks. Our project, "Business Co-pilot," aims to address this need through an interactive application built in Streamlit.

The user inputs a business idea in natural language (e.g., "I want to open a gourmet pizzeria in Rome"). The system then activates a pipeline of autonomous software agents to decompose the request, gather relevant data from both internal sources (a CSV database of Italian companies) and external sources (web search), and finally synthesize this information into a comprehensive strategic analysis.

The multi-agent architecture, orchestrated via LangChain and powered by local LLMs, allows for a clear separation of concerns, improving the system's modularity and reliability.

1.1 Project Objective

The primary objective was to evolve from a purely descriptive system (which only displays data) to an analytical and prescriptive one. To enrich the project, a symbolic analysis component has been integrated. This component does not merely display competition data but uses it to calculate a qualitative risk level (Low, Medium, High) through a Probability/Impact matrix.

2 Architecture and Technology Stack

The system is built on a pipeline architecture, where the output of one agent becomes the input of the next.

2.1 Technology Stack

The choice of stack was driven by the need for flexibility and local operation:

- **Streamlit:** For a fast and interactive user interface (UI).
- **Ollama (Gemma3:4b & Llama 3.2):** We specialized the models: `Gemma3:4b` for JSON extraction and sentiment analysis, and `Llama 3.2` for report generation and complex analysis.
- **LangChain & LangChain-Ollama:** The framework used to orchestrate the agents and connect them to the Ollama models.
- **ChromaDB & Sentence-Transformers:** The core of our RAG. Sentence-Transformers (specifically `nickprock/sentence-bert-base-italian-uncased` for better Italian understanding) generate embeddings for the CSV data, which are then stored and queried in ChromaDB.
- **Streamlit-Elements (Nivo):** To create interactive charts (Pie and Bar) not natively available in Streamlit.
- **DuckDuckGo Search (DDGS):** To provide real-time web search context.

3 Agent Roles and Functions

The system's effectiveness lies in the specialization of five distinct agents, executed in the following order:

3.1 Agent 1: DeconstructorAgent

Its sole task is to receive the user's raw input and translate it into a structured JSON format. This intent-understanding step is fundamental to correctly initiating the pipeline.

```
{
  "settore": "ristorazione",
  "tipo_business": "pizzeria gourmet",
  "target": "clientela locale e turistica interessata a prodotti di alta
    qualità e ingredienti ricercati",
  "dimensione_geografica": "Roma",
  "regione_specifica": "Lazio"
}
```

Listing 1: Example output from DeconstructorAgent

3.2 Agent 2: IntelligenceAgent

This is the data-gathering agent. It is divided into two main methods:

- **gather_data:** Executes RAG queries on ChromaDB (filtering by region and sector) and the web search. This method contains the crucial "ATECO translation" logic (`_find_best_ateco_description`).
- **synthesize_brief:** Writes a prose summary (the Market Brief) combining the RAG and web analysis.

3.3 Agent 3: DataExtractorAgent

This agent was created to solve JSON extraction failures. Its task is to receive the raw RAG data (text strings) from Agent 2 and transform them into clean JSON lists ready for visualization. Since the LLM proved unreliable at parsing complex strings, this agent also includes a fallback parser based on regular expressions to ensure robustness.

