EntreLocate: An AI Assistant for Business Location and Risk Assessment

Merve Bozdağ
Dept of Software Engineering,
TED University
Ankara, Türkiye
merve.bozdag@tedu.edu.tr

Selin Göç
Dept of Computer Engineering,
TED University
Ankara, Türkiye
selin.goc@tedu.edu.tr

Elif Sude Memiş
Dept of Software Engineering,
TED University
Ankara, Türkiye
elifsude.memis@tedu.edu.tr

Miray Aday
Dept of Computer Engineering,
TED University
Ankara, Türkiye
miray.aday@tedu.edu.tr

Kasım Murat Karakaya
Dept of Software Engineering,
TED University
Ankara, Türkiye
murat.karakaya@tedu.edu.tr

2025 10th International Conference on Computer Science and Engineering (UBMK) | 979-8-3315-9975-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/UBMK67458.2025.11206765

***Abstract*— EntreLocate is an AI-powered assistant designed to help entrepreneurs and small company owners find the best places to launch new ventures. To produce context-aware insights, the system integrates real-time spatial and socioeconomic data from OpenStreetMap (OSM) and other official datasets with Large Language Models (LLMs) through grounding approaches. EntreLocate provides thorough risk assessment scores, location-based recommendations, and procurement proposals by analyzing variables including competition density and demographics. The platform's interactive maps, automatic report production, and user-friendly bilingual interface are designed to improve decision-making and lower startup risks. A modular architecture and extensive testing guarantee usability, scalability, and dependability in a variety of circumstances.**

Keywords*— *software engineering, artificial intelligences applications, entrepreneurship, business location optimization, large language model, grounding, risk assessment, geospatial data, natural language processing

I. Introduction

In order to ensure long-term success, choosing the best site for a new business is essential, particularly for small business owners and novice entrepreneurs. A business's viability is greatly influenced by a number of factors, including area demand, socioeconomic dynamics, and local rivalry. However, without technological assistance, gathering and evaluating these factors can be challenging. The EntreLocate project intends to close this gap by offering an intelligent, AI-assisted system that empowers business owners to make informed choices regarding the location of their enterprises and the risks involved.

By examining socioeconomic conditions, local competitiveness, and financial risk considerations, the EntreLocate project aims to develop an AI-powered assistant that helps local business owners choose the best locations for their firms. The system's primary goal is to increase the sustainability and success rate of small enterprises by enabling data-driven decision-making.

Using real-world data, the EntreLocate project will use grounding approaches to increase the language model's response correctness and reliability. We use the Overpass API [1] to access OpenStreetMap (OSM) [1], which will serve as our geospatial data source. The system initially retrieves spatial and contextual data from OSM, such as nearby businesses, institutions, and organizations, in response to a user's input. Thus, by incorporating this structured data into the prompt, the Large Language Model (LLM) will be able to generate insights based on local and current facts instead of only its pre-trained knowledge. The model's reliance on external, up-to-date location data from OpenStreetMap may result in more accurate, reliable, and practical recommendations for enterprise placement and risk assessment.

The system also makes use of official statistical information gathered in PDF and Excel format from organizations like Turkish Statistical Institute (TUIK)[2], Small and Medium Enterprises Development Organization of Turkey (KOSGEB)[3], and the Ministry of Industry and Technology of the Republic of Turkey [4] in order to provide the model with more context. These records contain information on business opening/closing statistics, population demographics, and socioeconomic development at the district level. In order to generate grounded responses, pertinent content is retrieved and analyzed. In addition, it offers an interactive map to show similar businesses to the user visually.

EntreLocate provides a useful decision-support tool for business owners throughout Turkey by bridging the gap between real-world spatial and economic data and large language model capabilities.

II. Background

1) Retrieval-Augmented Generation (RAG)

Retrieval-augmented generation (RAG) is a natural language processing (NLP) method designed to address a fundamental limitation of LLMs: their reliance on static, internal knowledge. Although LLMs achieved noticeable

979-8-3315-9975-1/25/$31.00 ©2025 IEEE

success, they still face limitations in domain-specific knowledge-specific tasks, which can notably produce “hallucinations” [5]. RAG overcomes this by dynamically augmenting the LLM's prompt with relevant, external data retrieved in real-time.

