Technical University of Crete
School of Electrical and Computer Engineering

Diploma Thesis

Optimization of Enterprise Workflows through Automated Information Extraction from PDF Files using Large Language Models

Evangelos Athanasakis

Thesis Committee

Prof. Michail G. Lagoudakis (Supervisor)
Prof. Thrasyvoulos Spyropoulos
Dr. Vasileios Diakoloukas

Chania, October 2025

Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Βελτιστοποίηση Επιχειρησιακών Ροών Εργασίας μέσω Αυτοματοποιημένης Εξαγωγής Πληροφορίας από Αρχεία PDF με χρήση Μεγάλων Γλωσσικών Μοντέλων

Ευάγγελος Αθανασάκης

Εξεταστική Επιτροπή

Καθηγητής Μιχαήλ Γ. Λαγουδάκης (Επιβλέπων)
Καθηγητής Θρασύβουλος Σπυρόπουλος
Δρ. Βασίλειος Διακολουκάς

Χανιά, Οκτώβριος 2025

Abstract

The large volumes of files circulated in today's enterprise workflows have prompted the development of methods based on Artificial Intelligence (AI) techniques for automated information extraction, retrieval, and summarization. In this diploma thesis, methods for processing and extracting data from semi-structured Portable Document Format (PDF) documents are studied and implemented using Large Language Models (LLMs). The project is divided into two distinct parts. In the first part, the study focuses on information retrieval from Greek soil analyses, which are characterized by their heterogeneous structure and formatting. Various text extraction techniques are examined, both from natively digital and scanned documents, using Optical Character Recognition (OCR). The contribution of individual sub-modules in the processing pipeline, such as post-processing for text extraction error correction and translation from Greek to English, is then investigated to the accuracy and efficiency of the overall structure. Various information retrieval techniques are then compared, including the full-context prompting approach and Retrieval-Augmented Generation (RAG), with the goal of evaluating the efficiency of each processing flow. In the second part, the methodology is generalized to be applicable to PDF documents from any domain. To this end, three agents are developed: The Field Detection Agent identifies candidate fields, the Post- Processing Agent filters and normalizes the results, and the Prompt Builder Agent dynamically constructs prompts for the information retrieval phase. Different architectures created by these agents are examined for extracting the names of fields that can be retrieved from the document. The efficiency and accuracy of the best information retrieval method from the first part is then re-evaluated, along with a variation of the full-context prompting approach. The proposed approach allows for automatic, adaptive, and efficient information extraction from a variety of texts. Overall, the thesis contributes to both the evaluation and improvement of different processing flows for data extraction from Greek soil analyses and the development of a general and scalable multi-agent architecture for any domain. The proposed framework can be applied to various fields, enhancing the automation and accuracy of information extraction from PDF files.

Περίληψη

Οι μεγάλοι όγκοι αρχείων που διακινούνται στις σύγχρονες επιχειρησιακές ροές έχουν ωθήσει την ανάπτυξη μεθόδων που βασίζονται σε τεχνικές Τεχνητής Νοημοσύνης για αυτοματοποιημένη εξαγωγή, ανάκτηση και σύνοψη πληροφοριών. Στην παρούσα διπλωματική εργασία μελετώνται και υλοποιούνται μέθοδοι επεξεργασίας και εξαγωγής δεδομένων από ημιδομημένα έγγραφα PDF με χρήση Μεγάλων Γλωσσικών Μοντέλων (Large Language Modelds - llm). Η εργασία αναπτύσσεται σε δύο διακριτά μέρη. Στο πρώτο μέρος, το πεδίο μελέτης επικεντρώνεται στην ανάκτηση πληροφοριών από αναλύσεις Ελληνικών εδαφών, οι οποίες χαρακτηρίζονται από ετερογένεια στη δομή και τη μορφοποίησή τους. Εξετάζονται διάφορες τεχνικές εξαγωγής κειμένου, τόσο από εγγενώς ψηφιακά, όσο και από σκαναρισμένα, έγγραφα με χρήση Οπτικής Αναγνώρισης Χαρακτήρων (Optical Character Recognition - OCR). Στην συνέχεια, εξετάζεται η συνεισφορά επιμέρους υπομονάδων της ροής επεξεργασίας, όπως post-processing για διόρθωση λαθών κατά την εξαγωγή του κειμένου και μετάφραση από Ελληνικά σε Αγγλικά, στην ακρίβεια και την αποδοτικότητα της συνολικής δομής. Στη συνέχεια, συγκρίνονται διάφορες τεχνικές ανάκτησης πληροφορίας, όπως η προσέγγιση πλήρων συμφραζομένων (full-context prompting) και η Ανάκτηση Υποβοηθούμενη από Γνώση (Retrieval-Augmented Generation – RAG), με στόχο την αξιολόγηση της αποδοτικότητας κάθε ροής επεξεργασίας. Στο δεύτερο μέρος, η μεθοδολογία γενικεύεται, ώστε να μπορεί να εφαρμοστεί σε έγγραφα PDF από κάθε πεδίο εφαρμογής. Για τον σκοπό αυτό αναπτύσσονται τρείς πράκτορες (agents): Ο Πράκτορας Ανίχνευσης Πεδίων εντοπίζει υποψήφια πεδία, ο Πράκτορας Μετα-επεξεργασίας φιλτράρει και κανονικοποιεί τα αποτελέσματα, ενώ ο Πράκτορας Δημιουργίας Prompts κατασκευάζει δυναμικά prompts για τη φάση ανάκτησης πληροφορίας. Εξετάζονται διαφορετικές αρχιτεκτονικές που δημιουργούνται από αυτούς τους πράκτορες για την εξαγωγή των ονομάτων των πεδίων που μπορούν να ανακτηθούν από το έγγραφο. Στην συνέχεια, αξιολογείται εκ νέου η αποδοτικότητα της καλύτερης μεθόδου ανάκτησης πληροφορίας που προέκυψε από το πρώτο μέρος, καθώς και παραλλαγές της προσέγγισης πλήρων συμφραζομένων. Η προτεινόμενη προσέγγιση επιτρέπει την αυτόματη, προσαρμοστική και αποδοτική εξαγωγή πληροφορίας από ποικίλα κείμενα προερχόμενα από διαφορετικούς τομείς. Συνολικά, η εργασία συμβάλλει τόσο στην αξιολόγηση και βελτίωση διαφορετικών ροών επεξεργασίας για την εξαγωγή δεδομένων από αναλύσεις Ελληνικών εδαφών, όσο και στην ανάπτυξη μίας γενικής και επεκτάσιμης σε κάθε τομέα, πολυπρακτορικής αρχιτεκτονικής. Η προτεινόμενη υποδομή μπορεί να εφαρμοστεί σε ποικίλα πεδία εφαρμογής, ενισχύοντας την αυτοματοποίηση και την ακρίβεια στην εξαγωγή πληροφοριών από αρχεία PDF.

Acknowledgments

I would like to express my sincere gratitude to my advisors,
Professor Michail G. Lagoudakis and **Dr. Vasileios Diakoloukas**,
for their guidance, support, and valuable feedback throughout this project. I would also like to thank **Professor Thrasyvoulos Spyropoulos**,
member of the thesis committee for his helpful comments and participation.

I am especially grateful to my family and close friends,
whose constant encouragement, patience, and belief in me,
provided the strength and motivation to reach this milestone.

Contents

List of Figures

List of Acronyms

CAG	Context-Aware Generation
LLM	Large Language Model
LLMs	Large Language Models
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
OCR	Optical Character Recognition
PDF	Portable Document Format
RAG	Retrieval-Augmented Generation
RNN	Recurrent Neural Network

1 Introduction

In recent years, the digitization of information across the majority of industries has led to an increasing demand for data handling and automation. The increase in the amount of data has made the retrieval process even more complex and time-consuming. One such domain is agronomy, where soil analysis reports often arrive in PDF form, with inconsistent formatting and unstructured data. The latter has made the retrieval and processing of these data really challenging. In this diploma thesis, a local pipeline powered by Large Language Models (LLMs) is presented, which automates the extraction, transformation, and retrieval of information from Greek soil analysis reports, using full-context prompting, as well as Retrieval-Augmented Generation (RAG) techniques.

Beyond the specialized use case of soil analysis, the methodology is further tested, generalized, and optimized in order to handle arbitrary, domain-agnostic, PDF documents of diverse structure and content. This generalization involves a multi-agent architecture, where dedicated agents identify extractable fields and construct prompts that are then used in the next retrieval step. In this way, the thesis not only addresses a concrete agronomic challenge, but also demonstrates a scalable framework for automated knowledge extraction and retrieval from unstructured PDF data across domains.

1.1 Motivation and Objectives

The primary motivation for this work arises from the significant challenges that occur when managing large volumes of information contained in unstructured PDF documents. Traditionally, data from these sources are manually parsed into structured systems such as databases and spreadsheets. This manual process is time-consuming, error-prone, and inefficient, especially when dealing with documents that vary widely in layout and terminology.

The key drivers for researching this field are:

- **Time savings:** Automating the processing and retrieval of documents significantly reduces the time spent on repetitive tasks, such as manual data entry, and focuses time resources in other more vital tasks.
- **Error reduction:** By minimizing human interaction during data transcription and retrieval, the risk of errors is drastically reduced.
- **Database integration:** Structured information can be directly injected into existing systems and databases. This results in fully automating the process of data extraction.

While the initial motivation originates from the agronomic domain, the same challenges appear across industries, whenever information is stored in PDF documents. For this reason, the objectives of the thesis extend further than soil analysis reports and aim to achieve a generalizable approach that can handle arbitrary PDF content. This is achieved through a multi-agent pipeline that detects and extracts fields of interest, filters redundant or unsuitable values, and constructs context-specific prompts for Retrieval-Augmented Generation. As such, the work contributes both to domain-specific applications, such as soil analysis reports, and to broader advances in automated document parsing.

1.2 Problem Description

The core challenge of this project lies in extracting information from unstructured, diverse PDF files. In the agronomic domain, soil analysis reports are often generated by different laboratories, with significant variations in their formatting, language, terminology, and measurement units. Similarly, domain-specific papers and other types of technical documents present heterogeneous structures and inconsistent styles, making uniform processing difficult. In many cases, these files exist only as scanned documents (images), which require Optical Character Recognition OCR as a first step in processing.

However, text extraction alone (either from native PDFs or OCR-processed scans) is insufficient. Further challenges arise in parsing that content into structured fields, such as pH, electrical conductivity, organic matter, or nutrient values in the case of soil analysis, or more general fields, like Issuer Name, Receipt Date, Invoice Number, etc., in other document types. Additionally, these extracted values must be mapped to standardized database fields, validated for consistency, and stored in a retrievable format.

Finally, the goal is not only to store this information, but to enable intelligent retrieval across various types of documents — whether in the form of soil reports, archived PDFs, invoices, or blood test results.

1.3 Existing Approaches

Several traditional techniques have been employed to address similar problems in structured data extraction and retrieval. Among the most common methods are:

- **Regular expressions and rule-based parsers:** These represent some of the earliest approaches to information extraction, relying on manually crafted patterns or predefined grammars to identify key-value pairs in text. Regular expressions (regex, for short) can achieve high precision, when the input documents include consistent and predictable formatting. Rule-based parsers extend this philosophy by incorporating structural or grammatical constraints, offering slightly more flexibility than regex alone. However, both methods have limitations: small variations in layout, OCR noise, spacing, or terminology can easily break the rules, and extensive manual engineering and maintenance are required to adapt them to new domains. As noted by Chiticariu et al. (2013) [1], while regex-based and rule-based systems retain value in specialized, high-precision scenarios, they are insufficient for large-scale, domain-agnostic information extraction, motivating the transition to more adaptable machine learning and language model-based approaches.

While useful in specific situations, these approaches do not scale to the full complexity of heterogeneous PDF documents. This limitation becomes particularly evident in domains, such as soil analysis, where reports in Greek often exhibit diverse and inconsistent formats, making traditional extraction techniques insufficient.

Machine Learning and Document Understanding Approaches

To overcome the structure and domain limitation that the traditional rule-based methods introduce, recent research has focused on machine learning architectures.

- **Table detection algorithms:** Many documents contain key information in tabular form, making table recognition a critical step in information extraction. Traditional methods rely on heuristics, such as line detection, whitespace analysis, or font alignment, but these approaches often fail on irregular or borderless tables. Schreiber et al. (2017) [2] proposed DeepDeSRT, a deep learning framework for table structure recognition in document images. Their model first detects table regions and then recognizes row and column structures using convolutional neural networks, significantly improving accuracy over heuristic methods. While powerful, deep learning-based table recognition requires annotated training data and can be sensitive to noisy OCR outputs, limiting its immediate applicability across diverse domains without adaptation.

- **LayoutLM Family:** The LayoutLM models represents one of the first steps in document understanding using machine learning by explicitly modeling both textual content and two-dimensional layout information for capturing table content. The original LayoutLM, introduced by Xu et al. in 2020 [3], proposed combining text alongside coordinates of bounding boxes in pretraining enabling the encoding of semantics and spatial relationships. The introduction of LayoutLMv2 (Xu et al., 2021 [4]) further extended the architecture with multimodal pretraining that fused text, layout, and image embeddings, allowing for more robust cross-modal interactions and higher accuracy in visually-rich document tasks. The latest version, LayoutLMv3 (Huang et al., 2022 [5]), further unified the training process by adopting joint text and image masking strategies within a single pretraining framework. This variant can also perform Question-Answering tasks. These models provide great accuracy in layout understanding and introduce question-answering tasks, but they require large annotated datasets for fine-tuning.

- **Donut:** Donut (Document Understanding Transformer), introduced by Kim et al. in 2022 [6], is an OCR-free document understanding transformer. It directly processes raw document images without the need of OCR as a pre-process step, reducing error propagation from early extraction and allowing Donut to handle multilingual and noisy scanned documents more effectively. Donut represented a shift towards fully end-to-end document understanding pipelines. Despite its benefits, this models still needs a vast amount of training data for pre-training and cannot be used in zero-shot scenarios.

- **DocFormer:** DocFormer, proposed by Appalaraju et al. in 2021 [7], introduced a multimodal transformer architecture for end-to-end document understanding. Unlike earlier models that incorporated visual and layout features in a limited way, DocFormer employs a multimodal self-attention mechanism that mixes text, two-dimensional layout coordinates, and image embeddings at every layer of the transformer. This results in better understanding of the relationships between simple text, tables and in general, structural components in a document. Even though it achieves state-of-the-art performance on several tests, this model requires a large training dataset for task-specific finetuning and it is computationally heavy.

- **Complex Data Extraction using Document Intelligence and RAG:** A recent solution, proposed by Microsoft, combines Document Intelligence (DI) with retrieval-augmented generation (RAG) to address complex data extraction from heterogeneous documents, such as tax forms [8]. In this approach, DI is employed

to capture document layout, tables, and style information, while semantic chunking and vector-based retrieval ensure that only relevant portions of the document are passed to a large language model. The LLM is then prompted with both textual and layout/style metadata, allowing it to infer structured values, even in documents with diverse templates and formatting. This framework demonstrates the benefits of enriching retrieval with layout-aware and stylistic cues, significantly improving accuracy in form-like documents. However, it should be noted that Microsoft's Document Intelligence is not an open-source software, but it is integrated into the Azure ecosystem. As a result, API calls necessary imply that local execution is not possible.

1.4 Innovation and Contribution

The solution proposed in this thesis combines recent advances in LLMs and retrieval-based architectures (like RAG and its variants) to build a domain-agnostic, layout-agnostic pipeline for extracting key information from input PDF files.

The key innovations of this work include:

- **Domain-agnostic extraction:** Unlike regex or template-based systems that are sensitive to the format and the domain of the PDF, the Large Language Model (LLM)-based retrieval pipeline that is proposed can normalize and extract values across a wide range of document formats and domains — not just soil reports.
- **Handling unstructured and scanned PDFs:** Through the use of OCR and intelligent post-processing (that fixes extraction mistakes), the system can handle not only digital PDFs, but also image-based reports.
- **Multi-step pipeline architecture:** The proposed system leverages multiple stages — including text extraction, LLM-based correction and translation (Greek to English), and embedding-based retrieval, among others, to ensure accuracy and scalability.
- **Zero-shot invocation:** The presented pipeline does not require fine-tuning on domain-specific data. It can be deployed and immediately executed in any domain without additional instructions.
- **Local execution:** The pipeline is designed to run entirely on local infrastructure without reliance on external cloud services. This ensures data privacy, reduces dependency on third-party providers, and allows deployment in environments with restricted or offline access.

These contributions address the shortcomings of existing approaches and aim to build a generalized, domain-agnostic extraction pipeline for input PDFs. The performance of the proposed system is evaluated across multiple dimensions, including execution time, extraction accuracy, translation quality, retrieval relevance, and generalization ability.

