UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Taboada Dacosta, Sandra. *Skills Mobility Agent : A Multi-Agent AI System for Natural Language Internal Talent Matching*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Intel·ligència Artificial)

This version is available at https://ddd.uab.cat/record/336307

under the terms of the CC BY-SA license.

Skills Mobility Agent: A Multi-Agent AI System for Natural Language Internal Talent Matching

Sandra Taboada Dacosta

June 27, 2026

Abstract

This project presents a multi-agent AI system that enables HR managers to query a pool of 150 employees using natural language, automating candidate discovery, ranking, and skill-gap analysis. Developed as an internal initiative at Avanade, a global technology consulting company, the system is built on a dynamic orchestration layer powered by GPT-4o function calling, which routes queries to specialist sub-agents responsible for semantic search, weighted scoring, headcount filtering, and location-based lookup. Candidate ranking combines semantic similarity via Azure AI Search vector indexing with explicit skill coverage, using a weighted formula that prioritises profile–vacancy alignment. The system supports multi-turn conversation, enabling follow-up questions that reference prior context. Evaluation was conducted on a 150-question benchmark spanning seven query types, using two complementary automatic metrics: an LLM-based judge (avg. 0.75) and RAGAS answer relevancy (avg. 0.69). Results demonstrate that the agentic approach effectively handles diverse HR query types, with strong performance on out-of-scope refusals and strong results on candidate search and gap analysis, whilst multi-step queries present the greatest complexity.

Keywords: multi-agent systems, retrieval-augmented generation, semantic search, HR automation, internal mobility, LLM orchestrator, skill-gap analysis

1 Introduction

The growing complexity of talent management in consulting and professional services companies has made internal mobility a strategic priority. Growing organizations pose a challenge for HR managers who need to screen a larger and more geographically dispersed workforce to find the best fit candidates. Fitting talent to the various roles in consulting firms, especially those with international, experiential, and technical diversity, often involves going through spreadsheets and employing human judgment.

Progress in large language models (LLMs) and semantic search has recently led to promising possibilities for the automation of knowledge-intensive tasks. Retrieval-Augmented Generation (RAG) systems [1] integrate neural retrieval and generative language models for domain-specific question answering and have demonstrated strong empirical performance across multiple benchmarks [1, 2].

Additionally, multi-agent systems [3] have advanced this paradigm by decomposing intricate problems into specific sub-problems that coordinated AI agents handle, thereby increasing system modularity, robustness, and simplifying the understanding of the system. Given a synthetic workforce of 150 employees spread across six European nations, this system lets users perform queries like finding top candidates for open positions, gauging an employee's readiness for a different role, counting staff based on skill or location, and addressing follow-up questions in a conversational flow.

The project pursues three objectives:

1. Design and implement a dynamic multi-agent orchestration system that can handle at least five distinct HR query types through GPT-4o's native function calling without relying on a fixed execution pipeline.
2. Create a hybrid candidate ranking method that merges high-dimensional semantic vector searches with explicit skill-coverage scoring, while defining clear quality thresholds that gate final recommendations.
3. Evaluate the system against a 150-question benchmark using two complementary automatic metrics—an LLM-as-judge and RAGAS answer relevancy—to characterise performance across all supported query types and response time distributions.

• Contact E-mail: 1586299@uab.cat
• Supervised by: María Pilar Dellunde Clave
• Academic Year 2025/26

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [1] solves a fundamental problem of large language models which is that they are limited to the information contained in their training data. RAG addresses this by performing a retrieval process to find pertinent information in an external knowledge base and then a generation process where the retrieved information is given to the LLM as input and used to generate a response. This mechanism grounds generated content in concrete, up-to-date information rather than relying solely on the model's parametric knowledge. Implementations of RAG include combined retrieval methods and multi-stage re-ranking systems [2].

Currently, RAG is the core process: employee profiles are represented as vector embeddings and on receipt of a query they are retrieved on the basis of similarity of the embedded representations and given to the orchestrator to generate a response.