The core principle of RAG is a two-stage process: first, it retrieves information from an external, reliable knowledge source; second, it uses this information to generate a grounded, context-aware response. This method allows the model to generate more accurate and up-to-date results by utilizing information that becomes available after the training phase [6]. The RAG process can be broadly divided into two main phases:

Indexing (Offline Process): The external knowledge base is prepared by loading documents, splitting them into manageable chunks, converting these chunks into numerical vector representations using an embedding model, and storing them in a specialized vector database.

Retrieval Generation (Online Process): When a user submits a query, it is also converted into a vector. A similarity search is performed in the vector database to find the most relevant document chunks with the user query vector. This retrieved context is then combined with the original query and passed to the LLM, which generates the answer response.

a) Vector Database

A vector database holds numerical vectors generated by embedding models and enables frequent semantic searches based on vector similarity, using nearest-neighbor search techniques. Text representations are stored in vector databases, which are then used to retrieve relevant answer fragments. These retrieved pieces are then combined into a meaningful response with the help of large language models [7]. We used the ChromaDB vector database for our project.

b) Embedding Models

Natural language processing models cannot directly handle raw text because computers are unable to interpret natural language in its original form. To enable computation, the text must first be transformed into numerical values. This conversion process, known as 'embedding,' turns text into vectors that machines can understand. Every natural language processing model relies on this embedding step to process language inputs [8]. We used the “all-MiniLM-L6-v2” sentence transformer for embedding in our project. A brief explanation of the purpose of embeddings is provided above; Fig. 1 shows a Transformer-based autoencoder model for generating sentence embeddings.

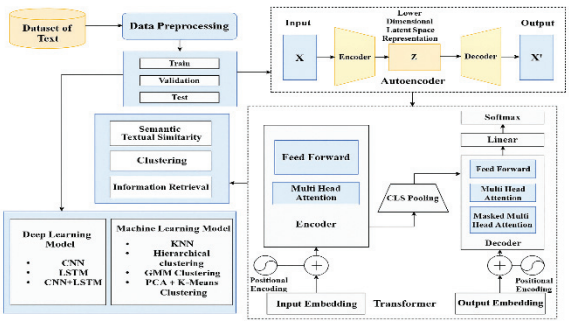

Fig. 1. Transformer-based autoencoder model for sentence embedding reproduced from [9].

c) Text Chunking

Text chunking is the process of breaking down huge documents or lengthy texts into smaller, more manageable pieces in order to improve retrieval performance and accuracy.

d) Similarity Metrics

RAG offers a framework that enables users to fetch relevant document chunks from a vector database through vector similarity search. Various methods exist for vector similarity search, including dot product, cosine similarity, Manhattan distance, and Euclidean distance. In this case, cosine similarity is employed to retrieve the top-k most relevant documents from the vector store. These top-k results include text segments. The retrieved documents, along with the user query, are then fed into LLM to generate the final output [10].

2) Large Language Models

A large language model (LLM) is a kind of artificial intelligence (AI) system designed to understand and produce text, among other capabilities. It's called "large" because it's trained on massive amounts of data. LLMs use machine learning techniques and are based on a specific kind of neural network known as a transformer. In simple terms, an LLM is a program that has learned from many examples to grasp and interpret complex content like human language [11].

3) OSM and Overpass API

OpenStreetMap (OSM) [1] and the Overpass API [1] were essential to the EntreLocate project's ability to retrieve real-time geospatial data for assessing the suitability of business locations and producing grounded risk assessments. Roads, buildings, and business kinds are all included in the comprehensive, community-contributed mapping data that OSM provides. The Overpass API enables effective, selective querying of this data without requiring the download of the entire dataset.

To retrieve data about the quantity and kinds of establishments (such as markets, pharmacies, and cafés) within a 250 m radius around a user-provided location, we built Overpass QL queries. These queries were run using HTTP requests, and the structured JSON data that was returned included semantic tags like "shop," "amenity," and "name," together with geographic coordinates. This information was compiled and analyzed in order to categorize neighboring entities and compute important metrics like infrastructure availability, business density, and local competition.