1.5 Thesis Outline

The thesis is structured into five discrete chapters:

- Chapter 2 contains all the theoretical background that is necessary for the description of the components that were tested. It includes both theory behind the technologies and available frameworks, as well as tools that deploy these technologies in real code.
- Chapter 3 describes all the components, such as text postprocessing, translation, retrieval methods, and agents, that were constructed and used in the experiments. It also presents the different architectures, in which these components were integrated.
- Chapter 4 presents all the experiments conducted in this thesis. It is divided into two parts. The first part focuses on foundational experiments with soil analysis reports, evaluating the impact of postprocessing, translation, and different retrieval methods. The second part extends the evaluation to domain-agnostic PDFs using a multi-agent architecture, in order to assess the generalizability of the proposed approaches.
- Chapter 5 presents a summary of the research conducted in this thesis, along with a discussion of the limitations of the proposed pipelines. It also highlights topics that indicate the need for further investigation and development in future research.

2 Theoretical Background

2.1 Soil Analysis Reports

Soil analysis is a fundamental process in agricultural and environmental science, providing essential information about the composition, fertility, and physicochemical properties of the soil. These reports are commonly used as indicators of the soil requirements and support in the proper nutrient replenishment, ensuring optimal conditions for crop growth and yield.

Soil analysis reports are typically provided in PDF format and include both structured and unstructured data, such as tables of chemical and physical properties, free-text comments, measurement units, and occasionally hand-written annotations or scanned pages. This variation in format introduces unique challenges for automated processing and digital understanding.

2.2 Challenges in Working with Soil Reports

Working with soil analysis reports involves several domain-specific and technical challenges. These include:

- **Language complexity**: Reports are written in Greek, often with domain-specific terminology and abbreviations that complicate translation and parsing.
- **Format inconsistency**: There is significant variability in structure from one report to another, as each laboratory follows a unique format. Some documents include clearly defined tables, while others may present data in semi-structured or narrative form. In many cases, a combination of both formats may be used.
- **Low-quality scans and OCR noise**: Reports are frequently digitized through scanning, resulting in image-based PDFs, where OCR practices need to be implemented. This introduces potential errors in character recognition, especially when dealing with tabular data or handwritten elements.
- **Multimodal content**: Reports may include a combination of text, tables, images, and graphical data, requiring flexible extraction techniques.
- **Unit and format variation**: Measurement values may be expressed in different units, number formats (e.g., decimal commas), or inconsistent column alignments.

These challenges highlight the need for a robust and flexible processing pipeline that can accurately extract, normalize, and interpret the contents of soil analysis documents in a scalable way.

2.3 PDF Text Extraction Techniques

Extracting text from PDF files is a non-trivial task, especially when dealing with scientific or semi-structured documents like soil analysis reports, invoices, blood test results, etc. The PDF format was designed primarily for accurate visual rendering, not for semantic clarity. As a result, document structure (e.g., tables, paragraphs, or headings) is not explicitly encoded, which presents significant challenges for automated processing.

A variety of text extraction libraries are available. While these tools have their strengths, most of them either fail to preserve the original layout or struggle with accurately identifying structured elements, like tables, and correctly extract their elements.

In this thesis, these libraries were selected based on their superior ability to preserve formatting and extract structured content as reliably as possible:

- **pdfPlumber** [9]: A powerful Python library designed for precise extraction of content from PDF files. It is especially effective at identifying and extracting content from tables.
- **unstructured.partition** [10]: A high-level library designed to segment documents into logical blocks.
- **PyMuPDF** (also known as `fitz`) [11] : A lightweight PDF and image processing library that enables word- and block-level text extraction while preserving layout.
- **Docling** [12]: A high-level wrapper around `pdfminer.six` focused on extracting semantically structured text with improved reading order and minimal layout distortion.

Although these tools perform well, further post-processing is often required due to inconsistencies in real-world documents. This includes:

- Reconstructing broken or multiline table rows.
- Merging fragmented or disordered text blocks.
- Aligning headers with the correct data values.
- Correcting cases where two or more words have been merged during extraction.

For that purpose, the output of the extraction libraries is then passed to a language model for further normalization and structuring. This multi-step pipeline maximizes the fidelity of the extracted information and ensures robust performance in subsequent tasks, such as translation and semantic retrieval.

2.4 Optical Character Recognition (OCR)

Many files are only available as scanned documents, saved as image-based PDFs without an embedded text layer. This makes direct text extraction impossible using standard tools (like the previously mentioned libraries). In such cases, Optical Character Recognition (OCR) is required to convert the visual representation of text into machine-readable form.

However, OCR support for Greek text is limited compared to more widely spoken languages, such as English. Only a few OCR libraries provide reliable recognition for Greek characters, which narrows the options available for high-quality extraction in this language.

For this purpose, the library **OCRmyPDF** [13] was used. Unlike traditional OCR tools that return raw text, OCRmyPDF enhances the original PDF by embedding the recognized text directly into the file as a hidden layer. This process effectively transforms image-only PDFs into *searchable PDFs*, enabling downstream tools to treat them as if they were originally digitally generated.

Internally, OCRmyPDF leverages the Tesseract OCR engine, performing the following operations:

- Firstly, for each page, it figures out the best **colorspace** and **resolution (DPI)** to ensure that, when it converts the page into an image, no information is lost. Colorspace is a way of organizing and representing colors using a set of numerical values (vector graphics). It is determined based on the pages content. For example, for an only black-and-white text, the monochrome colorspace will be selected, meanwhile, the presence of colorful photos and diagrams will lead to the selection of a full-color colorspace.
- It **rasterizes** each page of the input PDF. Rasterization takes the mathematical data from the vector graphic and converts it into a grid of colored pixels. This process transforms scalable vector elements, such as text and diagrams, into a fixed-resolution bitmap representation.
- **Tesseract** processes each rasterized page and detects text regions.
- The recognized text is stored as an **invisible layer** behind the original image.
- The resulting PDF maintains the original layout and appearance, but is now **searchable** and **selectable**, just like a native one.

After this transformation, the enhanced PDFs are passed to the same extraction pipeline described previously (i.e., using `pdfPlumber`). These tools can now operate as if the document was natively digital, allowing reliable extraction of structured content such as tables and narrative blocks.

This OCR-enhanced pre-processing step is crucial for ensuring consistency and robustness, when dealing with heterogeneous datasets that include both native and scanned documents.

2.5 Large Language Models (LLMs)

Large Language Models (LLMs) are a class of deep learning models trained on immense amounts of data, making them capable of understanding and generating natural language and other types of content. They have become foundational components in modern Natural Language Processing (NLP) tasks due to their ability to generate fluent text, follow instructions, and generalize across domains. Their architecture is primarily based on the *Transformer* [14], a neural network model introduced by Vaswani et al. in 2017.

2.5.1 Neural Networks for Language Modeling

At their core, LLMs are built upon artificial neural networks — specifically, deep learning architectures composed of multiple layers of linear transformations and nonlinear activations. Unlike traditional rule-based systems, neural networks learn statistical patterns from massive corpora of input data. This process is called **training** and is responsible for tuning the neural network's parameters (weights and biases) to achieve the best possible model output to the user's input queries.

Early models, such as Recurrent Neural Network (RNN) [15] and Long Short-Term Memory (LSTM) [16] networks, were designed to process sequences token by token, maintaining a hidden state to capture contextual information. While RNNs struggled with vanishing gradients during training, LSTMs introduced gating mechanisms to better preserve long-term dependencies. Despite these improvements, both architectures

suffer from limited parallelism and difficulty modeling long-range relationships due to the short reference window, which resulted in limited accuracy, long execution time, and ultimately led to the adoption of the Transformer architecture.

2.5.2 Tokenization and Input Representation

Before input can be processed by an LLM, it must first be broken down into individual units called **tokens**. Tokens can be words, subwords, or characters depending on the model's individual tokenizing rules.

Popular tokenization methods include:

- **Byte-Pair Encoding (BPE)**
  A subword tokenization algorithm that iteratively merges the most frequent pairs of characters or subwords. It allows the model to efficiently handle rare and compound words. BPE is used in models like GPT-2 and RoBERTa.

- **WordPiece**
  Similarly to BPE, WordPiece merges based on maximizing the likelihood of the training data. It breaks rare words into subword units, often marked with a prefix like `##`. It is used in BERT and related models.

- **SentencePiece**
  SentencePiece tokenizes raw text as a sequence of characters or bytes, independent of whitespace. It supports both BPE and unigram models. It is suitable for multilingual use and is being used in models like T5, XLNet, and LLaMA.

2.5.3 The Transformer's Advancements

The key advancements of the transformer architecture in comparison with previous ones are:

- **Parallelization and Faster Training:** The key problem with RNNs and LSTMs is that both of these architectures need to process the input tokens in sequence, one token after the other. As a result, parallelization strategies cannot be implemented. On the other hand, **transformers** use a self-attention mechanism that allows them to process all tokens in a sequence simultaneously. This enables them to take advantage of modern hardware like GPUs, drastically reducing training time and making it feasible to train on large datasets.

- **Better Understanding of Long-Range Dependencies:** The sequential processing of both RNNs and LSTMs creates another weakness, the **vanishing gradient** problem. Information found earlier in the input text diminishes as it has to be passed through many steps to reach the end. Transformers' self-attention mechanism solves this by directly linking every word in the input query to every other word in the sequence. This allows the model to instantly recognize the dependencies between all words, regardless of their position, which is critical for tasks, like understanding the context of a word in a long document.

- **Superior Performance on Modern NLP Tasks:** The previous strengths lead the Transformers to achieve state-of-the-art results across a wide range of NLP tasks, including machine translation, text summarization, and question answering.

2.5.4 The Transformer Architecture

The Transformer [14] is the foundation of nearly all modern LLMs. It is composed of encoder and decoder blocks, though most language models today (like GPT, LLaMA, Qwen) use only the decoder part for text generation.

The Transformer architecture, as shown in Figure 1, consists of the following key components:

- **Embedding:** Each token in the sentence is converted to an independent vector which contains information about its meaning and its position in the text. This step is necessary, since machine learning systems can only deal with numbers. Positional encoding is crucial, since transformers processes all tokens at once and have no inherent sense of word order.
- **Multi-Head Self-Attention:** This mechanism allows the model to weigh the importance of each token in a sequence relative to all others, regardless of position. Multiple attention heads operate in parallel, each learning different relationships.
- **Feed-Forward Network (FFN):** A position-wise fully connected network that processes the output of the attention mechanism.
- **Residual Connections:** These connections help preserve gradient flow during training, addressing the vanishing gradient problem.
- **Layer Normalization:** Applied after each sub-layer to stabilize and accelerate training.
- **Softmax:** This function converts a vector of numbers into a probability distribution over all possible words in the vocabulary. The word with the highest probability is the one that is selected as the model's prediction for the next word in the sequence.

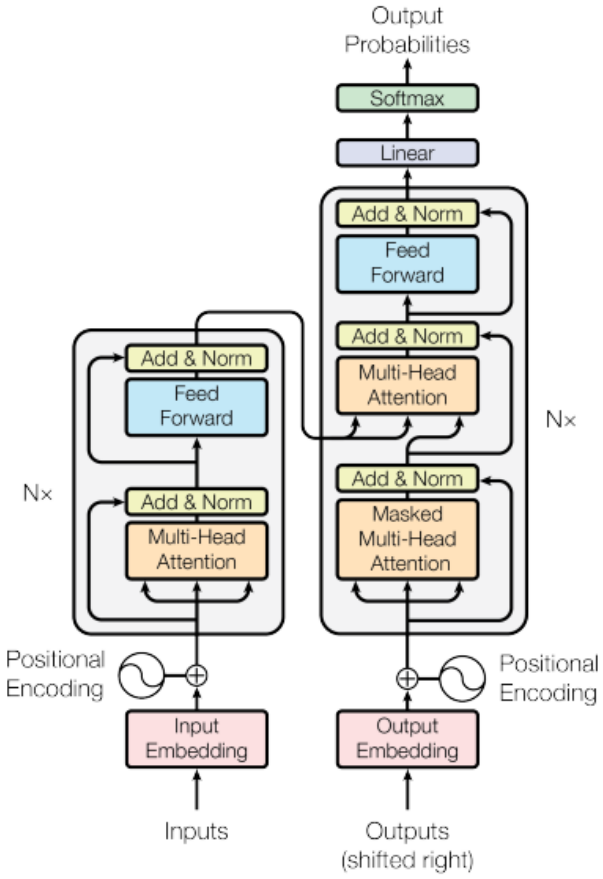


Figure 1: Overview of the Transformer architecture, including multi-head self-attention, feedforward layers, and residual connections. Adapted from Vaswani et al. (2017).

2.5.5 Self-Attention Mechanism

The self-attention mechanism is the key component of the Transformer architecture, enabling the model to capture contextual relationships between the tokens in a given sequence. It allows the network to attend to all positions in the input simultaneously, regardless of their relative distance.

For each input token, three distinct vectors are computed through learned linear transformations:

- **Query (Q):** Encodes the token used to query other tokens.
- **Key (K):** Encodes how relevant each token is when queried.
- **Value (V):** Contains the actual information to be aggregated.

For a sequence of n tokens, the model computes a Query, Key, and Value vector for each token. These are typically packed into matrices $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$, and $V \in \mathbb{R}^{n \times d_v}$, where d_k and d_v are the dimensions of the key and value vectors, respectively.

The attention weights are computed by taking the scaled dot-product between each query and all keys:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The dot product $QK^T \in \mathbb{R}^{n \times n}$ produces a matrix of similarity scores between all pairs of tokens. These scores are scaled by $\sqrt{d_k}$ to prevent large values that would lead to vanishing gradients, when passed through the softmax function. The softmax function converts the scores into normalized attention weights, indicating the importance of each token relative to the current one.

These weights are then applied to the corresponding value vectors, resulting in a weighted sum that reflects the contextualized representation of each token. This process allows every token to include information from the entire input query.

Each token is mapped to a unique integer index. These indices are then embedded into high-dimensional vectors using learned embedding matrices. Positional encodings are added to these embeddings to provide information about the order of tokens in the sequence.

2.5.6 Pretraining and Fine-Tuning

LLMs are typically pretrained on massive corpora using unsupervised objectives (unlabeled data) such as:

- **Causal Language Modeling (CLM):** Predicting the next token in a left-to-right fashion (e.g., GPT-style).
- **Masked Language Modeling (MLM):** Predicting randomly masked tokens within a sequence (e.g., BERT-style).

This process typically produces base models. Therefore, these base models can be fine-tuned on particular tasks (e.g., question answering, summarization), domain-specific data, or instructed through prompting (few-shot or zero-shot prompting).

2.5.7 Scaling Laws and Emergent Behavior

Research in LLMs has demonstrated that the model's performance and accuracy improve as the size of it increases. This phenomenon, known as *scaling laws*, relates three key factors:

- Model size (number of parameters)
- Training dataset size
- Compute budget (e.g., GPU time)

Model Size and Parameters The size of a model is computed by its number of **trainable parameters**, the parameters of the neural network, typically measured in millions or billions. These parameters include the weights and biases of all linear transformations (e.g., in attention layers, feedforward networks, embedding layers, etc.). In transformer-based models, each attention head, feedforward layer, and layer normalization contribute to the total amount of these parameters.

For example:

- GPT-2: $\sim$1.5 Billion parameters
- GPT-3: 175 Billion parameters

- LLaMA 2: 7, 13, and 65 Billion variants
- Qwen 2.5: up to 72 Billion parameters

As the number of parameters increases, the model becomes more capable of representing more complex patterns, relationships, and abstractions within language data.

Diminishing Returns and Trade-offs Although performance improves with scale, it does not do so indefinitely. Larger models:

- require exponentially more computational resources in order to perform inference on the input queries.
- are harder to deploy and fine-tune locally due to the large computational resources that they require.

Nevertheless, scaling remains a powerful strategy in LLM development, and continues to guide the design of new architectures.

2.5.8 Model Quantization

Quantization [17], [18] is a technique used to reduce the size and memory requirements of LLMs by representing model weights with lower numerical precision. Instead of using standard 16-bit or 32-bit floating-point values (FP16 or FP32), quantized models use 8-bit integers (INT8), 4-bit formats (e.g., Q4_0, Q4_K), or other compact representations.

Purpose and Benefits The main goals of quantization are:

- **Reduced memory requirements**: Enables large models (e.g., 7 or 13 Billion parameters) to run on smaller GPUs or CPUs.
- **Faster inference**: Smaller models lead to improved latency, especially when dealing with limited hardware resources.
- **Lower power consumption**: Useful in edge devices or low-resource environments.

Trade-offs While quantization significantly improves efficiency and execution time, it may lead to a slight reduction in model accuracy due to reduced numerical precision. However, modern quantization strategies can preserve most of the model's performance, while drastically improving usability.

Quantized models are especially useful in local LLM environments, where they enable deployment of high-parameter-count models on limited hardware.

2.5.9 Transformer-based Models

Many state-of-the-art LLMs today are based on the Transformer decoder-only architecture. Some widely used models include:

- **GPT (OpenAI)**
- **LLaMA (Meta)**

- **Qwen (Alibaba)**
- **Mistral and Mixtral (Mistral.ai)**

Each model is trained with variations in architecture, tokenizer design and training objectives, but all have the Transformer architecture as a base foundation.

