UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Utilization of Spatial Data and Generative Artificial Intelligence for Optimal Business Location Selection

Diploma Thesis

Giorgos Kakepakis

Supervisor: Elias Houstis

Month 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Utilization of Spatial Data and Generative Artificial Intelligence for Optimal Business Location Selection

Diploma Thesis

Giorgos Kakepakis

Supervisor: Elias Houstis

Month 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αξιοποίηση Χωρικών Δεδομένων και Γενετικής Τεχνητής Νοημοσύνης για Βέλτιστη Επιλογή Επιχειρηματικής Τοποθεσίας

Διπλωματική Εργασία

Γιώργος Κακεπάκης

Επιβλέπων/πουσα: Ηλίας Χούστης

Μήνας 2025

Approved by the Examination Committee:

Supervisor **Elias Houstis**
Emeritus Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Emmanouil Vavalis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Giorgos Petrakos**
Professor, Department of Spatial Planning, Urban Planning and Regional Development Engineering

Acknowledgements

I wish to express my sincere appreciation to the people who participated and contributed to the implementation of this project.

First of all, I would like to thank my supervisor, Emeritus Professor Elias Houstis of the Department of Electrical and Computer Engineering of the University of Thessaly, since this thesis was inspired by his innovative ideas. His knowledge and guidance, along with his unwavering willingness of assistance, have greatly influenced and enriched the development of this project.

I also want to express my thanks to Professor Emmanouil Vavalis for his additive influence and support on this thesis. The insights of a fellow professor of the Department of Electrical and Computer Engineering led to creative approaches and solutions.

The help of Professor Giorgos Petrakos should also be highlighted, as it provided a valuable connection with the Department of Spatial Planning, Urban Planning and Regional Development Engineering. The influence of a specialist in the field proved vital in refining and perfecting the project.

Last but not least, I would like to thank my family and close friends for the support and enthusiasm throughout the development process.

Thank you all for your support throughout the course of this work.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Giorgos Kakepakis

Diploma Thesis

Utilization of Spatial Data and Generative Artificial Intelligence for Optimal Business Location Selection

Giorgos Kakepakis

Abstract

Choosing the appropriate location is a critical success factor for every business activity. The purpose of this thesis is to develop a system that combines the power of spatial data with the capabilities of genetic artificial intelligence, with the aim of supporting the optimal choice of business location. The proposed system utilizes geospatial characteristics, such as population density, accessibility, as well as demographic or economic data of areas. Through a combination of spatial analysis and advanced language models (LLMs), the system extracts information from heterogeneous sources, processes it, and produces explanatory and personalized location suggestions for new businesses. The study includes the construction of two pipelines based on LLMs, the selection of appropriate geospatial features, the creation of a search index, the training of a location evaluation model, and the final generation of responses. At the same time, the effectiveness of the system is evaluated through accuracy metrics.

Keywords:

term, term, . . . , term

Διπλωματική Εργασία

Αξιοποίηση Χωρικών Δεδομένων και Γενετικής Τεχνητής Νοημοσύνης για Βέλτιστη Επιλογή Επιχειρηματικής Τοποθεσίας

Γιώργος Κακεπάκης

Περίληψη

Η επιλογή της κατάλληλης τοποθεσίας αποτελεί κρίσιμο παράγοντα επιτυχίας για κάθε επιχειρηματική δραστηριότητα. Σκοπός της παρούσας διπλωματικής μελέτης είναι η ανάπτυξη ενός συστήματος που συνδυάζει τη δύναμη των χωρικών δεδομένων με τις δυνατότητες της γενετικής τεχνητής νοημοσύνης, με στόχο την υποστήριξη της βέλτιστης επιλογής επιχειρηματικής τοποθεσίας. Το προτεινόμενο σύστημα αξιοποιεί γεωχωρικά χαρακτηριστικά, όπως πυκνότητα πληθυσμού, προσβασιμότητα, αλλά και δημογραφικά ή οικονομικά δεδομένα περιοχών. Μέσω ενός συνδυασμού χωρικής ανάλυσης και προηγμένων γλωσσικών μοντέλων (LLMs), το σύστημα αντλεί πληροφορίες από ετερογενείς πηγές, τις επεξεργάζεται και παράγει επεξηγηματικές και εξατομικευμένες προτάσεις τοποθεσιών για νέες επιχειρήσεις. Η μελέτη περιλαμβάνει την κατασκευή δυο pipelines βασισμένα σε LLMs, την επιλογή κατάλληλων γεωχωρικών χαρακτηριστικών, τη δημιουργία δείκτη αναζήτησης, την εκπαίδευση μοντέλου αξιολόγησης τοποθεσιών και την τελική παραγωγή απαντήσεων. Παράλληλα, αξιολογείται η αποτελεσματικότητα του συστήματος μέσω μετρικών ακρίβειας.

Λέξεις-κλειδιά:

όρος, όρος, . . . , όρος

Table of contents

List of figures

List of tables

Abbreviations

GIS	Geographic Information System
POI	Point Of Interest
AI	Artificial Intelligence
ML	Machine Learning
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
XAI	Explainable AI
QLoRA	Quantized Low-Rank Adaptation
UI	User Interface
API	Application Programming Interface
EDA	Explorarory Data Analysis
PCA	Principal Component Analysis
SHAP	Shapley Additive Explanations
JSON	JavaScript Object Notation
CSV	Comma-Separated Values
NACE	National Association of Colleges and Employers
ELSTAT	Hellenic Statistical Authority
QGIS	Quantum Geographic Information System

Chapter 1

Introduction

The selection of a business location is one of the most critical decisions that an entrepreneur must make, since it determines various parameters that greatly influence its growth, sustainability, and overall success. The location of an enterprise is directly connected to the proximity to points of interest, accessibility, competitive landscape, and demographic and socio-economic characteristics of the population. In addition, a potential relocation requires prohibitive financial costs and entails the risks of loss of the existing customer base, making it practically irreversible. As a result, choosing the optimal business location becomes a matter of significant importance.

1.1 Problem Definition and Thesis Scope

The selection of the optimal business location plays a crucial role for every enterprise, as it is indirectly responsible for many characteristics that influence the probability of its success. As mentioned above, choosing a location is responsible for the competitor density, distance from important places like transport stops, highways, or universities, and various other influential factors that need to be carefully examined. Since relocating a business is a process every entrepreneur aims to avoid, we treat it as a one-time decision. Therefore, it is of significant importance to make this decision wisely, taking into account multiple aspects and factors.

Spatial data, such as demographic, economic, and geographic information, have become increasingly available through open data sources such as government statistics. Research has consistently shown that integrating spatial data analysis in business location selection

improves outcomes by providing helpful insights from novel perspectives. However, most approaches that utilize spatial data for optimal location selection focus on manual rule-based methods that often fail to capture more complex underlying patterns to extract meaningful conclusions.

Simultaneously, the emergence of machine learning, and Large Language Models and Generative AI in particular, has resulted in state-of-the-art performance in many applications. This groundbreaking technology manages to capture non-linear relationships and achieve better performance than rule-based models. One of the most important advantages of the field lies in its adaptability, as ML algorithms can be applied to various kinds of datasets to specialize in specific domains.

The core idea of this thesis is to merge spatial data with state-of-the-art machine learning models, in order to leverage the adaptability of ML and the valuable insights of spatial data to enhance business location recommendation. To the best of our knowledge, limited research has focused on integrating the two fields. Since Generative AI tends to outperform traditional algorithms in the majority of its applications, we aim to investigate whether it can improve this task as well. The development of the final system will consist of handling heterogeneous spatial datasets, applying and comparing two cutting-edge LLM-based methods, and focusing on providing recommendations supported by justifications.

1.2 Contribution

This thesis makes several contributions to the field of optimal business location recommendation by utilizing spatial data in combination with state-of-the-art Generative AI models. They can be summarized as follows:

1. Constructed multiple datasets for the region of Magnesia (Volos, Greece) that combine business records with gepgraphic, economic, and demographic, information, on the level of neighborhood and municipal community.

2. Implemented and compared two LLM-based approaches for business location recommendation, in particular: fine-tuning a pre-trained LLM and designing a Retrieval-Augmented Generation (RAG) system.

3. Focused on providing explainability about the recommendations, by identifying the

most important characteristics of each candidate, and using their comparison with the all candidates as justifications.

4. Integrated both algorithms into a chatbot application, enabling interactive conversations that store user preferences and take them into account while providing recommendations

1.3 Thesis Overview

This thesis is structured in chapters, each focusing on a specific part of the study. The chapters are structured as follows:

- **Chapter 2: Literature Review / Background** - This chapter reviews the key foundations of this thesis, such as merging business location recommendation with GIS data, NACE business classification, and methods of implementing the fine-tuning and RAG systems.
- **Chapter 3: Data Collection and Preprocessing** - This chapter examines the process of collecting and processing information from heterogeneous spatial dataset sources, and analysing the three constructed datasets in order to create proxy ground truths.
- **Chapter 4: Methodology / System Implementation** - This chapter examines in detail the workflow of the modeling approaches used, such as the construction of the fine-tuning dataset and the LLM training, the creation of neighborhood descriptions used in RAG, the inference process of both approaches, and their integration in the chatbot application.
- **Chapter 5: Experiments and Results** - This chapter focuses on the evaluation and comparison of the fine-tuned LLM and Retrieval-Augmented Generation systems
- **Chapter 6: Conclusions and Discussion** This chapter reviews the results of the thesis, extract the final conclusions and mentions potential future directions

Chapter 2

Literature Review / Background

To achieve reach our goal, we aim to construct a complex system that consists of multiple modules based on various fields of knowledge. This paragraph provides the key foundations upon which this thesis is built.

2.1 Business Location Recommendation

The optimal selection of business locations is a factor that significantly influences the success and sustainability of any enterprise. The placement of a business determines various other parameters, such as customer accessibility, operational efficiency, and its overall profit in general. Prior studies [1] have emphasized that optimal location selection not only can reduce urban shrinkage, but also strengthen the resilience of cities. These benefits of choosing the appropriate location, combined with its irreversible nature once performed, motivated the exploration of state-of-the-art technologies that may help address the problem.

One field that has been increasingly utilized in recent years is Geographic Information Systems (GIS). As shown in prior work [2], spatial and GIS data enhance the location selection with important criteria, such as foot traffic, demographic patterns, and competitor distribution. The rapid increase of available data served as a catalyst for the evolution of machine learning to evolve. This application of artificial intelligence enables systems to automatically capture complex underlying patterns of data to make predictions in the future [3]. In recent years, groundbreaking approaches like Deep Learning, Transformers, and Large Language Models (LLMs) propose effective approaches to utilizing GIS data for optimal business location determination. Although limited research has been conducted on this topic, certain

studies [4] [5] suggest that LLMs can enhance the performance of location-based recommendation systems.

Building upon these foundations, the following paragraphs dive deeper in GIS data in location analysis, multiple LLM-based systems (fine-tuned LLMs, RAG) and AI explainability while also examining the NACE business classification protocol.

2.2 Geospatial Data and GIS in Location Analysis

A Geographic Information System (GIS) is a system designed to store, analyze, edit, and visualize geographic data [6]. It merges various types of data into a common framework, allowing users to capture and understand patterns, trends, and relationships in a spatial context. GIS data emerged in the early 1960s and initially focused on applications like map digitization and the conceptualization of spatial databases [6].

Since then, GIS has evolved into a versatile tool for spatial analysis, driven by breakthroughs in remote sensing data, both in terms of collection (e.g. satellite imagery, maps, demographic information), and accessibility (e.g. rise of location-based services, use of smartphones for real-time data collection). Certain fields that GIS data has been applied across include urban and infrastructure planning [6], environmental analysis and conservation [6], public health [7], enhancement of virtual assistants [8], public transportation accessibility and optimization [9], and, of particular relevance to this thesis, business intelligence [10]. Within business location analysis in particular, GIS enables the integration of spatial factors such as population density, transportation accessibility, and competitor distribution [11]. In addition, a rich ecosystem has been built around geospatial data, including a range of software platforms. In this thesis, QGIS, a powerful and open-source GIS software used for editing and visualizing geospatial data [12], was employed to extract and process valuable geographical information in order to enhance our datasets.

While Geographic Information Systems provide useful and interpretable tools, they mainly rely on static, rule-based algorithms that often fail to capture complex patterns. To address these limitations, machine learning frameworks have been increasingly integrated with GIS data, as discussed in the following sections.

2.3 Machine Learning in Business Location Analysis

Machine learning (ML) is a groundbreaking scientific field that focuses on the creation of models that are able to capture complex, non-linear underlying relationships, without the need to be explicitly programmed [13]. The learning process is driven by data, based on which the model adapts to a specific application domain. In the context of this thesis, machine learning models are integrated with GIS data to augment business location analysis, particularly by incorporating features such as geographic coordinates (latitude and longitude), distance to the city center, proximity to transportation stations, and accessibility to key facilities such as the university or the port. Despite the importance of these models, we do not use them directly to generate the final recommendations. Rather, we apply ML as a supporting tool for building proxy ground truths and providing interpretability about the importance of the features. The actual predictions are produced by LLM-driven pipelines, which we explain later.

2.3.1 Categories of Machine Learning

The field of machine learning is divided into certain categories, determined by the labeling of the training dataset.

- **Supervised ML**: These algorithms learn by using labeled examples to predict future events from unseen data [3]. Although there is a variety of traditional and reliable supervised ML models, in this work we use certain state-of-the-art tree-based methods. Specifically, Random Forest is an ensemble-based method which combines multiple decision trees using bagging [14], and XGBoost, a gradient boosting algorithm that sequentially builds ensembles of trees to minimize prediction error [15].

- **Unsupervised ML**: When labels are not available, unsupervised techniques are used discover underlying patterns within the dataset. In this thesis, K-Means clustering [16] was applied to group neighborhoods into sets with similar characteristics.

- **Semi-supervised ML**: In the course of this thesis, we frequently encountered the phenomenon of obtaining labels for only a small proportion of the data. Semi-supervised methods, such as Self-Training, aim to exploit limited labels to predict the missing ones, thus enhriching the dataset [17].

2.3.2 Explainable AI and SHAP

Modern ML models manage to capture non-linear underlying patterns that rule-based algorithms cannot detect. However, applying complex models comes with the cost of failing to understand their decision-making, as they often act as black boxes. In order to identify the features that affect the model output, Explainable AI (XAI) techniques have been developed to unravel some of these aspects.

SHAP is a prominent XAI method based on cooperative game theory, used to increase the transparency and interpretability of machine learning models [18]. It provides insights into the importance of every feature by assigning a value that represents its contribution to a specific prediction. Within this thesis in particular, we calculate the SHAP values of every feature per NACE code, in order to determine the most impactful factors for every business type. The most important features of the location are then included as justifications during the creation of the fine-tuning dataset, as we will see below.

2.4 NACE Codes and Economic Classification

NACE is the European standard classification of productive economic activities [19]. Providing a standardized categorization protocol is crucial for ensuring consistency across regions and time. Eurostat uses NACE to perform statistical reporting, as consistent classification enables better storing, processing, and merging heterogeneous data sources across different countries. The protocol follows a 4-level hierarchical structure, with business categorization getting increasingly detailed as the code gains precision. Specifically, the NACE code levels are the following:

- **Sections**: first level, identified by an alphabetical code
- **Divisions**: second level, identified by a two-digit numerical code
- **Groups**: third level, identified by a three-digit numerical code
- **Class**: fourth level, identified by a four-digit numerical code

For instance, the retail sale of clothing uses the following classification values per level:

- **Section G**: Wholesale and retail trade; repair of motor vehicles and motorcycles

- **Division 47**: Retail trade, except of motor vehicles and motorcycles
- **Group 47.7**: Retail sale of other goods in specialized stores
- **Class 47.71**: Retail sale of clothing in specialized stores

In the context of this thesis, we aim to collect multiple records of business activities in Volos. NACE will be used to effectively classify those enterprises and identify the ideal locations for each category. This thesis uses NACE Rev. 2, the current standard adopted in 2008.

2.5 LLMs and Fine-Tuning Techniques

Although traditional ML models have been integrated into recommendation systems, nowadays the focus has shifted to the new architecture of Large Language Models (LLMs). Below, we first provide the foundations of LLMs and then examine how they can be adapted to a specific field, such as location-based recommendation.

2.5.1 Introduction to Transformers & LLMs

Transformers are a groundbreaking machine learning technology that combines the methods of deep learning with the attention mechanism. Inspired by the 2017 paper ”Attention Is All You Need” [20], the attention mechanism is a method that enhances the vector representation of every text component with context information. In recent years, transformers have emerged as the state-of-the-art technology in machine learning, serving as the backbone of Large Language Models. Large Language Models (LLMs) are NLP models trained on extensive textual data that exhibit unparalleled understanding of natural language patterns [5]. Their ability to provide context-aware and personalized suggestions shows potential to improve recommendation systems. Multiple attempts have already been made to integrate LLMs in recommendation systems, such as GPT-based ChatBots that utilize ChatGPT [4]. Since general-purpose LLMs are trained on a broad web corpora, they are not specialized in an individual field. To achieve domain-specific knowledge, various adaptation techniques such as fine-tuning are applied.

2.5.2 Fine-Tuning Approaches

The motivation of this thesis is our desire to use an LLM-based approach in a specific task (business location recommendation). Training an LLM from scratch is computationally infeasible in practice, as it adapts billions of parameters and requires extensively large datasets and computational resources. The most frequent approach involves adapting an existing pre-trained model by modifying its parameters, based on domain-specific data [21]. This technique is called LLM fine-tuning, and has multiple variations which we analyze below.

Full-Model Fine-Tuning

Full-model fine-tuning focuses on updating the entire set of model weights to fit the specific task. While this approach can provide the maximum adaptability to the target domain, it remains resource-intensive and is generally impractical for simple research projects [22]. In order to make the LLM adaptation feasible, several techniques that perform fine-tuning more efficiently have been developed.

Parameter-Efficient Fine-Tuning (PEFT)

Parameter-efficient fine-tuning (PEFT) involves updating a relatively small proportion of the parameters while keeping the majority of the model frozen, in order to reduce computational cost and avoid overfitting. This approach drastically reduces the training parameters while retaining the general prior knowledge of the LLM.

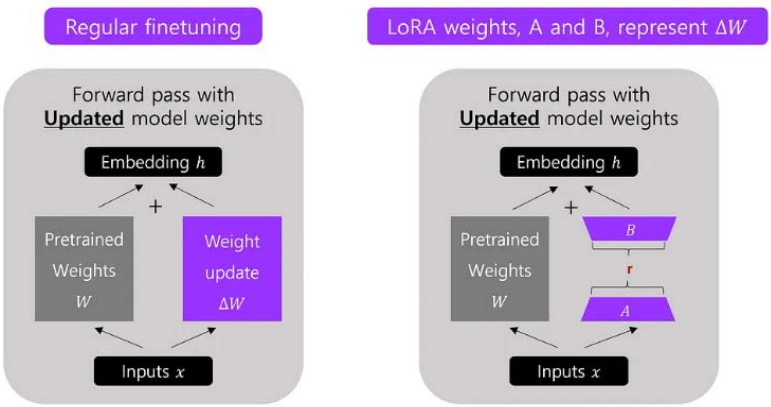


Figure 2.1: Full Fine-Tuning Vs LoRA

One popular emerging method that has been developed to achieve this goal is Low-Rank

Adaptation (LoRA). This technique injects low-rank matrices into the weight matrices of the transformer layers [22], which serve as the only trainable parameters during fine-tuning. The small proportion of trainable weights enables efficient adaptation with minimal overhead, that both preserves the pre-trained knowledge of the model and adapts to the desired domain. Figure 2.1 displays the difference between full fine-tuning and LoRA. An extension of LoRA, Quantized LoRA (QLoRA) further optimizes the fine-tuning process by quantizing the LLM in 4-bit precision. This significantly reduces the memory requirements for training, making it feasible to fine-tune LLMs on consumer-grade GPUs [22].

2.5.3 Role in Thesis

In the context of this thesis, the base model ”mistralai/Mistral-7B-Instruct-v0.2” was fine-tuned using QLoRA. In order to lower the memory footprint, we applied 4-bit NF4 quantization and gradient checkpointing, which only stores a few “checkpoint” activations computed during the forward pass of model training. The LoRA configuration contained a rank of 16, a scaling factor of 64, that controls the magnitude of the LoRA update relative to the frozen base model, and a dropout rate of 0.05 to prevent overfitting. A more detailed explanation of the fine-tuned LLM system is presented in section 4.2.

2.5.4 Limitations

While PEFT techniques significantly enable efficient fine-tuning, the approach of using a single model is prone to several disadvantages. Its main limitation resides in the static nature of the memory of the produced LLM. If we desire to update the knowledge of the model, we are forced to re-train it on updated data. Although the computational cost can be reduced by methods such as QLoRA, fine-tuning remains a time-consuming process we wish to avoid. The fine-tuned model can also suffer from hallucinations, as there is no guarantee that the generated output will always be accurate. To address these challenges, several alternative methods have been developed, such as Retrieval-Augmented Generation, which we explain in detail below.

2.6 Retrieval-Augmented Generation (RAG)

Fine-tuning an LLM to adapt to a specific task results in a specialized but relatively static model, prone to hallucinations. Retrieval-Augmented Generation (RAG) is a technique developed to solve these problems by leveraging dynamic knowledge.

2.6.1 Introduction to RAG

RAG is a method that enhances language model generation by incorporating external knowledge. Instead of relying on the memory stored in the model weights, we retrieve relevant information, typically documents or chunks of text, from external sources (e.g. files, databases), and feed it to the LLM to generate the final output [23]. This approach enables dynamic memory updating, as the knowledge of the model is stored outside of the LLM and can be modified. In addition, it drastically reduces the hallucinations, as the information is retrieved in real time from the outer source and passed to the LLM. Figure 2.2 displays the workflow of a RAG system.

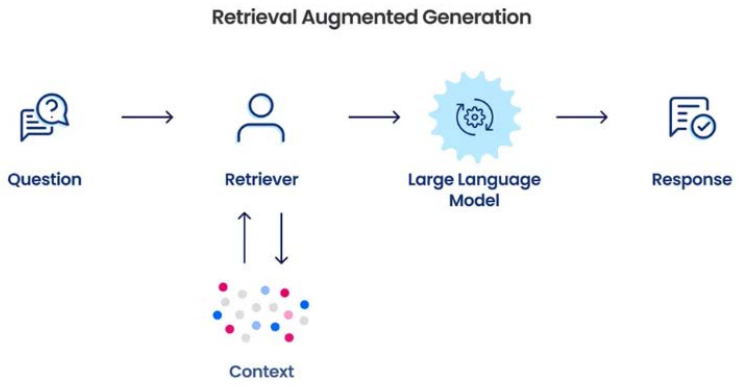


Figure 2.2: Retrieval-Augmented Generation Workflow

2.6.2 General RAG Workflow

The general workflow of a RAG system usually involves two main steps: retrieval and generation, each corresponding to a component, the retriever and the generator, respectively.

- **Retriever:** Responsible for fetching relevant information from external knowledge sources. It typically uses an embedder model, which encodes the user rquery in real time, and performs a similarity search with the pre-computed embeddings stored in

the sources. The most relevant documents are selected and returned for the generation step.

- **Generator:** A generative Large Language Model that receives the retrieved information and generates a context-aware coherent output. By basing the generation on the retrieved data the risk of hallucinations is mitigated.

2.6.3 Role in Thesis

Within this thesis, the retriever we use is ”BAAI/bge-large-en-v1.5”. This is a BERT-based encoder with 355 million parameters. The generator we select is ”Llama 3” with 8 billion parameters. Before deploying the RAG framework, we apply the embedder on the neighborhood and NACE descriptions to pre-compute their vector representations, so we can simply load them during runtime. The process involves encoding the user query in real-time, performing a similarity search with the NACE embeddings to extract the desired business type, and another similarity search, this time with the neighborhood embeddings, to suggest recommendations that align with user preferences. The selected locations are then fed into the generator along with their descriptions to construct the final user output. A more detailed explanation of the RAG pipeline system is presented in section 4.3.

2.6.4 Limitations

Although utilizing a RAG pipeline addresses certain challenges presented in the fine-tuning approach, it introduces its own set of limitations. Since the pipeline consists of multiple modules, the inference time is bigger, making the system slower. In addition, the structure of the system largely depends on the quality of the retriever and the generator, chaining it to their own weaknesses. Finally, it usually requires additional infrastructure for storing the pre-computed embeddings.

2.7 Introduction to Chatbots

Chatbots are advanced generative AI models that thrive at understanding and generating human language. Over the recent years, they have undergone a significant change in terms of quality and implementation, due to the emergence of Large Language Models (LLMs) [24].