4) Risk Assessment

EntreLocate's risk assessment process includes an in-depth review of a variety of factors, including regional socioeconomic indicators like demographic statistics (population density and age distribution), and the competitive landscape (business density and type). This multidimensional analysis is made easier by the semantic retrieval of related geographical data, which the LLM then synthesizes into meaningful risk assessments. These rankings accurately represent expected business concerns and possibilities, giving entrepreneurs specific, evidence-based information to help them make informed and intelligent location selections.

5) Multi-agent Communication and Workflow Design

Multi-agent systems are computing systems where multiple autonomous agents work together to solve complex tasks through various coordination approaches.

To effectively manage task delegation and maintain modularity in EntreLocate, an orchestrator model was implemented. The Orchestrator's job is to first create a dynamic plan for the analysis using an LLM. It then executes that plan by calling the correct specialist agents (e.g., RAG retriever, API caller, risk analyzer) in sequence, managing the flow of data between them to complete the entire workflow. Each sub-agent receives structured data in a predefined format and returns outputs that are combined and post-processed by the orchestrator to create a coherent system response. The orchestrator also ensures system robustness by managing fallbacks when an agent fails to provide valid output.

A clear and consistent data flow model was required for inter-agent communication. Data pipelines were designed to use standardized JSON structures that each agent could parse and return. This includes schemas for the following:

User input: retrieved from dropdown menu selections such as business type and location.

RAG input/output: embedding queries with metadata tags and returning the top k text fragments.

API responses: Formatted geographic data enriched with entity type and distance metrics from OpenStreetMap.

Uniformity in data representation ensures that agents can work independently but integrate seamlessly into the overall workflow. In Fig. 2, we can see the multi-agent workflow of the EntreLocate.

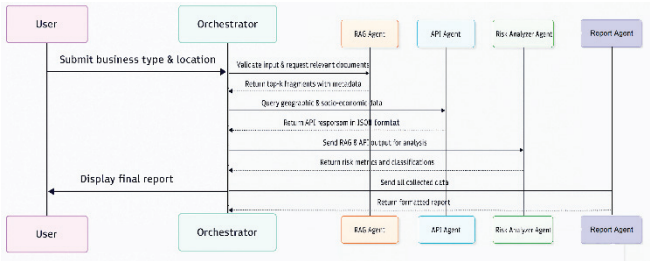

Fig. 2. Agent workflow.

6) RAG Knowledge Sources

Developing business assistant applications requires up-to-date data, which can be utilized with a comprehensive RAG structure. In this project, we used various documents written by the Turkish government, but due to insufficient resources, we created synthetic data that is compatible with real data. We gathered all the information from the PDF and Excel format reports we had, then added the generated synthetic data and compiled everything into a single Excel file. Finally, we parsed this file in the RAG system. Future developments aim to expand the system with more comprehensive data for our RAG system. Table I presents a breakdown of the dataset used to train our RAG model, categorized by domain, number of documents, and total size.

TABLE I. USED DATA SOURCES IN THE RAG SYSTEM

Domain	Document Type	Number of Documents	Size
Govt. Report	pdf	2	967,7 KB
TUIK Data	excel	2	772 KB
Synthetic Business open/close rate	excel	1	507 KB

III. RELATED WORK

In the field of AI-driven geospatial analysis, research relevant to our work follows two main approaches. The first approach applies general AI-enhanced analysis to high-level business planning, such as helping Small and Medium Enterprises (SMEs) evaluate market potential across large regions [12]. A second, more recent approach utilizes Large Language Models (LLMs) for specific, real-time tasks, like creating interactive guides for pedestrian navigation with OpenStreetMap (OSM) [13].

Our work connects these two areas. Instead of analyzing large regions, our system operates locally. In contrast to providing live directions, our model's purpose is to generate a detailed analytical report using a Retrieval-Augmented Generation (RAG) architecture. We synthesize competitor density from OSM with background information on the area's demographics and recent business openings and closings. This approach provides a detailed report on local risk and demand that has not been explored in previous studies.