2.5.10 Model Temperature and Output Diversity

One important hyperparameter in language generation tasks is the **temperature**, which influences the randomness of the model's predictions. Temperature values range from 0 to 1, with:

- low values (e.g., 0.1–0.3) producing more deterministic and focused outputs.
- high values (e.g., 0.7–1.0) yielding more diverse, creative, or exploratory results.

In the context of this thesis, low temperatures are typically preferred in stages such as table correction, extraction, and translation — where factual consistency and reproducibility are critical.

2.5.11 Multilingual Models and Language Biases

While many LLMs support multiple languages, their performance often varies substantially, depending on the source of their training data. English, being the dominant language in most pretraining data, receives the highest model attention and, as a result, yields the most reliable performance.

Although multilingual LLMs aim to generalize across languages, they still show uneven quality, when dealing with specialized domains and terminology in underrepresented languages like Greek. For this reason, this thesis implements a structured translation step from Greek to English, before applying retrieval tasks.

2.6 Agents in LLM Workflows

In modern LLM-based systems, **agents** [19] are autonomous or semi-autonomous components that combine a language model with specific instructions to complete a well-defined task. Each agent operates based on a structured prompt and typically performs a single role within a broader multi-step pipeline. Unlike basic prompt-based LLM calls, which perform a single task based on a static instruction, agents are modular components designed for reuse and composition within larger workflows. They encapsulate a specific role, maintain stricter prompt structures, and are often chained with other agents to enable multi-step reasoning. This architectural separation encourages interpretability, robustness, and easier debugging in complex pipelines.

Agents follow a simple cycle:

- **Receive input:** A text chunk or context relevant to a specific task alongside a prompt that includes all the instructions and guidelines that the agent needs to follow.
- **Execute reasoning:** The agent executes the prompt's instruction steps in the input data.

- **Produce output:** A structured, often JSON-formatted, result based on the task specification.

In this thesis, agents are not limited to structured prompt execution alone; they also integrate fixed Python logic as part of their operation. Each agent is implemented with a clearly defined role and executes a combination of LLM-based reasoning and deterministic Python functions. These functions are not optional tool calls triggered dynamically by the model, but are instead invoked as mandatory processing steps within the agent's logic. This design ensures predictable behavior and allows for hybrid workflows that combine the generative capabilities of LLMs with the precision and control of traditional programming.

Multi-Agent Workflows : [20] For executing more complex tasks, multiple agents can be combined to form *multi-agent workflows*, where each agent contributes a specialized capability and passes intermediate outputs to the next step. This architectural style enables complex pipelines that are easier to scale, maintain, and adapt to new domains.

Benefits of Agent-Based Design

- Encourages separation of tasks that leads to increased accuracy and easier debugging. LLMs often perform better, when they are assigned with specific and distinct tasks.
- Enables modular reuse in other pipelines or domains.

2.7 Translation from Greek to English

One of the core components of processing Greek PDF documents is the translation of Greek-language documents into English. This step is essential in order to take full advantage of the capabilities of LLMs, many of which are trained primarily on English data and achieve their best performance when operating in English texts.

2.7.1 Challenges in Domain-Specific Translation

Translation of Greek documents, like soil analysis reports, invoices and blood test results, presents unique challenges not typically encountered in general-purpose text translation:

- **Technical terminology**: Words like "αντίδραση εδάφους", "αγωγιμότητα", or "κορεσμένη πάστα" have specific scientific meanings. General-purpose translation APIs may incorrectly map them to misleading English terms.
- **Measurement units and context sensitivity**: Words like "dS/m", "mg/kg", or "meq/100g" are context-sensitive and need to be preserved precisely. Misplacement or conversion can lead to distortion of the results.

2.7.2 Neural machine translation systems

In addition to commercial services, several open-source neural machine translation systems have been developed and are widely used in research and practice. Two representative examples relevant to Greek–English translation are the Helsinki-NLP Opus-MT models and Facebook's M2M-100 model.

Helsinki-NLP Opus-MT [21, 22, 23] The Opus-MT models are part of the OPUS project and are based on the Marian neural machine translation framework. They are trained on large-scale parallel corpora collected from the OPUS repository, covering many language pairs including Greek–English. These models are lightweight and optimized for efficient deployment, making them suitable for local translation tasks.

Facebook M2M-100 [24, 25] The M2M-100 model, developed by Facebook AI, is the first fully multilingual neural machine translation system trained to support direct translation across 100 languages. Unlike traditional approaches that rely on English as an intermediate pivot, M2M-100 is trained on large multilingual corpora to perform direct translations between any two supported languages, including low-resource pairs, such as Greek–English.

2.7.3 LLM-Based Translation with Domain Awareness

To overcome the limitations of generic translation tools, LLMs were used to perform domain-aware translation. These models can be prompted in order to handle:

- preservation of domain specific terminology.
- consistent rendering of measurement units and abbreviations.
- table-aware translation using row-by-row structure.

2.8 Full-Context Prompting

In full-context prompting, the entire document's content is passed directly into the input prompt of the LLM alongside the user's question. The model is expected to process all the information in context and generate a relevant answer to the user's input.

2.8.1 Advantages of Full-Context Prompting

- **Simplicity**: Easy to implement without retrieval systems or indexes.
- **Context awareness**: The model has access to the complete document, enabling it to understand relationships between entities and create a response relevant to the whole document.
- **Effective for small documents**: Performs well and has low execution time, when the input document is short enough to fit within the model's context window.

2.8.2 Limitations

Despite its simplicity, full-context prompting has critical scalability limitations:

- **Token limits**: Most LLMs have fixed context windows (e.g., 4K, 8K, or 32K tokens). Documents that surpass this length cannot be fully passed to the model.
- **Context dilution**: As the input of the model rises, the harder it becomes for the model to focus on the relevant parts, reducing answer quality and leading to hallucinations.

- **Redundancy**: Repeating the entire context for each query is inefficient and unnecessary, when only parts of the document are relevant to each input query.

These limitations motivate the use of more efficient retrieval methods like **RAG** where only the most relevant sections of the document are dynamically selected and passed to the model.

2.9 Embeddings and Vector Databases

In order to overcome the limitations of full-context prompting, more efficient retrieval strategies are required, especially when dealing with long documents. One of the core components enabling such strategies is the use of **text embeddings**.

2.9.1 What Are Embeddings?

Embeddings are high-dimensional vector representations of text. Each sentence, paragraph, or document is mapped to a vector in a continuous vector space, where semantic similarity corresponds to geometric closeness. Texts with similar meaning will have embedding vectors that are close in terms of cosine similarity or Euclidean distance.

These vectors are generated using pretrained or fine-tuned neural networks, typically Transformer-based models trained specifically for embedding tasks. Embeddings can capture context, word order, and semantics.

2.9.2 Types of Embedding Models

There are several classes of embedding models, including:

- **General-purpose embedding models**: such as `all-MiniLM`, `sentence-transformers`, or `BGE-m3`, trained to represent sentence-level meaning across domains.
- **Multilingual embeddings**: capable of embedding text in multiple languages.

2.9.3 Chunking and Indexing Strategy

Since LLMs cannot process entire documents at once due to limited reference window, each document is first split into smaller segments, or **chunks**. Common strategies include:

- Fixed-size token windows (e.g., 256 or 512 tokens per chunk)
- Sentence or paragraph boundaries
- Content-aware chunking (e.g context in markdown format) that uses the document's internal formatting to define boundaries
- Overlapping windows to preserve continuity
- Custom chunking techniques

Each chunk is passed through the embedding model, converted into a vector and then stored in a **vector database**, where it can be indexed and retrieved based on different similarity techniques.

2.9.4 Vector Databases and Similarity Search

A vector database stores embeddings and provides efficient similarity-based retrieval. Given the input query (also embedded into a vector), the database returns the most similar chunks from the indexed documents.

Popular vector database options include:

- **ChromaDB**: Lightweight and Python-native, well suited for local experimentation.
- **FAISS (Facebook AI Similarity Search)**: High-performance and widely used in research.

The combination of semantic embeddings and vector search enables the retrieval of the most relevant document's chunks. As a result, the model processes only valuable information, leading to a significantly smaller input. This approach forms the base of the Retrieval-Augmented Generation (RAG) pipeline.

To simplify integration with LLM-based pipelines, higher-level retrieval abstractions, such as `VectorStoreIndex`, were also used in this thesis. This component facilitates indexing, chunk management, and similarity querying on top of the underlying vector store.

2.10 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) [26], [27] is a hybrid architecture that combines the strengths of information retrieval and large language models (LLMs) to answer questions or perform reasoning over large input data. Unlike Full-Context Prompting, which struggles with large inputs that come from big documents, RAG retrieves and ingests only the most relevant chunks of a document into the model's prompt.

2.10.1 Core Architecture

RAG is composed of two primary components:

- **Retriever**: Uses a vector database to retrieve the most relevant document chunks based on the user's query. This is achieved through semantic similarity using embeddings and techniques like cosine similarity.
- **Generator**: A language model (e.g., Qwen2.5, LLaMA 3.1) that takes both the user query and the retrieved context (returned from the retriever) as input and produces a coherent, context-aware response.

This architecture allows the system to scale to long or multi-document input data by decoupling content storage from generation.

2.10.2 Theoretical Motivation

The core motivation behind RAG stems from two fundamental limitations of LLMs:

1. **Limited context window**: LLMs have a maximum token limit that constrains the amount of input they can process. RAG addresses this by retrieving only the most relevant parts of a large input, allowing the model to operate effectively even on documents that exceed its native context capacity.

2. **Static training data**: LLMs are pretrained on fixed datasets and cannot access new or evolving information after training. RAG introduces an external source on new information that can be used in order to enhance the model's output by enabling access to up-to-date or domain-specific knowledge without requiring model retraining.

This augmentation of both memory and adaptability allows LLMs to generalize better, respond with increased accuracy, and be extended to use cases that demand real-time or long-range information access.

2.11 LangChain Framework

To implement the architectures and workflows described above — including document chunking, embedding, retrieval, and generation — modular frameworks are required to manage interactions between components. One such framework is **LangChain** [28], a Python-based toolkit designed for constructing applications that use large language models (LLMs).

2.11.1 Purpose and Capabilities

LangChain simplifies the development of LLM-based applications by providing high-level abstractions for:

- **Prompt engineering**: Structured and dynamic prompt templates for consistent model behavior.
- **Chaining**: Sequential or conditional linking of model calls, memory, tools, and logic steps.
- **Retrieval**: Native integration with vector databases and support for RAG-style pipelines.
- **Agents and tools**: Creation of tool-using agents that can be connected with python functions in order to execute tasks.
- **Model integration**: Provided interfaces for calling local or cloud-hosted language models.

Chaining A core abstraction in LangChain is the concept of **chains**. Chains are structured workflows in which multiple operations are composed to transform an input into a desired output.

LangChain supports several types of chains:

- **Simple chains:** Chains that involve only one-step flows that apply a prompt template to the model's input and return a model-generated response.
- **Sequential chains:** Linear pipelines in which each step's output becomes the next step's input.
- **Conditional chains:** Chains with branching logic based on the model's outputs.

Chains are used to construct a combined prompt, invoking the model, and parsing the output. Chaining improves modularity, code reuse, and the composability of components in complex workflows.

These components allow for building flexible, composable systems that integrate language models with search, summarization, and reasoning capabilities.

2.11.2 Relevance to Augmented Generation Architectures

LangChain is particularly suited for implementing RAG workflows. It provides built-in support for document loaders, text splitters, embedding pipelines, vector store connections, and retriever–generator interfaces. These modular components allow rapid experimentation with different chunking strategies, embedding models, and prompt configurations.

2.12 LlamaIndex Framework

LlamaIndex [29] is a high-level framework designed to simplify the process of connecting large language models (LLMs) to structured and unstructured data sources for retrieval and generation tasks. It plays a central role in building Retrieval-Augmented Generation (RAG) pipelines by offering modular components for document ingestion, indexing and querying.

Core Capabilities:

- **Document parsing and chunking:** Support for splitting documents into semantically meaningful chunks using sentence-based or token-based strategies.
- **Vector indexing:** Supports creation of vector stores through integrations with external databases (e.g., ChromaDB, FAISS) or internal abstractions like `VectorStoreIndex`.
- **Query engines:** Supports various retrieval methods such as similarity search, keyword filtering, or hybrid approaches, wrapped in a unified interface.

2.13 Ollama: Local Language Model Execution

To enable efficient and private inference with LLMs, especially in a local development environment, this work utilizes **Ollama** [30] — a platform for running LLMs on local machines with GPU acceleration and minimal configuration.

2.13.1 Purpose and Functionality

Ollama is an open-source tool that allows developers to download, run, and interact with pre-trained language models on their own hardware. It provides a lightweight API and command-line interface that supports:

- **Model deployment**: Easy downloading of a large set of models (e.g., `qwen`, `llama`, `mistral`, etc.) via simple terminal commands.
- **Local inference**: Execution of models on local CPUs or GPUs without requiring cloud access or API keys.
- **Hardware optimization**: Ollama optimizes the utilization of the resources of local machines depending on the requirements of the loaded model.

2.13.2 Benefits of Local Execution

Using Ollama for local inference offers several key advantages:

- **Data privacy**: No user data or prompt content is transmitted to external servers via API calls.
- **Cost efficiency**: Eliminates the need for pay-per-token or subscription-based API services.
- **Offline compatibility**: Models can be used entirely without internet access once downloaded.

2.13.3 Integration with LangChain and Embedding Pipelines

Ollama exposes a simple API that is fully compatible with LangChain's `LLM` and `ChatModel` interfaces. This allows it to function as the generator component in RAG and simple inference pipelines. It also supports running custom prompts, translation tasks, summarization, and few-shot reasoning directly within a reproducible local environment.

2.14 HuggingFace Transformers

An alternative to Ollama for working with LLMs locally is the **HuggingFace Transformers** ecosystem. HuggingFace [31] provides one of the most extensive open-source repositories of pretrained language models, including architectures such as `BERT`, `GPT-2`, `T5`, `LLaMA`, `Mistral`, and many more.

2.14.1 Model Variety and Customization

HuggingFace offers a significantly broader range of models compared to Ollama, spanning multiple domains, languages, sizes, and pretraining objectives. It also provides advanced tools for:

- Fine-tuning on custom datasets
- Quantization
- Adapter-based training (e.g., LoRA)
- Tokenizer customization and pretraining from scratch

This makes it a powerful platform for research, experimentation, and production-grade deployment.

2.14.2 Flexibility vs Simplicity

Despite its flexibility, the HuggingFace ecosystem is generally more complex to set up. Running models often requires:

- Manual installation of model weights and tokenizers
- Hardware configuration (e.g., CUDA, device mapping)

As a result, while HuggingFace is ideal for advanced customization and large-scale experimentation, it is less plug-and-play than Ollama, which automates the setup of the models.

2.15 System Specifications and Models Used

The development and experimentation conducted in this thesis were carried out on a high-performance local machine, configured to support efficient inference and processing of large language models (LLMs). The following hardware and software specifications define the environment in which all workflows were implemented:

2.15.1 System Specifications

- **Operating System**: Ubuntu 24.04 LTS (Kernel 6.8.0)
- **CPU**: Intel 12th Gen Core i9-12900, 24 threads
- **RAM**: 125 GiB DDR4
- **GPU**: NVIDIA GeForce RTX 3090, 24 GiB VRAM

This configuration allows for the local execution of both embedding and generation models using quantized formats, while maintaining fast retrieval and low inference latency.

2.15.2 Language Models Used

To support document processing, summarization, translation, and retrieval-augmented generation, the following LLMs were utilized:

- **Qwen2.5** [32]: Used for generation and translation tasks. These models offer high reasoning capabilities and were deployed locally using quantized variants via `Ollama`.
- **LLaMA 3.1** [33]: Used for baseline testing and local comparisons; selected for its balance between performance and resource usage.
- **Llama - Krikri** [34]: Used for testing in translation from Greek to English. Trained in both Greek and English dataset.

2.15.3 Embedding Models Used

Multiple embedding models were evaluated to support semantic retrieval across translated and original Greek-language documents. The selection aimed to balance performance, multilingual compatibility, and local inference efficiency.

- **sentence-transformers/all-mpnet-base-v2** [35]: A widely used, general-purpose English sentence embedding model. It served as the primary embedding model for translated documents.
- **all-MiniLM-L6-v2** [36]: A lightweight variant from the SentenceTransformers family, tested for efficiency comparisons in low-resource setups.
- **dimitriz/st-greek-media-bert-base-uncased** [37]: A Greek-specific sentence embedding model, evaluated for directly handling Greek documents without translation. Its performance was useful for comparison between the accuracy of the retrieval on the raw Greek docs versus the translated English equivalents.

All embeddings were generated using the HuggingFace Transformers and Sentence-Transformers libraries and integrated with the retrieval pipeline via vector databases such as ChromaDB.

3 Methodology

3.1 Description Of Components

3.1.1 Text Postprocessing Module

Motivation and Purpose As previously mentioned, text extraction from PDF documents is a non-trivial task. Inconsistent formatting, structured data, and even scanned documents make this procedure quite challenging. Text extraction libraries, like *PdfPlumber*, *Unstructured Partition*, etc., often make mistakes in the entities extraction (broken tables, wrong formatting, etc.). The issue is even bigger in scanned documents, which require OCR, as pre-processing step. To address these limitations, a post process module has been implemented.