While traditional chatbots used simple keyword-based rules to answer questions, modern chatbots enhance user experience by leveraging the state-of-the-art NLP ability of LLMs, enabling dynamic request-response conversations with personalized and context-aware responses. This has motivated the application of chatbots to various domains in order to provide new insights and hopefully better accuracy and performance.

While regular LLMs receive a text input and return a text output, the consecutive queries are independent and lack the context of the conversation. A chatbot wraps around an LLM by adding features that essentially transform the LLM into a complete application. The two LLM-based approaches we described in sections 2.5 and 2.6 are the brain of the system. We will add features such as dialog and memory management, an interface layer, and communication between the frontend and backend components, in order to create a user-friendly chatbot application.

Since chatbots are built upon LLMs, they inherit not only their strengths, but their limitations as well. They are also even harder to evaluate, as their predictions depend on the conversation history. In the context of this thesis, we aim to integrate a fine-tuned LLM approach and a RAG pipeline into a chatbot application, in order to exploit the advantages and disadvantages of each method.

2.8 Summary

In this chapter, we reviewed the research areas most relevant to the problem of business location recommendations, which form the basis of this thesis. We started by mentioning the motivation behind improving business location recommendation, and how GIS data enhance location analysis by providing geographic and demographic content. Next, we introduced machine learning and its integration with GIS, highlighting how ML models were used as supportive tools for generating proxy ground truths and providing interpretability. NACE codes were presented as a classification technique for economic activities. Moreover, we examined LLMs and domain-specific adaptation through fine-tuning, as well as the RAG framework and its components. Both approaches present different strengths and limitations, which we aim to exploit in the following chapters. Finally, we reviewed the process of transforming a simple LLM-based system into a chatbot application.

Chapter 3

Data Collection and Preprocessing

This chapter describes the data collection process and the preprocessing applied to make the data suitable for the models that will be used later. We will start by explaining how the multiple datasets were acquired, then see how they are joined along with the exploratory data analysis of the final dataset, and finally explore how the ground truths of the task were created.

3.1 Overview of Data Sources

The main datasets used in this thesis are the following 3: the Business Entries Dataset, the Municipal Communities Dataset, and the Neighborhoods Dataset. We will start by seeing how each of these was acquired and what preprocessing modification were made to prepare them for their Exploratory Data Analysis and later practical utilization.

3.1.1 Business Entries Dataset

Our first aim was to collect a dataset of business entries across the city of Volos. In order to do this we collected data from multiple sources and via different methods. Below, we will be breaking down each source and the process of merging them.

Foursquare (API Retrieval)

Foursquare is a global platform that provides information about places and human mobility [25]. In order to acquire data from this source, we used its public API. Specifically, we

created a grid of points around Volos 3.1 that represent the centroid of overlapping circles, in order to cover the area of the whole city 3.2.

Figure 3.1: Grid Centroids

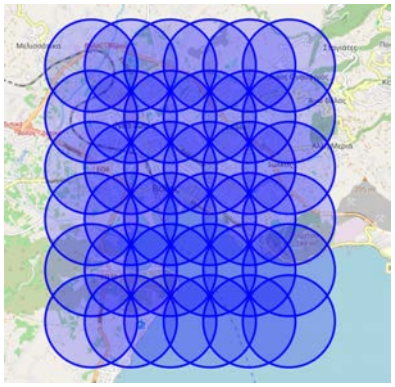


Figure 3.2: Total Area Covered

An API call was made for each circular area and the results were stored in a JSON file ('places_{i}.json'). Then, we merged the results and deleted the duplicates that occurred due to the area overlapping, and ended up with the final JSON file ('all_places_deduplicated.json'). Finally, we transformed the json file into a CSV file in order for the entries to later be able to merge with entries from other sources ('foursquare_data.csv'). This final file also contains the features 'Rating' and 'Reviews' which we manually added via the Foursquare API for each entry, and also the feature 'Postal Code' that was acquired via the Google Maps API. The total entries of this source were 307. The structure of the Foursquare entries was the following: Name, Address, Rating, City, Postal Code, Latitude, Longitude, Category, Country, Reviews.

Xrysos Odigos (Web Scraping)

Xrysos Odigos (Yellow Pages) is a website that contains information about businesses of various categories across Greece [26]. In order to collect data from this source, the Web Scraper Chrome Extension was used. The raw entries acquired were initially 4364. This source did not provide the 'Latitude' and 'Longitude' features, therefore we performed geocoding through the Google Maps API. The method was successful for 3420/4364 of the entries, so the remaining coordinates were inserted manually, using the Google Maps App. The data was saved in a CSV file ('xrysos-odigos-volos-ratings.csv'). The structure of the Xrysos Odigos entries was the following: Name, Address, Rating, City, Postal Code, Latitude, Longitude, Category, Country, Reviews, Description, Phone.

Greek Catalog (Web scraping)

Greek Catalog is an online business directory for Greece, similar to Xrysos Odigos [27]. In order to collect data from this source, we utilized the Web Scraper Chrome Extension once more. The raw entries collected were 46 this time. This source did not provide the ’Postal Code’ feature, therefore we performed geocoding through the Google Maps API. The data was saved in a CSV file (’greek-catalog-volos.csv’). The structure of the Greek Catalog entries was the following: Name, Address, Rating, City, Postal Code, Latitude, Longitude, Category, Country, Reviews.

Vacation Renter (Web scraping)

Vacation Renter is a travel accommodation aggregator that aggregates hotel listings from multiple booking platforms (Booking.com, Vrbo, etc.) [28]. In order to collect data from this source, the Web Scraper Chrome Extension was used one final time. The raw entries acquired were 24. This source did not provide the ’Latitude’, ’Longitude’ and ’Postal Code’ features either, therefore the Google Maps API as used to perform geocoding one final time. The ’Category’ features was also set to ’Hotel’ since this source specifies on this sector. The data was saved in a CSV file (’vacation-renter-volos.csv’). The structure of the data was the following: Name, Address, Rating, City, Postal Code, Latitude, Longitude, Category, Country, Description, Features.

Merging the Business Data Sources

On the table below, we can see a review of the structure of the individual data collected from the multiple sources mentioned above, as it has already been modified in order to facilitate their merging. Since the ’Phone’ and ’Features’ columns only exist in one dataset, we decided to drop them. Regarding the features ’Description’ and ’Reviews’ which are now the only missing features on some datasets, we simply set their values to Nan in these cases. The actual merging is simple, since all the individual datasets now comprise of the same structure. I ended up with a total of 307 + 4364 + 46 + 24 = 4741 entries, and 11 features.

Table 3.1: Attribute Presence Across Sources (highlighted were kept)

Attribute	Foursquare	Xrysos Odigos	Greek Catalog	Vacation Renter
Name	✓	✓	✓	✓
Address	✓	✓	✓	✓
Rating	✓	✓	✓	✓
City	✓	✓	✓	✓
Postal Code	✓	✓	✓	✓
Latitude/Longitude	✓	✓	✓	✓
Category	✓	✓	✓	✓
Country	✓	✓	✓	✓
Reviews	✓	✓	✓	–
Description	–	✓	–	✓
Phone	–	✓	–	–
Features	–	–	–	✓
Number of Entries	307	4364	46	24

Cleaning the Merged Business Dataset

Following the concatenation of the individual business datasets, a cleaning process was applied to ensure the final dataset's integrity and uniformity. Initially, 10 duplicates across the different datasets were identified and deleted. The 'Description' feature, that previously contained unintended line breaks, was reformatted for consistency. A total of 400 entries were found to not actually be located within the administrative boundaries of Volos, and therefore they were eliminated. After verifying the geographical accuracy, some wrongly performed Geocodings were corrected, either manually, or programatically via the Google Maps API, and 1 problematic entry was deleted.

Multiple entries lacked addresses, so we managed to find and manually insert a proportion, but deleted the 26 remaining. We also excluded 390 records lacking both address and geographic coordinates. Regarding the missing values of the feature 'Postal Code', we manually added them when they were available and removed 3 invalid entries. The feature 'City' also lacked 24 values, which we simply filled with "Βόλος Μαγνησίας". Finally, 4 'Category' values were missing, so we manually inserted them, and deleted 1 invalid record.

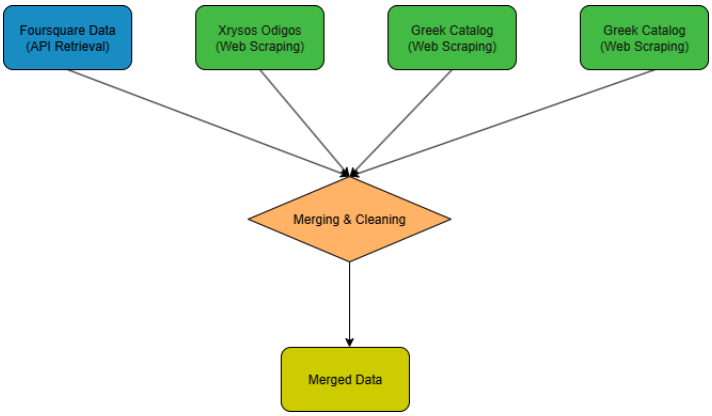


Figure 3.3: Merging & Cleaning the Business Data

Summing up all the records deleted in the cleaning process, we count: 10 + 400 + 1 + 26 + 390 + 3 + 1 = 831, and removing those from the original 4741 we are left with 3910 entries so far.

3.1.2 Municipal Communities Dataset

The next main dataset is the Municipal Communities Dataset. This information was gathered through the ELSTAT platform. This paragraph presents the process of acquiring the raw data and its preprocessing, leading to the creation of the final Municipal Communities Dataset.

Data Acquisition (ELSTAT)

ELSTAT is the official national statistics service of Greece [29]. It provides detailed demographic, social, and economic data at various administrative levels. Since the final model's recommendations are expected to be areas within Volos, we searched for the smallest geographic unit with available public datasets in ELSTAT. These units were the Municipal Communities, and we kept the ones within the boundaries of Magnesia (76). The platform provided demographic and economic information for each of them.

The demographic data A.2 contain the population count of each record, along with information about age, educational and marital status groups. More precisely, the age groups are divided to five-year intervals, while the education levels are categorized into six levels. Marital status is recorded as single, married, widowed, or divorced. The total features provided are 32.

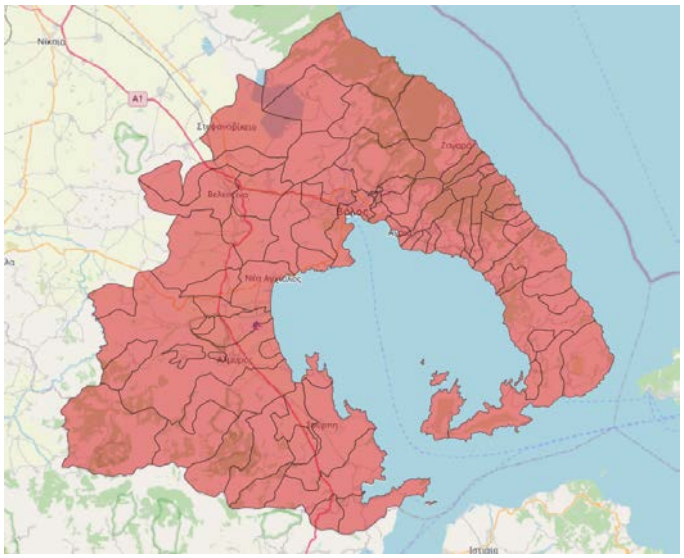


Figure 3.4: Municipal Communities within Magnesia

The economic features acquired from the ELSTAT platform A.3 are organized hierarchically. The root feature 'Σύνολο' is divided into 'Οικονομικά ενεργοί' and 'Οικονομικά μη ενεργοί'. 'Οικονομικά ενεργοί' is further categorized into 'Άνεργοι' and 'Απασχολούμενοι', while the latter is finally splitted into 'Πρωτογενής', 'Δευτερογενής', 'Τριτογενής'. The total features provided are 12.

The merging of demographic and economic features was simple, since they reference the same municipal community records. The economic and demographic datasets had 4 features in common ('ΠΕΡΙΦΕΡΕΙΑΚΗ ΕΝΟΤΗΤΑ', 'ΔΗΜΟΣ', 'ΔΗΜΟΤΙΚΗ ΕΝΟΤΗΤΑ', 'ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ'), so the total number of features ended up being 32 + 12 - 4 = 40. The shape of the dataset is therefore 76x40 so far.

Feature Engineering

After merging the demographic and economic datasets, preprocessing steps were applied to guide the feature selection and engineering so the final dataset is enhanced for later analysis. The first transformation applied was to normalize the raw count values for the demographic features of age, education, and marital status groups. This is done to put the features on a comparable scale and to be better fitted for their eventual integration with machine learning models, which benefit from normalized scales.

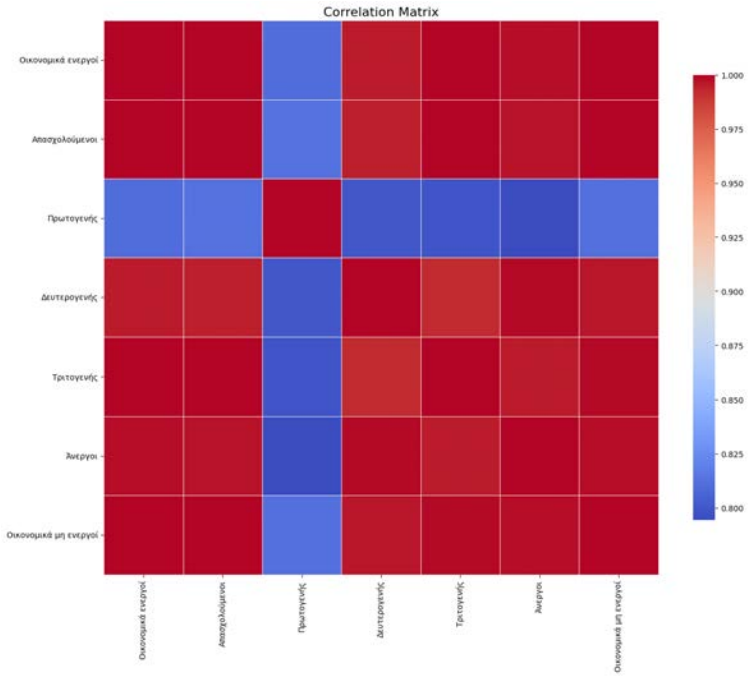


Figure 3.5: Correlation Matrix of Economic Features

The raw economic features were organized hierarchically, meaning that they contained redundant information. This can be verified by applying a correlation matrix that demonstrates their strong multicollinearity 3.5. Therefore, we used them to extract some meaningful economic features, like unemployment rate, labor force participation rate, and primary, secondary, and tertiary sector rate. The equations used to extract them from the old features are described in A.2.2.

In order to reduce the dimensionality of the dataset, we narrowed down the features within age ($15 \rightarrow 3$), educational level ($6 \rightarrow 3$) and economic feature groups ($8 \rightarrow 5$) A.2.2. The old features were eliminated. The marital status group was not changed.

In addition, the feature 'Area_km2' was added through the use of QGIS, which represents the total land area of the municipal community measured in square kilometers. This was possible because, apart from their tabular data, ELSTAT also provided the shapefiles for the geographical units.

I also added a feature with the English name of the Municipal Communities. The final Municipal Communities Dataset has 76 entries and 22 features A.4.

3.1.3 Neighborhoods Dataset

While the Municipal Community level data are indeed useful, their unit's size is large for the task we aim to achieve. Ideally, we would like smaller geographical units to represent the candidate locations. Therefore, we also created the Neighborhoods Dataset. This paragraph presents the process of acquiring data on the neighborhood level.

Data Acquisition (QGIS)

QGIS (Quantum Geographic Information System) is a free and open-source GIS software used for viewing, editing, and analyzing geospatial data [12]. To help acquire spatial data, it provides a QuickOSM plugin, which allows querying and downloading OpenStreetMap data based on specific tags and spatial filters. We used multiple filters (neighborhoods/villages/-suburbs/...) to make queries within Volos and therefore received multiple layers, both of point and polygon type. After processing them, we merged them into 1 point layer and stored them as a shapefile ('merged_neighborhood_points.shp').

In order for each business entry to be able to be matched to its corresponding neighborhood later, the shapefile should also possess a geomgetry attribute. To achieve this we applied Voronoi Polygons to transform the Point Layer to a Polygon Layer and saved it into another shapefile ('final_voronoi_output.gpkg'). The Voronoi Polygons tool partitions space into regions such that every location within a polygon is closer to its generating point (in this case, the neighborhood point layer) than to any other point [30]. In 3.6 we can see the initial point layer and the produced polygon layer.

Feature Engineering

Next, we followed a feature enhancement process, that provided some additional features by utilizing QGIS. We were able to extract the centroid coordinates ('x_centroid', 'y_centroid') and the polygon area ('area_km2') for each neighborhood.

We also collected the distance of every neighborhood from various Points Of Interest (POIs). The city center and the port were represented by fixed points, so the distance of each neighborhood to these POIs was measured as the distance between those fixed points and the neighborhood centroids.

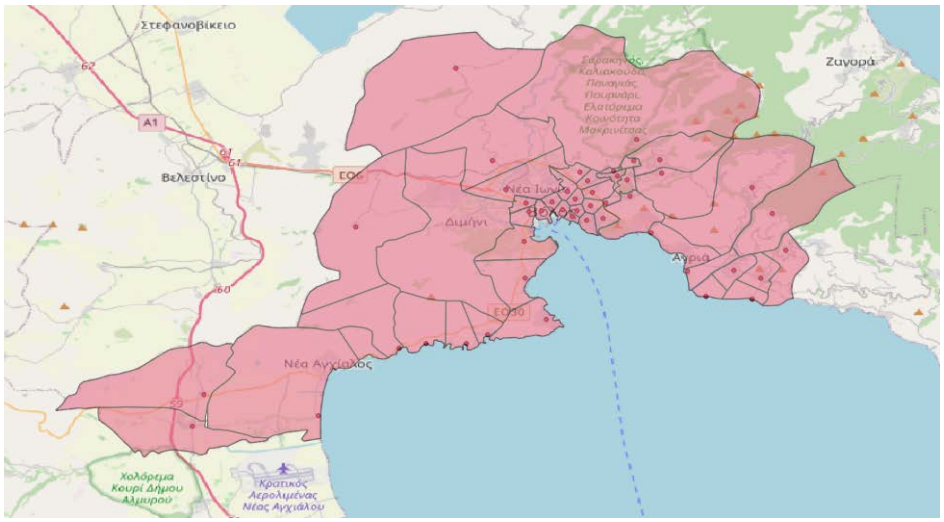


Figure 3.6: Neighborhoods within Magnesia

Certain POIs, such as main roads, transport stations, and universities, are represented by multiple points or lines. Consequently, I used the QuickOSM plugin to extract a shapefile for these cases. Then, for each neighborhood centroid, the nearest geometry was identified using the 'geopy' library. The final form of the Neighborhood dataset consists of 48 entries with 11 features so far A.5.

3.2 Datasets Integration

In the previous section, we analyzed how the 3 main datasets were created. In this section, we will see how they are integrated with each other and how the total features end up being greatly enriched in comparison to the initial business dataset. We will start by showing how the NACE code is assigned to each business and then see how we join it with its corresponding municipal community and neighborhood.

3.2.1 NACE Code Integration

In order to standardize business activity classification, the next step was the integration of the NACE code for each business entry. The workflow for this task involved the isolation of all the unique "Category" feature values, and their mapping to their corresponding NACE code. This mapping was performed manually to ensure accuracy, given the variability in category naming across sources. Once it was completed, it was joined with the merged dataset, to add the NACE code and description to each record. In the course of the aforementioned

process 4 invalid entries were spotted and deleted, leading to the final number of business records being 3906.

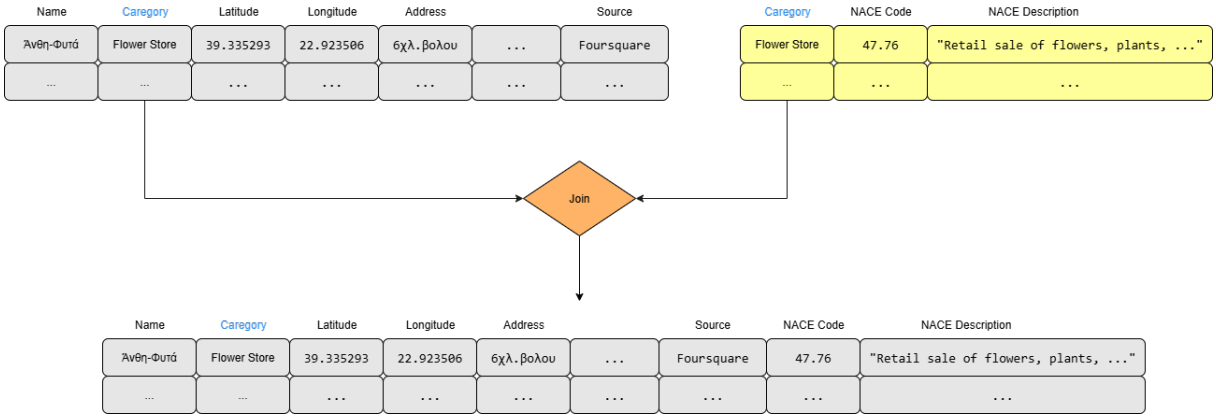


Figure 3.7: Integrating NACE code to the Business Data

Since we added these 2 features, plus we added a 'Source' feature that identifies where I extracted the specific entry from, the shape of the final business dataset is: 3906 entries and 14 columns A.1.

3.2.2 Municipal Communities Integration

Next, we integrated the business entries with the Municipal Communities Dataset. This process consists of two steps: assigning every business to its corresponding municipal community, and merging on this value with the other dataset. The first step was implemented by performing a spatial join between the business dataset and the polygon layer of Municipal Communities programmatically [31]. After adding the 'Municipal Community' feature to every entry, we simply join the 2 datasets on it.

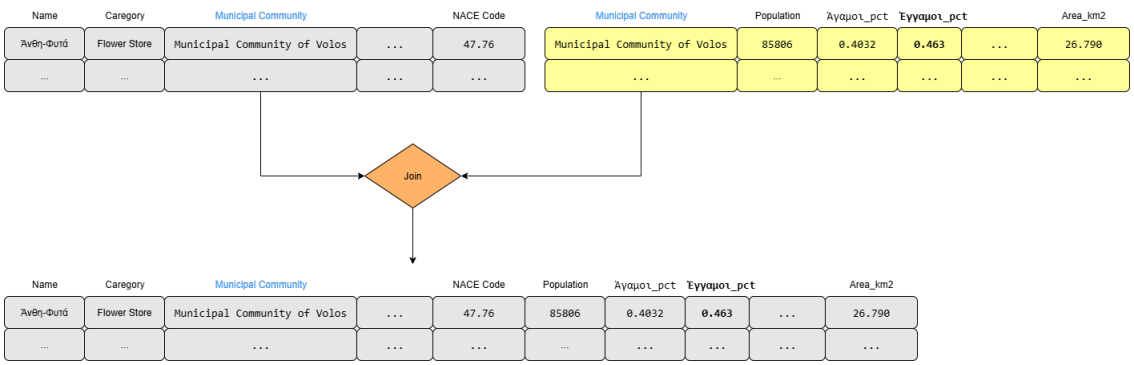


Figure 3.8: Integrating Municipal Community Features to the Business Data

The important part of this process was the 1st step, essentially the enhancement of the business data without necessarily joining the 2 datasets, since this can be done at any time later. The number of entries remains 3906, and the number of features of just the business data is 15 since we just added the 'Municipal Community' feature. If we want to inspect the total features of the joined set, they are 36 since we added the 22 features of the municipal community, and we dropped 'ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ ΕΛΛ' as keeping the Greek name was unnecessary.

3.2.3 Neighborhoods Integration

The workflow followed on this paragraph perfectly mirrors the previous one. We perform another spatial join [31], this time assigning each business to its corresponding neighborhood. The spatial join failed for 24 entries because all the neighborhoods belong to the municipality of Volos, while these specific businesses did not, so they were deleted. The valid entries can now be joined to acquire the Neighborhood Dataset features.