IV. METHODOLOGY

A. System Architecture

The EntreLocate architecture is founded on a multi-agent orchestration structure, allowing for scalability, modularity, and a clear separation of concerns. At its core is the Orchestrator, a master agent that, instead of following a rigid script, dynamically generates a multi-step execution plan using a Large Language Model (LLM) based on the user's initial request.

Upon receiving user input, the Orchestrator first crafts this plan, which dictates the precise sequence of agent calls. It then dispatches tasks to specialized, autonomous agents like the DataFetchAgent, the RAGRetrieverAgent, and the MarketRiskAgent. These agents communicate via standardized JSON, enabling decoupled, maintainable, and easily extensible components managed by the Orchestrator.

The architecture integrates external services (e.g., Overpass API) and internal modules (e.g., ChromaDB, risk scoring) through agents, with the Orchestrator managing data flow. The ReportGeneratorAgent compiles outputs into a user-facing PDF, enabling the system to handle complex business queries through modular, step-by-step reasoning.

B. Information Sources and Datasets

This project utilized multiple datasets from authoritative Turkish governmental and international sources to ensure robust analysis and reliable findings. The Turkish Statistical Institute's population dataset formed one of the demographic foundations of this research; distributions are focused on cities related to their age and gender. The Turkish Statistical Institute's Distribution of household consumer expenditures by type provides insights into the demand in the business field. In an effort to determine the level of occupation or congestion in the desired sector, we also examine the data of recently established and closed firms from KOSGEB [3]. The Socio-Economic Development Index of Districts (SEDI) reports [14], which are published by the General Directorate of Development Agencies of the Ministry of Industry and Technology, provide an important viewpoint on the socioeconomic development level of each district in Turkey. These sources, along with real-time geographic data from

OpenStreetMap [1], serve as the foundation for current and grounded location-based risk and opportunity assessment.

C. Development Environment

The EntreLocate system was implemented in a modular, agent-based architecture, allowing for independent development and debugging. The system was implemented using modern tools and libraries in a reproducible and collaborative environment.

1) Programming Language & Environments

Python was employed as the primary development language due to its versatility and broad support for AI, NLP, and data science tools. Development was done in both Google Colab, for prototyping, embedding generation, and vector DB population, and Visual Studio Code (VS Code) for module development with structure and logging.

2) Core Libraries and Technologies

Google Generative AI SDK (google.generativeai) was utilized to interact with Gemini 1.5 Flash [15], a multimodal LLM that carried out structured prompt completion and business logic.

Sentence-Transformers enabled high-quality embedding generation for text segments, particularly with the all-MiniLM-L6-v2 model.

ChromaDB was used as the vector database for rapid similarity search.

Overpass API was used to retrieve real-time geospatial data from OpenStreetMap.

FPDF was used to generate downloadable PDF reports with user-specific risk analysis and recommendations.

python-dotenv was used for secure management of environment variables, i.e., API keys.

3) Agent Framework

The system uses modular multi-agent architecture, with each Python class performing a specialized role: DataFetchAgent retrieves geospatial data from OpenStreetMap, RAGRetrieverAgent extracts and synthesizes socio-economic data from ChromaDB, MarketRiskAgent analyzes business risks, and ReportGeneratorAgent compiles the final PDF report.

The Orchestrator dynamically creates execution plans and coordinates agent interactions, ensuring smooth data flow and task sequencing based on user input.

All agents inherit from a common base class that supports API retries, LLM token tracking, and metadata capture (e.g., rag_synthesis_confidence, token usage, and cost), which are included in the final report for transparency.

4) Logging and Monitoring

Module-level custom logging was implemented for each agent to facilitate real-time updating, debugging, and monitoring of token usage for Gemini calls. The orchestrator and agents also displayed status, errors, and runtime performance for facilitating traceability and testing.

5) User Interface and Output

The Flask web framework was used to create the EntreLocate system's user interface, as shown in Fig. 3, offering a scalable and lightweight backend infrastructure for managing user interactions. Users interact with the system by using additional question buttons and selecting location details.

The user begins by selecting four major parameters via the interface: Province, District, Street, and Business Type.