2.2 Multi-Agent LLM Systems

Decomposing complicated reasoning problems over several collaborating large language models (LLMs) has become quite popular [3]. Systems like AutoGen [4] allow for the assembly of agent networks that communicate, assign smaller tasks, and have a common state. One of the fundamental choices in the architecture of agentic systems is the routing architecture: fixed pipelines (always calling the next agent) are predictable, while dynamic orchestration via OpenAI function calling [5, 6] is more flexible and can manage a broader range of input types. Function calling is described in the GPT-4 technical report [5]; ReAct [7] is cited for its reasoning-action synergy framework. This work adopts the latter approach, using GPT-4o's native function calling [6] to allow the orchestrator to decide at runtime which agents to invoke and in which order.

2.3 Semantic Search and Vector Databases

Dense retrieval using pre-trained embedding models has become the standard approach for semantic similarity tasks [8]. Embeddings from models such as text-embedding-3-small map text into high-dimensional vector spaces where cosine similarity approximates semantic relevance. Efficient approximate nearest-neighbor (ANN) search at scale is enabled by algorithms such as HNSW (Hierarchical Navigable Small World graphs) [9], which provide sub-linear query time with high recall. Azure AI Search implements HNSW natively, making it well-suited for production-grade semantic retrieval over structured document corpora.

2.4 AI in HR and Talent Matching

Internal mobility consistently delivers better outcomes than external hiring in organisational contexts studied primarily in North American firms [10]. Automated matching of candidates to job descriptions has been studied extensively in the information retrieval and NLP literature [11]. However, most prior systems operate as offline ranking tools rather than interactive conversational agents. The present work differs by integrating semantic retrieval, scoring, and gap analysis into a unified conversational interface capable of multi-turn dialog.

2.5 Evaluation of LLM-Based Systems

System evaluation employs two complementary approaches: LLM-as-judge [5, 12], where a separate LLM evaluates response quality using a detailed rubric; and RAGAS [13], which measures answer relevancy by generating reverse questions from the answer and computing their similarity to the original query. Both approaches are used in this work to provide complementary views of system quality.

3 Methodology and Work Plan

3.1 Development Approach

The project followed an iterative, milestone-driven development methodology divided into five phases. Each phase produced a testable increment and informed the design decisions of the next. This approach was chosen because the scope of an open-ended conversational HR system is difficult to fully specify upfront: early testing of the orchestration layer revealed routing edge cases that required additional agent types, and early evaluation runs identified the need for multi-step query support not included in the initial specification.

3.2 Development Phases

Phase 1 – Infrastructure and Data Preparation: Azure OpenAI, Azure AI Search, and Azure Blob Storage resources were provisioned. The 150-employee synthetic dataset was designed, generated, and validated. The employee indexing pipeline was implemented and all profiles were vectorised using text-embedding-3-small and uploaded to the HNSW-configured Azure AI Search index.

Phase 2 – Core Agent Implementation: The orchestrator (AG1) and the five specialist agents (AG3, AG4, AG6, AG7, AG8) were implemented as Python functions registered as OpenAI tools. The deterministic scoring formula and the three quality-gate thresholds were defined and unit-tested against hand-crafted employee profiles. The work modality derivation rule was implemented and validated for all country combinations.

Phase 3 – Web Interface and Streaming: The Flask application was developed with the asynchronous SSE streaming pattern, allowing the client to receive real-time agent activation events. The conversation history mechanism was integrated into the messages array submitted to the Azure OpenAI API, enabling multi-turn follow-up queries.

Phase 4 – Evaluation Framework: The 150-question benchmark was designed to cover all seven query types, with deterministic ground-truth answers derivable from the employee CSV. The LLM-as-judge evaluator was implemented and the RAGAS answer relevancy was integrated via the RAGAS framework [13].

Phase 5 – Analysis and Documentation: Results were analysed and visualised; the system was subsequently pre-

sented at the Avanade Spain AI Hub; and the final paper was written.

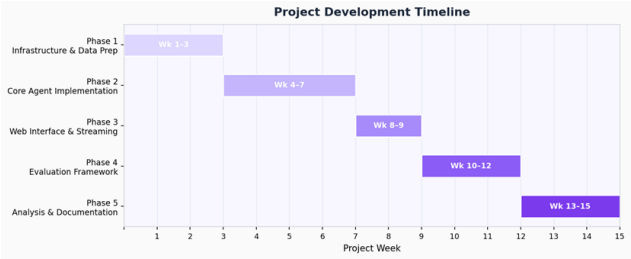


Fig. 1: Project development timeline showing the five phases across 15 weeks.