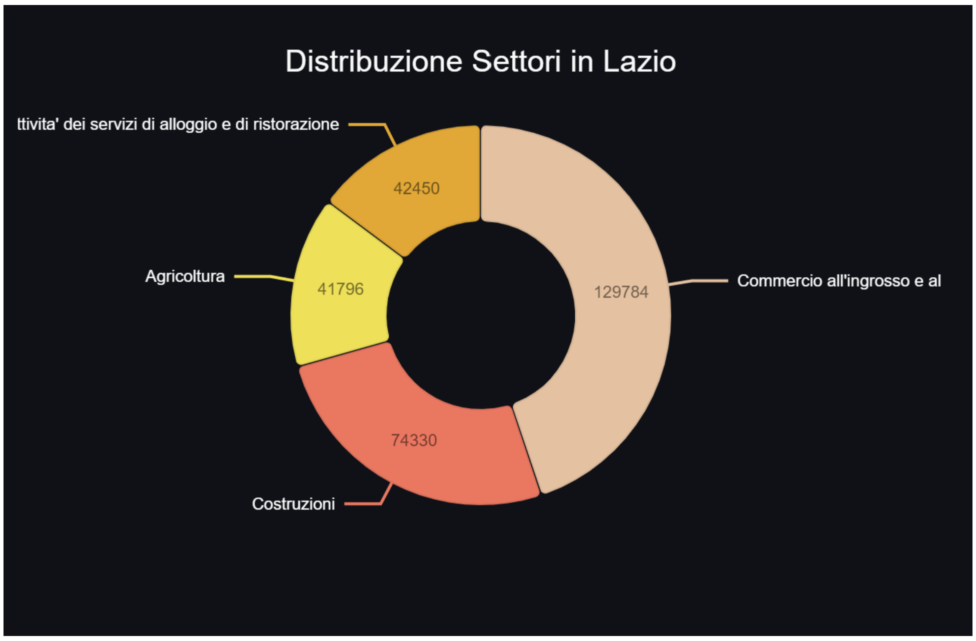


Figura 1: Example Pie Chart generated by Nivo, showing local market distribution.

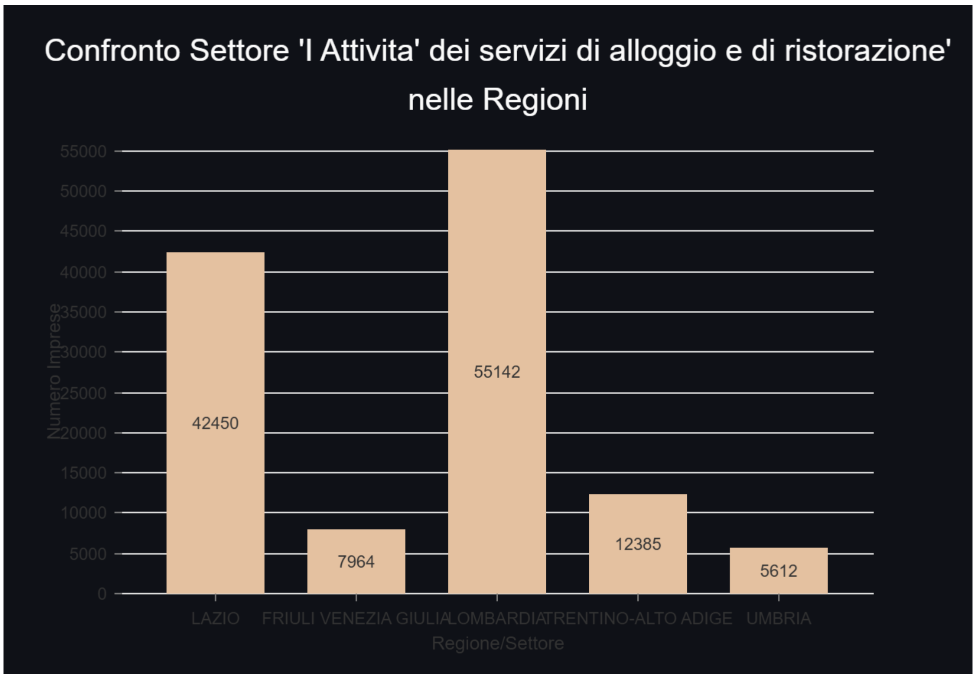


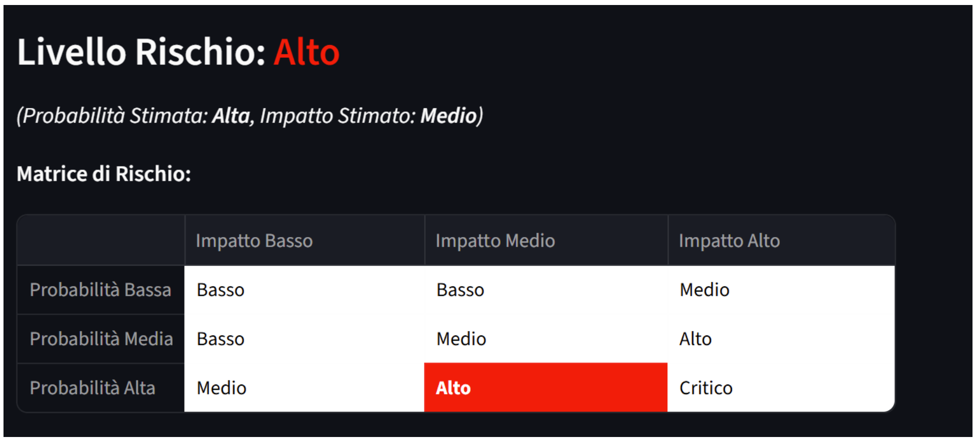
Figura 2: Example Bar Chart generated by Nivo, illustrating national sector comparison across regions.