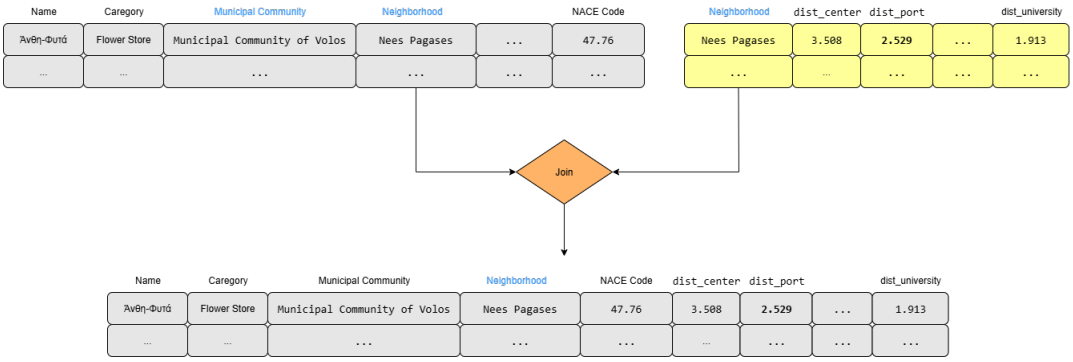


Figure 3.9: Integrating Neighborhood Features to the Business Data

As was the case above, the important improvement through this process was the integration of the 'Neighborhood' feature on the business entries. The shape of the business dataset is now 3882 records, due to the 24 invalid spatial joins, and 16 columns since adding 'Neighborhood'. If we look at the fully joined dataset, after joining both on the 'Municipal Community' and 'Neighborhood' features, and keeping only the useful columns for a business entry (dropping 'Geometry', 'Centroid_x','Centroid_y', 'Neighborhood_Greek','ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ ΕΛΛ'), their number is 42. In we can see the final schema of the relationships of the 3 main datasets.

3.2.4 Integrating Municipal Communities to Neighborhoods

In the previous two sections we saw how we connect an individual business to its municipal community and neighborhood. However, there is also a connection between the two geographical units themselves, since each neighborhood essentially lies within a municipal community. We can also add the 'Municipal Community' feature to the Neighborhood Dataset by performing a spatial join [31] on its centroids with the municipal communities. The added feature results in a 48x12 shape for the neighborhood dataset. After we do this, we can join the neighborhood with the features of its corresponding municipal community. In 3.10 we can see the final relationship of the 3 main datasets. The arrows act as foreign keys that allow joining on the specific features.

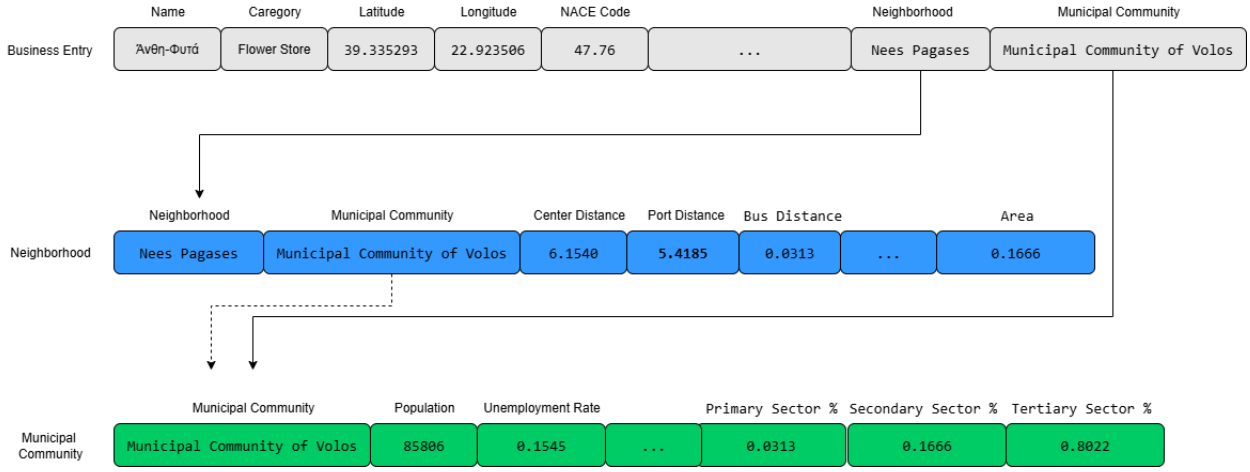


Figure 3.10: Relations of the 3 Datasets

3.3 Exploratory Data Analysis

In the previous sections, we described how we obtained the 3 main datasets and how they are related in order to be joined. In the current paragraph, we will analyze the 3 main datasets individually and then the final joined dataset.

3.3.1 EDA - Business Dataset (before joining)

We started by inspecting the features of the pure business dataset, before joining it with the other ones. As we mentioned previously, the shape of this dataset is 3882x16. Most of its features are spatial as they provide information about its location. As is obvious, all entries

in the ’Country’ field are set to ’Greece’. The overwhelming majority of the ’City’ column contains the value ’Βόλος Μαγνησίας’ (82.86%), while the next most frequent value is ’Νέα Ιωνία Μαγνησίας’ (11.23%) A.1. The entries contain 12 unique ’Postal Code’ values, with ’38221’, ’38333’ and ’38334’ being the most common A.2. In 3.11 we can see a clustered version of the businesses on a map, as specifying every point would hinder the clarity of the visualization. We see that most records are located within the city center.

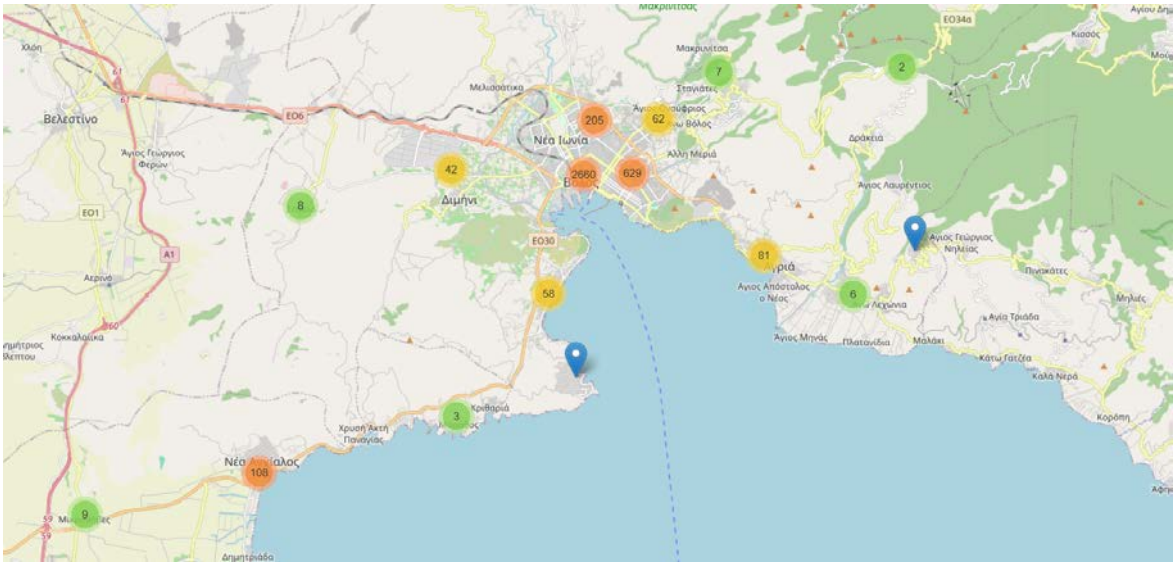


Figure 3.11: Clustered Businesses Visualization

Figure 3.12: ’Rating’ Feature Heatmap

Although the ’Rating’ values are extremely sparse (only 5.87% present), the distribution

of their values is right-skewed A.6. The figure 3.12 shows a heat map of the ratings in the city, highlighting their highest values mainly in the center and the Palaia area. Regarding the categorization of the business type, the 'Category' feature consisted of 1233 unique values A.5, while the 'NACE Code' feature only had 306. This is expected since the categogies were the raw values we acquired along with the business entries, while the official categorization method that we will use throughout this thesis is the NACE protocol. In 3.13 we can see the most frequent of business types, based on the NACE Code Description. We see that they comprise medical activities, food service and beverage activities, education, and more.

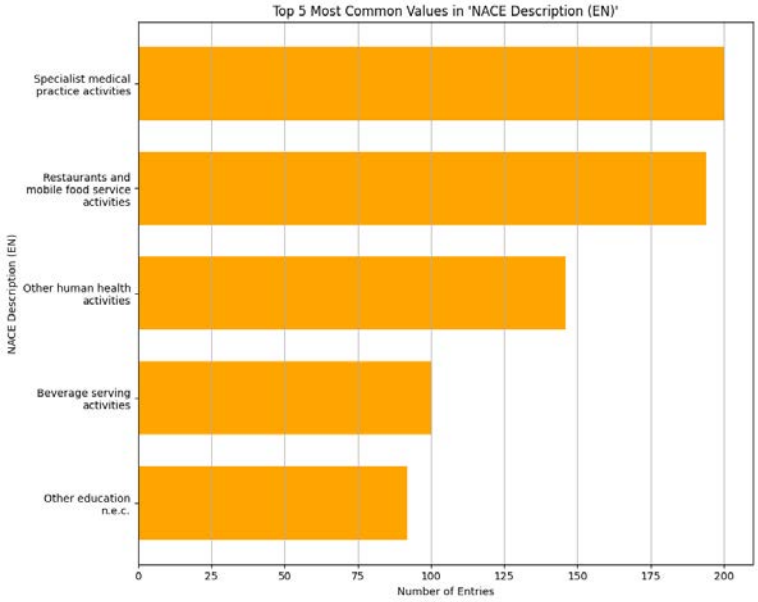


Figure 3.13: 'NACE Code' Bar Chart

Finally, looking into the 'Neighborhood' feature, we can see that its distribution A.3 is much less skewed than the one of 'Municipal Community' A.4. This is logical because the latter feature refers to a much larger geographical unit which contains the majority of the businesses. The neighborhoods provide a better and informative separation of the areas, and this is why they will be used as the candidates of the recommendation system.

3.3.2 EDA - Municipal Communities Dataset

Subsequently, the Municipal Communities dataset was analyzed, which consists of 76 entries with 23 features. Although 75% of the spatial units host fewer than 902 people, the municipal communities of Volos and Nea Ionia are outliers, hosting 85.806 and 31.683 respectively. Figure 3.14 demonstrates the value distribution of the feature, after having re-

moved the 2 extreme outliers mentioned to provide a better visualization. The 'Area_km2' also contains outliers A.7. An attempt was made to identify a correlation between the population and the area features, no sufficient statistical evidence existed to assume such a relationship. The distribution of 'unemployment_rate' visually suggests a normal distribution approximation. Several statistical tests were applied and did not reject this suspicion while a QQ-Plot strengthened it A.8.

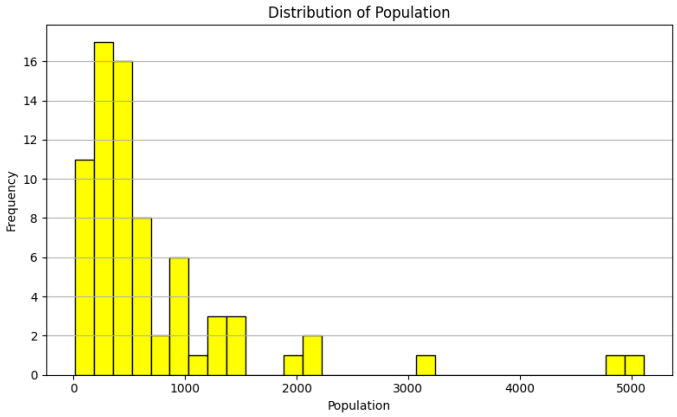


Figure 3.14: 'Population' Feature Histogram

The age, education level, marriage and economic sector groups columns comprise feature subsets that suffer internally from multicollinearity since they are perfectly linearly dependent (A.14, A.15, A.16, A.17). Specifically, the percentages of high and low education exhibit a strong negative correlation, as do the percentages of married and unmarried individuals, and the shares of the primary and tertiary sectors. Examining the mean age feature values across all municipal communities shows that the majority of the population is between 10 and 65 years of age A.9. In addition, on average, the majority of the community people posess a low educational level, and higher education levels become progressively less common. A.10. The mean percentages of married and unmarried citizens correspond to approximately 48.89% and 35.93% respectively. Most people are employed in the tertiary sector, followed by the primary and then the secondary sector A.11.

There is a moderate negative correlation between the 'Population' and the 'age_65_plus' columns, which indicates that the most populated municipal communities consist of fewer older people A.12. Another relationship I identified was between the unemployment rate and the education level. In particular, it presented a positive correlation with higher education levels 3.15 and a negative correlation with lower education levels 3.16, both indicating that areas with higher education correspond to lower employment rates. Finally, there is a positive

relationship between the proportion of citizens aged 15–64 and labor force participation, which is expected since this age group constitutes the working-age population. A.13.

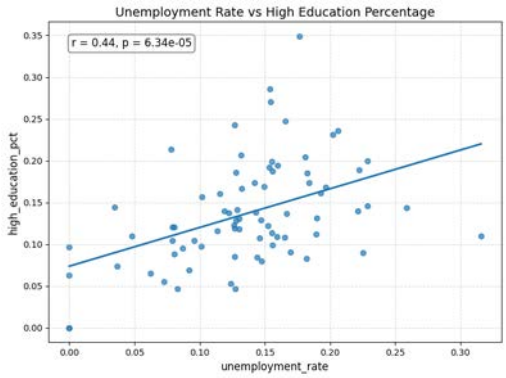


Figure 3.15: Scatter Plot: Unemployment Rate VS High Education Level

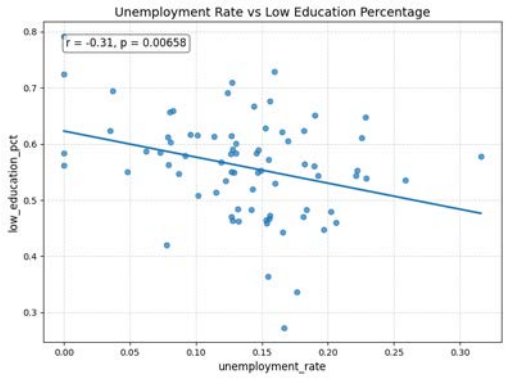


Figure 3.16: Scatter Plot: Unemployment Rate VS Low Education Level

3.3.3 EDA - Neighborhoods Dataset

Next, we explored the features of the Neighborhoods Dataset, which consists of 48 entries and 12 total columns. The ’Neighborhood_Area_km2’ column represents the area of each neighborhood and contains more outliers than the municipal community area A.18. This is caused by the large number of small neighborhoods, particularly those located near the city center, while simultaneously peripheral areas are divided into fewer segments, as we can see in Figure3.6. The largest areas—Makrinitsa, Glafyra, Dimini, Dimitriada, and Sesklo—also appear as outliers A.19α′.

Regarding the distance from the city center, Agios Vasilios, Analipsi, Metamorfosi, Agios Nikolaos and Epta Platania - Oksigono are the most central neighborhoods A.19β′. After inspecting the relationship of the neighborhood distance from the city center with various other features, we identified that it presents a positive correlation with many of them. A weak but statistically significant positive relationship with the distance from main roads A.20α′ confirms the suspicion that central areas tend to have more highways. Its moderate correlation with the distance from transport stops 3.17 suggests that public transport is more accessible in central areas. It also presents a nearly perfect correlation with the distance from university 3.18, which shows that, on average, universities are relatively absent from peripheral areas. Another almost perfect linear relationship exist between the distance from the city center and from the city port, which is expected since they are located closely. Finally, a significant

direct relationship exists with the neighborhood area A.20β′, which reflects the shape of the produced Voronoi Polygons and confirms that indeed the peripheral areas are divided into fewer and therefore larger segments.

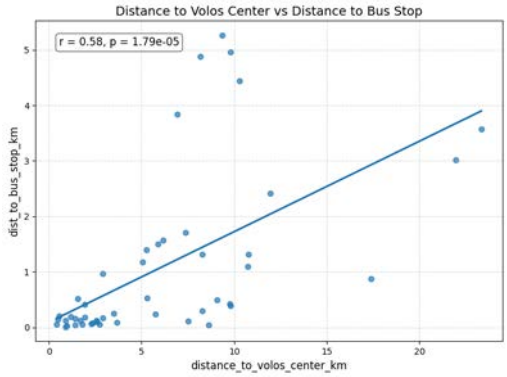


Figure 3.17: Scatter Plot: Distance from Center VS Distance from Transport Stops

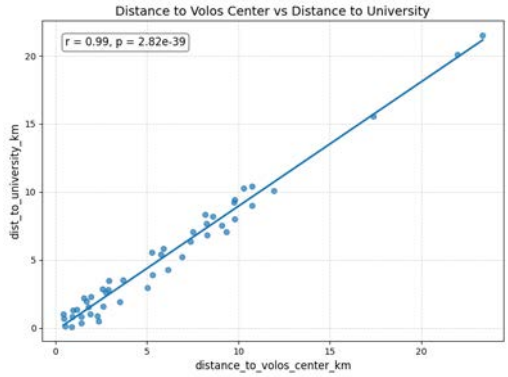


Figure 3.18: Scatter Plot: Distance from Center VS Distance from University

3.3.4 EDA - Joined Dataset

In the three previous sections we inspected and analyzed the 3 main datasets individually. To identify patterns among features originating from different datasets, we present below the analysis of the final joined dataset. It consists of 3882 entries and 42 total features A.6. Inspecting the missing data we see that the dataset is complete, except from the potential label columns (’Rating’, ’Reviews’, ’Description’), which are very sparse A.21. First, we attempted to relate the business category with several joined features. In order to use fewer categories to get the bigger picture, I grouped the NACE codes by division instead of class and found their mean values across some features. These graphs do not represent a cause and effect relationship between the NACE code and the feature, they just state the context by highlighting observed associations of the columns.

Regarding the unemployment rate, although the top values are similar, it is most dominant in the manufacture of wood, electricity and air conditioning supply, land transport, manufacture of computer and electronic products, and manufacture of machinery and equipment 3.19. We also see a similar pattern when analyzing the high educational level per NACE division, with the highest values being in the following sections: travel agency, advertising, insurance, activities of membership organisations and management consultancy activities 3.20.

Despite the severe sparseness of ratings, they present a significant negative relationship

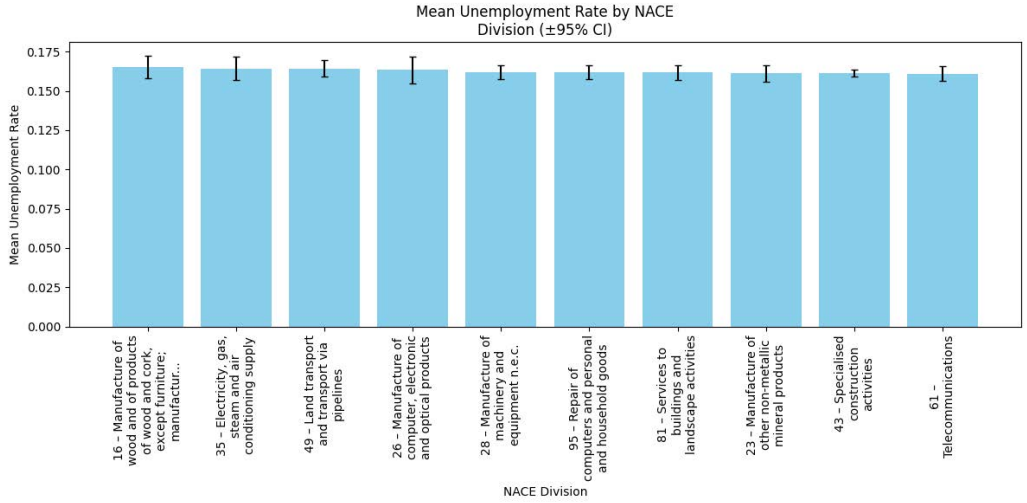


Figure 3.19: Mean Unemployment per NACE Division Bar Chart

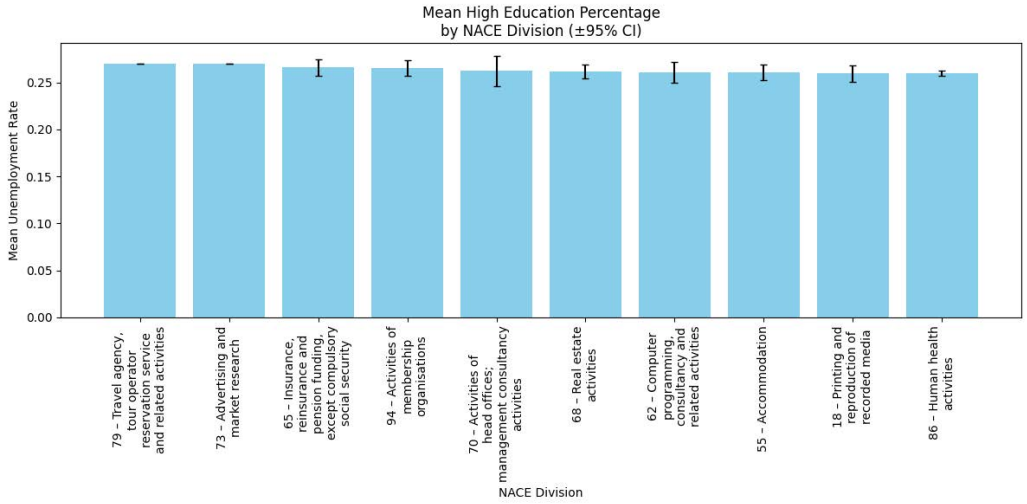


Figure 3.20: Mean High Education per NACE Division Bar Chart

with the unemployment rate. To acquire this I calculated the mean rating per municipal community and compared it with the unemployment rate. This indicates that areas that contain highly rated businesses tend to also have a lower unemployment rate value. As before, this only provides context for interpreting rating values that tend to occur alongside specific unemployment rates. I also tried to identify a connection between the rating and the distance from center, but there was no sufficient statistical evidence that supported this suspicion.

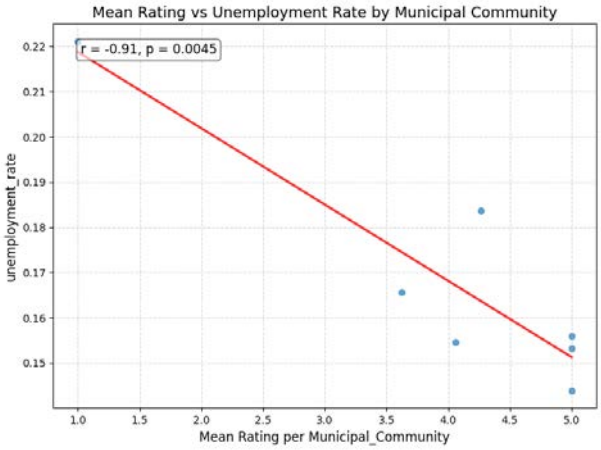


Figure 3.21: Scatter Plot: Mean Rating VS Unemployment Rate

3.4 Proxy Ground Truth Creation

In order to utilize machine learning for creating a recommendation system of optimal business locations, we need a criterion to rank the candidate areas for every category to use the top ones as labels. The intuitive approach would be to filter the entries for every NACE class, and use the rating average of every neighborhood to sort the best candidates. This is not feasible, however, due to the sparsity of the labels, so we were forced to create a proxy ground truth myself.

Following principles of weak supervision and ensemble learning, three complementary approaches were implemented: supervised (learning from labeled instances), unsupervised (capturing latent data structure), and semi-supervised (expanding limited labels). To mitigate the bias of any single method and to also exploit the distinct aspect of the problem space that each approach captures, their outputs were ensembled into a unified ranking. While this does not provide the definitive ground truth, it establishes a consistent and interpretable proxy based on which the models can be trained and evaluated. For every approach, we implemented an attempt that only uses the neighborhood columns, and one that also includes the municipal community features. The two attempts were compared by measuring the amount of central suggestions they provide, and the best was kept.

3.4.1 Supervised Learning Approach

To address the lack of explicit labels, a supervised machine learning approach was first developed. The objective was to approximate the “success likelihood” of each location for a

given business category (NACE code).

Model Dataset Creation

The worflow begins by creating the dataset that we aim to train the model on. The entries of the dataset consist of every possible (NACE class, Neighborhood) pair, therefore 306x48 = 14688 records. We join the entries with the features of its corresponding neighborhood, and in a second attempt, also with their corresponding municipal communities. The target column does not relate with the ratings, rather it is a new binary value, indicating the presence or absence of a business of this type in a given neighborhood 3.22. The dataset presents a considerable class imbalance as only 11.58% of the label values are 1.

NACE Code	Neighborhood	Center Distance	Port Distance	Bus Distance		Area	Target
47.76	Nees Pagases	6.1540	5.4185	0.0313	...	0.1666	1
47.76	Agios Nikolaos	4.125	1.112	0.007	...	1.143	0
47.76	...	...	...	...	...	...	...