3.3 Tools and Technologies

The implementation language is Python 3.11 throughout. All cloud services are on the Microsoft Azure platform, chosen for its native integration with the Azure OpenAI deployment required by the project.

Core runtime: Python 3.11, Flask 3.0 (web server), python-dotenv (configuration).

AI services: Azure OpenAI — GPT-4o (gpt-4o-2024-05-13) for orchestration, formatting, and LLM evaluation; text-embedding-3-small (1536 dimensions) for vector generation.

Search and storage: Azure AI Search (HNSW index, cosine similarity), Azure Blob Storage (log persistence).

Evaluation: RAGAS and custom LLM-as-judge rubric.

Development: Git (version control), Visual Studio Code, Postman (API testing).

4 System Design

4.1 Overview

The Skills Mobility Agent follows a hub-and-spoke multi-agent architecture (see Fig. 2). A central orchestrator agent (AG1) receives all user queries and uses function calling to dispatch work to five specialized sub-agents. The system maintains a conversation history buffer that allows later queries to reference entities mentioned in earlier turns.

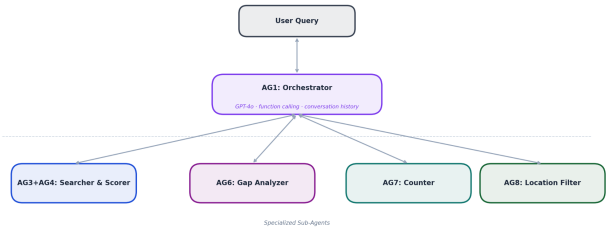


Fig. 2: Multi-agent architecture of the Skills Mobility Agent.

4.2 Agent Descriptions

AG1 Orchestrator: The primary interface for user interactions, utilising GPT-4o to act as a centralised HR coordinator. It leverages five distinct tools through a maximum of six iterative calls per request to manage intricate workflows. This component maintains a comprehensive dialogue state and applies a professional linguistic filter to finalise outputs.

AG3 Semantic Searcher: This agent transforms role requirements into vector embeddings using text-embedding-3-small and performs semantic retrieval over the Azure AI Search index. Candidate profiles are ranked according to cosine similarity, while additional constraints such as location, seniority, or work modality are applied to refine the results.

AG4 Scorer: Responsible for the secondary refinement of candidate pools through the application of a weighted scoring algorithm. It identifies specific competence deficits, benchmarks individuals against established quality requirements, and generates strategic advice based on these evaluative thresholds.

AG6 Gap Analyser: Focuses on individualized readiness assessments by comparing specific staff records against defined skill sets. By extracting data from the primary repository, it calculates alignment percentages and issues recommendations based on a minimum coverage benchmark of 80%.

AG7 Counter: Facilitates quantitative queries regarding workforce distribution. It applies multi-criteria Boolean logic to the dataset to provide precise headcounts categorised by geographic presence, professional seniority, and employment modality.

AG8 Location Filter: Returns the full list of employee records filtered by country and/or work modality. It is used for availability queries rather than ranking.

4.3 Data Model

Each employee record contains the following fields:

Field	Description
empleado_id	Unique identifier (E001–E150)
job_role	Current job title
departamento	Department
hard_skills	Comma-separated list of skills
experience_level	Senior / Mid / Entry
pais_operacion	Country of operation
modalidad	Derived work modality (on/near/off-shore)
embedding_descripcion	1536-dim vector (Azure AI Search)

TABLE 1: Employee record fields.

Note: Field names follow the source dataset's Spanish naming convention; English equivalents are given in the Description column.

The embedding is generated from a concatenation of the employee's role, department, and skills, capturing their overall professional profile in a single vector representation.