3.4 Agent 4: RiskScoringAgent

This agent implements the required analytical component:

1. Performs sentiment analysis on the web results (using `ollama_model`).
2. Calculates raw Key Risk Indicators (KRIs).

3. Maps these KRIs to qualitative scales of Probability (based on local competition) and Impact (based on dominance and sentiment).
4. Determines a final risk level ("Low," "Medium," "High," "Critical") using a 3×3 matrix.



Livello Rischio: Alto

*(Probabilità Stimata: **Alta**, Impatto Stimato: **Medio**)*

Matrice di Rischio:

	Impatto Basso	Impatto Medio	Impatto Alto
Probabilità Bassa	Basso	Basso	Medio
Probabilità Media	Basso	Medio	Alto
Probabilità Alta	Medio	**Alto**	Critico

Figura 3: Example Risk Matrix visualisation, mapping Probability and Impact to a qualitative risk level.

3.5 Agent 5: AnalystAgent

This is the final agent. It receives all previous outputs (context, brief, chart data, and the risk assessment) and combines them into a final strategic report, which includes a SWOT analysis (Strengths, Weaknesses, Opportunities, Threats). The report is streamed to the user word-by-word.

4 Technical Challenges and Solutions

4.1 RAG Reliability (Semantic Mismatch)

The biggest challenge was ensuring that RAG queries returned correct data.

- **Problem 1 (Semantics):** The user writes "pizzeria," but the database contains "Attività dei servizi di alloggio e di ristorazione."
- **Solution 1:** We implemented the `_find_best_ateco_description` function. This function uses semantic search to find the most similar documents, but we added keyword matching logic on the top five results to select the most relevant ATECO description, drastically increasing translation accuracy.
- **Problem 2 (Filters):** Queries were failing silently.
- **Solution 2:** Debugging revealed two errors:
  1. **Case mismatch:** Agent 1 produced "Campania," but the CSV contained "CAMPANIA." The solution was to apply `.upper()` to the `regione_filtro` variable before querying ChromaDB.
  2. **Multiple filters:** ChromaDB does not accept direct multiple filters (e.g., `where={"regione": "X", "settore": "Y"}`). The solution was to adopt the correct syntax `where={"$and": [{"regione": "X"}, {"settore": "Y"}]}`.

4.2 LLM Reliability (JSON Extraction)

The LLMs proved unreliable at generating complex JSON or remaining silent.

- **Problem (Chatty LLM):** The model would add text before and after the JSON (e.g., "Here is the JSON: {...} I hope this helps!"), breaking `json.loads`.
- **Solution:** We implemented a robust `extract_json_from_response` function that does not use greedy regex. Instead, it balances curly braces to extract only the first valid JSON block, ignoring all other text.
- **Problem (Lazy LLM):** Asking the LLM to read raw RAG data and build the correct Nivo JSON (Agent 3) often failed.
- **Solution:** We simplified the task. The LLM now only needs to extract the data into a list format. The Python code then manually builds the final `charts_data_object` for Nivo.

5 Results and User Interface

The final application provides the user with a comprehensive analytical dashboard in seconds. The output is divided into clear sections:

- **Context Extracted:** The structured JSON from Agent 1.
- **Market Charts (Nivo):** The Pie chart (local sector distribution) and Bar chart (national sector comparison) from Agent 3's data.
- **Risk Assessment:** The symbolic risk matrix and qualitative rating (e.g., "Medium Risk") from Agent 4.
- **Market Brief (RAG):** The prose summary of RAG and web data from Agent 2.
- **Final Strategic Report:** The streamed, in-depth analysis from Agent 5, including the SWOT.

6 Conclusions

The project successfully achieved its goal of creating an analytical business assistant. The multi-agent architecture proved flexible, allowing us to easily add the `RiskScoringAgent` component as required, evolving the project from descriptive to evaluative.

The main challenges centered on the interface between symbolic/structured logic (our CSV database and filters) and semantic logic (embeddings and LLMs). The implemented solutions (ATECO translation, robust parsers, `$and` filters) have created a resilient pipeline.