Figure 3.22: ML Model Dataset Creation

Top-10 Extraction

The model used is a Random Forest with 100 decision trees. Prior to training, the input features were scaled to the range [0,1] using 'MinMaxScaler' to ensure comparability with other models in the pipeline. Once trained on the dataset, the model is inferenced to obtain prediction probabilities for every entry. These probability values are then used as ranking criteria for the neighborhoods within each NACE class, with the top 10 being selected as the proxy ground truths. Both attempts resulted in similar lists, but the second was chosen as it yielded a slightly higher percentage of central recommendations (81.48% over 81.05%). Although promising, this approach is limited by its implicit assumption that the presence of a business in a location implies a high probability of success, which is not always the case in real world scenarios.

3.4.2 Unsupervised Learning Approach

A second method that uses unsupervised learning was applied, with the goal of identifying the "ideal conditions" for each business category. Unlike the supervised method, this

approach does not rely on labels but instead exploits the feature space structure.

Clustering

First, we filter the business entries for every NACE class and apply k-means clutering on them. The optimal value of the k parameter is determined by applying the eblow method. After the clusters are created, we determine the ”ideal cluster” by comparing them based on the value of a ”judge feature”, in our case the ratings. Each cluster centroid represents the average profile of the businesses in that group, therefore the centroid of the ideal cluster is assumed to describe the ”average ideal conditions” for the specific business type.

Top-10 Extraction

Once the ideal cluster centroid is obtained, we can calculate its Euclidean distance into the same feature space with all the candidate neighborhoods and use it as the ranking criterion. The top 10 recommendations are the 10 neighborhoods closer to the centroid. In 3.23 we see a 2D PCA projection of the high-dimensional feature space containing the candidates and their distance from the ideal cluster centroid for the NACE class 10.52. Although this time the recommendations do not occur solely based on the current business presence, the identification of the “ideal” cluster relies on a single judge feature, oversimplifying the multidimensional nature of business success.

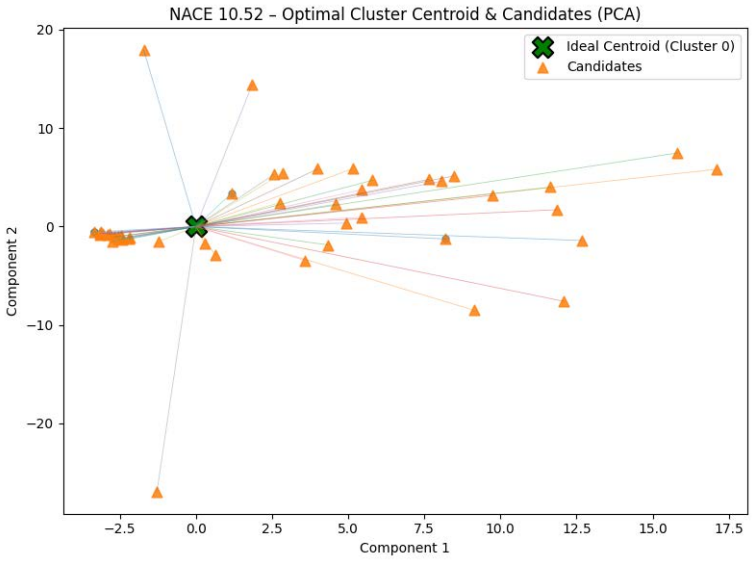


Figure 3.23: Ideal Cluster Centroid & Candidates (2D PCA)

3.4.3 Semi-Supervised Learning Approach

The third method adopted was a semi-supervised learning (SSL) technique, specifically self-training. SSL methods leverage both the limited labeled data and the large pool of unlabeled data by iteratively expanding the training set with pseudo-labels.

Self Training

The model used is a Random Forest Regressor with 100 estimators. The process of self training start with training the initial model on the small set of labeled data (5.87%). The trained model is then inferenced to produce labels (pseudo-labels) for the unlabeled entries. The pseudo-labels that are predicted ”confidently” get integrated to the training set. The model is then re-trained on the updated training set, and the procedure repeats until either all the labels enter the set or no confident predictions occur 3.24. The confidence of the predictions is determined by comparing its variance to a predetermined threshold explicitly set at 0.1.

Top-10 Extraction

After the ratings are enhanced, we can simply use the mean rating value of each neighborhood as the ranking mechanism to find the optimal locations for each business type. This time only neighborhood features were used since they provided a slightly higher percentage of central locations than the full feature approach. This method resulted to less than 10 recommendations for specific NACE classes since such business types did not exist within every neighborhood, so no rating was created. This is not a problem because the final proxy truths will be determined by combining all three approaches.

Name	Caregory	Latitude	Longitude		Neighborhood	Rating	
Άνθη-Φυτά	Flower Store	39.335293	22.923506	...	Nees Pagases	4.7	Real Label
Μέταλλο και ξύλο	Furniture	39.339432	22.923224	...	Agios Nikolaos	3.5	Pseudo-Labels
Frago Cargo	Shipping	39.332756	22.929457	...	Nees Pagases	4.5	

Figure 3.24: Real & Pseudo Labels

3.4.4 Ensemble Method

The techniques outlined in the previous sections capture and contribute an additional view of the problem, but they also suffer from multiple limitations. Following principles of ensemble learning, I combined the individual weaker predictions to obtain the final proxy ground truths. Below, we show a comparison of the individual results and then present their combined outcome.

Comparison of Results

Firstly, I compared the three individual proxy truth results to see if they agree or not. I computed the agreement count per NACE class, which essentially combines the total suggestions from all the sources along with their frequency. For example, NACE class 23.70 ”cutting, shaping and finishing of stone” presented the following results: 3 votes for Agioi Anargiroi, 3 votes for Nea Ionia, 2 votes for Analipsi, and 1 vote for Epta Platania - Oksigono. In Figure 3.25 we can see the histogram of the unique suggestions across all NACE classes. A total of 3 suggestions suggest a prefect agreement of the three methods, whereas with more suggestions their results tend to diverge. We see that the majority of different recommendations gathers at 4-6, implying a partial agreement between the techniques.

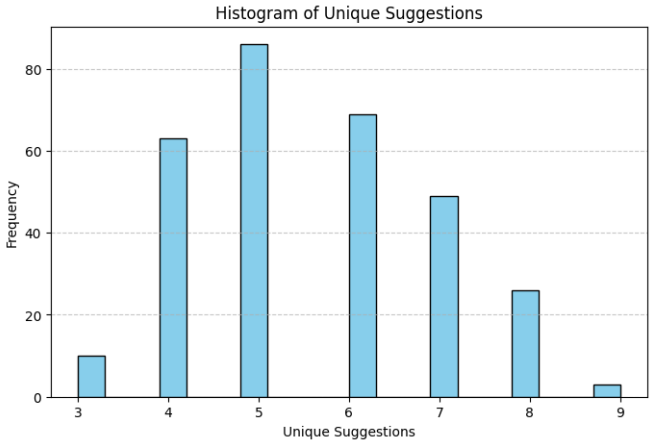


Figure 3.25: Unique Suggestions of all Methods Histogram

Venn Diagrams can provide a more intuitive understanding of the similarity of the individual suggestions A.5, but they only relate to a specific NACE class and not the wider picture as the histogram did. I also calculated the Jaccard similarity of every NACE class between the pairs of the three aproaches. Figure 3.26 demonstrates the average Jaccard similarity across

all NACE classes, and we see that the unsupervised and semi-supervised approach provide the most similar results. Spatial similarity within 1 kilometer was also measured, which measures how geographically close the suggestions are, even when they differ. This also showed that the supervised and unsupervised approaches were the most dissimilars, while the highest similarity was observed between the supervised and semi-supervised approaches.

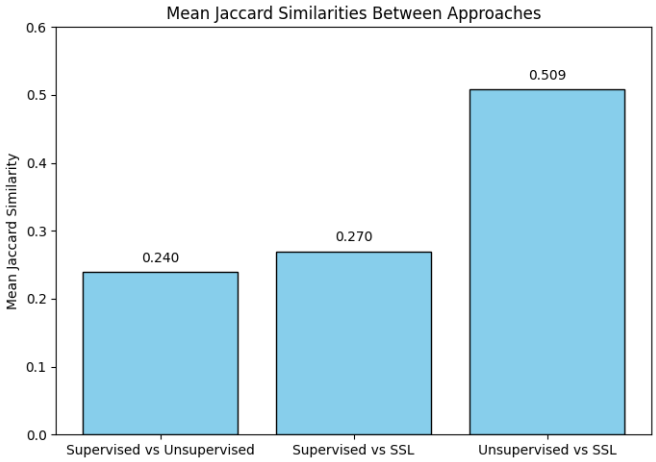


Figure 3.26: Mean Jaccard Similarities Bar Chart

Combination of Results

To combine the outputs of multiple approaches into a single consensus ranking, a weighted rank aggregation procedure was implemented. Throughout this thesis we have assumed that the central locations tend to be favoured in general, therefore the weight that each approach gets assigned is based on the proportion of its recommendations that fell within the municipality of Volos after being normalized so they add up to 1. In table 3.2 we see the central areas pectentage and the corresponding weight of each approach.

Approach	**Central Suggestions Percentage**	**Weight**
Supervised	81.48%	$w_s = 0.3110$
Unsupervised	93.79%	$w_u = 0.3580$
Semi-supervised	86.67%	$w_{ss} = 0.3308$

Table 3.2: Central Areas Percentage and Weight of Approaches

In addition to method weights, the rank position of each suggested neighborhood was

also weighted. Specifically, the following values were assigned: Top1 = 10, Top2 = 9, …, Top10 = 1. This happens to ensure that a candidate appearing higher in a method's ranking contributed more strongly to the final consensus. In order to obtain the final neighborhood score, we simply perform the weighted sum of the method and rank position 3.1:

$$score(N_i) = w_s \cdot top_{s,i} + w_u \cdot top_{u,i} + w_{ss} \cdot top_{ss,i} \tag{3.1}$$

This cumulative score is the final ranking criterion. For every NACE class the candidates were sorted by their scores in descending order and the 10 highest were selected as the final proxy ground truth. In Table 3.3 we provide some indicative examples of the final top 3 estimated locations for certain NACE classes.

NACE Class	**NACE Description**	**Top 1**	**Top 2**	**Top 3**
1.13	Growing of vegetables and melons, roots and tubers	Metamorfosi	Kato Lehonia	Palaia
43.22	Plumbing, heat and air conditioning installation	Agia Paraskeui	Nea Ionia	Agios Georgios
68.32	Management of real estate on a fee or contract basis	Agios Nikolaos	Agios Vasilios	Agios Konstantinos
79.11	Travel agency activities	Metamorfosi	Agios Konstantinos	Analipsi

Table 3.3: Final Top-3 Recommendations of NACE Classes

Chapter 4

Methodology / System Implementation

In this chapter we describe the modeling approaches that were used for the creation of the business location recommendation system. We begin by discussing the general motivation behind each method that led to their implementation and comparison, and then proceed to further analyze each in detail.

4.1 Overview of Modeling Approaches

Given the recommandation system we aim to implement, and the ambition to achieve this through the use of generative AI, two state-of-the-art approaches were suggested for the task, each with different strengths, limitations, and ways of utilizing the gathered data. An initial approach involves fine-tuning a Large Language Model (LLM) to specialize in predicting optimal locations while also providing a contextual and user-personalized response. An alternative approach that has emerged in recent years is the construction of a Retrieval-Augmented Generation (RAG) system. In this framework, the actual decision-making process takes place outside the LLM, which is instead used to generate the chatbot-like user output.

Both methodologies present different strengths, limitations and implementation details. In general, a fine-tuned LLM is expected to provide a faster inference that can generalize better on unseen prompts due to its wider knowledge as a pre-trained model. The RAG workflow ,on the other hand, supposedly reduces the hallucinations since its memory does not lie inside the LLM but it fetches data from databases at runtime, and can possibly lead to better personalization due to its more interpretable re-ranking ability.

4.2 Fine-tuned LLM

This section examines the approach of fine-tuning an LLM to specialize on the task of providing the three best locations to open a business. We will first analyze the motivation behind this method, then show the detailed process of creating the fine-tuning and inference datasets, and finally describe the fine-tuning attempts and their comparison based on their results on the inference dataset.

4.2.1 Motivation

Traditional recommendation systems, such as collaborative filtering, content-based filtering, and hybrid approaches, have long been prominent in the field of data science. Despite their indisputable effectiveness, limitations like the cold start problem (struggling when limited historical data is available) or the lack of the explainability of the recommendations have led to the gradual adoption of LLMs in the field. Although several attempts in general recommendation tasks (e.g. movies, products) have tested the suitability and success of LLMs, their application in spatial data and specifically business location recommendation is still very limited. In addition, the potential to provide contextual and personalized suggestions along with justification represents a novel yet promising approach that led to the idea of using LLMs for this problem. In the following paragraph, we show the creation of the essential datasets, the detailed fine-tuning, and the evaluation of the final model.

4.2.2 Fine-Tuning Dataset Creation

This section describes analytically the construction of the fine-tuning dataset. This constitutes the core of the methodology, as the content and structure of the final model's outputs are learned directly based on its training data.

Synthetic NACE Categories

The final goal is a model that responds to an every-day description of the business that the user is interested in, (”Where can I open a cafe in Volos?”), not one that specifically mentions its NACE class (”Where can I open a business of NACE class 56.3?”), since this encoding is too technical, or even unknown to the user. Therefore, training prompts should consist of such category descriptions and not the NACE classes themselves. In order to obtain

such descriptions generative AI was used. In particular, ChatGPT was produced 3 synthetic categories for each NACE class 4.1. Each synthetic category will correspond to 1 entry on the fine-tuning dataset, resulting in 306x3 = 918 entries.

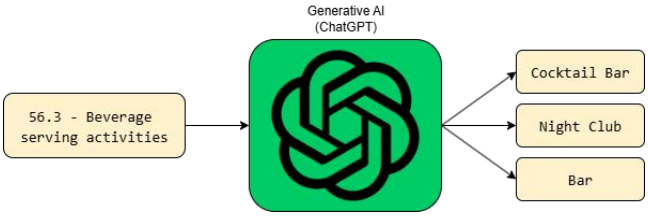


Figure 4.1: Synthetic Categories Creation

Feature Importance per NACE Class

Looking forward, we aim to involve justification for every suggestion provided, as one of the primary motivations of using an LLM was explainability. Since the features describe various aspects of the suggestion (e.g. distance from city center, distance from main roads, area), we aim to use their values for the appropriateness of the recommendation. This requires deciding which features to include in the model's response, since using them all would generate unnecessarily long outputs. To make this decision, feature importance was found and used as a ranking criterion to select features from highest to lowest. The ranking was found individually for every NACE class, since it is assumed that feature importances change between different business types. To acquire the feature importances per NACE class, I used the 'shap' library.

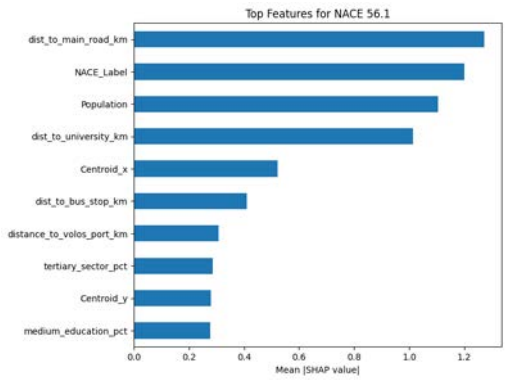


Figure 4.2: SHAP Values of NACE Class 56.10

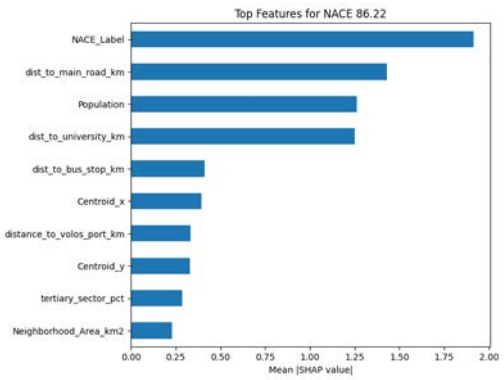


Figure 4.3: SHAP Values of NACE Class 86.22

An XGBoost machine learning model was trained on a dataset consistsing of all possible

(NACE Class, Neighborhood) pairs, resulting in 306 x 48 = 14.688 entries. Each entry is joined with its corresponding neighborhood and municipal community features (apart from 'Neighborhood_Greek', 'Geometry' and 'ΠΕΡΙΦΕΡΕΙΑΚΗ ΕΝΟΤΗΤΑ', 'ΔΗΜΟΣ', 'ΔΗΜΟΤΙΚΗ ΕΝΟΤΗΤΑ', 'ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ ΕΛΛ' respectively). The label used is binary, indicating whether the specific neighborhood belongs to the top 3 recommendations for the corresponding NACE class.

After model training, we extract its SHAP values to get insight on the feature importances. While the initial SHAP values refer to individual entries, they can be aggregated by averaging either across all NACE classes or even within each NACE class individually. In figures 4.2, 4.3 we see the descending order of SHAP values for specific NACE classes, while in 4.4 we see the mean feature importances across all columns. Beyond the NACE class itself, the largest average SHAP values are associated with population, distance to main roads, and distance to universities. The individual rankings will be the feature selection criterion when creating the training outputs.

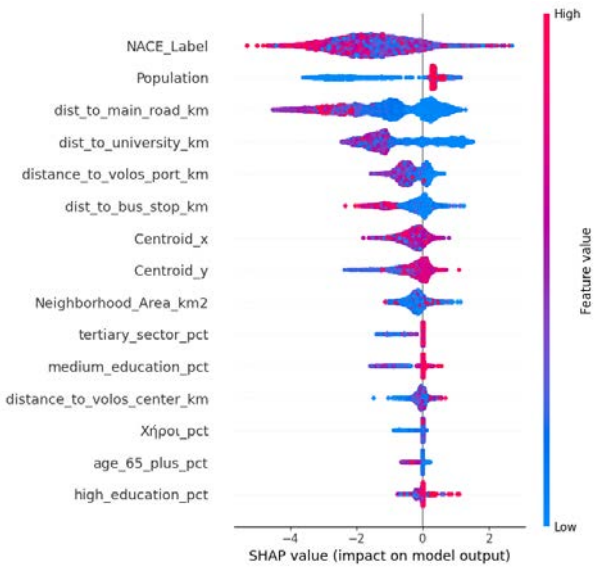


Figure 4.4: Average SHAP Values of all Features

Feature Statistics & Feature Justification

The SHAP values displayed in the previous section described which features will be justified, but did not explain how. To provide informative justifications for each feature, I calculated specific statistics per feature, namely the 25th, 50th, and 75th percentiles (q25, q50, q75). During feature justification, each feature value is compared with these percentiles to

determine its corresponding interval. For every interval, 3 justification templates have been generated by ChatGPT, one of which is randomly selected to provide variation. From this point forward, we will refer to this process as ”feature justification”. Figure 4.5 visually demonstrates the different templates used based on the value and statistics.

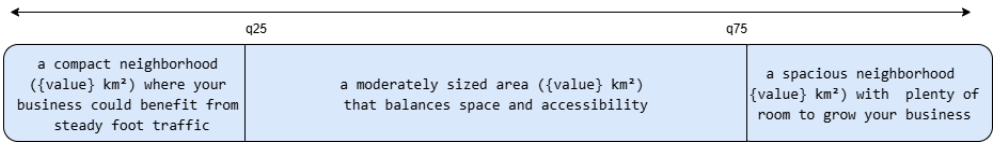


Figure 4.5: Feature Justification for ’Neighborhood_Area_km2’

Neighborhood Justification

During the construction of training completions, we aim to provide explainable outputs which justify each recommendation. Building on the feature justification method described above, a complete reasoning about a neighborhood can be implemented by aggregating the justifications of several features. At this stage, it becomes essential to select certain features, as using them all would result in an overly large output.

Six features were used to provide explainability to the output. Certain features, assumed to be consistently important, were treated as ”fixed”. Examples include neighborhood area, proximity to the city center, and distance from highways B.1. During neighborhood justification, 4 out of 7 fixed features are randomly selected and justified. The two remaining features were selected according to their SHAP values, specifically choosing the two most important features not included among the fixed ones. This combination of fixed features with SHAP importance balances consistency with variety of the training output.

Prompt Construction

The creation of training prompts is relatively straightforward. They are based on five synthetic templates generated by ChatGPT, each parametrised by the business category. When constructing the fine-tuning dataset the synthetic category is passed as input. Table 4.1 displays the synthetic prompt templates. We generate one template per synthetic category, resulting in a total amount of 918.

Prompt Templates
”Where should I open a **{category}** in Volos?”
”What is the best neighborhood in Volos for a **{category}**?”
”I’m looking to start a **{category}**. Which area would be ideal in Volos?”
”Suggest a location in Volos for a **{category}**.”
”Which place would you recommend for opening a **{category}** business in Volos?”

Table 4.1: Synthetic Prompt Templates

Completion Construction

The construction of training completions involves the justification mechanisms mentioned above. For every synthetic category, we extract the proxy top 3 recommendations of its corresponding NACE class as estimated in section 3.4. The recommendations are transformed to explanatory text by applying the neighborhood justification method described previously 4.2.2, and combined to form the final completion. The workflow is shown intuitively in Figure 4.6.

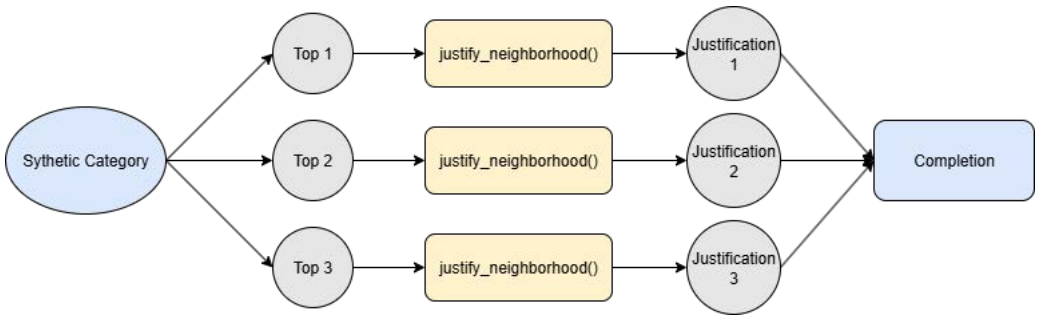


Figure 4.6: Completion Construction Workflow

Prompt-Completion Example

To illustrate the fine-tuning dataset creation process, we present a prompt-completion pair generated for the synthetic category ”greenhouse farming” that corresponds to the NACE class 1.13 in 4.7. The total prompt-completion pairs are 918 as mentioned before.

Prompt-Completion Example

Prompt: *Where should I open a greenhouse farming business in Volos?*

Completion: *Top neighborhoods for a Greenhouse farming business (NACE 1.13: Growing of vegetables and melons, roots and tubers) in Volos:*

1. **Metamorfosi** – right near the university (0.14 km) — great if you want to attract students or young professionals; a large local population, which can bring in steady foot traffic; a compact neighborhood (0.56 km^2) where your business could benefit from steady foot traffic; excellent central location (0.54 km), making your business easy to find and reach; accessible without being too exposed to street noise or congestion (0.16 km); fairly accessible by public transport (0.20 km), even if not right next to a bus stop.

2. **Kato Lehonia** – a balanced location (2.07 km^2) — not too dense, not too open; a busy neighborhood with a strong base of potential customers nearby; close enough (8.62 km) to be convenient, while still offering some breathing room from downtown traffic; great access to main roads (0.01 km) — easy for customers to reach you and for deliveries to run smoothly; not close to campus (8.21 km), which could work well for businesses focused on the neighborhood itself; excellent access to public transit (0.04 km), which can boost foot traffic to your business.

3. **Agios Konstantinos** – a moderately sized area (0.82 km^2) that balances space and accessibility; it's very close to the city center (1.41 km), which means great visibility and easy access; a reasonable distance from transit (0.16 km) — close enough for most customers to reach you without much hassle; lots of people live in the area — great for attracting walk-in customers; reasonably close to main roads (0.13 km), giving you decent accessibility without heavy traffic; a lively spot near the university (0.82 km) that could bring in a lot of student activity.

Figure 4.7: Prompt-Completion Example

4.2.3 Inference Dataset Creation

The construction of the inference dataset is similar to that of the training dataset, differing only in the synthetic categories used. All NACE classes were divided into two categories based on their frequency on the business entries dataset: classes that characterize at least 50 businesses were considered main classes, otherwise they are treated as secondary. The main classes are 53 and comprise almost 63.32% of the total businesses. During the inference data creation only main classes are used. This is done to provide a more realistic model evaluation, as keeping only frequent business types results in an inference dataset representative of real-world examples. As before, 3 synthetic categories per class were generated, accounting for 3x53 = 159 total entries. The construction of the prompts and completions is identical to that described previously.