System Design and Implementation The Post processing module follows the workflow described below:

- **Input**: The input of this module is the raw output of the text extraction libraries which contains text in Greek, English or event both languages.
- **Chunking**: The input text, needs to be split into smaller parts in order to fit the model's context window and achieve better results. For that purpose, *SentenceSplitter* library of *LlamaIndex* (Section. 2.12) framework is used to split the text into chunks based on sentence boundaries, ensuring that each segment remains within the model's token limit, while preserving contextual integrity.
- **Model:** The core component of this module is a LLM, which processes the input chunks sequentially. Deterministic outputs are essential to ensure consistent behavior across runs, enable reliable merging of processed chunks, and support reproducible evaluation of the overall pipeline. In order to achieve this behavior, the model's temperature parameter is set to zero.
- **Prompt:** Instructions are provided through a structured prompt to the LLM. The prompt instructs the model to correct OCR and parsing errors, reconstruct logical structure (sections, tables, paragraphs), and apply consistent formatting using Markdown syntax. It emphasizes precision, strict preservation of all technical content (including Greek scientific terms and units), and avoids any hallucinated or missing information.
- **Output:** After the processing of every document's chunk, all outputs are merged together. The result is a unified, cleaned and corrected text formatted in Markdown.

3.1.2 Translation Module

Motivation and Purpose In many occasions, the input PDFs contain text in Greek. As previously discussed, LLMs tend to perform better when the input query is provided in English due to the predominance of English-language datasets used during their training. While many translation solutions exist, such as cloud-based APIs (e.g., Google Translate) or local open source models (e.g., Helsinki-NLP 2.7.2, Facebook M2M-100 2.7.2), they present certain limitations, such as data privacy control, limited understanding of domain-specific terminology and inability to sustain input formatting

in the output. The goal of the translation module is to tackle all these limitations by providing a robust, private and domain-aware translation.

System Design and Implementation The translation module operates according to the following workflow:

- **Input**: Receives in the input the Greek text.
- **Chunking**: Similarly with the *Post-Process* Section (3.1.1), the input text, needs to be split into chunks. For that purpose, the *SentenceSplitter* library is used again to split the text in chunks.
- **Model:** The central component of this module is a multilingual LLM capable of translating text from Greek to English without altering the content's format.
- **Prompt:** The multilingual LLM is guided by a prompt in order to perform translation of the technical document from Greek to English. The instructions emphasize precision, requiring the model to preserve original formatting, tables, and numerical units, while avoiding any commentary or additions. This ensures that the translated output remains faithful to the structure and content of the original document.
- **Output:** After all document chunks are processed, the translated outputs are sequentially merged into a unified English version that preserves the original structure and formatting, rendered in Markdown.

3.1.3 Full Context Retrieval Component

Purpose The following component is implementing the **Full Context Prompting**. In this approach, the entire content of the document is provided to the model's input alongside the user's prompt. This approach leads the model to generate responses with full contextual awareness, but faces significant challenges, as the length of the input document increases and reaches the model's context window.

Inputs

- **documents:** List of document objects, that contain the PDF's content.
- **prompts:** A list of string prompts, defining the fields or values to be extracted from the document, alongside instructions about the desired output format that the LLM has to follow.
- **model_name:** The name of the language model to be used for inference (e.g., `llama3.1`, `Qwen2.5`, etc.).

Tunable Parameters

- **model:** Specifies the language model used for inference. The choice of model directly affects response quality, reasoning ability, and inference cost.
- **temperature:** Affects the randomness of the model's output. Lower values (e.g., 0.0) produce more deterministic and consistent results, while higher values increase diversity.

Processing Workflow

1. Initialize the LLM using the `OllamaLLM` wrapper with the given `model_name` and the desired temperature.
2. Concatenate the contents of all document objects into a single unified text block, separated by line breaks.
3. For each prompt:
    - Construct a complete query by appending the prompt to the full document text.
    - Invoke the LLM with the constructed prompt and clean the response (e.g., remove code fences).
    - Attempt to parse the output into a dictionary using a robust JSON parser.

Output The component returns a dictionary containing all extracted key-value pairs, aggregated across all prompts.

3.1.4 Classic RAG with VectorStoreIndex

Purpose This component implements a classic Retrieval-Augmented Generation (RAG) pipeline using `VectorStoreIndex`, a high-level abstraction that internally manages chunking, embedding, and indexing of documents. Instead of processing the entire document at once, the text is divided into overlapping chunks and becomes embedded into a vector space. Given a prompt, the most semantically relevant chunks are retrieved and passed to the language model as context. This approach offers a balance between contextual accuracy and scalability, particularly in multi-document or long-text scenarios.

Compared to low-level vector store solutions, like ChromaDB or FAISS, the VectorStoreIndex provides a higher-level abstraction that builds on top of these backends, prioritizing ease of use and seamless integration within the LlamaIndex framework (Section 2.12).

Inputs

- **documents:** A list of text document objects to be indexed and searched.
- **prompts:** A list of user-defined queries to guide retrieval and extraction.
- **model_name:** The name of the language model used for inference.
- **embedding_model:** The name of the sentence embedding model used for vectorization.

Tunable Parameters

- **k (top-k retrieval):** Defines the number of most relevant chunks to retrieve for each query. A higher value increases the chance of including useful context but may also introduce irrelevant information and increase the input leading to reduced accuracy.
- **model:** Specifies the language model used for generating responses.

- **temperature:** As described previously, affects the randomness of the model's output.
- **embedding model:** Controls how textual chunks are embedded into the vector space. Different embedding models yield different semantic representations and influence retrieval accuracy.
- **chunk size:** Determines the length (in tokens) of each text chunk during document preprocessing. Larger chunks preserve more context, but may increase the input with irrelevant information.
- **chunk overlap:** Specifies the number of overlapping tokens between consecutive chunks. This helps maintain continuity and avoid missing context at chunk boundaries.

Processing Workflow

1. Initialize the embedding model via `HuggingFaceEmbeddings`, and load the LLM using `OllamaLLM` with a fixed temperature.
2. Configure global parameters (embedding model, LLM).
3. Split each document into overlapping chunks using `SentenceSplitter`.
4. Create the vector index using `VectorStoreIndex.from_documents()`.
5. For each prompt:
    - Retrieve the top-k most relevant chunks using semantic similarity and the input embedding model.
    - Combine the retrieved chunks with the prompt to form a complete query.
    - Invoke the LLM and parse the output using a robust JSON parser.
    - Merge valid outputs into a shared result dictionary.

Output A dictionary containing the aggregated key-value results extracted across all prompts, based on semantically retrieved context chunks.

3.1.5 Chunk-Level Retrieval

Purpose This component performs targeted information extraction using prompts generated for each individual text chunk. Each chunk is paired with one or more prompts. This design combines the benefits of RAG-style retrieval (targeted context for each field) and Full-Context Prompting, as the entire chunk—containing the field of interest—is passed directly to the LLM as part of the input.

Inputs

- **chunks:** A list of text segments derived from the source document.
- **prompts:** A list of dictionaries, where each entry includes:
    - `chunk`: The index of the chunk this prompt is associated with.

- `group`: A group ID used to indicate the position of a field group within a chunk. Since only a limited number of fields (e.g., 5) are included in each prompt for readability and context fitting, multiple prompts may be generated per chunk. The `group` index tracks which subset of fields the prompt covers.
- `prompt`: A structured instruction for the language model, tailored to a specific field group.

Example:

```
{
  "chunk": 3,
  "group": 2,
  "prompt": "You are an extractor agent. Extract the
            following fields from the text..."
}
```

- **model_name:** The name of the language model to be invoked for inference.

Tunable Parameters

- **model:** Specifies the language model used for generating responses.
- **temperature:** As described previously, affects the randomness of the model's output.

Processing Workflow

1. For each chunk:
   - Identify and sort all prompts associated with that chunk.
   - For each prompt:
     - Combine the prompt with the chunk text.
     - Invoke the language model with the combined input.
     - Parse the model output using a relaxed JSON parser.
     - Remove any fields with empty values.
     - Merge valid outputs into a global result dictionary using a recursive update function.

Output Returns a merged dictionary (`combined_results`) containing extracted key-value pairs across all chunks and prompts.

3.1.6 Field Detection Agent

Purpose The Field Detection Agent is responsible for identifying which fields in a document chunk are likely to contain extractable values. Its goal is to isolate short, meaningful fields—such as numerical measurements, codes, names, and contact details—that are suitable for later retrieval tasks. By filtering out non-informative or long-text elements, this agent prepares a targeted set of fields for further processing and extraction.

Inputs

- **chunks:** Text segments derived from the original document.
- **seen fields:** Since the agent is limited to chunk level only, the previously extracted fields from previous chunks are passed through the input for duplication checks.
- **prompt:** A structured instruction template that guides the agent to select useful fields, with domain-specific rules for documents, like invoices or lab reports.
- **model:** A language model used to produce the responses.

Processing Workflow

1. Load the field detection prompt containing instructions and return format rules.
2. For each text chunk:
    - Pass the chunk into the prompt, along with an explicit instruction to return a well-formed JSON object.
    - Invoke the LLM using `temperature = 0.0` to ensure deterministic field lists.
    - Capture and parse the returned JSON containing the predicted list of `fields`.
    - After the processing of the chunk, it compares the extracted fields from this chunk with fileds extracted from other chunks using regular expressions. If one fields is already extracted previously, it is not included again.

Output A JSON object for each chunk with the format:

```
{
"fields": ["field1", "field2", ...]
}
```

3.1.7 Field Postprocessing Agent

Purpose The Field Postprocessing Agent acts as a validation and filtering layer, following initial field detection. Its role is to refine the list of candidate fields produced by the *Field Detection Agent* (Section 3.1.6) by eliminating duplicates (synonyms are not tracked by regular expressions), non-extractable entries, and unclear or ambiguous fields. This step aims to improve the reliability and simplicity of the extraction process by reducing the number of calls that the system needs to be executed for unnecessary fields.

Inputs

- **text chunk:** The same document chunk originally passed to the Field Detection Agent.
- **potential fields:** The output of the Field Detection Agent that consists of a list of field names, to be validated and cleaned.
- **prompt:** A structured instruction that guides the agent to identify which fields are semantically valid and practically extractable.
- **model:** A deterministic language model (`temperature = 0.0`) invoked to ensure stable field filtering decisions.

Processing Workflow

1. The text chunk and its corresponding list of potential fields are inserted into a predefined validation prompt.
2. The prompt instructs the model to validate each field using the following criteria:
   - **Semantic duplication:** Remove fields that are synonyms or variants of earlier fields in the same list of the chunk's fields.
   - **Presence in text:** Remove fields whose values are not explicitly present or clearly implied in the text.
   - **Value feasibility:** Remove fields with values that are too complex, multi-paragraph, or unstructured to extract reliably.
3. The agent returns a structured JSON response categorizing fields into:
   - `fields_to_keep`: A list of valid fields to retain.
   - `fields_to_remove`: A list of excluded fields, each with a justification.

Output A structured JSON object for each chunk:

```
{
  "fields_to_keep": ["field1", "field2"],
  "fields_to_remove": [
    {"name": "fieldX", "reason": "Brief justification for removal"}
  ]
}
```

3.1.8 Prompt Builder Agent

Purpose The Prompt Builder Agent is responsible for generating structured and domain-aware prompts that are later used to guide the final extraction phase. Based on the filtered field list, it creates a clear and well-structured prompt for the relevant extraction method to follow. This ensures that all downstream extraction tasks follow a consistent format and return JSON output.

Inputs

- **Discovery JSON:** A dictionary containing:
  - `fields`: A list of validated field names to be extracted from the document.
- **prompt template:** A prewritten instruction prompt defining schema rules, response format, and extraction constraints.
- **model:** A deterministic language model (`temperature = 0.0`) used to guarantee consistent and schema-compliant prompt generation.

Processing Workflow

1. For each document chunk:
    - Retrieve the list of unique fields that were extracted by previous agents.
    - Group the fields into smaller batches of at most 5 fields each, to avoid overly long or ambiguous prompts.
    - For each field group, construct a discovery JSON object with the following structure:

```
{
    "fields": ["Field1", "Field2", ..., "Field5"]
}
```

    - Inject the discovery JSON into the prompt template.
    - Send the final prompt to the LLM, which generates a structured instruction prompt related to the fields.
    - Store the generated prompt in:
        - An in-memory list of all prompts.
        - A corresponding output text file named `prompt_builder_output_<chunk>_<group>.txt`.

Output A plain-text instruction string for each field group, formatted for extraction use. Each prompt includes:

- A one-line description of the Extractor's role.
- The full JSON output schema with all field names.
- Extraction rules covering value, unit, and method.
- Handling instructions for missing values (use of empty strings).
- An explicit requirement to return only valid JSON (no explanations, markdown, or commentary).

3.2 Architecture

3.2.1 Soil Analysis Architectures

In this section, different pipelines are evaluated for extracting values, units, and methods from soil analysis PDFs. The experiments are conducted in two stages. First, the pipelines are tested on small PDFs. Then, the best-performing pipeline is extended and adapted for large PDFs using retrieval methods.

Configuration A: Base Extraction + Full Context Prompting
In the first configuration, the extracted text is used directly as input to the Full Context Prompting approach described in Section 2.8.

Execution Flow

1. The text is extracted from the soil analysis PDF using a PDF extraction library (e.g PdfPlumber, Partition etc.)
2. The extracted text is provided as a whole to the model via Full Context Prompting, alongside a single prompt containing all the requested fields to be extracted.
3. The model outputs the values, units and methods in a structured JSON format.

Configuration B: Base Extraction + Postprocessing + Full Context Prompting
This configuration extends Configuration A by introducing the `Postprocessing Module`, described in Section 3.1.1, before Full-Context Prompting.

Execution Flow

1. The text is extracted from the soil analysis PDF using a PDF extraction library (e.g PdfPlumber, Partition etc.)
2. The text is passed to the `Postprocessing Module` to clean the extracted text (removing noise, fixing OCR issues, normalizing structure).
3. The processed text is provided as a whole to the model via Full Context Prompting alongside a single prompt containing all the requested fields to be extracted.
4. The model outputs the values, units and methods in a structured JSON format.

Configuration C: Base Extraction + Postprocessing + Translation + Full Context Prompting
This configuration extends Configuration B by using the `Translation Module` introduced in Section 3.1.2 in order to translate the cleaned, post-processed text from Greek to English.

Execution Flow

1. The text is extracted from the soil analysis PDF using a PDF extraction library (e.g PdfPlumber, Partition etc.)
2. The text is passed to the Postprocessing module to clean the extracted text (removing noise, fixing OCR issues, normalizing structure).
3. The processed text is translated to English using the `Translation Module`.
4. The processed-translated text is provided as a whole to the model via Full Context Prompting, alongside a single prompt containing all the requested fields to be extracted.
5. The model outputs the values, units and methods in a structured JSON format.

Configuration D: Base Extraction + Postprocessing + Translation + Prompt Expansion + Full Context Prompting
This configuration extends Configuration C by separating the fields into multiple prompts. That way, each prompt contains a subset of the total requested fields, leading to smaller and more targeted prompts.

Execution Flow

1. The text is extracted from the soil analysis PDF using a PDF extraction library (e.g PdfPlumber, Partition etc.)
2. The text is passed to the Postprocessing module to clean the extracted text (removing noise, fixing OCR issues, normalizing structure).
3. The processed text is translated to English using the Translation Module.
4. The processed-translated text is provided as a whole to the model via Full Context Prompting.
5. The requested fields are separated into many prompts leading to many, but smaller, input prompts to the model.
6. The model iterates through all the input prompts (queries) and performs multiple invokes, producing many JSON outputs. Finally, a large JSON is returned, containing all the info from the small ones.

Configuration E: Base Extraction + Postprocessing + Translation + Prompt Expansion + RAG
This configuration adapts Configuration D for large PDFs by replacing Full-Context Prompting, which is context-limited, with RAG, a method described in Section 3.1.4.

Execution Flow

1. The text is extracted from the soil analysis PDF using a PDF extraction library (e.g PdfPlumber, Partition etc.)
2. The text is passed to the Postprocessing module to clean the extracted text (removing noise, fixing OCR issues, normalizing structure).
3. The processed text is translated to English using the Translation Module.
4. The processed-translated text is split into multiple chunks and stored in a vector database.
5. The requested fields are separated into many prompts leading to many but smaller input prompts.
6. For each input prompt, the most relevant chunks are retrieved.
7. The retrieved chunks, alongside the corresponding prompt is passed to the the model's input in order to generate a structured JSON response.
8. This process is conducted for every input prompt. Finally, a large JSON is returned containing all the info from the small ones.

3.2.2 MultiAgent Architectures

In this section, the different pipelines that can be constructed using different configurations of Agents and Retrieval methods are presented.

Configuration A: Field Detection Agent + Prompt Builder Agent + Classic RAG
In the first configuration, the Field Detection Agent (Section 3.1.6) is directly connected with the Prompt Builder Agent (Section 3.1.8) and the Classic RAG approach, introduced in Section 3.1.4.