4.4 Candidate Scoring Formula

The final candidate score is computed as a weighted combination of three signals, as shown in Eq. (1):

$$\begin{aligned} \text{score}_{\text{final}} &= 0.60 \times \text{semantic_similarity} \\ &+ 0.28 \times \text{required_skills_coverage} \\ &+ 0.12 \times \text{desired_skills_coverage} \end{aligned} \tag{1}$$

- **semantic_similarity**: Cosine similarity between the vacancy and candidate embeddings.
- **required_skills_coverage**: Fraction of required skills present in the candidate's skill list.
- **desired_skills_coverage**: Fraction of desired skills present in the candidate's skill list.

The weights reflect the relative importance of each signal: semantic alignment is the primary driver, while explicit skill coverage provides a secondary refinement. Three quality thresholds gate the final recommendation:

- Semantic similarity ≥ 0.60
- Required skills coverage ≥ 0.50
- Final weighted score ≥ 0.55

If the top-ranked candidate fails any threshold, the system provides a remedial suggestion (internal upskilling, broadened search criteria, or external recruitment).

4.5 Work Modality Derivation

Work modality (on-shore, near-shore, off-shore) is derived dynamically at query time from the employee's country and the vacancy's country, rather than stored as a static field. The derivation rule is:

- Same country $\rightarrow$ on-shore
- Different countries, both within the defined European set (Spain, France, Germany, Portugal, Netherlands, Italy) $\rightarrow$ near-shore
- Otherwise $\rightarrow$ off-shore

This approach avoids data redundancy and correctly handles scenarios where the same employee may have different modality classifications depending on the vacancy location.

5 IMPLEMENTATION

5.1 Infrastructure

The system is deployed on Microsoft Azure and relies on three managed services:

- **Azure OpenAI**: provides access to GPT-4o and text-embedding-3-small.
- **Azure AI Search**: hosts the employee vector index with HNSW-based approximate nearest-neighbor search.
- **Azure Blob Storage**: used for persistent logging of agent execution traces.

5.2 Employee Indexing Pipeline

Before the system can serve queries, each employee profile must be vectorized and indexed. The indexing performs the following steps:

1. Reads 150 employee records from the employee CSV dataset described in Section III-A.
2. Constructs a text description for each employee by concatenating role, department, and skills.
3. Generates a 1536-dimensional embedding for each description using text-embedding-3-small via the Azure OpenAI API.
4. Creates an Azure AI Search index with the schema described in Table 1, configuring the embedding_description field with HNSW vector search (cosine similarity metric, 1536 dimensions).
5. Uploads documents in batches of 50.

This process is executed once during setup. The resulting index supports vector (k-NN) search as well as hybrid search with additional filtering conditions.

5.3 Web Interface and Streaming

The user-facing interface is a chat UI served by a Flask application. Interaction follows an asynchronous SSE (Server-Sent Events) pattern:

1. The user submits a message via `POST /api/chat`, which enqueues the query and returns a `task_id`.
2. The client opens a `GET /api/stream/<task_id>` endpoint and receives a stream of structured events as the agent executes.
3. Events include `agent_start`, `tool_start`, `tool_end`, `agent_end`, `error`, and `end`, providing real-time transparency into which agents are active.

This design allows the UI to render agent activity progressively rather than waiting for the full response, improving perceived responsiveness, as agent progress is surfaced incrementally rather than delivered as a single delayed response.

5.4 Conversation History

The full message history (user messages, assistant messages, tool call objects, and tool results) is maintained client-side and submitted with each new query. The orchestrator receives this history as part of the messages array passed to the Azure OpenAI chat completions API, enabling it to resolve conversational references such as "the first candidate mentioned" or "their skill gaps" without requiring a dedicated memory agent.

6 Ethics and Regulatory Compliance

6.1 EU AI Act Compliance

The Skills Mobility Agent was designed with the requirements of Regulation (EU) 2024/1689 (the EU AI Act) [14] in mind. Under Annex III of the Act, AI systems used for employment decisions — including candidate selection and internal mobility recommendations — are classified as high-risk applications. Although the present system operates as a research prototype using a synthetic dataset, the implementation anticipates the compliance obligations of Articles 12–14 that would apply upon production deployment.

Article 12 mandates that high-risk AI systems maintain automatically generated logs sufficient to ensure traceability throughout the system lifecycle. Every conversation is persisted as a structured JSON log file in the `logs/` directory. Each record contains: timestamp (UTC ISO-8601), task ID (UUID), the full user message, the ordered list of agents activated, all intermediate tool call inputs and outputs, and the final response text. This log format enables post-hoc audit of every system decision.