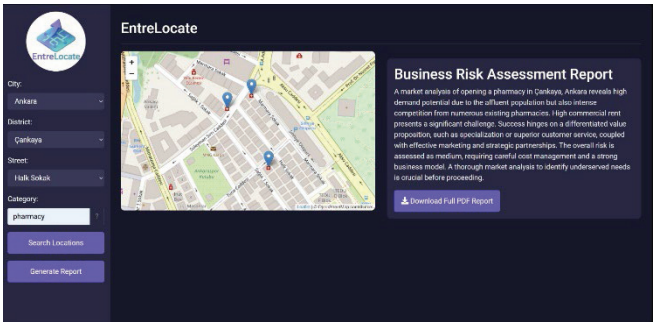

Fig. 3. User interface of EntreLocate.

The backend processing pipeline receives a query from the Flask server, retrieves pertinent socioeconomic and geographic information, and passes it to the language model along with the user input. In addition to risk scores, business appropriateness evaluations, and regional insights, the model produces context-aware recommendations.

After being returned in structured JSON format, the results are shown as a summary in a neutral language query which is more understandable by users on the front end. Buttons for optional actions like "Download Report," "Recommend Business," and "Suggest Location" are among the interactive elements of the UI. Users can export the data in a structured report format for future planning and reference using a PDF creation tool.

The EntreLocate system follows a structured multi-agent pipeline initiated through user interactions on a graphical interface. The procedure is outlined as follows:

6) First Risk Assessment Output

Upon receiving the user's input, the Orchestrator first generates a dynamic, step-by-step plan. It then initiates the execution of this plan by sequentially calling the necessary agents:

The DataFetchAgent queries the Overpass API to retrieve a list of competing businesses and relevant entities within a 250-meter radius of the specified district.

The RAGRetrieverAgent accesses the ChromaDB vector database, using a precise metadata filter (not a semantic search) to find the specific socio-economic data for the district. It then uses an LLM to synthesize this raw data into a structured JSON format.

The MarketRiskAgent combines the data from the previous two agents to perform a detailed qualitative analysis, generating a market narrative, strategic recommendations, and a set of graded factor scores (e.g., for demand and competition).

A separate ScoreCalculator function then takes these graded scores and applies a weighted formula to produce a single, final numeric risk score and recommendation level.

Finally, all of this combined output is passed to the ReportGeneratorAgent, which compiles the entire analysis—

including the narrative, scores, and strategic advice—into a comprehensive, downloadable PDF report.

7) On-Demand Function Triggers

After the initial risk output has been shown, the user can trigger additional activity by clicking the report generation button. The system takes the current risk assessment and context, renders it as a styled PDF document from HTML templates, and presents it for downloading.

D. Parameters and Configurations

The specific hyperparameter settings used to configure the Gemini model are detailed in Table II.

TABLE II. USED PARAMETERS WITH VALUES AND PURPOSE

Parameter	Value	Purpose
chunk_size	512 tokens	Maximum length of text chunks in the vector database
chunk_overlap	10 tokens	Overlap between adjacent chunks to preserve semantic flow
top_p	0,95	The smallest set whose cumulative probability exceeds the specified threshold
top_k	64	Number of nearest neighbor results retrieved from the vector DB
osm_radius	250 meters	Search radius for nearby features using Overpass API
llm_temperature	0,1	Reduces randomness for more consistent, grounded answers
api_timeout	25 seconds	Prevents long delays from external API calls

chunk_size & chunk_overlap: These controls were tuned to preserve coherency in document meaning and within the token constraints of large language models.

osm_radius: For this field, the radius of 250 meters was chosen in order to balance local importance and data density. It is an estimate of the typical area where small businesses assess rivals and similar products.

top_k = 64: This retrieval factor returns the top 64 results with high relevance without flooding the LLM context window with too many documents.

Low Temp for LLMs: The temperature was kept at 0,1 so that it would be stable, particularly for recommendation prompts based on factual grounding.

Api_timeout: To ensure timely execution and prevent system hangs, a 25-second timeout threshold is imposed on external API calls.