4.2.4 Fine-Tuning Process

After showing the construction of the training dataset, we can finally examine the fine-tuning process. Below, we mention details like computational recourses, the base model, and the fine-tuning techniques applied.

Hardware & Base Model

The training was carried out on the high-performance computing cluster of University of Thessally, that utilizes an NVIDIA A100 GPU. Given our hardware capabilities, ”mistralai/Mistral-7B-Instruct-v0.2” was chosen for its balance between strong instruction-following capabilities and computational efficiency.

Quantization & QLoRA

Fine-tuning was preformed using the QLoRA (Quantized Low-Rank Adaption) method, which combines quantization of the base model with integration of lightweight trainable adepter layers. In order to fit the 7 billion parameter within the available GPU memory, the base model weights were loaded in 4-bit NF4 quantization and kept frozen during training. Since these weights remain unchanged, LoRA adapters, specificlly 16-rank matrices, were inserted as trainable parameters. When the parameters of the model are printed, we get the values shown on Table 4.2. The total parameters are displayed to be around 3.75 instead of

7 billion due to the compressed representation after quantization. In addition, the tiny percentage of trainable parameters demonstrates the effect of QLoRA adapters, resulting in a significantly more lightweight model with only 0.1813% trainable weights.

Total Parameters	3.758.895.104
Trainable parameters	6.815.744
Trainable %	0.1813%

Table 4.2: Model Parameters after QLoRA

Tokenization & Label Masking

Before performing the model fine-tuning, each prompt-completion example was converted to compatible formatting for the Mistral-Instruct tokenizer, using the template 4.1. The formatted string is then tokenized using a fixed length, either by truncation or padding, set to 512 after determining that 95% of the formatted strings result in 462 or less tokens.

```
<s>[INST] prompt [/INST] completion </s>
```
(4.1)

To ensure the model learns to predict only the completion and not the prompt, the labels of the instruction's tokens are masked out by setting their value to -100 so they get ignored when computing the crossentropy loss function.

Training Configuration

As we previously mentioned, 4-bit NF4 quantization was used along with QLoRA adapters of rank 16. The scaling factor, which determines the influence of the adapters, is set to 64, which is a balanced, standard choice. During the training we used an effective batch size per device of 2 and a gradient accumulation of 4, resulting in a total effective batch size of 8. The learning rate was set to 2e-5. The complete training configurating is displayed on table B.2. Regarding the number of epochs, three attempts were made with 3, 5, and 8 epochs respectively. The results will be further explored in chapter 5.

4.2.5 Fine-Tuned LLM Workflow Summary

So far we described the complete fine-tuning process. After training the model, its workflow is relatively straight forward. The LLM is simply fed a text input, and produces the predicted output. We showcase the workflow here to compare it later with the RAG pipeline, whose workflow is a bit more complex. Figure 4.8 provides an intuitive visualization of the system.

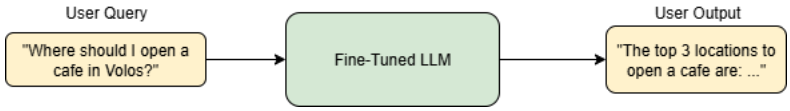


Figure 4.8: Fine-Tuned LLM Workflow

4.2.6 Inference Process

After performing model fine-tuning, the resulting LLM is used on the inference dataset described in 4.2.3. This process was also run on the NVIDIA A100 GPU of University of Thessally. To ensure computational efficiency, inference was performed in batches of size 8. The outputs are stored, as they will be used later to evaluate the model performance by comparing them to the completions created in section 4.2.2.

4.3 RAG with Re-Ranking Pipeline

This section explores the second state-of-the-art method that was implemented, a Retrieval-Augmented Generation workflow. First, we examine the motivation for exploring this method despite having already implemented the LLM fine-tuning approach. Followingly, we will show the creation of neighborhood descriptions that will be utilized in the framework. Finally, we will present the complete RAG pipeline, including the selected embedder, its application, and the integration with a generative LLM to produce the final output.

4.3.1 Motivation

While fine-tuning an LLM as described in section 4.2 indeed utilizes generative AI in an innovative way to address the problem of business location recommendation, it presents several important constraints. Adding proxy truths for new NACE classes or modifying existing

ones requires re-running the entire fine-tuning process, which is both time consuming and computationally expensive. The desire to avoid this makes the approach static. Moreover, the justifications may not always include valid numbers, since LLMs are prone to hallucinations. Furthermore, since the model essentially learns to predict three static recommendations per NACE class, personalization may be difficult if users have specific preferences. A Retrieval- Augmented Generation can potentially handle these problems because its memory is not stored inside the LLM and can be updated without extensive training In addition, the system outside of the LLM can adjust the final recommendations based on specific user desires, potentially allowing for better personalization.

4.3.2 Neighborhood Description Creation

Looking forward, we aim to compare the initial user query with the candidate locations to recommend the appropriate ones. Since the query is a text input which can be embedded into a dense vector representation, the first step was to create a text description per neighborhood that will later be also embedded and compared with the user query. The creation of descriptions is based on the core concepts used in the fine-tuning dataset construction 4.2.2, specifically the neighborhood justification method which essentially integrates multiple feature justifications.

Feature Statistics & Feature Justification

This process is identical to the feature justification workflow explained in section 4.2.2. The same statistic values are calculated for every feature, resulting in a specific text justification depending on the comparison of the individual feature's value in comparison with its percentiles. Once more, we randomly select one out of three templates per precentile interval for variety.

Neighborhood Description

The generation of descriptions is very similar to the construction of neighborhood justifications described previously 4.2.2, with some minor modifications. The major difference is that, in this case, producing large text is not a problem. While previously these justifications were used as LLM outputs and therefore had to be constrained, in this case, we aim to embed

Neighborhood Description

Agios Nikolaos: *Neighborhood description:*
A cozy location (0.35 km^2) that might work well for storefronts or quick-service ideas; it's very close to the city center (0.90 km), which means great visibility and easy access; very close to the port (0.53 km), which makes transport and logistics a lot easier; great access to main roads (0.00 km) — easy for customers to reach you and for deliveries to run smoothly; excellent access to public transit (0.01 km), which can boost foot traffic to your business; close to campus (0.09 km), which means steady foot traffic from a younger, educated crowd; a busy neighborhood with a strong base of potential customers nearby; a neighborhood with many single adults (40%) — ideal for modern, fast-moving business models; fewer married residents (46%), which could favor services tailored to individuals or younger adults; not a heavily elderly area (8%), which could favor modern, fast-paced, or tech-forward businesses.

Figure 4.9: Description for Agios Nikolaos

them. Hence, including as much information as possible about the neighborhood is not only acceptable but beneficial.

In more detail, the same fixed features are used B.1, but a justification is provided for all of them rather than only a subset. The total number of features used is also increased, resulting in a larger amount of secondary features as well. In 4.9 we provide an indicative example of a neighborhood description.

4.3.3 Embedding and Retrieval Process

The RAG pipeline uses an embedder model that transforms text descriptions into high-dimensional dense vector representations. This enables similarity search between user queries and neighborhood descriptions.

Embedder Model

To perform the embeddings, we selected the model "BAAI/bge-large-en-v1.5". This is a BERT-style transformer encoder with about 335 million parameters, optimized for semantic

similarity search and RAG pipelines. It uses a vocabulary size of 30.522 tokens, 24 transformer layers, and the generated embeddings are of dimension 1024.

Embedding NACE Descriptions

As we will explain later, after acquiring the user query, the first step is to determine the NACE class of the business the user wants to open. This will be done by performing a vector similarity search between the query embedding and all NACE class descriptions. To avoid embedding all NACE descriptions in real time, we precompute the vector representations once and store them, allowing them to be retrieved during runtime. They are stored in the JSON file 'nace_descriptions_embeddings.json'.

Embedding Neighborhood Descriptions

After determining the NACE code, we could simply use its corresponding proxy truths to provide the recommendations. In order to allow better personalization, however, we utilize a re-ranking method that initially fetches the top 10 suggestions, and re-ranks them based on vector similarity between them and the user query. This requires embedding the neighborhood descriptions created in section 4.3.2. As we previously did with the NACE descriptions, the embedding generation of neighborhood descriptions is performed once, and stored in the JSON file 'neighborhoods_descriptions_embeddings.json'.

Retrieval Process Output

The re-ranking process results in an updated sorted list of top 10 recommendations. Since the structure of the final output should consist of three suggestions, we simply select the three most similar neighborhoods. Figure 4.10 demonstrates the complete retrieval process as described in the previous sections.

4.3.4 LLM Integration

The retrieval process extracts the final suggestions that are intended to be presented to the user. In order to present them in a coherent, user-friendly recommendation, we utilize an LLM that receives the raw recommendations and descriptions and produces the final output message. In contrast with the fine-tuned LLM approach, in this case the LLM does not decide the actual recommendations, it acts only as the output generator.

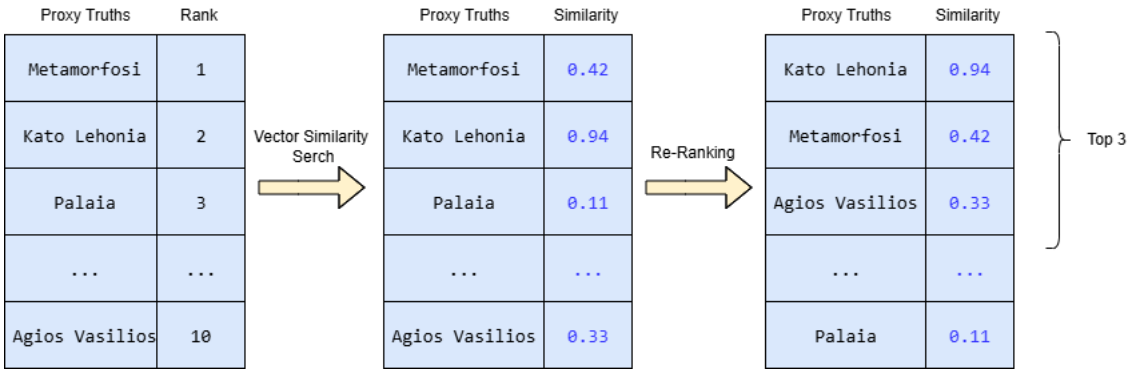


Figure 4.10: Retrieval Process

Selected LLM

For user output generation, Llama 3 (8 billion parameters) was used. The model is run locally via Ollama, and thus is loaded in 4-bit quantization to ensure compatibility with the local hardware. It supports a context length of 8192 tokens and embedding dimensionality of 4096. During retrieval-augmented generation, the model is simply fed a specifically structured input and inferred to obtain the user output.

LLM Prompt Construction & Final Output

To ensure clarity and user-friendly outputs, a structured template was constructed to guide the final output of the LLM. The template initially specifies the user query with the extracted NACE class and its description. Next, it mentions the three recommendations with their descriptions, and finishes by providing certain explicit output instructions. Figure 4.11 shows an indicative example of the LLM prompt constructed to provide recommendations to the query ”Where in Volos should I start a bakery business?”. The descriptions are incomplete to fit the whole prompt. After the LLM prompt is constructed, we simply feed it to the model and infer it to acquire the final output. This is the final response that the user receives from our system.

4.3.5 RAG Pipeline Workflow Summary

In the previous sections we examined the individual parts of the RAG framework. Here, we present a summary of the complete workflow. The system first receives the user query that asks about a specific type of business. Using the embedder, the entry is converted into a high-dimensional vector in real time. We find the corresponding NACE class of the business type

LLM Prompt Example

The user wants to open a business of type: ”Where in Volos should I start a bakery business?”
NACE Code: 46.36 — Wholesale of sugar and chocolate and sugar confectionery
Based on data analysis, here are the top 3 recommended neighborhoods in Volos:

Hiliadou – a balanced location (1.03 km²) — not too dense, not too open, you’ll be right near the heart of Volos (1.17 km) — convenient for both customers and deliveries, ...

Karagats – a cozy location (0.67 km²) that might work well for storefronts or quick-service ideas, it’s very close to the city center (0.98 km), which means great visibility and easy access, ...

Metamorfosi – a cozy location (0.56 km²) that might work well for storefronts or quick-service ideas, you’ll be right near the heart of Volos (0.54 km) — convenient for both customers and deliveries, ...

Please present the 3 neighborhoods in a numbered list format like this:
1. Neighborhood Name: Summary of strengths...
2. Neighborhood Name: Summary of strengths...
3. Neighborhood Name: Summary of strengths...
Each point should be 2–4 sentences long, clearly explaining why the neighborhood is a good fit for the business. Avoid ranking or comparisons — describe each one positively and independently.

Figure 4.11: LLM Prompt Example

by calculating the similarity between the query embedding, and the embedding of all NACE descriptions, which are pre-computed and stored locally. Next, we fetch the top 10 proxy recommendations of the NACE class, and apply vector semantic similarity once more, this time between the query and the neighborhood embeddings which are also available locally. The similarities are used as the re-ranking criterion to acquire personalized suggestions.

The three neighborhoods with higher similarity scores are the final recommendations. In order to return a cohesive and context-aware message to the user, we pass the recommendations and their descriptions to a specifically structured template, and feed this to the Llamma 3 LLM to produce the final output. Figure 4.12 visually shows the complete pipeline.

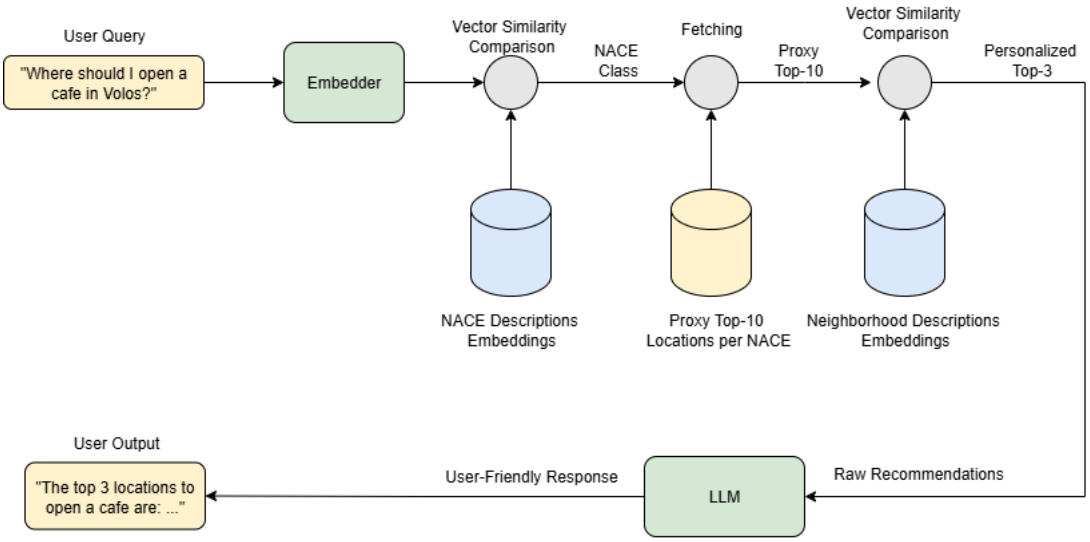


Figure 4.12: RAG Pipeline

4.3.6 Inference Process

After implemention the complete RAG pipeline, it is used on the inference dataset described in 4.2.3. In contrast to the fine-tuned LLM inference, this process is run locally since the loaded embedder is of reasonable size (355 million parameters) and the loaded LLM is quantized. The outputs are stored, as they will be used later to evaluate the pipeline performance by comparing them to the completions created in section 4.2.2.

4.4 Chatbot Interface

The previous sections 4.2 and 4.3 essentially describe the process of constructing frameworks that receive a text input and return a text output. In order to create a user-friendly application we create a chatbot-like interface. This section describes the frontend and backend development of the application, highlighting the mechanism that transforms simple LLMs into chatbots by memorizing the conversation.

4.4.1 Frontend Development

The frontend construction regards the form of the final user interface. Below we describe details of the fontend implementation. We start by mentioning why we chose the 'Streamlit' library, then showcase the interface design, and finally explain how the chat memory is stored in the frontend.

Streamlit Framework

Streamlit is an open-source Python framework designed to develop application frontends in a simple and efficient manner. Unlike traditional web frameworks, like React or Django, which require extensive knowledge of front-end technologies, Streamlit supports direct implementation in Pythn scripts. This makes it a great option for deploying machine learning models, where our primary goal is to provide a simple user interface rather than create a complex application. In the context of this thesis, Streamlit was utilized to implement the chatbot interface, as it provides a built-in session management and a simple layout system that can be easily integrated with a Python backend.

User Interface Design

The application design aims to mirror a state-of-the-art interface that provides a user-friendly experience. The page is initialized with the title "Business Location Chatbot Demo" and a greeting message from the AI assistant. At the bottom of the screen, an input box expects the input of the user. To improve readability, the conversation messages were designed with simple HTML and CSS embedded in Streamlit B.1, B.2. The assistant messages are shown in gray and aligned to the left, with the title "Assistant:". In contrast, the user messages ap-

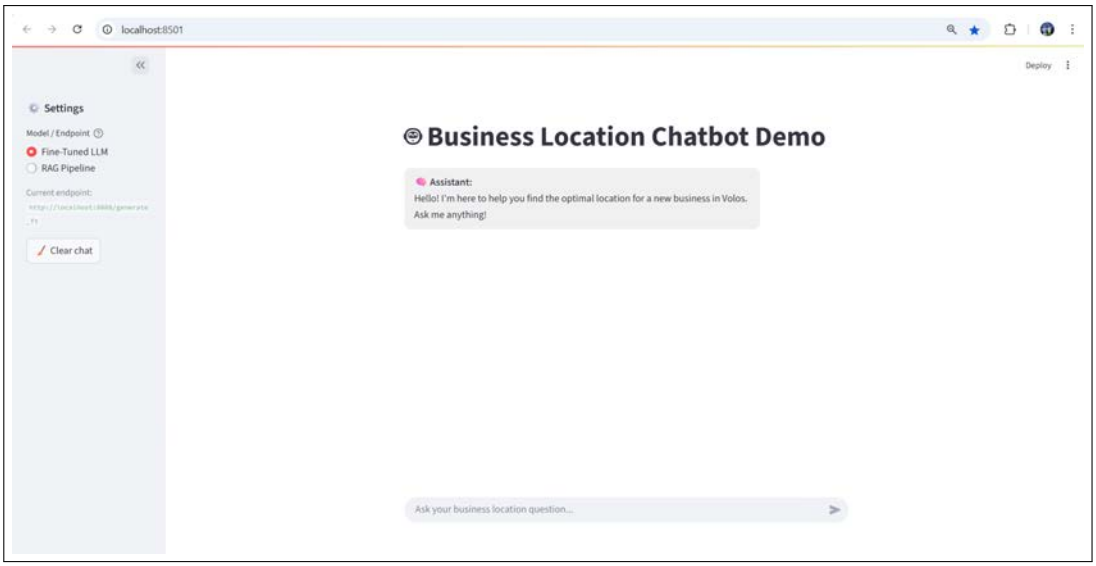


Figure 4.13: User Interface Design

pear in blue color, aligned to the right, and using the title ”You:”. The interface dynamically updates after every new message. To ensure a continuous flow of conversation, when a user query is submitted, a spinner animation is displayed until the request is completed, and the response is displayed as well. The interface also contains a sidebar, that allows the user to select the backend system used, which we further explain in section 4.4.2. Figure 4.13 shows an indicative screenshot of the application design, along with the greeting message.

Session & Memory Management

During Streamlit sessions, the Python script is re-executed on every user interaction. In order to save the conversation history, we use the ’st.session_state’ object, which contains messages within the same session. The messages are stored as JSON objects, with fields: ’role’, which identifies wether this is a user or assistant message, and ’content’, which contains the actual message. At the beginning of a new session, the session state is initialized only with the greeting message. Every new message, whether from the user or the assistant, is integrated into the state. This mechanism essentially stores the conversation memory in the frontend. The complete history is then forwarded to the backend and processed from there, as we will analyze below.

4.4.2 Backend Development

The backend acts as the bridge between the user interface and the AI models. It is responsible for handling requests from the frontend, producing the output by using the systems we implemented, and returning it as a response. Below, we explain the framework selected to build the backend and the endpoint functions for each method. It is important to highlight that, while the inference of the RAG framework ran locally 4.3.6, in order to use one server for the application, we run both approaches remotely on the computational cluster of University of Thessally.

FastAPI Framework

FastAPI is a modern Python framework, specifically designed for building APIs efficiently. Compared to traditional frameworks such as Flask or Django, FastAPI is lightweight and more simple. One of its main advantages is the support of asynchronous requests, which makes it suitable for long-running processes. This makes it ideal for our case, as querying a language model can be modeled in this framework.

Fine-Tuned LLM Pipeline

The fine-tuned LLM approach is wrapped in the FastAPI endpoint '/generate'. When the Python script starts running, we load the LLM ('mistralai/Mistral-7B-Instruct-v0.2') in 4-bit quantization and attach the LoRA adapters created during fine-tuning on top of it. Once the model is active, it listens for user queries. The process is almost identical to the simple inference of the fine-tuned LLM system, with minor changes. In order to transform a simple LLM module into a chatbot, we simply have to modify the system input.

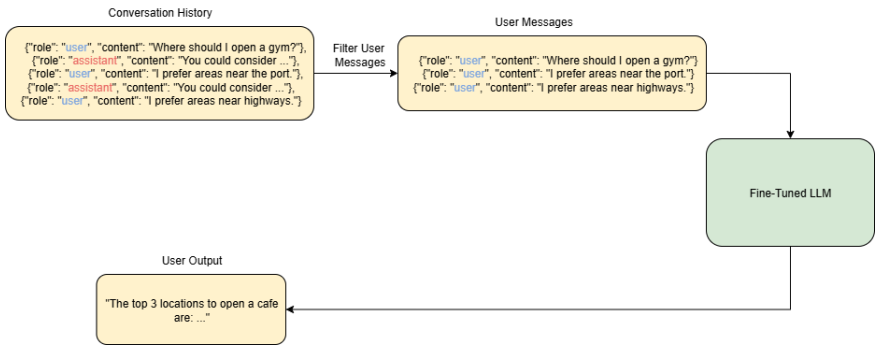


Figure 4.14: Backend Fine-Tuned LLM Pipeline

As described previously, when a user query is received, a request is made from the frontend to the backend, containing the complete conversation history. The fine-tuned LLM extracts only the user messages and concatenates them to create the complete input. We treat the first user message as the 'intent', since we assume that it contains the desired business type. The rest of them are treated as 'constraints', based on the assumption that the follow-up messages specify preferences of the user. By combining the intent and constraints we create a prompt that always contains the desired NACE class, and also is up to date with every user specification. Figure 4.14 demonstrates this workflow.

Retrieval-Augmented Generation Pipeline

The Retrieval-Augmented Generation pipeline is wrapped in the FastAPI endpoint '/generate_rag'. In the beginning of the Python script, we load the embedder model ('BAAI/bge-large-en-v1.5') with its tokenizer, and we fetch the pre-computed embeddings of the NACE and neighborhoods descriptions described in section 4.3.3. The process again mirrors the inference of the system.