Article 13 requires sufficient transparency for users to interpret system outputs. The web interface presents itself explicitly as an AI-powered tool; agent activation events (`agent_start`, `tool_end`, `agent_end`) are streamed to the client in real time so users can observe which agents contributed to each response. The deterministic scoring formula makes every ranking decision fully inspectable: HR users can examine the per-candidate score breakdown stored in the tool result to understand why a specific candidate was ranked above another, satisfying the explainability requirement without relying on black-box model outputs for the final ranking step.

Article 14 requires that humans maintain appropriate oversight of high-risk AI systems. The Skills Mobility Agent is explicitly designed as a decision-support tool: all recommendations are presented with numeric scores and supporting evidence, and are intended to inform — not replace — the professional judgement of HR managers. The out-of-scope refusal mechanism (AG1 declines queries outside the HR domain, achieving an LLM Judge score of 0.97 on out-of-scope queries) further constrains the system to its intended use context, reducing the risk of misuse.

6.2 Data Protection and GDPR

The employee dataset used throughout this project was generated synthetically using controlled random sampling; no real personal data was involved at any stage. The 150 employee records contain only professionally relevant attributes — job role, department, technical skills, experience level, and country of operation — none of which constitute special-category data under Article 9 of the GDPR.

6.3 Fairness and Non-Discrimination

A central ethical concern in automated talent matching is the risk of perpetuating or amplifying workforce biases. The scoring formula uses only professionally relevant signals: semantic similarity between vacancy and candidate embeddings, and skill coverage fractions. No demographic attributes — gender, age, nationality, or ethnicity — are present in the dataset or used in any scoring step. The use of ESCO v1.1 [15] as the skill taxonomy further mitigates lexical bias: ESCO provides EU-standardised, language-neutral skill descriptions that reduce the risk of systematically advantaging candidates whose skills happen to use particular terminology.

It is important to acknowledge, however, that the text-embedding-3-small model is a general-purpose model trained on large web corpora, which may encode societal biases present in its training corpus. If the model associates certain skill descriptions with particular demographic groups, these associations could subtly influence cosine similarity scores. This is a known limitation of dense retrieval systems [8] and underscores the necessity of human review of all recommendations, particularly for sensitive or senior roles. Future work could evaluate the system for demographic parity across role types as part of a broader fairness audit.

7 Evaluation

7.1 Test Suite Design

A benchmark of 150 questions was constructed to cover the full range of query types supported by the system. Questions were designed to have deterministic, verifiable ground-truth answers derivable from the employee CSV dataset, enabling automatic correctness labelling. The distribution across seven question types is shown in Fig. 3.

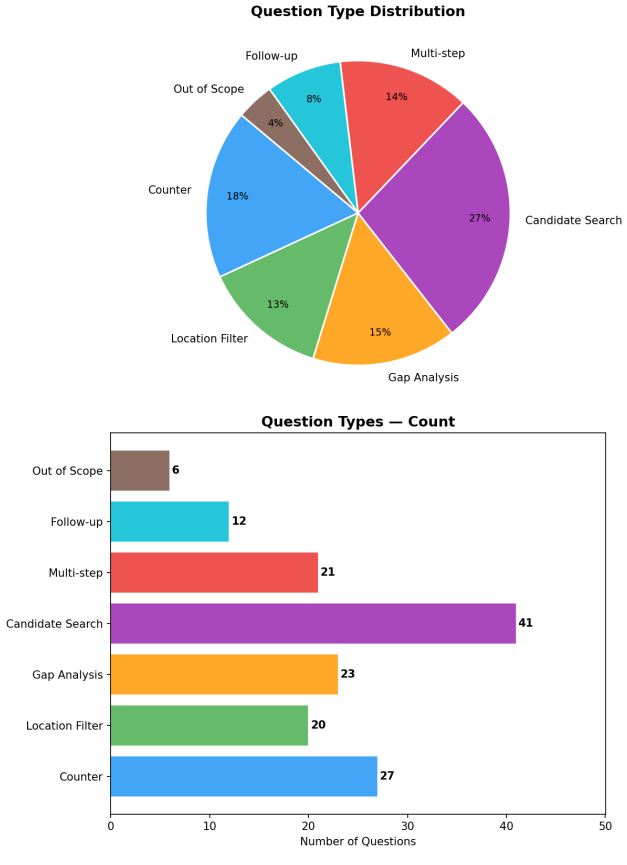


Fig. 3: Test suite composition by question type.