Execution Flow

1. The input text is chunked using a sentence-based splitter.
2. Each chunk is passed to the **Field Detection Agent**, which outputs a JSON object containing a list of detected `fields`.
3. The **Prompt Builder Agent** receives this as input and produces a single prompt for each 5 fields.
4. The Prompt Builder Agent returns structured prompts that include:
   - Domain-specific extraction instructions
   - A JSON schema requiring `value`, `unit`, and `method` for each field
   - Strict output formatting rules (e.g., return only JSON, no commentary)
5. The final prompt list is stored per chunk and per group and are passed to the Classic RAG, alongside the document's text in order to retrieve the data.

Configuration B: Field Detection Agent + Prompt Builder Agent + Chunk Level Retrieval
The following configuration extends Configuration A by introducing Chunk-Level Retrieval, as presented in Section 3.1.5, instead of Classic RAG.

Execution Flow

1. The input text is chunked using a sentence-based splitter.
2. Each chunk is passed to the **Field Detection Agent**, which outputs a JSON object containing a list of detected `fields`.
3. The **Prompt Builder Agent** receives this as input and produces a single prompt for each subset of 5 fields.
4. The Prompt Builder Agent returns structured prompts that include:
   - Domain-specific extraction instructions
   - A JSON schema requiring `value`, `unit`, and `method` for each field
   - Strict output formatting rules (e.g., return only JSON, no commentary)
5. The final prompt list is stored per chunk and per group and is passed to the Chunk Level Retrieval method, alongside the document's chunks in order to retrieve the data.

Configuration C: Field Detection Agent + Field Postprocessing Agent + Prompt Builder Agent + Classic RAG
In this setup, an additional Field Postprocessing Agent is inserted between the Field Detection and Prompt Builder agents. The output of the Prompt Builder Agent is been directed to the Classic RAG for retrieval purposes.

Execution Flow

1. The input text is chunked using a sentence-based splitter.
2. Each chunk is passed to the **Field Detection Agent**, which outputs a JSON object containing the `domain` and a list of detected `fields`.
3. The field list is passed to the **Field Postprocessing Agent**, which removes fields that are:
   - Semantically duplicated
   - Not present or implied in the text
   - Too complex for reliable extraction
4. The **Prompt Builder Agent** receives the new list as input and produces a single prompt for each subset of 5 fields.
5. The Prompt Builder Agent returns structured prompts that include:
   - Domain-specific extraction instructions
   - A JSON schema requiring `value`, `unit`, and `method` for each field
   - Strict output formatting rules (e.g., return only JSON, no commentary)
6. The final prompt list is stored per chunk and per group and is passed to the Classic RAG, alongside the document's text in order to retrieve the data.

Configuration D: Field Detection Agent + Field Postprocessing Agent + Prompt Builder Agent + Chunk Level Retrieval
This configuration mirrors Configuration C, with the Field Postprocessing Agent placed between the Field Detection and Prompt Builder agents. The only difference lies in the retrieval stage, where Chunk-Level Retrieval is applied instead of Classic RAG.

Execution Flow

1. The input text is chunked using a sentence-based splitter.
2. Each chunk is passed to the **Field Detection Agent**, which outputs a JSON object containing the `domain` and a list of detected `fields`.
3. The field list is passed to the **Field Postprocessing Agent**, which removes fields that are:
   - Semantically duplicated
   - Not present or implied in the text
   - Too complex for reliable extraction

4. The **Prompt Builder Agent** receives the new list as input and produces a single prompt for each subset of 5 fields.

5. The Prompt Builder Agent returns structured prompts that include:

    - Domain-specific extraction instructions
    - A JSON schema requiring `value`, `unit`, and `method` for each field
    - Strict output formatting rules (e.g., return only JSON, no commentary)

6. The final prompt list is stored per chunk and per group and are passed to the Chunk Level Retrieval method, alongside the document's chunks in order to retrieve the data.

4 Experiments

4.1 Setup

4.1.1 Overview

This section presents the experimental evaluation of the proposed pipeline, which is divided into two main parts. The **first part** focuses on foundational experiments conducted on soil analysis documents regarding text extraction, post-processing and cleanup, translation, prompting strategies and multiple retrieval methods.

The **second part** introduces a more advanced, agent-based system for field extraction, incorporating modular agents that operate sequentially to detect, refine, and construct prompts for information retrieval. Each stage of the pipeline is evaluated separately, using structured ground truth data, and final extraction accuracy is measured across several criteria.

4.1.2 Evaluation Criteria

This section outlines the evaluation metrics and criteria applied to each individual component of the system. The goal is to assess performance both on individual components and to the end-to-end pipeline.

Text Extraction and Postprocessing

To assess the quality of both raw extracted and preprocessed text from PDF files, the following criteria were used:

- **Preservation of content:** Whether all key textual elements were extracted without omissions.
- **Line and word breaks:** Frequency of word merging.
- **Structure awareness:** Whether tables or lists were preserved or broken.
- **Manual inspection:** Human review to assess naturalness and fidelity.

Translation

To evaluate translation quality from Greek to English:

- **Terminology preservation:** Accuracy of scientific and domain-specific terms.
- **Formatting consistency:** Whether the translated output retains Markdown, tables, and structure.
- **Fluency and correctness:** Manual judgment based on language quality.
- **BLEU Score:** Automated evaluation metric used to compare the translations with manually curated reference translations. Note: While BLEU provides a general measure of translation similarity, it does not account for formatting, structure preservation, or terminology correctness, which are critical in technical domains, like soil analysis.

- **BERT Score:** An evaluation metric that measures semantic similarity between the system output and reference translations using contextual embeddings from pretrained language models (e.g., BERT). Unlike BLEU, which relies on exact n-gram overlap, BERTScore captures meaning, even when different wording is used. However, it has limitations: its performance depends on the quality and language coverage of the underlying embeddings, it may overestimate adequacy when outputs are fluent, but partially incorrect.

- **Embedding Score:** The embedding-based similarity score measures the semantic closeness between the input and the reference translations by comparing their vector representations. Both candidate and reference texts are encoded into dense embeddings and the similarity is calculated with cosine similarity. Unlike BLEU, which relies on surface n-gram overlap, and BERTScore, which aligns tokens, this metric operates at the sentence level and captures meaning, even when different wording is used. However, it is sensitive to the choice of embedding model and may overlook finer structural or formatting differences. The model *paraphrase-multilingual-MiniLM* [38, 39] was used for that purpose.

Field Extraction

Field extraction is evaluated against ground-truth alias lists of fields that must be extracted from a specific document. This list is manually constructed for each tested PDF.

Retrieval Methods

Each retrieval variant (full context, chunk-level, vector-based RAG) is evaluated by:

- **Extraction accuracy:** Measures the overall correctness of the extraction process. It includes (i) the correctness of retrieved values, units, and methods per field, (ii) the coverage of ground-truth fields successfully extracted, and (iii) the consistency of preserving the requested structured output format (e.g., valid JSON) without errors or extra text.

- **Efficiency:** Measured in terms of execution time.

For that purpose, for each tested PDF, an individual prototype JSON file has been created that contains the fields that need to be extracted together with their value, unit, method. The same json is used for the evaluation of the Field Extraction. To account for natural variation in field names, values, units and methods, each target entry in the prototype JSON is accompanied by a list of acceptable aliases and formats — for example, fields, like `pH`, `pHValue`, or `pHAmount` are considered equivalent, as are numeric variations, like `0`, `0.0`, or `0.00`, and name-related fields, such as `Name` or `Surname`.

End-to-End Pipeline

The full system is also evaluated on:

- **Extraction accuracy:** Combines completeness and correctness of the final JSON output. It accounts for (i) the number of correct fields matched against the full ground truth, and (ii) the aggregated accuracy of values, units, and methods.

- **Efficiency:** Measured, again, in terms of execution time. This includes the time from the text extraction until the produced output in json. These metrics help evaluate the scalability and deployability of each method under real-world conditions.

5.1.3 Input Documents

To support the evaluation of each pipeline component, two sets of input PDFs were used. These datasets are split according to the experimental parts:

- **Part I:** Soil analysis reports in Greek, scanned and digital.
- **Part II:** Cross-domain PDFs, including invoices, lab sheets, and soil reports.

Each set contains diverse document layouts and data densities, enabling robust evaluation. Screenshots and document-specific notes are provided in the corresponding experiment sections.

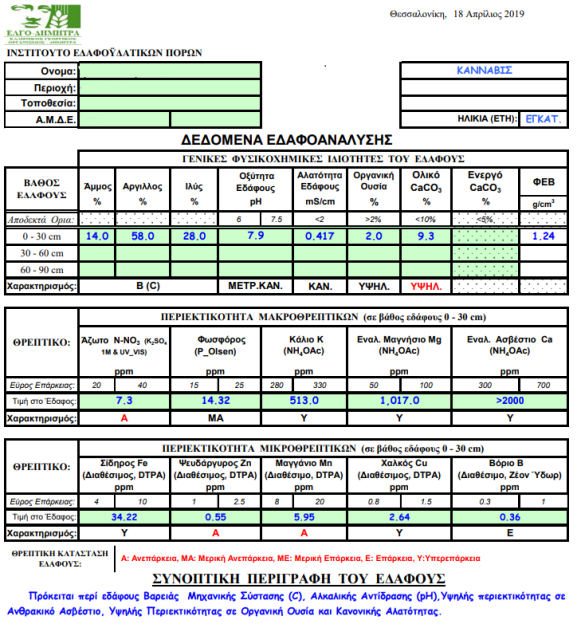
4.2 Part I: Foundational Experiments on Soil Analysis Reports

4.2.1 Dataset Description

The following PDFs were used in Part I of the experiments. All files are Greek soil analysis reports, featuring scientific terminology, tables with numeric measurements, and varying degrees of formatting complexity. To assess system performance across different input lengths, the dataset is divided into:

- **Small Reports:** 10 files, both 1 page PDFs and single page snippets from large ones used to test the system in small input.
- **Large Reports:** 10 files of 3–5 pages, used for testing scalability and accuracy in large inputs.

Note: Some small reports were cropped or extracted as single-page segments from the larger documents.



Θεσσαλονίκη, 18 Απρίλιος 2019

ΙΝΣΤΙΤΟΥΤΟ ΕΔΑΦΟΫΔΑΤΙΚΩΝ ΠΟΡΩΝ

Ονομα:	
Περιοχή:	
Τοποθεσία:	
Α.Μ.Δ.Ε.	

ΚΑΝΝΑΒΙΣ

ΗΛΙΚΙΑ (ΕΤΗ): ΕΓΚΑΤ.

ΔΕΔΟΜΕΝΑ ΕΔΑΦΟΑΝΑΛΥΣΗΣ

ΓΕΝΙΚΕΣ ΦΥΣΙΚΟΧΗΜΙΚΕΣ ΙΔΙΟΤΗΤΕΣ ΤΟΥ ΕΔΑΦΟΥΣ

ΒΑΘΟΣ ΕΔΑΦΟΥΣ	Άμμος %	Αργιλλος %	Ιλύς %	Οξύτητα Εδάφους pH	Αλατότητα Εδάφους mS/cm	Οργανική Ουσία %	Ολικό $CaCO_3$ %	Ενεργό $CaCO_3$ %	ΦΕΒ g/cm³
Αποδεκτά Όρια:				6 7.5	<2	>2%	<10%	<5%	
0 - 30 cm	14.0	58.0	28.0	7.9	0.417	2.0	9.3		1.24
30 - 60 cm									
60 - 90 cm									
Χαρακτηρισμός:		B (C)		ΜΕΤΡ.ΚΑΝ.	ΚΑΝ.	ΥΨΗΛ.	ΥΨΗΛ.		

ΠΕΡΙΕΚΤΙΚΟΤΗΤΑ ΜΑΚΡΟΘΡΕΠΤΙΚΩΝ (σε βάθος εδάφους 0 - 30 cm)

ΘΡΕΠΤΙΚΟ:	Άζωτο N-NO_3 (K$_2$SO$_4$ 1M & UV_VIS) ppm		Φωσφόρος (P_Olsen) ppm		Κάλιο K (NH_4OAc) ppm		Εναλ. Μαγνήσιο Mg (NH_4OAc) ppm		Εναλ. Ασβέστιο Ca (NH_4OAc) ppm	
Εύρος Επάρκειας:	20	40	15	25	280	330	50	100	300	700
Τιμή στο Έδαφος:	7.3		14.32		513.0		1,017.0		>2000	
Χαρακτηρισμός:	A		MA		Y		Y		Y	

ΠΕΡΙΕΚΤΙΚΟΤΗΤΑ ΜΙΚΡΟΘΡΕΠΤΙΚΩΝ (σε βάθος εδάφους 0 - 30 cm)

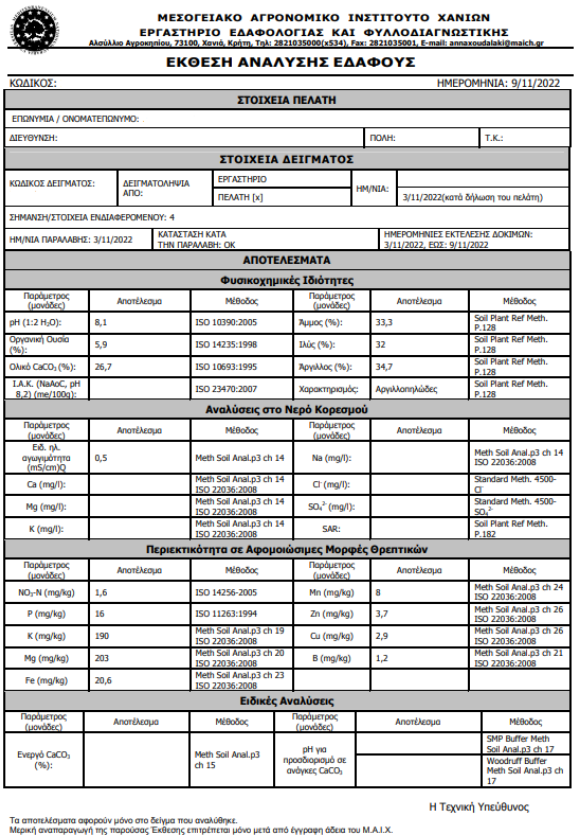
ΘΡΕΠΤΙΚΟ:	Σίδηρος Fe (Διαθέσιμος, DTPA) ppm		Ψευδάργυρος Zn (Διαθέσιμος, DTPA) ppm		Μαγγάνιο Mn (Διαθέσιμο, DTPA) ppm		Χαλκός Cu (Διαθέσιμος, DTPA) ppm		Βόριο B (Διαθέσιμο, Ζέον Ύδωρ) ppm	
Εύρος Επάρκειας:	4	10	1	2.5	8	20	0.8	1.5	0.3	1
Τιμή στο Έδαφος:	34.22		0.55		5.95		2.64		0.36	
Χαρακτηρισμός:	Y		A		A		Y		E	

ΘΡΕΠΤΙΚΗ ΚΑΤΑΣΤΑΣΗ ΕΔΑΦΟΥΣ: Α: Ανεπάρκεια, ΜΑ: Μερική Ανεπάρκεια, ΜΕ: Μερική Επάρκεια, Ε: Επάρκεια, Υ:Υπερεπάρκεια

ΣΥΝΟΠΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΕΔΑΦΟΥΣ

Πρόκειται περί εδάφους Βαρειάς Μηχανικής Σύστασης (C), Αλκαλικής Αντίδρασης (pH),Υψηλής περιεκτικότητας σε Ανθρακικό Ασβέστιο, Υψηλής Περιεκτικότητας σε Οργανική Ουσία και Κανονικής Αλατότητας.

(a) First PDF.