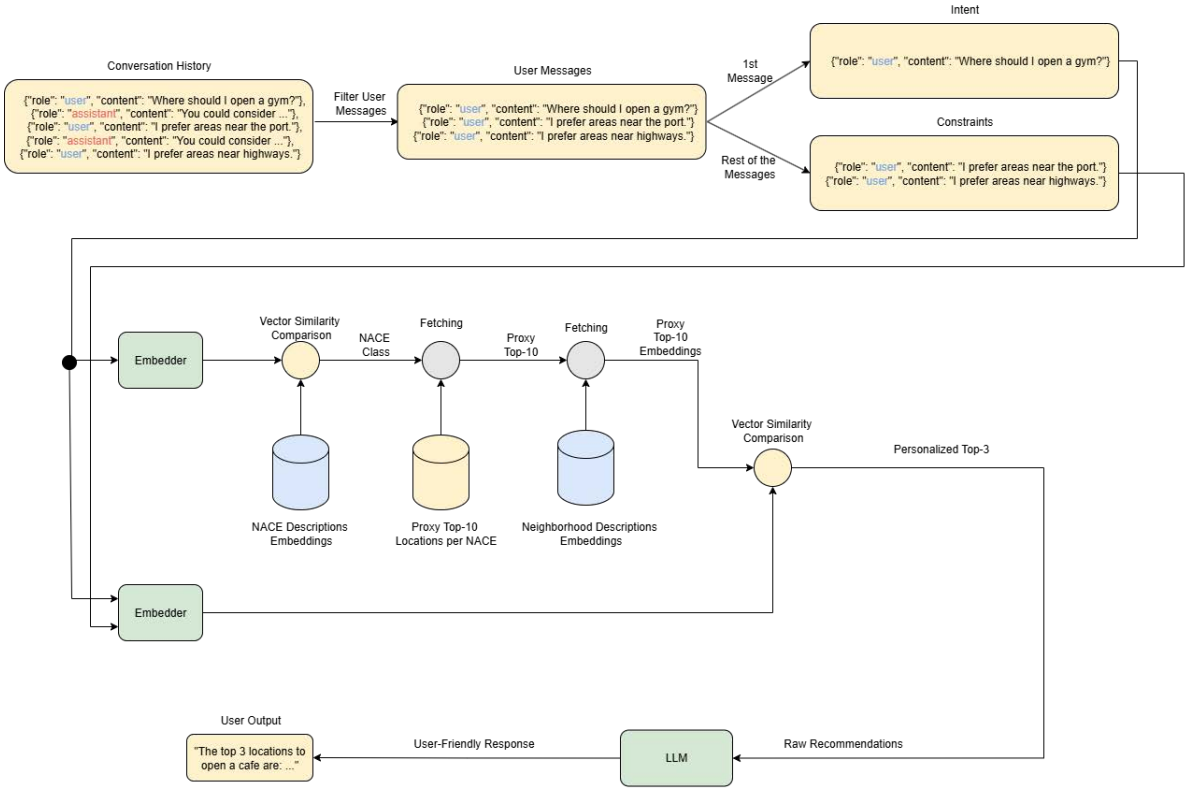


Figure 4.15: Backend RAG Pipeline

The RAG system also extracts only user messages. We use just the intent to specify the

NACE class, as the constraints should not affect the type of the business, only the actual recommendations. During the re-ranking, we embed the intent along with the constraints to fetch the final suggestions. Figure 4.15 shows the workflow visually.

4.4.3 Frontend-Backend Communication

The frontend communicates with the backend using synchronous HTTP POST requests to FastAPI endpoints. FastAPI provides a lightweight and JSON-friendly communication protocol that can easily be integrated with the Streamlit interface. Two main endpoints are available: '/generate', for deploying the fine-tuned LLM approach described in section 4.4.2, and '/generate_rag' for the RAG system pipeline explained in section 4.4.2. The request messages send a JSON payload that contains the complete conversation history, together with a fixed system prompt that guides the output of the selected system 4.16.

"You are a helpful assistant that suggests optimal business locations in Greece based on demographics and area characteristics."

Figure 4.16: System Prompt for LLM Guidance

The complete communication process can be summarized as follows: The frontend receives the user input, integrates it to the conversation, forwards the updated conversation history to the backend, which processes it, generates a response, and returns it. This approach ensures a clear and modular role for each side of the system. Figure 4.17 intuitively displays the communication workflow between the frontend and the backend.

4.4.4 Chatbot Demo

The previous sections described the process of transforming an LLM-based system, which simply receives a text input to a text output, to a chatbot application, with memory for the conversation history and a complete fontend-backend communication system. Below, we present an example of a conversation for every approach to demonstrate the results.

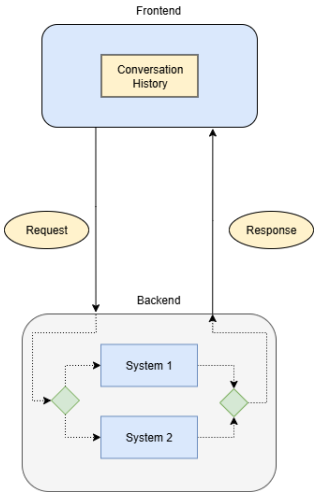


Figure 4.17: Frontent-Backend Communication

Fine-Tuned LLM Approach

Figure 4.18 shows an example of a conversation using the fine-tuned LLM. We see that the structure of the output resembles the training data, which is expected, since the LLM learns to follow this specific pattern. In this case, we see a mistaken prediction, since the constraint about areas close to the ports actually results in a number 1 option, whose port distance is larger than options 2 and 3.

RAG Pipeline Approach

Figure 4.19 shows an instance of the conversation that occurs by the RAG pipeline. In contrast with the fine-tuned LLM, this output follows a slightly different reasoning format because the generator is not trained on following the structure of the training data. This time the chatbot actually changes its prediction in the correct direction, as the port distance constraint produces Palaia as the first recommendation, which is closer to the port than the old recommendation.

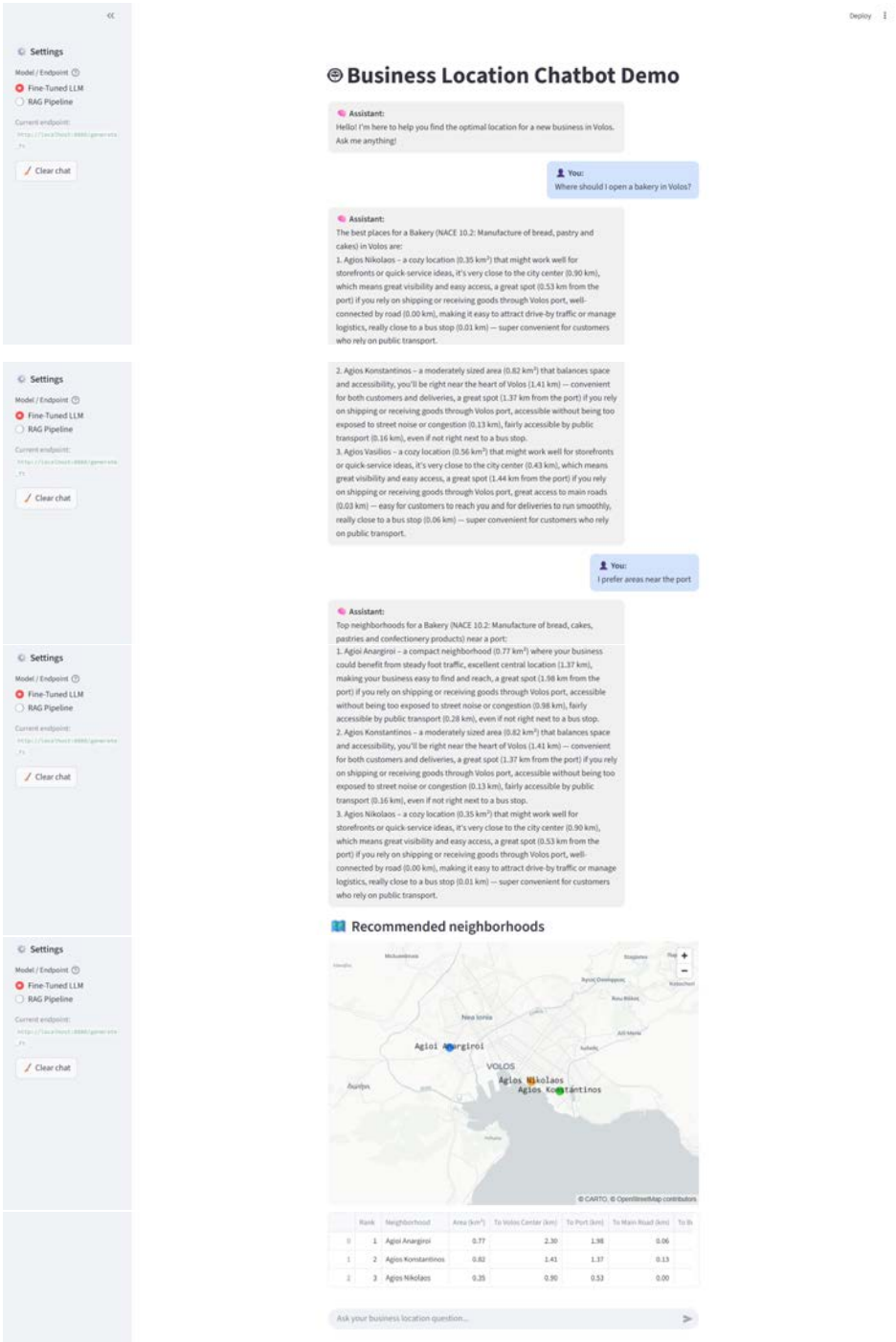


Figure 4.18: Chatbot Inference - Fine-Tuned LLM Approach

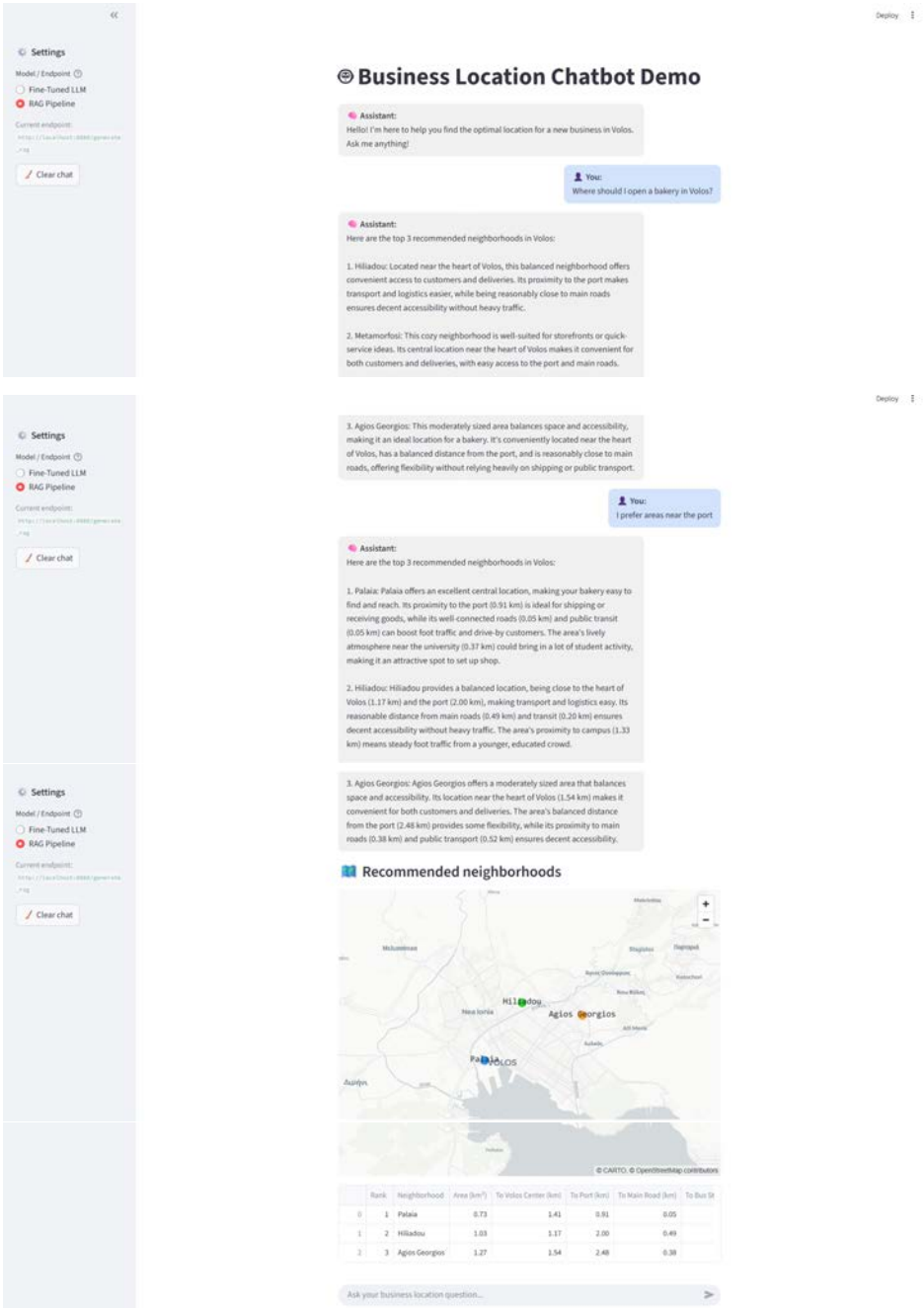


Figure 4.19: Chatbot Inference - RAG Pipeline Approach

Chapter 5

Experiments and Results

In this chapter, we describe the evaluation and comparison of the fine-tuned LLM and Retrieval-Augmented Generation systems. We begin by specifying the environment configurations, then proceed to mention the evaluation metrics, and finally calculate those metrics to evaluate and compare both approaches.

5.1 Experimental Setup

Here, we present details of the setup of the evaluation process. First and foremost, we aim to evaluate two approaches: the fine-tuned LLM, and the RAG system. We tested both frameworks on simple input-output pairs that aim to measure their quality as regular LLMs, not as chatbots. They were tested on the same dataset, specifically the 'inference dataset', constructed in section 4.2.3, which contains 159 entries Each entry consists of one prompt, which will be used as the input to each system, and its corresponding proxy truth output, which will be compared to the system prediction.

During evaluation, the fine-tuned LLM inference was performed on a remote server provided by the University of Thessal equipped with an NVIDIA A100 GPU with 40 GB of VRAM and 128 GB of RAM. This was performed on an Ubuntu environment using Python 3.10.12 and the Transformers library from Hugging Face. The RAG experiments were carried out locally (HP OMEN Laptop 15-en0xxx) on Windows 11 operating system, equipped with an AMD Ryzen 7 4800H CPU 16 GB RAM, without the use of a dedicated GPU. The environment contained Python 3.9.18, with the Ollama library employed as well.

5.2 Evaluation Metrics

To evaluate the performance of both approaches, several complementary accuracy metrics were utilized. Certain metrics examine semantic similarity, while others measure recommendation quality. In addition, we provide an efficiency metric that measures system inference speed.

5.2.1 Recommendation Metrics

To evaluate the quality of system recommendations, we apply metrics that measure how close they are to their corresponding proxy ground truths. These metrics are widely used in recommendation systems and information retrieval evaluation.

- **Match@3:** captures whether at least one of the three provided recommendations belongs in the proxy truth set, providing a measure of minimal success
- **Precision@3:** evaluates the proportion of the three predicted locations that are correct, emphasizing on the on the quality of the recommendations
- **Recall@3:** measures the proportion of the proxy ground truths that the top three recommendations manage to capture, emphasizing on the system's ability to cover the expected solutions

5.2.2 Semantic Metrics

In addition to evaluating the actual system recommendations, it is also important to measure the semantic similarity between the predicted completions and the proxy ground truths. This could evaluate the system justification, ensuring that different wording that results in similar context does not get penalized. To acquire these metrics, we used BERT-score, which generates embeddings from the pre-trained transformer model 'RoBERTa-large'. For every token of the predicted output, the most similar token of its corresponding reference is found, and the following three metrics are calculated:

- **Precision**: measures the proportion of the predicted output's tokens that have a semantically close match in its corresponding proxy truth

- **Recall**: measures the proportion of the proxy truth's tokens that are covered in the generated completion
- **F1**: the harmonic mean of precision and recall metrics, providing a balanced similarity score

5.2.3 Efficiency Metric

In addition to evaluating the quality of the recommendations, we also considered measuring the efficiency of each system. This metric demonstrates the computational trade-offs between the two approaches.

- **Inference Time per Query**: the average inference time of the system, measures practical system responsiveness

5.3 Results of Fine-Tuned LLM

For the fine-tuned LLM approach, I attempted multiple trainings that use a different number of epochs. Specifically, we used 3, 5, and 8 epochs. This was done to ensure that the training process was neither interrupted too soon nor led to model overfitting. Figure 5.1 demonstrates estimations of the metrics explained in section 5.2.3. The complete values of the evaluation metrics are displayed in table C.1.

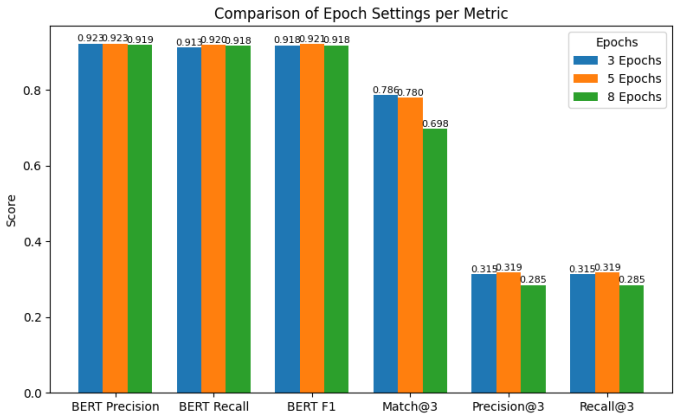


Figure 5.1: Fine-Tuned Models Comparison

As we see, after 8 epochs every metric decreases. This indicates that the model starts to overfit and should not be further trained on our data. The semantic similarity metrics are

generally higher than the recommendation metrics for every attempt. This implies that the justifications are satisfying and semantically close to the proxy truths across all the models. If we observe the BERT-F1 value, which acts as a harmonic mean of the other two metrics, it slightly peaks at 5 epochs. Regarding the recommendation metrics, we see that Match@k has higher values than Precision@3/Recall@3, which is expected since it is a less strict metric. Although the highest Match@k value occurs at 3 epochs, if we also look at Precision@3/Recall@3, the complete recommendation accuracy peaks at 5 epochs. This indicates that the model more often retrieves at least one correct community, but is less precise overall.

The overall trend shows that the metrics tend to increase from 3 to 5 epochs, but the performance decreases significantly at 8 epochs. This indicates that the model can be further trained on the fine-tuning data after 3 epochs, but overfits at 8 epochs. Overall, 5 epochs represent the optimal fine-tuning configuration among the tested attempts. This is the fine-tuned LLM we select to be integrated to the chatbot system.

5.4 Results of RAG System

The RAG pipeline is parametrized by certain components, as the embedder and generative LLM models used. In our case, we chose to modify the neighborhood descriptions to demonstrate the dynamic nature of the system. As described in section 4.3.2, we can specifically select the number of feature justifications included in the neighborhood description. We implemented three attempts, with 10, 15, and 20 justifications, respectively. Since the descriptions represent the system memory, modifying them also shows the ability of the framework to change its memory without the need for additional training. Figure 5.2 compares the metrics for the different attempts. The complete values of the evaluation metrics are displayed in table C.2.

We observe that the quality drops as the justifications, and therefore the length of neighborhood descriptions, increases. In our RAG pipeline, the accuracy of the system is determined by the accuracy of NACE class prediction, which in turn depends on the embedder model. This implies that, in general, the embedder performance worsens as the size of the descriptions increases. Mutiple factors could contribute to this tendency. Firstly, the encoder has a token limit, beyond which everything gets truncated. If important information is added to the end of the description, it can get cut off and never get embedded, unless we use chunk-

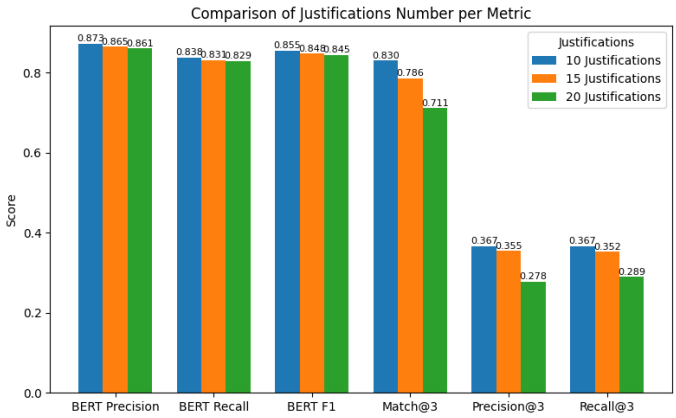


Figure 5.2: RAG Systems Comparison

ing, which we do not in our case. In addition, the description embedding is created by averaging the embeddings of its tokens. Adding irrelevant tokens results in weaker description embeddings. This suggests that the extra details are either not that relevant to the descriptions, or introduce semantic noise. In our chatbot, we will keep the descriptions that include 10 justifications.

5.5 Comparison of Approaches

To compare the two systems, we selected the best attempt to represent each approach. These are: the LLM fine-tuned on 5 epochs, and the RAG pipeline that uses neighborhood descriptions that include 10 justifications. In addition to the semantic and recommendation metrics demonstrated in the previous paragraph, we include the efficiency metric to highlight the computational trade-offs of the two approaches. Figure 5.3 demonstrates the semantic and recommendation metrics between the two systems. These values can also be found in table C.3. In addition, the efficiency metric is shown in Figure 5.4, while table C.4 contains the analytical time-related values.

Overall, BERT scores are higher for the fine-tuned system. In contrast, the recommendation metrics tend to favor the RAG pipeline. The efficiency metric also demonstrates the quicker inference of the fine-tuned model. Both approaches present significant strengths and limitations, which will be further analyzed below.

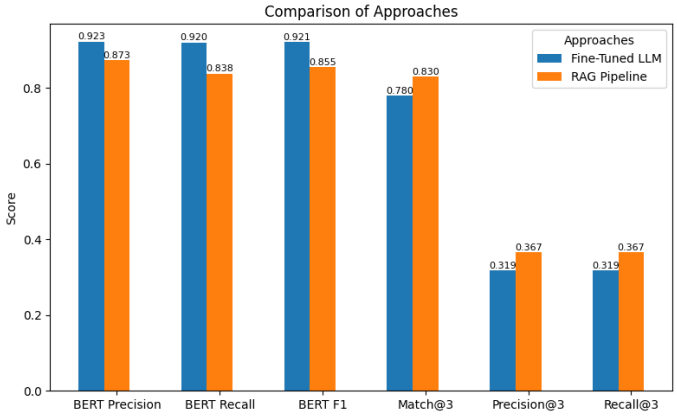


Figure 5.3: Comparison of Approaches - Semantic & Recommendation Metrics

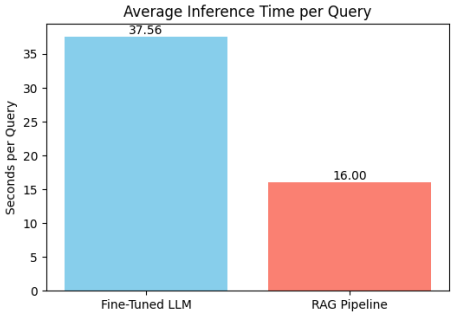


Figure 5.4: Comparison of Approaches - Efficiency Metric

5.6 Chatbot Performance

In the previous sections. we evaluated and compared the performance of our two LLM-based approaches. Although these results are insightful, the performance of the system is slightly affected when they are integrated into a chatbot application. In particular, we see that the RAG approach is actually faster during conversation.

This happens because the inference of the evaluation dataset used a batch size of 8, which utilized the available GPU much more efficiently than the current one-to-one conversation setting. This can also be attributed to the difference in input size between the two approaches. As explained in 4.4.2, the fine-tuned LLM input is the combination of all previous user messages, which increases in length as the conversation progresses. In contrast, the RAG input is structured upon a fixed template, regardless of the past messages 4.4.2. Because the response time of an LLM is proportional to the input size, the fine-tuned LLM carries a greater computational burden.

5.7 Conclusions

In the previous paragraph, we presented the results of the comparison of the two methods. Here, we interpret these metrics to extract general insights on the strengths and weaknesses of each system. We also mention limitations met throughout this thesis, and possible future directions that could enhance the project even more.

5.7.1 Summary of Key Findings

The result of the comparison showed that the semantic similarity is better captured by the fine-tuned LLM. This approach also provides faster inference, which is expected since it uses only one model and utilizes the GPU better by using a batch size of 8. In contrast, the RAG framework demonstrated greater adaptability, along with high accuracy in capturing recommendations. This happens because the accuracy of the RAG system depends entirely on the embedder performance, which provides the correct recommendations if it captures the NACE class successfully. The semantic metrics probably have worse values because RAG does not use the pure proxy justifications, but the reasoning is influenced by the generative LLM. Both approaches have distinct advantages depending on the deployment scenario. The dilemma between them demonstrates a clear trade-off between static efficiency and dynamic adaptability.

The chatbot application demonstrates a different outcome regarding the speed of the fine-tuned LLM approach. Since the structure of the conversation contains a single query, we cannot use batching and therefore utilize the GPU fully. On the other hand, the flexibility of the RAG pipeline allows the designer to select different generator models. In our case, we use Ollama, which optimizes the Llama3 LLM. Additionally, a chatbot application uses memory to provide context to follow-up replies. While the RAG system is based on a pre-defined template, the fine-tuned LLM integrates past user messages, resulting in a larger input, which burdens its performance.

5.7.2 Strengths and Limitations of each Approach

The core idea of the Fine-tuned LLM system is to train an LLM to directly generate business location recommendations with justifications. Since the framework relies on a single large model, predicting new outputs requires only one inference, making the process fast. In

addition, considering that the model is trained on domain-specific data, it learns patterns that allow it to generalize to unseen prompts. However, given that the knowledge of the model depends almost entirely on the fine-tuning data, keeping the model up to date (e.g. learning new NACE classes or updated proxy truths) requires re-training, which is a computationally expensive process. Furthermore, since the generated completions depend on the weights of the model, it is prone to hallucinations, by providing incorrect data in the justifications. Finally, given that the model learns static suggestions for every NACE class, it may lack in utilizing user preferences to provide personalised recommendations.