7.2 Evaluation Metrics

Two complementary metrics were used to assess system quality:

LLM Judge. A GPT-4o evaluator with temperature 0 was prompted with a detailed rubric to score each system response on a scale from 0.0 to 1.0. The rubric defines six score levels:

- 1.0: Perfect, correct, complete, and well-formatted.
- 0.8: Good, correct with minor omissions.
- 0.6: Partial, partially correct or incomplete.
- 0.4: Poor, major errors or significant omissions.
- 0.2: Very poor, mostly wrong.
- 0.0: Fail, incorrect or system error.

A special rule scores all correctly refused out-of-scope queries as 1.0. This metric captures factual correctness and response quality as judged by an expert HR evaluator persona.

RAGAS Answer Relevancy. Using the RAGAS framework [13], each system response is evaluated by generating multiple reverse questions (typically three per response). Reverse questions are generated from the answer via GPT-4o; the cosine similarity between the original question and each reverse question is then computed. A higher score indicates that the answer directly and completely addresses what was asked, without introducing irrelevant content. This metric is independent of ground truth labels and captures a different dimension of quality from the LLM judge.

7.3 Results

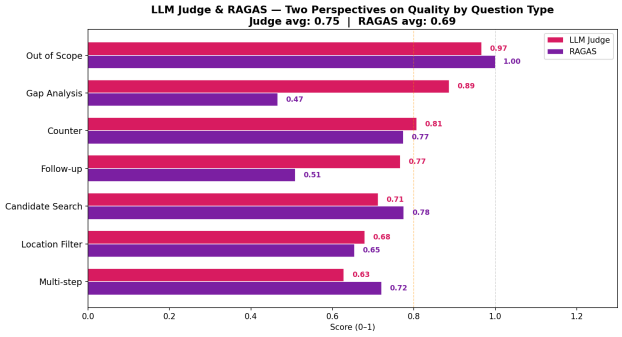


Fig. 4: LLM Judge and RAGAS scores by question type (n=150).

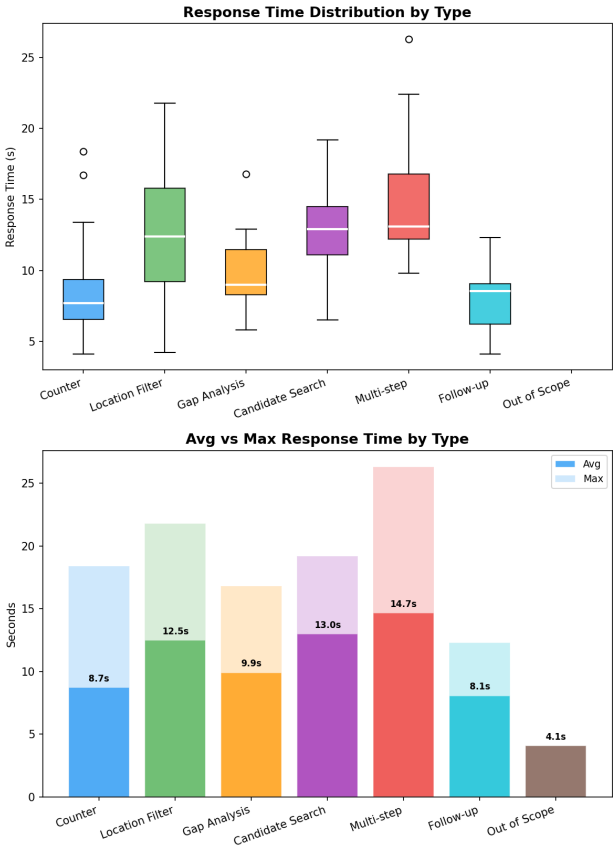


Fig. 5: Response time distribution by question type (n=150). Box plots show median (white line) and interquartile range (IQR); whiskers extend to 1.5× IQR from Q1/Q3; points beyond are individual outliers.