E. Testing and Performance

TABLE III. TOKEN USAGE

	Report 1	**Report 2**	**Report 3**	**Report 4**
Location	Ankara/ Çankaya /Rabat Street	Ankara/ Beypazarı/Ayvaşık 2 Street	Antalya/ Kemer/ Karapınar Street	Amasya /Suluova/Şener Street
Job Type	pharmacy	pharmacy	pharmacy	pharmacy
Total Tokens Used	24,032	6,432	6,967	4,829
Total LLM Cost($)	$0,002134	$0,000803	$0,000883	$0,000695
Report Generation Time(s)	15,25	13,76	15,65	18,28

As detailed in Table III, one of the key metrics is the total number of tokens used, which measures all tokens consumed in AI for a single analysis, serving as the main contributor to cost.

a) Total LLM Cost:

This metric calculates the total monetary cost of all AI calls in USD, clearly revealing the system's economic efficiency and the actual cost per report.

b) Report Generation Time:

This metric measures the total time in seconds from the start of a request to the final report delivery, reflecting the user-perceived speed and overall system responsiveness.

Token usage and cost of Report-1 are 3-5x higher than others because Çankaya is a dense urban center. The API query returned large data, so the workload for analysis, the token count, and the cost increased.

Other places have fewer competitors, so they are less expensive.

Generation time can be longer due to API latencies (Geocoding, Overpass, Gemini). A momentary delay in any of these network calls affects total time more than LLM's processing effort.

F. User Feedback and Validation

Through informal demonstrations and task-driven evaluations with early users, we gathered preliminary user feedback to assess EntreLocate's usability and efficacy. The purpose of this validation was to determine areas for future development and to comprehend the system's practical impact.

Users said that, particularly in metropolitan regions with several documentation, the location-based choices matched their expectations reasonably well. The system's risk evaluations were seen as useful and practical in denser neighborhoods like Çankaya, where more detailed data was

accessible. In addition, users liked the opportunity to choose criteria such as province, district, and company type, and considered the dropdown-based input method to be simple to use. Also, the downloadable PDF reports and the organized summary in neutral language were appreciated for being understandable and instructive. Lastly, some users said that when data was sparse or distributed unevenly among districts, suggestions might be ambiguous.

V. Discussion

The development of EntreLocate highlighted significant design features in creating an AI assistant that integrates large language models (LLMs), retrieval-augmented generation (RAG), and structured data. The modular multi-agent design improved stability, scalability, and testability.

While our RAG module grounds the LLM in real-world data, inconsistent retrieval quality sometimes led to ambiguous outputs, creating a need for improved data chunking and prompt engineering to ensure accuracy.

We used a JSON structure to integrate diverse data sources (e.g., Excel, PDFs, APIs). Internal tests demonstrated reliable outputs and responsive, explicit interface behavior in addition to the absence of quantitative user testing. Initial user feedback was positive, praising the intuitive controls. Planned enhancements to further improve usability include interactive visualizations, multilingual support, and real-time feedback loops.

While our analysis is based on real-world data from OpenStreetMap and official statistics, we lacked reliable local data on business turnover rates. This challenge was compounded by the Overpass API, which provided incomplete results for smaller towns. To overcome this, we generated synthetic data exclusively for business turnover rates at the district level. This method resulted in a complete and realistic model of local market dynamics.

VI. Conclusion

EntreLocate provides precise, data-driven location suggestions by integrating AI, RAG, and LLMs to enhance business decision-making. The technology minimizes uncertainty and improves strategic planning by clearly establishing insights in real-time sources such as OpenStreetMap and demographic data. Its creative strategy not only gives users greater ownership but also boosts the local economy. Expanding geographical availability, adding more real-time data, and guaranteeing easy integration with popular business tools will be the main goals of future projects.

Our project's open-source implementation can be accessed via this GitHub repository [16].

Acknowledgement

This study was carried out as part of the SENG 472-Building AI Applications with Large Language Models course, under the supervision of Dr. Kasım Murat Karakaya at TED University.