ΜΕΣΟΓΕΙΑΚΟ ΑΓΡΟΝΟΜΙΚΟ ΙΝΣΤΙΤΟΥΤΟ ΧΑΝΙΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΔΑΦΟΛΟΓΙΑΣ ΚΑΙ ΦΥΛΛΟΔΙΑΓΝΩΣΤΙΚΗΣ
Αλεξάλλου Αγροκηπίου, 73100, Χανιά, Κρήτη, Τηλ: 2821035000(κ534), Fax: 2821035001, E-mail: annaxoudalaki@maich.gr

ΕΚΘΕΣΗ ΑΝΑΛΥΣΗΣ ΕΔΑΦΟΥΣ

ΚΩΔΙΚΟΣ: ΗΜΕΡΟΜΗΝΙΑ: 9/11/2022

ΣΤΟΙΧΕΙΑ ΠΕΛΑΤΗ

ΕΠΩΝΥΜΙΑ / ΟΝΟΜΑΤΕΠΩΝΥΜΟ:

ΔΙΕΥΘΥΝΣΗ: ΠΟΛΗ: Τ.Κ.:

ΣΤΟΙΧΕΙΑ ΔΕΙΓΜΑΤΟΣ

ΚΩΔΙΚΟΣ ΔΕΙΓΜΑΤΟΣ:	ΔΕΙΓΜΑΤΟΛΗΨΙΑ ΑΠΟ:	ΕΡΓΑΣΤΗΡΙΟ / ΠΕΛΑΤΗ [x]	ΗΜ/ΝΙΑ:	3/11/2022(κατά δήλωση του πελάτη)

ΣΗΜΑΝΣΗ/ΣΤΟΙΧΕΙΑ ΕΝΔΙΑΦΕΡΟΜΕΝΟΥ: 4

ΗΜ/ΝΙΑ ΠΑΡΑΛΑΒΗΣ: 3/11/2022	ΚΑΤΑΣΤΑΣΗ ΚΑΤΑ ΤΗΝ ΠΑΡΑΛΑΒΗ: ΟΚ	ΗΜΕΡΟΜΗΝΙΕΣ ΕΚΤΕΛΕΣΗΣ ΔΟΚΙΜΩΝ: 3/11/2022, ΕΩΣ: 9/11/2022

ΑΠΟΤΕΛΕΣΜΑΤΑ

Φυσικοχημικές Ιδιότητες

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
pH (1:2 H_2O):	8,1	ISO 10390:2005	Άμμος (%):	33,3	Soil Plant Ref Meth. P.128
Οργανική Ουσία (%):	5,9	ISO 14235:1998	Ιλύς (%):	32	Soil Plant Ref Meth. P.128
Ολικό $CaCO_3$ (%):	26,7	ISO 10693:1995	Άργιλλος (%):	34,7	Soil Plant Ref Meth. P.128
Ι.Α.Κ. (NaAoC, pH 8,2) (me/100g):		ISO 23470:2007	Χαρακτηρισμός:	Αργιλλοπηλώδες	Soil Plant Ref Meth. P.128

Αναλύσεις στο Νερό Κορεσμού

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
Ειδ. ηλ. αγωγιμότητα (mS/cm)Q	0,5	Meth Soil Anal.p3 ch 14	Na (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008
Ca (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008	Cl (mg/l):		Standard Meth. 4500-Cl
Mg (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008	SO_4^{2-} (mg/l):		Standard Meth. 4500-SO_4^{2-}
K (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008	SAR:		Soil Plant Ref Meth. P.182

Περιεκτικότητα σε Αφομοιώσιμες Μορφές Θρεπτικών

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
NO_3-N (mg/kg)	1,6	ISO 14256-2005	Mn (mg/kg)	8	Meth Soil Anal.p3 ch 24 ISO 22036:2008
P (mg/kg)	16	ISO 11263:1994	Zn (mg/kg)	3,7	Meth Soil Anal.p3 ch 26 ISO 22036:2008
K (mg/kg)	190	Meth Soil Anal.p3 ch 19 ISO 22036:2008	Cu (mg/kg)	2,9	Meth Soil Anal.p3 ch 26 ISO 22036:2008
Mg (mg/kg)	203	Meth Soil Anal.p3 ch 20 ISO 22036:2008	B (mg/kg)	1,2	Meth Soil Anal.p3 ch 21 ISO 22036:2008
Fe (mg/kg)	20,6	Meth Soil Anal.p3 ch 23 ISO 22036:2008			

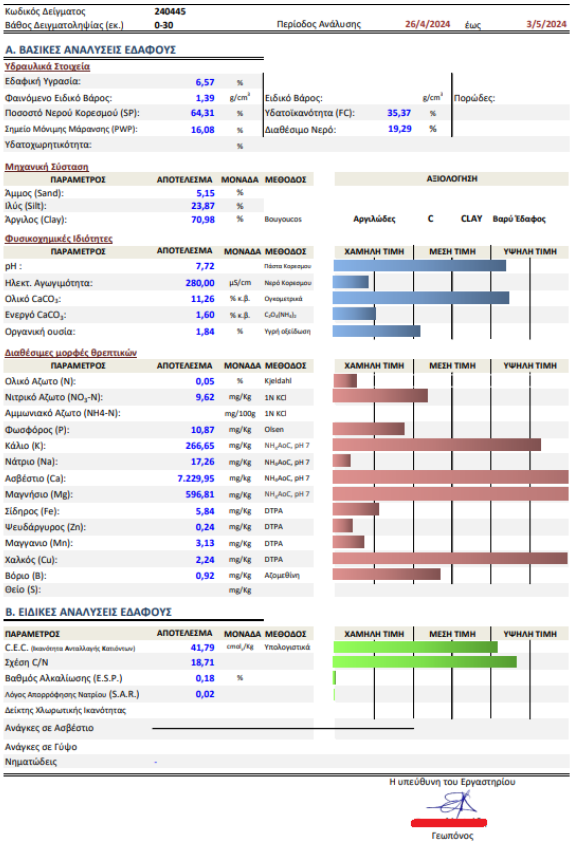
Ειδικές Αναλύσεις

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
Ενεργό $CaCO_3$ (%):		Meth Soil Anal.p3 ch 15	pH για προσδιορισμό σε ανάγκες $CaCO_3$		SMP Buffer Meth Soil Anal.p3 ch 17
					Woodruff Buffer Meth Soil Anal.p3 ch 17

Η Τεχνική Υπεύθυνος

Τα αποτελέσματα αφορούν μόνο στο δείγμα που αναλύθηκε.
Μερική αναπαραγωγή της παρούσας Έκθεσης επιτρέπεται μόνο μετά από έγγραφη άδεια του Μ.Α.Ι.Χ.

(b) Second PDF.



Κωδικός Δείγματος	240445		
Βάθος Δειγματοληψίας (εκ.)	0-30	Περίοδος Ανάλυσης	26/4/2024 έως 3/5/2024

Α. ΒΑΣΙΚΕΣ ΑΝΑΛΥΣΕΙΣ ΕΔΑΦΟΥΣ

Υδραυλικά Στοιχεία

Εδαφική Υγρασία:	6,57	%			
Φαινόμενο Ειδικό Βάρος:	1,39	g/cm³	Ειδικό Βάρος:		g/cm³ Πορώδες:
Ποσοστό Νερού Κορεσμού (SP):	64,31	%	Υδατοϊκανότητα (FC):	35,37	%
Σημείο Μόνιμης Μάρανσης (PWP):	16,08	%	Διαθέσιμο Νερό:	19,29	%
Υδατοχωρητικότητα:		%			

Μηχανική Σύσταση

ΠΑΡΑΜΕΤΡΟΣ	ΑΠΟΤΕΛΕΣΜΑ	ΜΟΝΑΔΑ	ΜΕΘΟΔΟΣ
Άμμος (Sand):	5,15	%	
Ιλύς (Silt):	23,87	%	
Άργιλος (Clay):	70,98	%	Bouyoucos

ΑΞΙΟΛΟΓΗΣΗ: Αργιλώδες C CLAY Βαρύ Έδαφος

Φυσικοχημικές Ιδιότητες

ΠΑΡΑΜΕΤΡΟΣ	ΑΠΟΤΕΛΕΣΜΑ	ΜΟΝΑΔΑ	ΜΕΘΟΔΟΣ	ΧΑΜΗΛΗ ΤΙΜΗ	ΜΕΣΗ ΤΙΜΗ	ΥΨΗΛΗ ΤΙΜΗ
pH :	7,72		Πάστα Κορεσμού			
Ηλεκτ. Αγωγιμότητα:	280,00	μS/cm	Νερό Κορεσμού			
Ολικό $CaCO_3$:	11,26	% κ.β.	Ογκομετρικά			
Ενεργό $CaCO_3$:	1,60	% κ.β.	$C_2O_4(NH_4)_2$			
Οργανική ουσία:	1,84	%	Υγρή οξείδωση			

Διαθέσιμες μορφές θρεπτικών

ΠΑΡΑΜΕΤΡΟΣ	ΑΠΟΤΕΛΕΣΜΑ	ΜΟΝΑΔΑ	ΜΕΘΟΔΟΣ	ΧΑΜΗΛΗ ΤΙΜΗ	ΜΕΣΗ ΤΙΜΗ	ΥΨΗΛΗ ΤΙΜΗ
Ολικό Αζωτο (N):	0,05	%	Kjeldahl			
Νιτρικό Αζωτο (NO_3-N):	9,62	mg/Kg	1N KCl			
Αμμωνιακό Αζωτο (NH4-N):		mg/100g	1N KCl			
Φωσφόρος (P):	10,87	mg/Kg	Olsen			
Κάλιο (K):	266,65	mg/Kg	NH_4AoC, pH 7			
Νάτριο (Na):	17,26	mg/Kg	NH_4AoC, pH 7			
Ασβέστιο (Ca):	7.229,95	mg/Kg	NH_4AoC, pH 7			
Μαγνήσιο (Mg):	596,81	mg/Kg	NH_4AoC, pH 7			
Σίδηρος (Fe):	5,84	mg/Kg	DTPA			
Ψευδάργυρος (Zn):	0,24	mg/Kg	DTPA			
Μαγγανιο (Mn):	3,13	mg/Kg	DTPA			
Χαλκός (Cu):	2,24	mg/Kg	DTPA			
Βόριο (B):	0,92	mg/Kg	Αζομεθίνη			
Θείο (S):		mg/Kg				

Β. ΕΙΔΙΚΕΣ ΑΝΑΛΥΣΕΙΣ ΕΔΑΦΟΥΣ

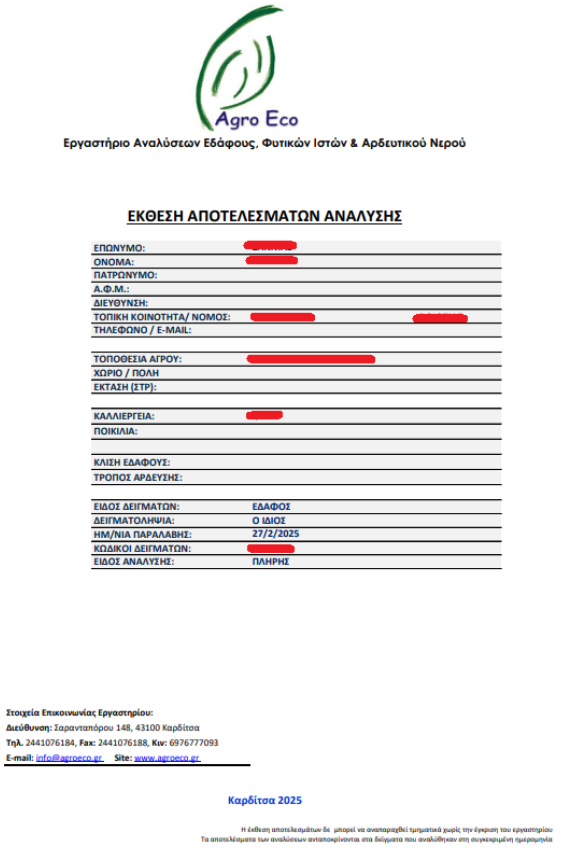
ΠΑΡΑΜΕΤΡΟΣ	ΑΠΟΤΕΛΕΣΜΑ	ΜΟΝΑΔΑ	ΜΕΘΟΔΟΣ	ΧΑΜΗΛΗ ΤΙΜΗ	ΜΕΣΗ ΤΙΜΗ	ΥΨΗΛΗ ΤΙΜΗ
C.E.C. (Ικανότητα Ανταλλαγής Κατιόντων)	41,79	cmol_/Kg	Υπολογιστικά			
Σχέση C/N	18,71					
Βαθμός Αλκαλίωσης (E.S.P.)	0,18	%				
Λόγος Απορρόφησης Νατρίου (S.A.R.)	0,02					
Δείκτης Κλωριωτικής Ικανότητας						
Ανάγκες σε Ασβέστιο						
Ανάγκες σε Γύψο						
Νηματώδεις	-					

Η υπεύθυνη του Εργαστηρίου

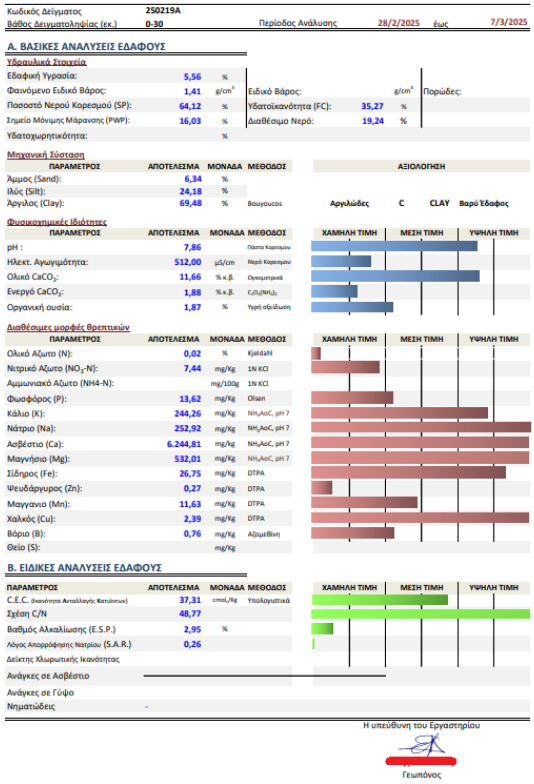
Γεωπόνος

(c) Third PDF.

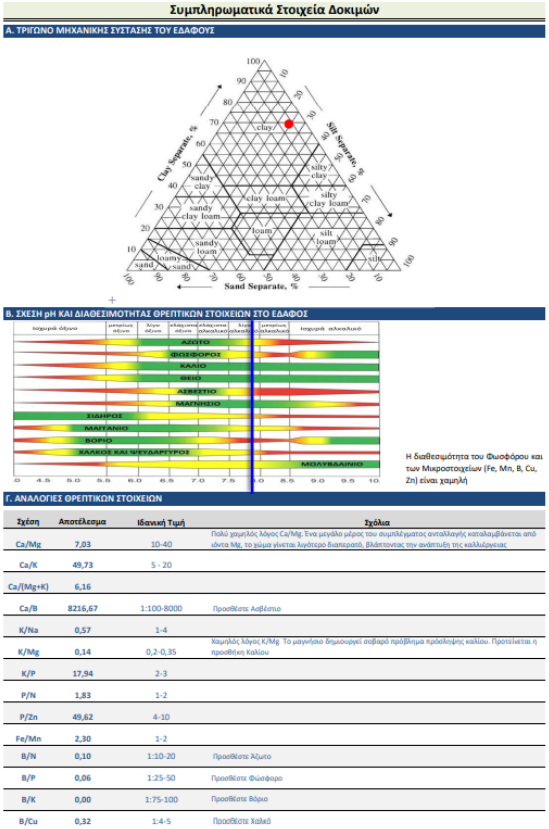
Figure 2: Examples of Small Input PDFs



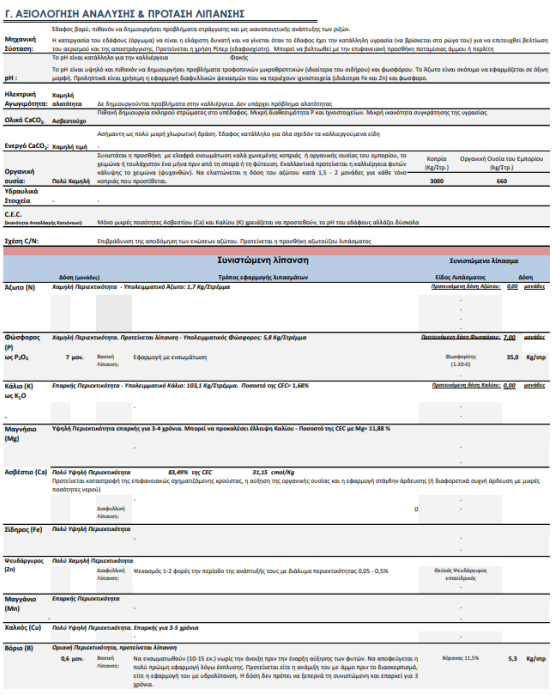
(a) First Page.



(b) Second Page.

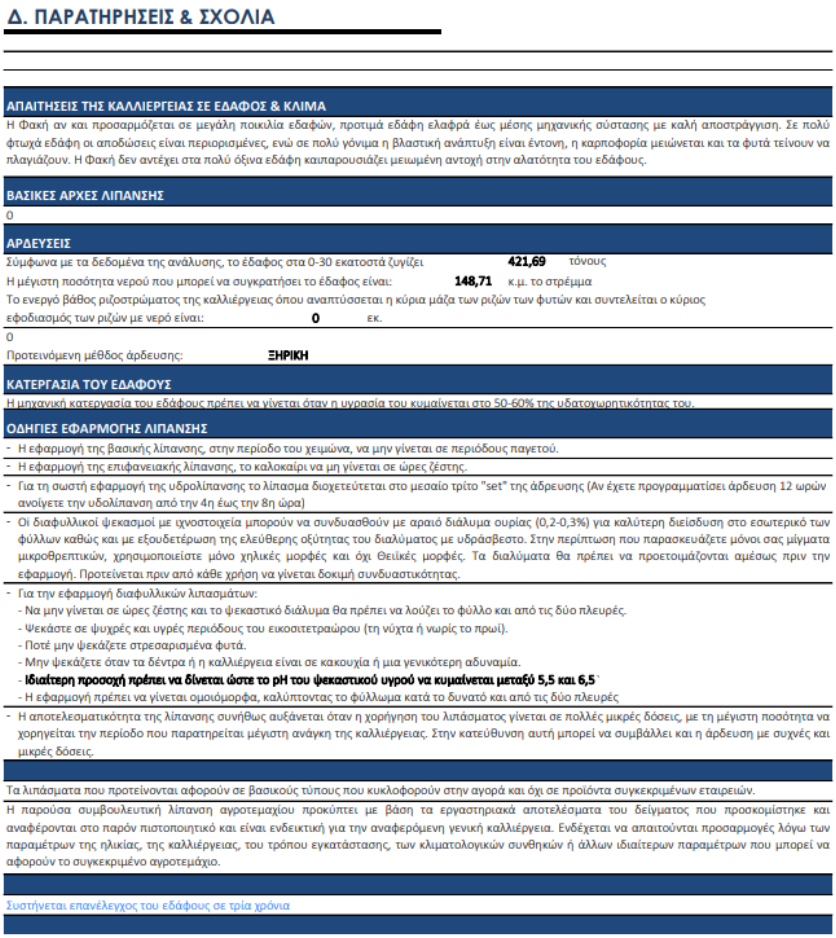


(c) Third Page.