Overall results across all 150 test cases are shown in Fig. 4, which presents LLM Judge and RAGAS scores broken down by question type. The system achieved an overall LLM Judge mean of 0.75 and a RAGAS mean of 0.69.

Out-of-scope queries achieved a near-perfect LLM Judge score of 0.97, confirming that the orchestrator reliably refuses off-topic requests. Gap analysis queries scored highest among functional types (0.89), reflecting the precision of the skill-matching logic. Counter queries scored 0.81 and follow-up queries scored 0.77, demonstrating that headcount filtering and conversation history resolution are well-handled. Candidate search queries scored 0.71, reflecting the core strength of the semantic retrieval and scoring pipeline. Location filter queries scored 0.68. Multi-step queries scored lowest among functional types (0.63), attributable to the compounding complexity of chaining multiple tool calls, where any intermediate imprecision propagates to the final response.

RAGAS scores follow a different pattern. Out-of-scope queries achieved a perfect score of 1.00, as refusal responses are concise and directly address the question. Candidate search (0.78) and counter queries (0.77) also scored strongly, reflecting that these responses are focused and directly relevant to what was asked. Multi-step queries scored 0.72 despite having the lowest LLM Judge score, suggesting that, even when the answer is incomplete, the response remains on-topic. Location filter queries scored 0.65. The largest divergence from the LLM Judge appears in gap analysis (0.47) and follow-up queries (0.51), where responses

are longer and richer in context, which reduces the RAGAS relevancy score even though the content is factually correct and useful.

Response times ranged from approximately 4 seconds (out-of-scope refusals, which require only a single orchestrator decision) to approximately 26 seconds (complex multi-step queries requiring multiple sequential tool calls), with a typical range of 8–12 seconds for candidate search queries. This is acceptable for an HR decision-support tool where queries are not expected in real-time interactive settings.

8 Discussion

8.1 Metric Divergence: LLM Judge vs. RAGAS

A notable finding across the evaluation is the divergence between the two metrics for certain query types. Gap analysis queries achieve the highest LLM Judge score but one of the lowest RAGAS scores. The LLM judge evaluates factual correctness and completeness: a response that correctly identifies missing skills and provides an actionable upskilling plan receives a high score. RAGAS answer relevancy, by contrast, measures how focused the response is on the question by generating reverse questions and checking their similarity to the original query. Gap analysis answers are typically long — they include skill lists, gap explanations, development recommendations, and seniority assessments — and this richness reduces the RAGAS score even when the content is genuinely useful to the HR professional.

Follow-up responses resolve conversational references ("the third candidate", "their skill gaps") by repeating context from the conversation history. This additional context makes the response richer from the perspective of a human reader, but lowers the RAGAS relevancy score because it introduces content beyond a literal reading of the follow-up question. This metric divergence demonstrates the importance of combining multiple evaluation approaches: neither metric alone provides a complete picture of system quality for this domain.

8.2 Query Type Performance Analysis

Out-of-scope queries achieved a near-perfect LLM Judge score of 0.97, confirming that the orchestrator robustly refuses off-topic requests. This is a critical safety property: an HR decision-support tool that can be manipulated into answering unrelated queries represents both a misuse risk and a reliability concern. The high refusal accuracy is attributable to the GPT-4o orchestrator, which was prompted with an explicit scope boundary and consistently declines queries about geography, general knowledge, or non-HR domains.

Multi-step queries scored lowest among functional types (LLM Judge 0.63). These queries require the orchestrator to sequence multiple tool calls — typically a candidate search followed by a gap analysis on the top result. The lower score reflects the compounding effect of intermediate imprecision: if the initial search returns a marginally relevant candidate, the subsequent gap analysis is applied to a suboptimal target, propagating the error. This highlights a known challenge in multi-agent systems: error accumulation across a chain of dependent tool calls.

Mitigating this would require either re-ranking with lookahead (i.e., prospectively assessing candidate quality before committing to the subsequent tool call), or adding a verification step between the search and gap-analysis stages.