The RAG pipeline, on the other hand, uses embeddings to first extract the NACE class, and then retrieve relevant neighborhoods and re-rank them, based on their similarity to the user query. This method makes better use of the user specifications, and reduces hallucinations, as the generative LLM is fed the justifications in real time. Moreover, the flexible structure of this system allows us to simply update the neighborhood descriptions without the need to re-train. Nevertheless, this method presents a number of limitations. Its main disadvantage is its slower inference, due to the overhead of the real-time embedding of the user query and the retrieval process. It also requires additional infrastructure to store the descriptions and the embeddings, and its dependence on the embedder quality can harm the total system performance.

The integration of the approaches into a chatbot application introduced new parameters about their appropriateness. As the conversation structure follows a one-to-one message, a single-threaded application does not fully utilize the GPU resources. However, as the fine-tuned LLM inference demonstrated, concurrent queries could enable the full potential of the GPU parallel computing. This could correspond to a scenario of a real-life server that receives multiple queries simultaneously and utilizes a batch to process them in parallel.

Both systems provide significant and promising results, but we cannot definitively argue that one is superior to the other. Rather, each method is better suited for different conditions and circumstances. For example, startups that still determine the best recommendation per NACE class and require dynamic updates might prefer the RAG pipeline. If the most important concern is speed, or we desire to withstand concurrent queries or batched workloads, the fine-tuned LLM might be the best solution. Table C.5 describes the trade-offs between the approaches in detail.

Chapter 6

Conclusions and Discussion

In this chapter, we provide a comprehensive review of the results of this thesis and draw our final conclusions about the bigger picture. We also mention several future directions that can potentially enhance the application even more.

6.1 Summary of Key Findings

This thesis began by constructing the three main datasets of the project. This process involved merging heterogeneous data structures from multiple sources, extracting additional information from software applications, and combining and analyzing the final dataset, which joins business records with geographical, economic, and demographic characteristics. This dataset provided the basis for creating proxy ground truths to evaluate business location suitability.

Two state-of-the-art frameworks were introduced and developed. Firstly, a fine-tuned LLM was trained on a constructed domain-specific dataset. Next, a RAG pipeline that required the creation of descriptions of neighborhoods for the retrieval was implemented. The fine-tuning dataset and neighborhood descriptions provided justifications about the recommendations, created through the use of explainable AI methods. Both approaches were evaluated on the same data with the same metrics.

To transform the LLM-based systems into user-friendly frameworks that support personalization, a chatbot application that offers the option for both predictive systems was implemented. The two approaches acted as the application backend, which communicated with a user interface that stores and transfers the entire conversation history.

Overall, the two approaches demonstrated a trade-off between efficiency and accuracy. While the fine-tuned LLM can easily generalize to unseen data efficiently and fully utilize the GPU resources during concurrent requests, the RAG system provides better adaptability, lowers hallucination risk, and enables dynamic memory updating without the need for re-training. Each approach is therefore suited to different deployment contexts. The user can select the backend system depending on the nature of the application.

6.2 Limitations

In the course of this project, several challenges were faced. First and foremost, the recommendations per NACE class are approximations that were estimated by ensembling three weaker approaches. If actual ground truths were available, the results of the recommendation system would be much more reliable and valuable. In addition, although we believe that the dataset used represents the business tendencies of the city, it is relatively small (less than 4.000 entries). A larger dataset could potentially lead to different, more realistic conclusions. Our hardware constraints also limit the potential system accuracy. Moreover, larger and better models, either generative for fine-tuning, or embedding encoder for retrieval, could improve the performance of the systems.

6.3 Future directions

To improve the recommendation system, there are multiple potential enhancements that could be explored in the future. Acquiring ground truths would be a fundamental improvement in order to escape from estimating the optimal locations and provide realistic recommendations. More base models could be explored for the fine-tuned LLM approach, potentially with more parameters. The model quality is also largely influenced by the fine-tuning dataset, which can be enhanced if more business data is gathered. Using more synthetic categories per NACE class to create a training dataset with more entries may also lead to a more domain-specific model. The fine-tuning dataset could also be improved through the use of prompt engineering, as is the case for the creation of the neighborhood descriptions used in RAG. The RAG pipeline could be improved if we explored different embedder models. In addition, since current embeddings are stored locally in files, techniques like chunking and utilizing

vector databases could possibly allow the system to use larger neighborhood descriptions successfully. Finally, regarding the chatbot application, a database could be set up to store and retrieve past conversation sessions. In addition, the current structure uses one server for both approaches, which could be modified into multiple servers to make the application more modular and scalable.

6.4 Conclusions

This thesis focused on integrating spatial information such as economic, demographic, and geographical data with generative AI technologies like LLMs for optimal business recommendation. By comparing two cutting-edge algorithms -LLM fine-tuning and Retrieval-Augmented Generation- we identified complementary strengths and weaknesses that mirror potential use cases in real-world examples. We believe this thesis can serve as a foundation for further improving business recommendation by combining spatial information with state-of-the-art methodologies.

Bibliography

[1] Stella Manika, Konstantinos Karalidis, and Aspa Gospodini. Mechanism for the optimal location of a business as a lever for the development of the economic strength and resilience of a city. *Urban Science*, 5(4), 2021.

[2] Shuihua Han, Linlin Chen, Zhaopei Su, Shivam Gupta, and Uthayasankar Sivarajah. Identifying a good business location using prescriptive analytics: Restaurant location recommendation based on spatial data mining. *Journal of Business Research*, 179:114691, 2024.

[3] Nasim Tohidi and Rustam B. Rustamov. A review of the machine learning in gis for megacities application. In Rustam B. Rustamov, editor, *Geographic Information Systems in Geospatial Intelligence*, chapter 3. IntechOpen, Rijeka, 2020.

[4] Tamim Mahmud Al-Hasan, Aya Nabil Sayed, Faycal Bensaali, Yassine Himeur, Iraklis Varlamis, and George Dimitrakopoulos. From traditional recommender systems to gpt-based chatbots: A survey of recent developments and future directions. *Big Data and Cognitive Computing*, 8(4), 2024.

[5] Leeway Hertz. Build content-based recommendation for entertainment using llms.

[6] Afonso Oliveira, Nuno Fachada, and João Carvalho. Data science for geographic information systems. 04 2024.

[7] Andrew Rundle, Michael Bader, and Stephen Mooney. Machine learning approaches for measuring neighborhood environments in epidemiologic studies. *Current Epidemiology Reports*, 9:1–8, 06 2022.

[8] Carlos Granell, Paola Pesántez-Cabrera, Luis M. Vilches-Blázquez, Rosario Achig, Miguel Luaces, Alejandro Cortiñas, Carolina Chayle, and Villie Morocho. A scop-

ing review on the use, processing and fusion of geographic data in virtual assistants. *Transactions in GIS*, 25, 01 2021.

[9] Sultan Alamri, Kiki Adhinugraha, Nasser Allheeib, and David Taniar. Gis analysis of adequate accessibility to public transportation in metropolitan areas. *ISPRS International Journal of Geo-Information*, 12(5), 2023.

[10] Christos Kyriakos and Manolis Vavalis. Business intelligence through machine learning from satellite remote sensing data. *Future Internet*, 15(11), 2023.

[11] Pasupuleti Pavani, Sanjay Reddy Komatireddy, Venkata Teja Yarasuri, Vivek Reddy Police, Sunil Kumar Tanneru, and Hassan M. Al-Jawahry. Geographic information systems applications in business decision-making. In *Proceedings of the 2024 International Conference on Sustainable Materials, Environment and Engineering (ICSMEE'24)*, volume 529 of *E3S Web of Conferences*, page 04006. EDP Sciences, 2024.

[12] Qgis - spatial without compromise. `https://qgis.org/`. Accessed: 28 Aug. 2025.

[13] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

[14] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[16] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023.

[17] Nathan Rosidi. Semi-supervised learning: Techniques & examples. `https://www.stratascratch.com/blog/semi-supervised-learning-techniques-and-examples/`, oct 2024. Accessed: 2025-06-21 @ 17:50.

[18] Vinícius Trevisan. Using shap values to explain how your machine learning model works. `https://towardsdatascience.com/using-shap-values-to-explain-how-your-machine-learning-model-works-`

b5e0d0e1e3c0, jan 2022. Medium – Towards Data Science; 7 min read. Accessed: 2025-06-21.

[19] Eurostat. *NACE Rev. 2: Statistical classification of economic activities in the European Community*. Office for Official Publications of the European Communities, Luxembourg, 2008.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[21] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[22] Joe El Khoury and *GenAI Engineer*. Pre-training and fine-tuning large language models for recommender systems: An overview of techniques and models. https://medium.com/@jelkhoury880/pre-training-and-fine-tuning-large-language-models-for-recommender-systems-an-overview-of-b38d8b07c114, jul 2024. Medium blog post; 27 min read; Accessed: 2025-06-21 (if that's when you accessed it).

[23] Tejpal Kumawat. Retrieval-augmented generation (rag) from basics to advanced. https://medium.com/@tejpkumawat/retrieval-augmented-generation-rag-from-basics-to-advanced-9c915a5f4cbf, feb 2024. Medium blog post; 12 min read. Accessed: 2025-06-21.

[24] Sophie Hundertmark. Llm-chatbots — an introduction to the new world of bots, October 2024. Medium, accessed 2025-09-01.

[25] Foursquare platform. https://foursquare.com/. Accessed: 11 Aug. 2025.

[26] Xrysos odigos – online business directory of greece. https://www.xo.gr/dir-loc-geo/Nomos%20Magnisias/Volos/. Accessed: 11 Aug. 2025.

[27] Greekcatalog.net - business professional directory. https://greekcatalog.net/loc/%CE%B2%CF%8C%CE%BB%CE%BF%CF%82/. Accessed: 11 Aug. 2025.

[28] Vacation renter - a travel accommodation aggregator. https://www.vacationrenter.com/. Accessed: 11 Aug. 2025.

[29] Elstat - the official national statistics service of greece. https://www.statistics.gr/. Accessed: 12 Aug. 2025.

[30] David Crowther. Qgis – voronoi polygons. https://ukcommunity.arkance.world/hc/en-us/articles/21549919137682-QGIS-Voronoi-Polygons. Accessed: 28 Aug. 2025.

[31] geopandas.sjoin. https://geopandas.org/en/stable/docs/reference/api/geopandas.sjoin.html#geopandas-sjoin. Accessed: 28 Aug. 2025.

APPENDICES

Appendix A

Datasets Structure & Analysis

A.1 Business Entries Dataset

This appendix contains information about the business dataset after unifying the entries from multiple sources, filling the missing values, and adding the NACE code:

A.1.1 Dataset Structure

This section provides the structure of the pure business dataset (before joining).

Table A.1: Business Dataset Columns (highlighted features are used for joining)

Column Name	**Description**
Name	Business name as listed in the source dataset
Category	General business category (e.g., restaurant, retail, service)
Latitude	Geographic latitude coordinate of the business location
Longitude	Geographic longitude coordinate of the business location
Address	Full street address of the business
Country	Country where the business is located
City	City where the business is located
Postal Code	Postal code of the business location
Rating	Average rating score given by customers, if available
Reviews	Number of customer reviews, if available
Source	Origin of the business entry

Continued on next page

Table A.1 (continued)

Column Name	Description
Description	Short textual description of the business
NACE Code	Official NACE classification code for the business activity
NACE Description (EN)	English description of the NACE code activity
Municipal Community	The municipal community of the business location
Neighborhood	The neighborhood of the business location

A.1.2 Dataset Analysis

Feature Values Distributions

Here we see the most frequent values of the business dataset features that were not explicitly mentioned in the paragraph 3.3.1.

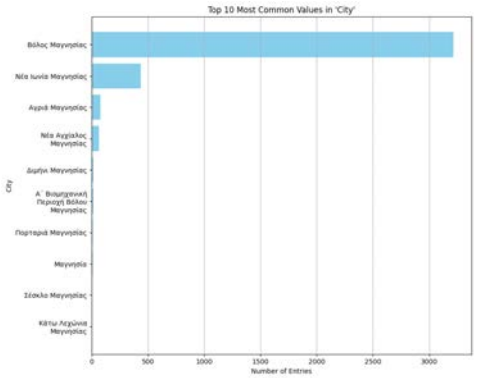


Figure A.1: 'City' Bar Chart

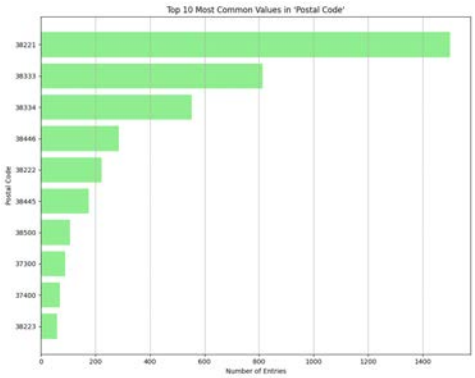


Figure A.2: 'Postal Code' Bar Chart

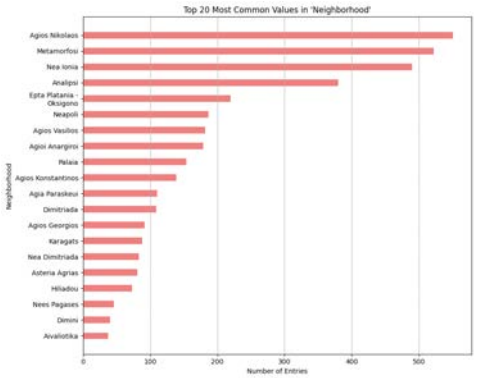


Figure A.3: 'Neighborhood' Bar Chart

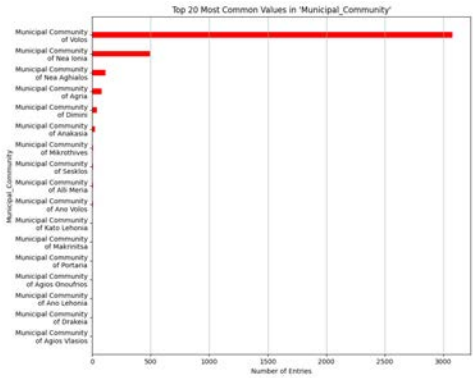


Figure A.4: 'MC' Bar Chart

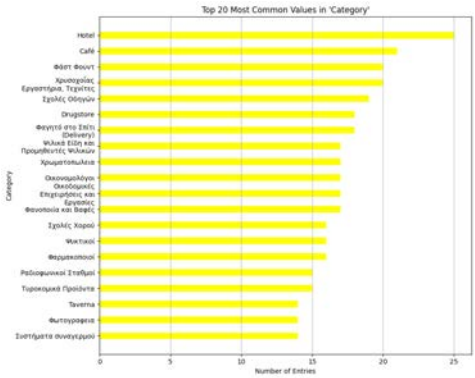


Figure A.5: 'Category' Bar Chart

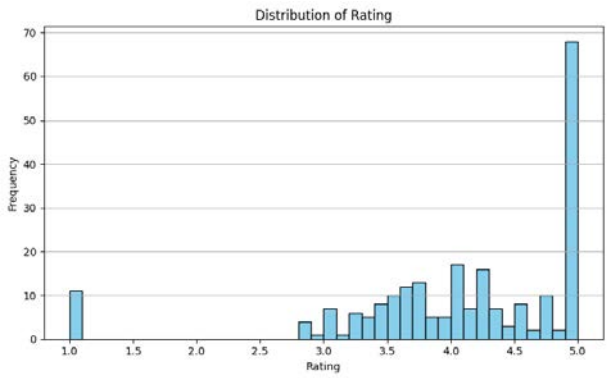


Figure A.6: 'Rating' Histogram

A.2 ELSTAT Datasets

This appendix contains information about the data extracted from the ELSTAT platform: their raw structure, visualizations about their analysis, and more.

A.2.1 Datasets Structure

Here we present the complete structure of the raw demographic and economic data extracted from the ELSTAT platform.

Raw Demographic Data

Table A.2: Demographic Dataset Columns

Column Name	Description
ΠΕΡΙΦΕΡΕΙΑΚΗ ΕΝΟΤΗΤΑ	Administrative regional unit where the municipal community belongs
ΔΗΜΟΣ	Municipality name
ΔΗΜΟΤΙΚΗ ΕΝΟΤΗΤΑ	Administrative municipal unit
ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ	Administrative municipal community name (Greek)
Population	Total population count
0-4	Population aged 0 to 4 years
5-9	Population aged 5 to 9 years
10-14	Population aged 10 to 14 years
15-19	Population aged 15 to 19 years

Continued on next page

Table A.2 (continued)

Column Name	Description
20-24	Population aged 20 to 24 years
25-29	Population aged 25 to 29 years
30-34	Population aged 30 to 34 years
35-39	Population aged 35 to 39 years
40-44	Population aged 40 to 44 years
45-49	Population aged 45 to 49 years
50-54	Population aged 50 to 54 years
55-59	Population aged 55 to 59 years
60-64	Population aged 60 to 64 years
65-69	Population aged 65 to 69 years
70-74	Population aged 70 to 74 years
75+	Population aged 75 years and older
higher_education	Individuals with higher education (university or equivalent)
post_secondary_non_uni	Individuals with post-secondary non-university education
high_school	Individuals with completed high school education
lower_secondary	Individuals with completed lower secondary education
primary_education	Individuals with completed primary education
no_formal_education	Individuals with no formal education
not_classified	Individuals whose education level is not classified
Άγαμοι	Individuals who are unmarried
Έγγαμοι	Individuals who are married
Χήροι	Individuals who are widowed
Διαζευγμένοι	Individuals who are divorced

Raw Economic Data

Table A.3: Economic Dataset Columns

Column Name	**Description**
ΠΕΡΙΦΕΡΕΙΑΚΗ ΕΝΟΤΗΤΑ	Administrative regional unit where the municipal community belongs
ΔΗΜΟΣ	Municipality name
ΔΗΜΟΤΙΚΗ ΕΝΟΤΗΤΑ	Administrative municipal unit
ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ	Administrative municipal community name (Greek)
Σύνολο	Total reference population for the economic activity statistics
Οικονομικά ενεργοί	Economically active population (employed + unemployed individuals)
Απασχολούμενοι	Employed individuals within the economically active population
Πρωτογενής	Number of employed individuals in the primary economic sector
Δευτερογενής	Number of employed individuals in the secondary economic sector
Τριτογενής	Number of employed individuals in the tertiary economic sector
Άνεργοι	Unemployed individuals within the economically active population
Οικονομικά μη ενεργοί	Economically inactive population (students, retirees, homemakers, etc.)

Processed Demographic & Economic Data

Table A.4: Processed Demographic & Economic Dataset Columns

Column Name	Description
ΠΕΡΙΦΕΡΕΙΑΚΗ ΕΝΟΤΗΤΑ	Administrative regional unit where the municipal community belongs
ΔΗΜΟΣ	Municipality name
ΔΗΜΟΤΙΚΗ ΕΝΟΤΗΤΑ	Administrative municipal unit
ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ ΕΛΛ	Municipal community name in Greek
ΔΗΜΟΤΙΚΗ ΚΟΙΝΟΤΗΤΑ	Municipal community name in English
Population	Total population count
Άγαμοι_pct	Percentage of unmarried individuals
Έγγαμοι_pct	Percentage of married individuals
Χήροι_pct	Percentage of widowed individuals
Διαζευγμένοι_pct	Percentage of divorced individuals
age_0_14_pct	Percentage of population aged 0–14 years
age_15_64_pct	Percentage of population aged 15–64 years
age_65_plus_pct	Percentage of population aged 65 years and older
low_education_pct	Percentage of population with no formal, primary, or lower secondary education
medium_education_pct	Percentage of population with upper secondary or post-secondary non-university education
high_education_pct	Percentage of population with higher education (university or equivalent)
unemployment_rate	Ratio of unemployed individuals to economically active population
labor_force_participation_rate	Ratio of economically active population to total population
primary_sector_pct	Percentage of employed individuals in the primary sector
secondary_sector_pct	Percentage of employed individuals in the secondary sector
tertiary_sector_pct	Percentage of employed individuals in the tertiary sector

Continued on next page

Table A.4 (continued)

Column Name	Description
Area_km2	Area of the municipal community in square kilometers

A.2.2 Datasets Analysis

Here, we present the visualizations provided by analyzing the demographic and economic data gathered from the ELSTAT platform. The results of these inspections led to the chosen transformation methods applied in order to prepare the data for later use.

Feature Values Distributions

Below, we see the most frequent values of the ELSTAT dataset features that were not explicitly mentioned in the paragraph 3.3.2.