(d) Fourth Page.

Figure 3: First example of a large input PDF spanning multiple pages.



Δ. ΠΑΡΑΤΗΡΗΣΕΙΣ & ΣΧΟΛΙΑ

ΑΠΑΙΤΗΣΕΙΣ ΤΗΣ ΚΑΛΛΙΕΡΓΕΙΑΣ ΣΕ ΕΔΑΦΟΣ & ΚΛΙΜΑ

Η Φακή αν και προσαρμόζεται σε μεγάλη ποικιλία εδαφών, προτιμά εδάφη ελαφρά έως μέσης μηχανικής σύστασης με καλή αποστράγγιση. Σε πολύ φτωχά εδάφη οι αποδώσεις είναι περιορισμένες, ενώ σε πολύ γόνιμα η βλαστική ανάπτυξη είναι έντονη, η καρποφορία μειώνεται και τα φυτά τείνουν να πλαγιάζουν. Η Φακή δεν αντέχει στα πολύ όξινα εδάφη καιπαρουσιάζει μειωμένη αντοχή στην αλατότητα του εδάφους.

ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΛΙΠΑΝΣΗΣ

0

ΑΡΔΕΥΣΕΙΣ

Σύμφωνα με τα δεδομένα της ανάλυσης, το έδαφος στα 0-30 εκατοστά ζυγίζει **421,69** τόνους

Η μέγιστη ποσότητα νερού που μπορεί να συγκρατήσει το έδαφος είναι: **148,71** κ.μ. το στρέμμα

Το ενεργό βάθος ριζοστρώματος της καλλιέργειας όπου αναπτύσσεται η κύρια μάζα των ριζών των φυτών και συντελείται ο κύριος εφοδιασμός των ριζών με νερό είναι: **0** εκ.

0

Προτεινόμενη μέθοδος άρδευσης: **ΞΗΡΙΚΗ**

ΚΑΤΕΡΓΑΣΙΑ ΤΟΥ ΕΔΑΦΟΥΣ

Η μηχανική κατεργασία του εδάφους πρέπει να γίνεται όταν η υγρασία του κυμαίνεται στο 50-60% της υδατοχωρητικότητας του.

ΟΔΗΓΙΕΣ ΕΦΑΡΜΟΓΗΣ ΛΙΠΑΝΣΗΣ

- Η εφαρμογή της βασικής λίπανσης, στην περίοδο του χειμώνα, να μην γίνεται σε περιόδους παγετού.
- Η εφαρμογή της επιφανειακής λίπανσης, το καλοκαίρι να μη γίνεται σε ώρες ζέστης.
- Για τη σωστή εφαρμογή της υδρολίπανσης το λίπασμα διοχετεύεται στο μεσαίο τρίτο "set" της άδρευσης (Αν έχετε προγραμματίσει άρδευση 12 ωρών ανοίγετε την υδολίπανση από την 4η έως την 8η ώρα)
- Οι διαφυλλικοί ψεκασμοί με ιχνοστοιχεία μπορούν να συνδυασθούν με αραιό διάλυμα ουρίας (0,2-0,3%) για καλύτερη διείσδυση στο εσωτερικό των φύλλων καθώς και με εξουδετέρωση της ελεύθερης οξύτητας του διαλύματος με υδράσβεστο. Στην περίπτωση που παρασκευάζετε μόνοι σας μίγματα μικροθρεπτικών, χρησιμοποιείστε μόνο χηλικές μορφές και όχι Θειϊκές μορφές. Τα διαλύματα θα πρέπει να προετοιμάζονται αμέσως πριν την εφαρμογή. Προτείνεται πριν από κάθε χρήση να γίνεται δοκιμή συνδυαστικότητας.
- Για την εφαρμογή διαφυλλικών λιπασμάτων:
  - Να μην γίνεται σε ώρες ζέστης και το ψεκαστικό διάλυμα θα πρέπει να λούζει το φύλλο και από τις δύο πλευρές.
  - Ψεκάστε σε ψυχρές και υγρές περιόδους του εικοσιτετραώρου (τη νύχτα ή νωρίς το πρωί).
  - Ποτέ μην ψεκάζετε στρεσαρισμένα φυτά.
  - Μην ψεκάζετε όταν τα δέντρα ή η καλλιέργεια είναι σε κακουχία ή μια γενικότερη αδυναμία.
  - **Ιδιαίτερη προσοχή πρέπει να δίνεται ώστε το pH του ψεκαστικού υγρού να κυμαίνεται μεταξύ 5,5 και 6,5**
  - Η εφαρμογή πρέπει να γίνεται ομοιόμορφα, καλύπτοντας το φύλλωμα κατά το δυνατό και από τις δύο πλευρές
- Η αποτελεσματικότητα της λίπανσης συνήθως αυξάνεται όταν η χορήγηση του λιπάσματος γίνεται σε πολλές μικρές δόσεις, με τη μέγιστη ποσότητα να χορηγείται την περίοδο που παρατηρείται μέγιστη ανάγκη της καλλιέργειας. Στην κατεύθυνση αυτή μπορεί να συμβάλλει και η άρδευση με συχνές και μικρές δόσεις.

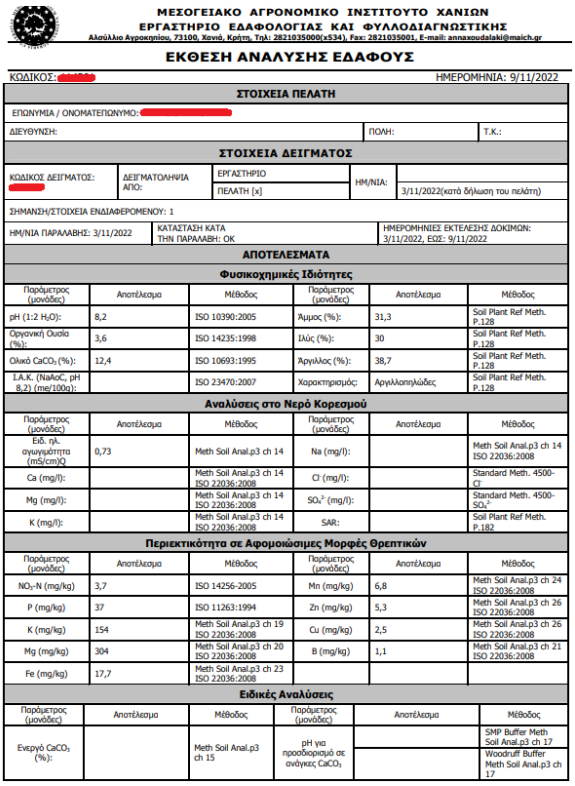
Τα λιπάσματα που προτείνονται αφορούν σε βασικούς τύπους που κυκλοφορούν στην αγορά και όχι σε προϊόντα συγκεκριμένων εταιρειών.

Η παρούσα συμβουλευτική λίπανση αγροτεμαχίου προκύπτει με βάση τα εργαστηριακά αποτελέσματα του δείγματος που προσκομίστηκε και αναφέρονται στο παρόν πιστοποιητικό και είναι ενδεικτική για την αναφερόμενη γενική καλλιέργεια. Ενδέχεται να απαιτούνται προσαρμογές λόγω των παραμέτρων της ηλικίας, της καλλιέργειας, του τρόπου εγκατάστασης, των κλιματολογικών συνθηκών ή άλλων ιδιαίτερων παραμέτρων που μπορεί να αφορούν το συγκεκριμένο αγροτεμάχιο.

Συστήνεται επανέλεγχος του εδάφους σε τρία χρόνια

(e) Fifth Page.

Figure 3: (continued) First example of a large input PDF spanning multiple pages.



ΜΕΣΟΓΕΙΑΚΟ ΑΓΡΟΝΟΜΙΚΟ ΙΝΣΤΙΤΟΥΤΟ ΧΑΝΙΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΔΑΦΟΛΟΓΙΑΣ ΚΑΙ ΦΥΛΛΟΔΙΑΓΝΩΣΤΙΚΗΣ
Αλσύλλιο Αγροκηπίου, 73100, Χανιά, Κρήτη, Τηλ: 2821035000(κ534), Fax: 2821035001, E-mail: annaxoudalaki@maich.gr

ΕΚΘΕΣΗ ΑΝΑΛΥΣΗΣ ΕΔΑΦΟΥΣ

ΚΩΔΙΚΟΣ: [redacted] ΗΜΕΡΟΜΗΝΙΑ: 9/11/2022

ΣΤΟΙΧΕΙΑ ΠΕΛΑΤΗ

ΕΠΩΝΥΜΙΑ / ΟΝΟΜΑΤΕΠΩΝΥΜΟ: [redacted]

ΔΙΕΥΘΥΝΣΗ:	ΠΟΛΗ:	Τ.Κ.:

ΣΤΟΙΧΕΙΑ ΔΕΙΓΜΑΤΟΣ

ΚΩΔΙΚΟΣ ΔΕΙΓΜΑΤΟΣ: [redacted]	ΔΕΙΓΜΑΤΟΛΗΨΙΑ ΑΠΟ:	ΕΡΓΑΣΤΗΡΙΟ / ΠΕΛΑΤΗ [x]	ΗΜ/ΝΙΑ:	3/11/2022(κατά δήλωση του πελάτη)

ΣΗΜΑΝΣΗ/ΣΤΟΙΧΕΙΑ ΕΝΔΙΑΦΕΡΟΜΕΝΟΥ: 1

ΗΜ/ΝΙΑ ΠΑΡΑΛΑΒΗΣ: 3/11/2022	ΚΑΤΑΣΤΑΣΗ ΚΑΤΑ ΤΗΝ ΠΑΡΑΛΑΒΗ: ΟΚ	ΗΜΕΡΟΜΗΝΙΕΣ ΕΚΤΕΛΕΣΗΣ ΔΟΚΙΜΩΝ: 3/11/2022, ΕΩΣ: 9/11/2022

ΑΠΟΤΕΛΕΣΜΑΤΑ

Φυσικοχημικές Ιδιότητες

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
pH (1:2 H_2O):	8,2	ISO 10390:2005	Άμμος (%):	31,3	Soil Plant Ref Meth. P.128
Οργανική Ουσία (%):	3,6	ISO 14235:1998	Ιλύς (%):	30	Soil Plant Ref Meth. P.128
Ολικό $CaCO_3$ (%):	12,4	ISO 10693:1995	Άργιλλος (%):	38,7	Soil Plant Ref Meth. P.128
Ι.Α.Κ. (NaAoC, pH 8,2) (me/100g):		ISO 23470:2007	Χαρακτηρισμός:	Αργιλλοπηλώδες	Soil Plant Ref Meth. P.128

Αναλύσεις στο Νερό Κορεσμού

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
Ειδ. ηλ. αγωγιμότητα (mS/cm)Q	0,73	Meth Soil Anal.p3 ch 14	Na (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008
Ca (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008	Cl (mg/l):		Standard Meth. 4500-Cl
Mg (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008	SO_4^{2} (mg/l):		Standard Meth. 4500-SO_4^{2}
K (mg/l):		Meth Soil Anal.p3 ch 14 ISO 22036:2008	SAR:		Soil Plant Ref Meth. P.182

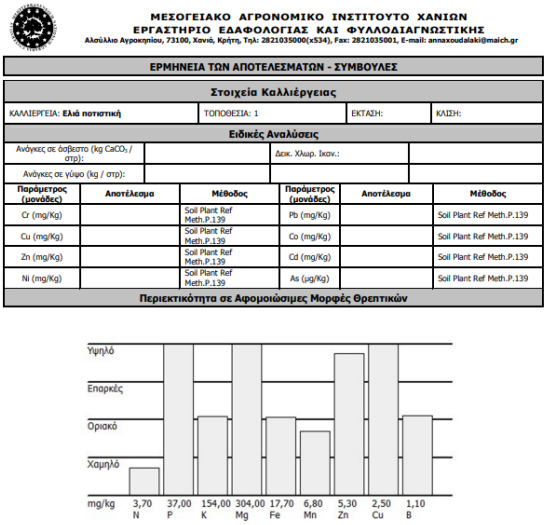
Περιεκτικότητα σε Αφομοιώσιμες Μορφές Θρεπτικών

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
NO_3-N (mg/kg)	3,7	ISO 14256-2005	Mn (mg/kg)	6,8	Meth Soil Anal.p3 ch 24 ISO 22036:2008
P (mg/kg)	37	ISO 11263:1994	Zn (mg/kg)	5,3	Meth Soil Anal.p3 ch 26 ISO 22036:2008
K (mg/kg)	154	Meth Soil Anal.p3 ch 19 ISO 22036:2008	Cu (mg/kg)	2,5	Meth Soil Anal.p3 ch 26 ISO 22036:2008
Mg (mg/kg)	304	Meth Soil Anal.p3 ch 20 ISO 22036:2008	B (mg/kg)	1,1	Meth Soil Anal.p3 ch 21 ISO 22036:2008
Fe (mg/kg)	17,7	Meth Soil Anal.p3 ch 23 ISO 22036:2008			

Ειδικές Αναλύσεις

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
Ενεργό $CaCO_3$ (%):		Meth Soil Anal.p3 ch 15	pH για προσδιορισμό σε ανάγκες $CaCO_3$		SMP Buffer Meth Soil Anal.p3 ch 17
					Woodruff Buffer Meth Soil Anal.p3 ch 17

(a) First Page.



ΜΕΣΟΓΕΙΑΚΟ ΑΓΡΟΝΟΜΙΚΟ ΙΝΣΤΙΤΟΥΤΟ ΧΑΝΙΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΔΑΦΟΛΟΓΙΑΣ ΚΑΙ ΦΥΛΛΟΔΙΑΓΝΩΣΤΙΚΗΣ
Αλσύλλιο Αγροκηπίου, 73100, Χανιά, Κρήτη, Τηλ: 2821035000(κ534), Fax: 2821035001, E-mail: annaxoudalaki@maich.gr

ΕΡΜΗΝΕΙΑ ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ - ΣΥΜΒΟΥΛΕΣ

Στοιχεία Καλλιέργειας

ΚΑΛΛΙΕΡΓΕΙΑ: **Ελιά ποτιστική**	ΤΟΠΟΘΕΣΙΑ: 1	ΕΚΤΑΣΗ:	ΚΑΛΣΗ:

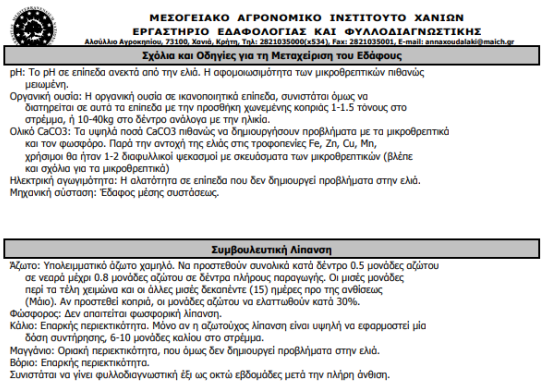
Ειδικές Αναλύσεις

Ανάγκες σε άσβεστο (kg $CaCO_3$ / στρ):		Δεκ. Χλωρ. Ικαν.:	
Ανάγκες σε γύψο (kg / στρ):			

Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος	Παράμετρος (μονάδες)	Αποτέλεσμα	Μέθοδος
Cr (mg/Kg)		Soil Plant Ref Meth.P.139	Pb (mg/Kg)		Soil Plant Ref Meth.P.139
Cu (mg/Kg)		Soil Plant Ref Meth.P.139	Co (mg/Kg)		Soil Plant Ref Meth.P.139
Zn (mg/Kg)		Soil Plant Ref Meth.P.139	Cd (mg/Kg)		Soil Plant Ref Meth.P.139
Ni (mg/Kg)		Soil Plant Ref Meth.P.139	As (μg/Kg)		Soil Plant Ref Meth.P.139

Περιεκτικότητα σε Αφομοιώσιμες Μορφές Θρεπτικών

(b) Second Page.