Candidate search queries (LLM Judge 0.71, RAGAS 0.78) show a balanced performance profile where both metrics agree, suggesting the system produces focused and factually accurate responses. This is the most frequent query type in the benchmark (41 of 150) and represents the core capability of the system, so the consistent performance across both metrics is a positive signal for the system's primary use case.

9 Conclusions

This paper presented the Skills Mobility Agent, a multi-agent AI system for natural language internal talent matching in a consulting firm context. The system demonstrates that a dynamic function-calling orchestration architecture, combined with semantic vector search and weighted skill-based re-ranking, can handle the diverse query types that arise in real-world HR decision support, achieving an overall LLM Judge mean of 0.75 across all query categories — from candidate search and skill-gap analysis to headcount reporting and conversational follow-ups.

The evaluation on a 150-question benchmark showed an LLM Judge average of 0.75 and a RAGAS answer relevancy average of 0.69. The system's logging of all agent activations and tool calls supports the traceability requirements of the EU AI Act [14] for AI systems deployed in high-stakes decision contexts such as HR. The dual-metric evaluation approach provided complementary insights: the LLM judge captured correctness and completeness, while RAGAS measured answer focus and relevance.

Several directions for future work are worth pursuing. First, the employee dataset could be expanded beyond 150 profiles and enriched with dynamic data such as current project assignments, availability, and performance history. Second, the system could be extended with proactive alerts — for example, notifying HR managers when a vacancy has remained open for a threshold period and no internal candidate meets the quality thresholds. Finally, a fine-tuned embedding model trained on HR-specific vocabulary (roles, certifications, frameworks) could improve the quality of semantic similarity estimates beyond what a general-purpose model provides.

Acknowledgments

I would like to express my sincere gratitude to María Pilar Dellunde Clave for her support, guidance, and valuable supervision throughout the development of this Final Degree Project as my university advisor.

I would also like to thank Avanade for providing me with the opportunity to carry out this project within the company. In particular, I would like to express my appreciation to Ignacio García for his continuous support and for reviewing the progress of the project on a weekly basis, as well as to

Anna de Llorens for making this experience possible. I am especially grateful to both of them for giving me the opportunity to present my work to the entire AI Hub, an experience that greatly enriched the development of this project.

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[2] Y. Gao, Y. Xiong, X. Gao, *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[3] Z. Xi, W. Chen, X. Guo, *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.

[4] Q. Wu, G. Bansal, J. Zhang, *et al.*, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.

[5] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[6] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[7] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations*, 2023.

[8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019, pp. 3980–3990.

[9] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[10] M. Bidwell, "Paying more to get less: The effects of external hiring versus internal mobility," *Administrative Science Quarterly*, vol. 56, no. 3, pp. 369–407, 2011.

[11] F. Qin, Z. Chen, Y. Zhang, *et al.*, "A survey of job recommendation and talent matching: Methods, trends, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 9873–9893, 2023.

[12] L. Zheng, W.-L. Chiang, Y. Sheng, *et al.*, "Judging LLM-as-a-judge with MT-bench and chatbot arena," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[13] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, 2024, pp. 150–158.

[14] European Parliament and Council of the European Union, "Regulation (eu) 2024/1689 laying down harmonised rules on artificial intelligence (artificial intelligence act)," 2024.

[15] European Commission, "ESCO: European skills, competences, qualifications and occupations (v1.1)," 2023, https://esco.ec.europa.eu.

APPENDIX

A.1 Agent Activation Frequency

Across the 150 test cases, the orchestrator activated agents with the following frequency:

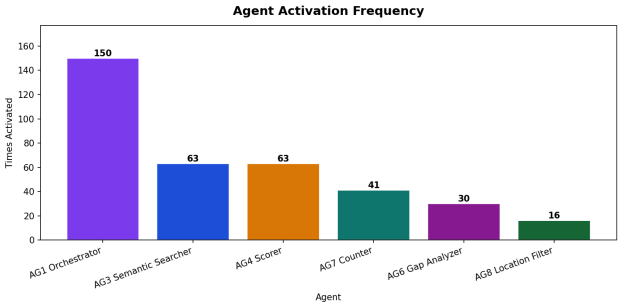


Fig. 6: Agent activation frequency across 150 test cases (n=150).