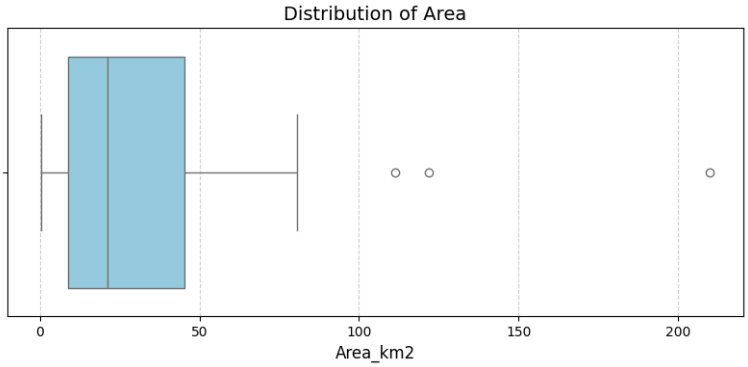


Figure A.7: ’Area_km2’ Feature Boxplot

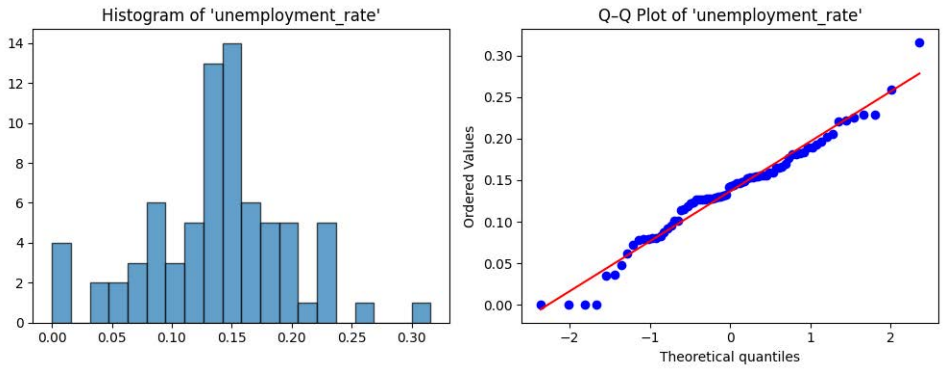


Figure A.8: ’unemployment_rate’ Feature Histogram & QQ-Plot

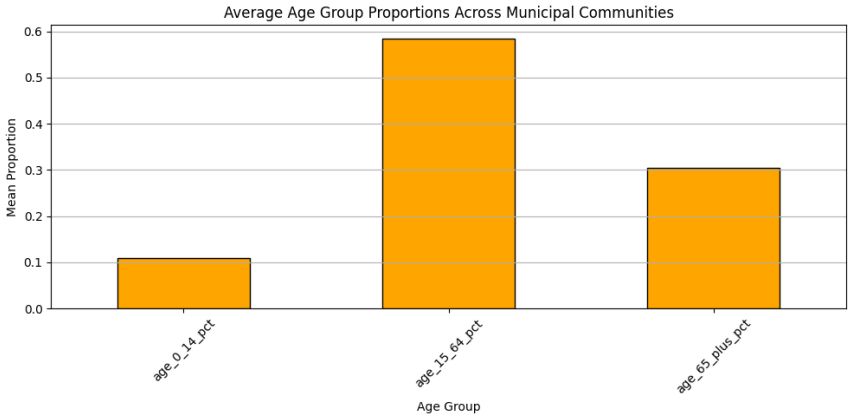


Figure A.9: Mean Age Group Features Bar Chart

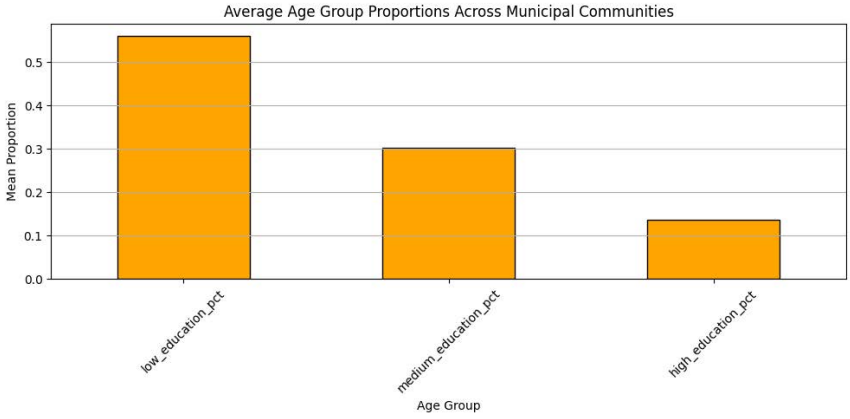


Figure A.10: Mean Education Group Features Bar Chart

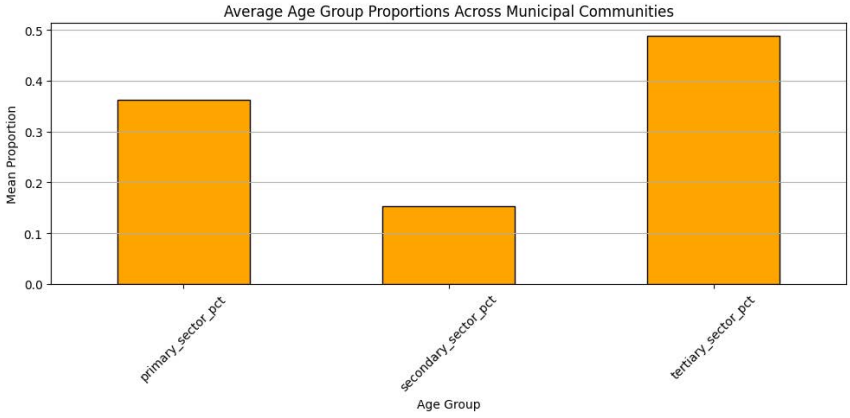


Figure A.11: Mean Sector Group Features Bar Chart

Scatter Plots

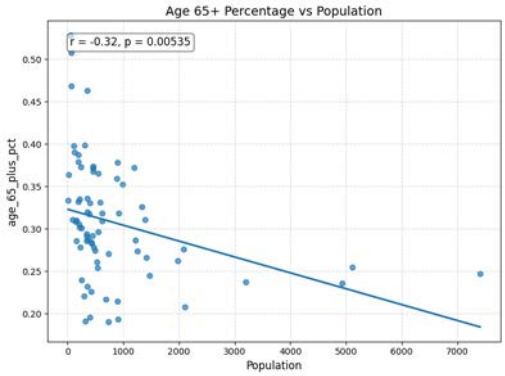


Figure A.12: Population VS Age 65+

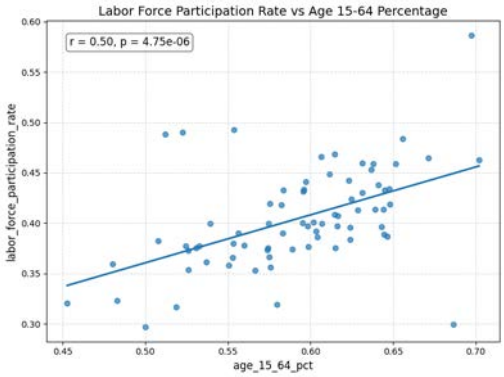


Figure A.13: Labor Force VS Age 15-64

Correlation Matrices

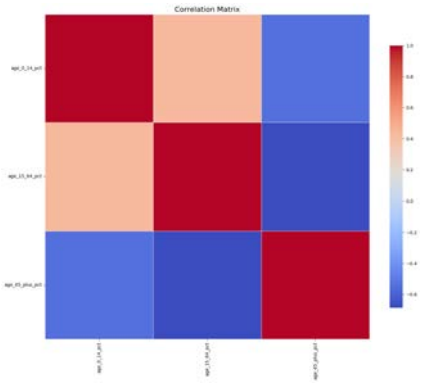


Figure A.14: Age Groups

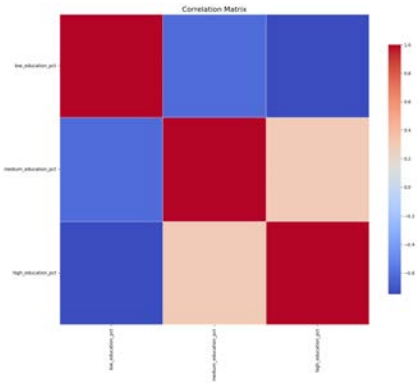


Figure A.15: Educational Level Groups

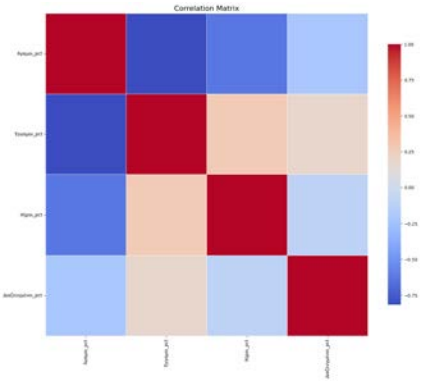


Figure A.16: Martial Status Groups

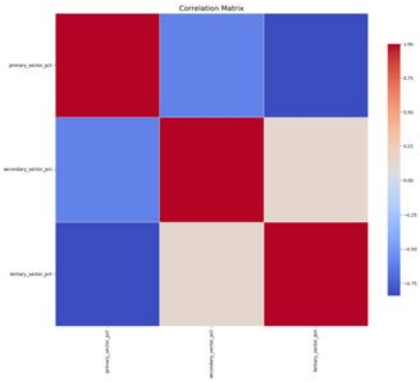


Figure A.17: Sector Groups

Feature Engineering

Below is presented the specific modifications performed in the feature groups in order to reduce the dimensionality:

- Age Groups:
  - $age\_0\_14\_pct = (0-4)_{\%} + (5-9)_{\%} + (10-14)_{\%}$ (younger people)
  - $age\_15\_64\_pct = (15-19)_{\%} + (20-24)_{\%} + (25-29)_{\%} + (30-34)_{\%} + (35-39)_{\%} + (40-44)_{\%} + (45-49)_{\%} + (50-54)_{\%} + (55-59)_{\%} + (60-64)_{\%}$ (working people)
  - $age\_65\_plus\_pct = (65-69)_{\%} + (70-74)_{\%} + (75+)_{\%}$ (elderly people)
- Educational Level Groups:
  - $low\_education\_pct = no\_formal\_education_{\%} + primary\_education_{\%} + lower\_secondary_{\%}$
  - $medium\_education\_pct = high\_school_{\%} + post\_secondary\_non\_uni_{\%}$
  - $high\_education\_pct = higher\_education_{\%}$
- Economic Hierarchical Features:
  - $unemployment\_rate = \frac{\text{Άνεργοι}}{\text{Οικονομικά ενεργοί}}$
  - $labor\_force\_participation\_rate = \frac{\text{Οικονομικά ενεργοί}}{\text{Σύνολο}}$
  - $primary\_sector\_pct = \frac{\text{Πρωτογενής}}{\text{Απασχολούμενοι}}$
  - $secondary\_sector\_pct = \frac{\text{Δευτερογενής}}{\text{Απασχολούμενοι}}$
  - $tertiary\_sector\_pct = \frac{\text{Τριτογενής}}{\text{Απασχολούμενοι}}$

A.3 Neighborhoods Dataset

This section provides information about the structure of the Neighborhoods Dataset after extracting additional features from QGIS:

A.3.1 Datasets Structure

Here we present the structure of the neighborhood dataset.

Table A.5: Neighborhood Dataset Columns (highlighted features are used for joining)

Column Name	Description
Neighborhood_Greek	Neighborhood name in Greek
Neighborhood	Neighborhood name in English
Centroid_x	Longitude coordinate of the neighborhood centroid
Centroid_y	Latitude coordinate of the neighborhood centroid
Neighborhood_Area_km2	Total area of the neighborhood polygon in square kilometers
Geometry	Polygon geometry representing the neighborhood boundary
distance_to_volos_center_km	Distance from the neighborhood centroid to the Volos city center (km)
distance_to_volos_port_km	Distance from the neighborhood centroid to the Volos port (km)
dist_to_main_road_km	Distance from the neighborhood centroid to the nearest main road (km)
dist_to_bus_stop_km	Distance from the neighborhood centroid to the nearest bus stop (km)
dist_to_university_km	Distance from the neighborhood centroid to the University of Thessaly (km)
Municipal Community	Administrative municipal community in which the neighborhood is located

A.3.2 Dataset Analysis

Below we show various visualizations that occured during the EDA of the neighborhood dataset.

Feature Values Distributions

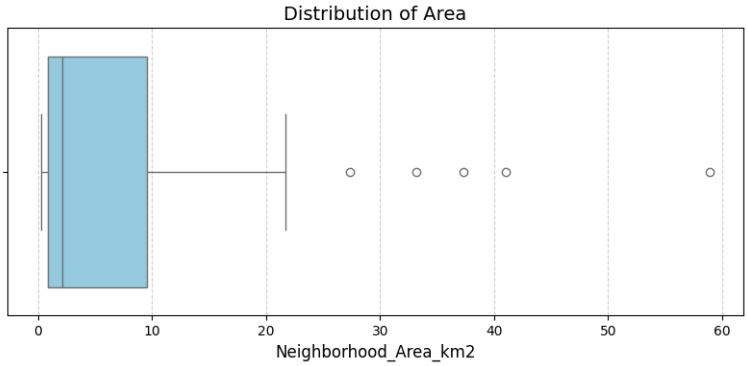
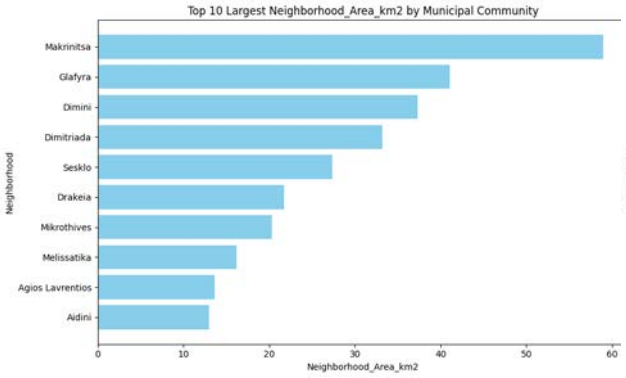
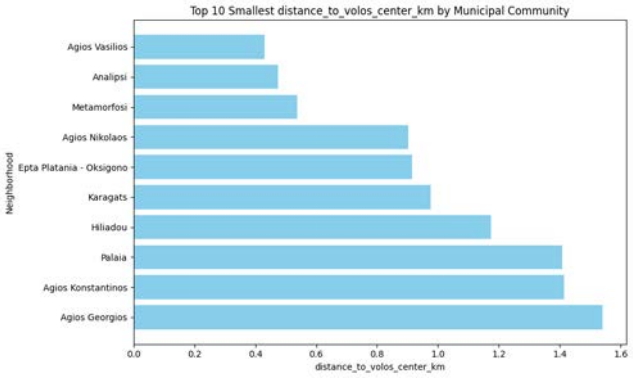


Figure A.18: 'Neighborhood_area_km2' Feature Boxplot

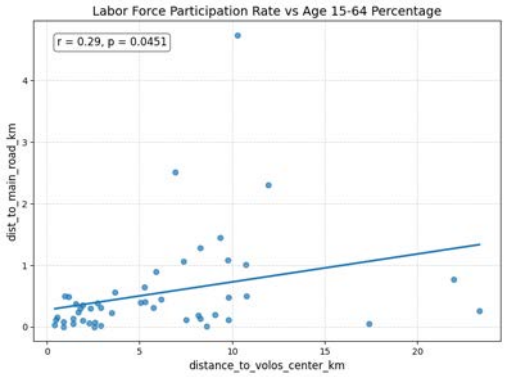


(α′) Neighborhood area Bar Chart

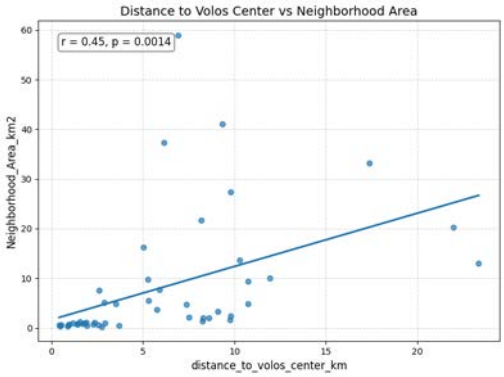


(β′) 'Distance to Center Bar Chart

Scatter Plots



(α′) Distance from Center VS Distance from Main Roads



(β′) Distance from Center VS Neighborhood Area

A.4 Joined Dataset

This appendix contains information about the business dataset after joining it both with the neighborhood and municipal community features.

A.4.1 Dataset Structure

This section provides the structure of the joined business dataset.

Table A.6: Joined Dataset Columns

Column Name	Description
Name	Business name as listed in the source dataset
Category	General business category (e.g., restaurant, retail, service)
Latitude	Geographic latitude coordinate of the business location
Longitude	Geographic longitude coordinate of the business location
Address	Full street address of the business
Country	Country where the business is located
City	City where the business is located
Postal Code	Postal code of the business location
Rating	Average rating score given by customers, if available
Reviews	Number of customer reviews, if available
Source	Origin of the business entry
Description	Short textual description of the business
NACE Code	Official NACE classification code for the business activity
NACE Description (EN)	English description of the NACE code activity
Municipal Community	The municipal community of the business location
Neighborhood	The neighborhood of the business location
ΠΕΡΙΦΕΡΕΙΑΚΗ ΕΝΟΤΗΤΑ	Greek regional unit where the business is located
ΔΗΜΟΣ	Municipality where the business is located
ΔΗΜΟΤΙΚΗ ΕΝΟΤΗΤΑ	Municipal unit of the business location
Population	Total population of the municipal community
Άγαμοι_pct	Percentage of unmarried individuals in the population

Continued on next page

Table A.6 (continued)

Column Name	Description
Έγγαμοι_pct	Percentage of married individuals in the population
Χήροι_pct	Percentage of widowed individuals in the population
Διαζευγμένοι_pct	Percentage of divorced individuals in the population
age_0_14_pct	Percentage of population aged 0–14
age_15_64_pct	Percentage of population aged 15–64
age_65_plus_pct	Percentage of population aged 65 and older
low_education_pct	Percentage of population with low education level
medium_education_pct	Percentage of population with medium education level
high_education_pct	Percentage of population with high education level
unemployment_rate	Unemployment rate in the municipal community
labor_force_participation_rate	Labor force participation rate in the municipal community
primary_sector_pct	Employment percentage in the primary sector
secondary_sector_pct	Employment percentage in the secondary sector
tertiary_sector_pct	Employment percentage in the tertiary sector
Area_km2	Total area of the municipal community in square kilometers
Neighborhood_Area_km2	Total area of the neighborhood in square kilometers
distance_to_volos_center_km	Distance from the neighborhood center to Volos city center (km)
distance_to_volos_port_km	Distance from the neighborhood center to Volos port (km)
dist_to_main_road_km	Distance from the neighborhood center to the nearest main road (km)
dist_to_bus_stop_km	Distance from the neighborhood center to the nearest bus stop (km)
dist_to_university_km	Distance from the neighborhood center to the nearest university (km)

A.4.2 Missing Data Analysis

Here we present a heatmap of the missing values across all the features of the joined dataset.

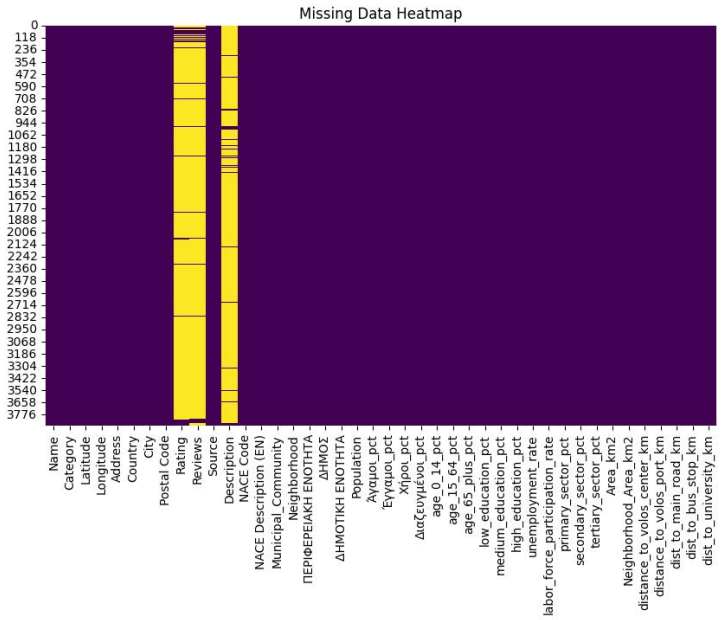


Figure A.21: Missing Data Heatmap

A.5 Proxy Ground Truths

This appendix contains information about the procedure of creating the proxy ground truths. Below we present some indicative Venn diagrams of the agreement count between the three different proxy truth creation methods. Not every single one will be shown since their total number is 306 (1 per NACE class).

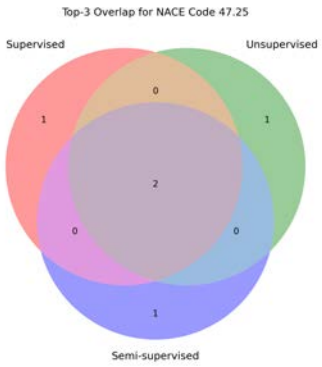


Figure A.22: Venn Diagram for NACE Class 47.25 (Partial Similarity)

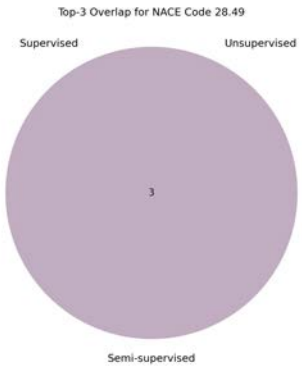


Figure A.23: Venn Diagram for NACE Class 28.49 (Perfect Similarity)

Appendix B

Fine-tuned LLM and Chatbot

This appendix contains information about the workflow of fine-tuning an LLM to specialize in suggesting optimal business locations, and its transformation to a chatobt application.

B.1 Fixed Features of Fine-Tuning Dataset

During the fine-tuning dataset construction, a proportion of the following features will always be selected as justification, regardless of their SHAP importance.

Fixed Features
”Neighborhood_Area_km2”
”distance_to_volos_center_km”
”distance_to_volos_port_km”
”dist_to_main_road_km”
”dist_to_bus_stop_km”
”dist_to_university_km”
”Population”

Table B.1: Fixed Fine-Tuning Features

B.2 Training Configuration

Below, we present the detailed description of the configuration parameters chosen to perform the LLM fine-tuning.

Table B.2: Fine-tuning configuration of the QLoRA setup.

Component	**Configuration**
Quantization (bitsandbytes)	
Weight precision	4-bit (NF4)
Compute dtype	FP16
Double quantization	Enabled
Base model	`mistralai/Mistral-7B-Instruct-v0.2`
LoRA adapters	
Rank (r)	16
Scaling factor (α)	64
Dropout	0.05
Bias	None
Task type	Causal LM
Trainable parameters	$\sim$6.8M ($\approx$0.18% of total)
Training arguments	
Batch size (per device)	2
Gradient accumulation	4 (effective batch size = 8)
Learning rate	2e-5
Optimizer	Paged AdamW (32-bit)
LR scheduler	Cosine
Warmup steps	10
Epochs	3 (with further runs at 5 and 8)
Precision	bf16
Logging steps	25
Checkpointing	Save every 100 steps (keep max 2)
Gradient checkpointing	Enabled

B.3 Custom CSS & HTML Messages

Below, we demonstrate the custom CSS/HTML templates used for the user and assistant messages.

Listing B.1: Rendering User Messages

```
def render_user_message(message):
    return f"""
    <div style="display: flex; justify-content: flex-end;">
        <div style="background-color:#D2E3FC; border-radius:10px;
           padding:10px 15px; margin:8px 0; max-width:80%; text-
           align:left;">
            <b>You:</b><br>{message}
        </div>
    </div>
    """
```

Listing B.2: Rendering Assistant Messages

```
def render_assistant_message(message):
    return f"""
    <div style="background-color:#F1F0F0; border-radius:10px;
       padding:10px 15px; margin:8px 0; width:fit-content; max-
       width:80%;">
        <b>Assistant:</b><br>{message}
    </div>
    """
```

Appendix C

Evaluation of Approaches

This appendix contains information about the evaluation of the two approaches. We show analytically the calculated metric values.

C.1 Fine-Tuned LLM Results

The following table demonstrates the calculated values of the evaluation metrics for the fine-tuned LLM approach.

Table C.1: Evaluation metrics across different training epochs

Epochs	BERT Precision	BERT Recall	BERT F1	Match@3	Precision@3	Recall@3
3 Epochs	0.9229	0.9133	0.9181	0.7862	0.3145	0.3145
5 Epochs	0.9228	0.9202	0.9215	0.7799	0.3187	0.3187
8 Epochs	0.9188	0.9179	0.9183	0.6981	0.2851	0.2851

C.2 RAG Pipeline Results

The following table demonstrates the calculated values of the evaluation metrics for the RAG pipeline approach.

Table C.2: Evaluation metrics across different approaches

Attempts	BERT Precision	BERT Recall	BERT F1	Match@3	Precision@3	Recall@3
10 Justifications	0.8727	0.8381	0.8550	0.8302	0.3669	0.3669
15 Justifications	0.8648	0.8314	0.8477	0.7862	0.3548	0.3522
20 Justifications	0.8608	0.8290	0.8446	0.7107	0.2778	0.2893

C.3 Approaches Comparison

The following table demonstrates the calculated values of the evaluation metrics for the fine-tuned LLM and RAG pipeline approaches. The best attempt of each system is selected.

C.3.1 Semantic & Recommendation Metrics

Below we compare the metrics that measure semantic similarity and recommendation accuracy.

Table C.3: Semantic & Recommendation Metrics across both Approaches

Approach	BERT Precision	BERT Recall	BERT F1	Match@3	Precision@3	Recall@3
Fine-Tuned LLM	0.9228	0.9202	0.9215	0.7799	0.3187	0.3187
RAG Pipeline	0.8727	0.8381	0.8550	0.8302	0.3669	0.3669

C.3.2 Efficiency Metrics

Below, we compare the metrics that measure the runtime duration of each approach.

Table C.4: Efficiency Metrics across both Approaches

Approach	Average Time per Query (s)	Total Runtime (mm:ss)
Fine-Tuned LLM	16.00	12:31
RAG Pipeline	37.56	42:24

C.3.3 Strengths and Limitations

Below, we provide a table that analytically describes the advantages and disadvantages of the two frameworks.

Table C.5: Fine-tuned LLM vs. RAG with Re-Ranking

Aspect	Fine-tuned LLM	RAG with Re-Ranking Pipeline
Core Idea	Train an LLM (Mistral-7B-Instruct with QLoRA) to directly generate top-3 business location recommendations with justifications.	Use embeddings to retrieve relevant neighborhoods and re-rank based on similarity to the user query; LLM (Llama 3) only formats the final output.
Strengths	• Fast inference after training (if GPU is utilized) • Learns domain-specific patterns • Can generalize to unseen but similar prompts	• Dynamic: no need to retrain when data updates • Reduces hallucinations (data fetched at runtime) • Easier to personalize results to user preferences
Limitations	• Static: retraining required for new NACE classes or updated proxy truths • Risk of hallucination in justifications • High initial computational cost • One-to-one conversation does not fully utilize the GPU	• Slower inference (retrieval + embedding overhead) • Dependent on embedding quality • Requires additional infrastructure (vector DB / storage)
Explainability	Built-in through feature & neighborhood justification (SHAP + percentile templates).	Explainability derived from neighborhood descriptions (feature statistics & contextual text).

Continued on next page

Aspect	Fine-tuned LLM	RAG with Re-Ranking Pipeline
Adaptability	Less flexible – retraining needed when data or categories change.	Highly flexible – can add new businesses, neighborhoods, or user constraints without retraining.
Computational Resources	Requires HPC (A100 GPU) for fine-tuning; optimized via QLoRA (0.18% trainable params).	Moderate GPU/CPU needed for embeddings and inference; feasible to run locally (quantized models).
Personalization	Limited – produces 3 static recommendations per NACE class.	Strong – query-specific re-ranking allows user-tailored recommendations.
Best Use Case	When fast inference on fixed, stable data with concurrent queries or batched workloads is needed, and compute resources are available.	When frequent updates, user-specific preferences, and reduced hallucinations are priorities.