References

[1] OpenStreetMap Wiki, "Overpass API." [Online]. Available: https://wiki.openstreetmap.org/wiki/Overpass_API

[2] Türkiye İstatistik Kurumu (TÜİK), "Hanehalkı tüketim harcamalarının türlerine göre dağılımı." [Online]. Available: https://data.tuik.gov.tr/Bulten/Index?p=Hanehalki-Tuketim-Harcamasi-Istatistikleri-2023-49920

[3] KOSGEB, "Rapor ve istatistikler – yeni açılan ve kapanan işletmeler," Küçük ve Orta Ölçekli İşletmeleri Geliştirme ve Destekleme İdaresi Başkanlığı. [Online]. Available: https://www.kosgeb.gov.tr/site/tr/genel/detay/349/rapor-ve-istatistikler

[4] T.C. Sanayi ve Teknoloji Bakanlığı, "T.C. Sanayi ve Teknoloji Bakanlığı Resmi Web Sitesi." [Online]. Available: https://www.sanayi.gov.tr

[5] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, Dec. 2023. [Online]. Available: https://arxiv.org/abs/2312.10997

[6] B. Tural, Z. Örpek, and Z. Destan, "Retrieval-augmented generation (RAG) and LLM integration," in *Proc. 8th Int. Symp. Innovative Approaches in Smart Technologies (ISAS)*, Istanbul, Turkiye, 2024, pp. 1-5, doi: 10.1109/ISAS64331.2024.10845308.

[7] G. Şahin, K. Varol, and B. K. Pak, "LLM and RAG-based question answering assistant for enterprise knowledge management," in *Proc. 9th Int. Conf. Computer Science and Engineering (UBMK)*, Antalya, Turkiye, 2024, pp. 1-6, doi: 10.1109/UBMK63289.2024.10773564.

[8] K. Jung, G. Lee, D. Lee, S. Lee, and J.-J. Kim, "Analysis of transformer decoder architecture and KV cache behavior during LLM inference," in *Proc. Int. Conf. Electronics, Information, and Communication (ICEIC)*, Osaka, Japan, 2025, pp. 1-4, doi: 10.1109/ICEIC64972.2025.10879650.

[9] S. N. Pearl and M. S. Arefin, "Developing a transformer-based autoencoder model for sentence embedding," in *Proc. Int. Conf. Electrical, Computer and Communication Engineering (ECCE)*, Chittagong, Bangladesh, 2025, pp. 1-6, doi: 10.1109/ECCE64574.2025.11012946.

[10] P. Joshi, A. Gupta, P. Kumar, and M. Sisodia, "Robust multi model RAG pipeline for documents containing text, table & images," in *Proc. 3rd Int. Conf. Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India, 2024, pp. 993-999, doi: 10.1109/ICAAIC60222.2024.10574972.

[11] Cloudflare, "What is a large language model (LLM)?," *Cloudflare Learning Center*. [Online]. Available: https://www.cloudflare.com/learning/ai/what-is-large-language-model/. [Accessed: Jun. 1, 2025].

[12] G. Mallela, R. Sahu, and M. K. Dash, "AI-enhanced geospatial analysis for global small and medium enterprises market," in *Proc. 8th Int. Conf. I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Kirtipur, Nepal, 2024, pp. 1660-1665, doi: 10.1109/I-SMAC61858.2024.10714641.

[13] G. I. Yussif, M. Abdelatti and A. Hendawi, "Harnessing Crowdsourced Mobile Data and LLM for Dynamic and Accessible Pedestrian Routing," *2025 26th IEEE International Conference on Mobile Data Management (MDM)*, Irvine, CA, USA, 2025, pp. 109-112, doi: 10.1109/MDM65600.2025.00057.

[14] T.C. Sanayi ve Teknoloji Bakanlığı, "İlçelerin Sosyo-Ekonomik Gelişmişlik Sıralaması (SEGE-2022)." [Online]. Available: https://www.sanayi.gov.tr/merkez-birimi/b94224510b7b/sege. [Accessed: Jun. 26, 2025].

[15] S. Pichai, "Our next-generation model: Gemini 1.5," *Google Blog*. [Online]. Available: https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/. [Accessed: Jun. 26, 2025].

[16] ZebraFinchIAN, "EntreLocate_Project." *GitHub repository*. [Online]. Available: https://github.com/ZebraFinchIAN/EntreLocate_Project