ΜΕΣΟΓΕΙΑΚΟ ΑΓΡΟΝΟΜΙΚΟ ΙΝΣΤΙΤΟΥΤΟ ΧΑΝΙΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΔΑΦΟΛΟΓΙΑΣ ΚΑΙ ΦΥΛΛΟΔΙΑΓΝΩΣΤΙΚΗΣ
Αλσύλλιο Αγροκηπίου, 73100, Χανιά, Κρήτη, Τηλ: 2821035000(κ534), Fax: 2821035001, E-mail: annaxoudalaki@maich.gr

Σχόλια και Οδηγίες για τη Μεταχείριση του Εδάφους

pH: Το pH σε επίπεδα ανεκτά από την ελιά. Η αφομοιωσιμότητα των μικροθρεπτικών πιθανώς μειωμένη.
Οργανική ουσία: Η οργανική ουσία σε ικανοποιητικά επίπεδα, συνιστάται όμως να διατηρείται σε αυτά τα επίπεδα με την προσθήκη χωνεμένης κοπριάς 1-1.5 τόνους στο στρέμμα, ή 10-40kg στο δέντρο ανάλογα με την ηλικία.
Ολικό CaCO3: Τα υψηλά ποσά CaCO3 πιθανώς να δημιουργήσουν προβλήματα με τα μικροθρεπτικά και τον φωσφόρο. Παρά την αντοχή της ελιάς στις τροφοπενίες Fe, Zn, Cu, Mn, χρήσιμοι θα ήταν 1-2 διοφυλλικοί ψεκασμοί με σκευάσματα των μικροθρεπτικών (βλέπε και σχόλια για τα μικροθρεπτικά)
Ηλεκτρική αγωγιμότητα: Η αλατότητα σε επίπεδα που δεν δημιουργεί προβλήματα στην ελιά.
Μηχανική σύσταση: Έδαφος μέσης συστάσεως.

Συμβουλευτική Λίπανση

Άζωτο: Υπολειμματικό άζωτο χαμηλό. Να προστεθούν συνολικά κατά δέντρο 0.5 μονάδες αζώτου σε νεαρά μέχρι 0.8 μονάδες αζώτου σε δέντρα πλήρους παραγωγής. Οι μισές μονάδες περί το τέλη χειμώνα και οι άλλες μισές δεκαπέντε (15) ημέρες προ της ανθίσεως (Μάιο). Αν προστεθεί κοπριά, οι μονάδες αζώτου να ελαττωθούν κατά 30%.
Φώσφορος: Δεν απαιτείται φωσφορική λίπανση.
Κάλιο: Επαρκής περιεκτικότητα. Μόνο αν η αζωτούχος λίπανση είναι υψηλή να εφαρμοστεί μία δόση συντήρησης, 6-10 μονάδες καλίου στο στρέμμα.
Μαγγάνιο: Ορισκή περιεκτικότητα, που όμως δεν δημιουργεί προβλήματα στην ελιά.
Βόριο: Επαρκής περιεκτικότητα.
Συνιστάται να γίνει φυλλοδιαγνωστική έξι ως οκτώ εβδομάδες μετά την πλήρη άνθιση.

(c) Third Page.

Figure 4: Second Example of Large Input PDF

4.2.2 Text Extraction Tests

The purpose of this subsection is to evaluate the effectiveness of different text extraction methods on the soil analysis reports. These methods are grouped into two categories: **native PDF text extraction** and **OCR-based extraction** for scanned or image-based inputs. Since the quality of text extraction has a direct impact on downstream modules such as postprocessing, translation, and field extraction, a reliable evaluation is essential. Automated similarity metrics (BLEU, BERTScore, and embedding-based scores) were used to provide an objective measure of performance, while manual inspection was also carried out to assess aspects that automated metrics cannot capture, such as table layout and formatting consistency. This combined evaluation ensures a balanced view of the strengths and weaknesses of each extraction method.

Test Categories

- **Native PDF Extraction:** Methods that extract text directly from native PDF files.
- **OCR-based Extraction:** Methods that extract text from scanned PDF documents using OCR.

4.2.2.1 Native PDF Extraction Methods

Methods Evaluated. For native PDF text extraction, the libraries pdfPlumber, Unstructured, PyMuPDF (fitz) and Docling were used and tested.

Experimental Setup.

- **Input:** Native, text-based soil analysis reports in Greek.
- **Procedure:** Each report was processed using all four methods. Outputs were saved as plain text for manual review and evaluation.

Results Table 2 presents the evaluation results of the four native PDF text extraction libraries using BLEU, BERTScore, and embedding-based similarity. While all libraries exhibited similar performance according to the automatic metrics, manual inspection of multiple extracted texts indicated that **pdfPlumber** was the superior extraction library in terms of completeness and layout preservation. For this reason, **pdfPlumber was selected as the default extraction method in the subsequent experiments.**

Library	BLEU Score	BERTScore	Embedding Score
pdfPlumber	0.37	0.57	0.82
Unstructured	0.33	0.57	0.81
PyMuPDF (fitz)	0.38	0.57	0.82
Docling	0.16	0.55	0.82

Table 2: Evaluation of native PDF extraction libraries using BLEU, BERTScore, and embedding-based similarity.

4.2.2.2 OCR-Based Extraction Methods

Methods Evaluated.

- **Pytesseract:** Classical OCR engine built on Tesseract, evaluated on Greek inputs. Handles clean scanned documents, but limited layout awareness.
- **OCRmyPDF + pdfPlumber:** First applies OCRmyPDF to convert scanned PDFs into searchable ones with the help of Tesseract, then uses pdfPlumber for accurate, layout-preserving extraction.

Experimental Setup.

- **Input:** Scanned, image-based soil analysis reports in Greek.
- **Procedure:** Each PDF was processed with both OCR pipelines. The extracted text was saved for manual review and evaluation.

Results. Table 3 presents the evaluation results of the two scanned PDF text extraction libraries using BLEU, BERTScore, and embedding-based similarity. While, again, all libraries exhibited similar performance according to the automatic metrics, manual inspection of multiple extracted texts indicated that OcrMyPdf + pdfPlumber was the best extraction library in terms of completeness and layout preservation. For this reason, this pipeline was selected to handle scanned documents.

Library	BLEU Score	BERTScore	Embedding Score
OcrMyPdf + PdfPlumber	0.18	0.56	0.82
Pytesseract	0.16	0.57	0.83

Table 3: Evaluation of Scanned PDF extraction libraries using BLEU, BERTScore, and embedding-based similarity.

Discussion.

- All three metrics (BLEU, BERTScore, Embedding) gave broadly similar results across the libraries, suggesting no strong differences under purely automatic evaluation.
- Manual inspection revealed that pdfPlumber was superior for native PDFs and that the OCRmyPDF + pdfPlumber pipeline was best for scanned documents, primarily due to better handling of table structures and layout preservation.
- Although functional, OCR-based extraction on scanned documents is limited by the challenges of processing the Greek language and is highly sensitive to the quality of the scans, with noise or low resolution significantly reducing accuracy.

4.2.3 Text Post-Processing Tests

Methods Evaluated. For this experiment, the Postprocessing module described in Section 3.1.1 is being evaluated in cleaning, correcting and formatting the extracted text that the text extraction libraries produce.

Experimental Setup.

- **Input:** The raw text that the text extraction libraries produce.
- **Parameters:** *Chunk Size* = 650, *LLM* = Qwen2.5:32B [32], *LLM Temperature* = 0
- **Procedure:** Each raw text is passed to the PostProcessing module. The output is the processed text.

Outputs and Examples Some visual examples of the texts before and after the post processing are given below:

#	Raw extracted text	Postprocessed output	Time (s)
1	Ηκατεργασίατουεδάφους (όργωμα)ναείναιηελάχιστη δυνατήκαιναγίνεταιόταντο έδαφοςέχειτηνκατάλληλη υγρασία	Μηχανική Η κατεργασία του έδαφους (όργωμα) να είναι η λιγότερο δυνατή και να γίνεται όταν το έδαφος έχει την κατάλληλη υγρασία	8.4
2	Συνιστώνται2-3εφαρμογές στηνκαλλιεργητικήπερίοδο σεδόση30-60γρ.στα100λίτρανερό	Συνιστώνται 2-3 εφαρμογές στην καλλιεργητική περίοδο, με δόση 30-60 γρ. στα 100 λίτρα νερό.	11.4

Table 4: Comparison of raw extracted text and postprocessed output, with emphasis on the text extraction, with corresponding processing times.

#	Raw extracted text	Postprocessed output	Time (s)
1	ΓΕΝΙΚΕΣ ΦΥΣΙΚΟΧΗΜΙΚΕΣ ΙΔΙΟΤΗΤΕΣ ΤΟΥ ΕΔΑΦΟΥΣ Οξύτητα Αλατότητα Οργανική Ολικό Ενεργό ΒΑΘΟΣ ΦΕΒ Άμμος Αργιλλος Ιλύς Εδάφους Εδάφους Ουσία CaCO CaCO 3 3 ΕΔΑΦΟΥΣ % % % pH mS/cm % % % g/cm3 Αποδεκτά Οριa: 6 7.5 <2 >2% <10% <5% 0 - 30 cm 14.0 58.0 28.0 7.9 0.417 2.0 9.3 1.24 30 - 60 cm 60 - 90 cm Χαρακτηρισμός: Β (C) ΜΕΤΡ.ΚΑΝ. ΚΑΝ. ΥΨΗΛ. ΥΨΗΛ.	# ΓΕΝΙΚΕΣ ΦΥΣΙΚΟΧΗΜΙΚΕΣ ΙΔΙΟΤΗΤΕΣ ΤΟΥ ΕΔΑΦΟΥΣ ## Οξύτητα, Αλατότητα, Οργανική Ολικό Ενεργό ΒΑΘΟΣ | ΦΕΒ | Άμμος | Αργιλλος | Ιλύς | Εδάφους | Ουσία CaCO3 | CaCO3 | pH | mS/cm | % Αποδεκτά Οριa: | 6 | 7.5 | <2 | >2% | <10% | <5% | - | - | - | - | - | - 0 - 30 cm | - | 14.0 | 58.0 | 28.0 | 7.9 | 0.417 | 2.0 | 9.3 | 1.24 | - | - | - 30 - 60 cm | - | - | - | - | - | - | - | - | - | - | - | - 60 - 90 cm | - | - | - | - | - | - | - | - | - | - | - | - ## Χαρακτηρισμός Β (C) ΜΕΤΡ.ΚΑΝ. ΚΑΝ. ΥΨΗΛ. ΥΨΗΛ.	16.8
2	pH (1:2 HO): 8,1 ISO 10390:2005 Άμμος (%): 33,3 2 P.128 Οργανική Ουσία Soil Plant Ref Meth. 5,9 ISO 14235:1998 Ιλύς (%): 32 (%): P.128 Soil Plant Ref Meth. Ολικό CaCO (%): 26,7 ISO 10693:1995 Άργιλλος (%): 34,7 3 P.128	| Παράμετρος | Αποτέλεσμα | Μέθοδος | |---|---|---| | pH (1:2 H_2O) | 8,1 | ISO 10390:2005| | Άμμος (%) | 33,3 | | ## Soil Plant Ref Meth. | Παράμετρος | Αποτέλεσμα | Μέθοδος | |---|---|---| | Οργανική Ουσία (%) | 5,9 | ISO 14235:1998 | | Ιλύς (%) | 32 | | ## Soil Plant Ref Meth. | Παράμετρος | Αποτέλεσμα | Μέθοδος | |---|---|---| | Ολικό $CaCO_3$ (%) | 26,7 | ISO 10693:1995 | | Άργιλλος (%) | 34,7 | |	10.4
3	ΠΑΡΑΜΕΤΡΟΣ ΑΠΟΤΕΛΕΣΜΑ ΜΟΝΑΔΑ ΜΕΘΟΔΟΣ ΧΑΜΗΛΗ ΤΙΜΗ ΜΕΣΗ ΤΙΜΗ ΥΨΗΛΗ ΤΙΜΗ C.E.C. (Ικανότητα Ανταλλαγής Κατιόντων) 43,65 cmolc/Kg Υπολογιστικά Σχέση C/N 17,79 Βαθμός Αλκαλίωσης (E.S.P.) 2,58 % Λόγος Απορρόφησης Νατρίου (S.A.R.) 0,25	| Παράμετρος | Result | Unit | Method | |---|---|---|---| | C.E.C. (Ικανότητα Ανταλλαγής Κατιόντων) | 43.65 | cmolc/Kg | Calculated | | Σχέση C/N | 17.79 | | | | Βαθμός Αλκαλίωσης (E.S.P.) | 2.58 | % | | | Λόγος Απορρόφησης Νατρίου (S.A.R.) | 0.25 | | |	18.6

4	ΟΡΕΠΤΙΚΟ: Σίδηρος Fe Ψευδάργυρος Zn Μαγγάνιο Mn Χαλκός Cu Βόριο B (Διαθέσιμος, DTPA) (Διαθέσιμος, DTPA) (Διαθέσιμο, DTPA) (Διαθέσιμος, DTPA) (Διαθέσιμο, Ζέον Υδωρ) ppm ppm ppm ppm ppm Εύρος Επάρκειας: 4 10 1 2.5 8 20 0.8 1.5 0.3 1 Τιμή στο Έδαφος: 34.22 0.55 5.95 2.64 0.36	## ΠΕΡΙΕΚΤΙΚΟΤΗΤΑ ΜΙΚΡΟΘΡΕΠΤΙΚΩΝ (σε βάθος εδάφους 0 - 30 cm) ### ΘΡΕΠΤΙΚΟ: - Σίδηρος Fe - Ψευδάργυρος Zn - Μαγγάνιο Mn - Χαλκός Cu - Βόριο B #### Measurement Methods: - (Διαθέσιμος, DTPA) - (Διαθέσιμος, DTPA) - (Διαθέσιμο, DTPA) - (Διαθέσιμος, DTPA) - (Διαθέσιμο, Ζέον Υδωρ) | ppm | ppm | ppm | ppm | ppm | |-----|-----|-----|-----|-----| | Εύρος Επάρκειας: 4 | 10 | 1 | 2.5 | 8 | | 20 | 0.8 | 1.5 | 0.3 | 1 | ### Soil Values - Τιμή στο Έδαφος: 34.22, 0.55, 5.95, 2.64, 0.36	18.9

Table 5: Comparison of raw extracted text and postprocessed output, with emphasis on the format, with corresponding processing times.

Observations and Analysis.

- As it is clearly observed in the output examples, the PostProcessing module significantly improves the text alignment, readability, word separation and preserves scientific notation and technical expressions. As a result, the text becomes more suitable for future processing by machine learning models and greatly facilitates the accurate extraction of structured information, such as numerical values, units, and field labels.
- Although the correction applied by the module is generally beneficial, there are cases—such as the final example in the table—where the initial extraction is so severely misaligned or inaccurate that even the PostProcessing module cannot fully recover the original structure or meaning.
- Finally, the main disadvantage of this module is the high execution time, since the execution of a -locally deployed- LLM is necessary.

4.2.4 Text Translation Tests

Methods Evaluated. This section evaluates different translation methods. More specifically, the `Translation Module` introduced in Section 3.1.2, the `Google API` and the HuggingFace models `Helsinki-NLP/opus-mt-tc-big-el-en` (Section 2.7.2) and `facebook/m2m100` (Section 2.7.2).

Experimental Setup.

- **Input:** Representative outputs of the PostProcessing module, containing the post processed text.
- **Parameters:** The only tunable parameters for this experiment is *Chunk Size* = 650, *LLM* = Qwen2.5:32B [32], *LLM Temperature* = 0 for the `Translation Module`. More models got tested (ilsp/Llama-Krikri [34, 40]) but due to the weak performance, especially to the preservation of input text layout, their results are not presented.

- **Evaluation:** The evaluation focuses on translation accuracy, preservation of domain-specific terminology, and retention of the original formatting, with reference to established metrics, such as BLEU and BERTScore, and a custom method using embedding similarity to evaluate the translation.
- **Procedure:** Each post-processed text is passed to the input of all 4 different translators. The output is the translated content from Greek to English.

Outputs and Examples Some visual examples of the texts before and after the translation step, for the different translators are given below:

#	Post-Processed Text	Translated Output	Time (s)
1	Το έδαφος στα 0-30 εκατοστά ζυγίζει 421,69 τόνους	The soil in the 0-30 cm layer weighs 421.69 tons	4.5
2	Το ενεργό βάθος ριζοστρώματος της καλλιέργειας όπου αναπτύσσεται η κύρια μάζα των ριζών των φυτών και συντελείται ο κύριος εφοδιασμός των ριζών με νερό είναι: 0 εκ.	The active root zone depth where the main mass of plant roots develops and where the main root watering takes place is: 0 cm.	5.1

Table 6: Translation of simple text from Greek to English using the **Translation Module**.

#	Post-Processed Text	Translated Output	Time (s)
1	# Φυσικοχημικές Ιδιότητες | ΠΑΡΑΜΕΤΡΟΣ | ΑΠΟΤΕΛΕΣΜΑ | ΜΟΝΑΔΑ | ΜΕΘΟΔΟΣ | |---|---|---|---| | pH | 7,86 | | Πάστα Κορεσμού | | Ηλεκτ. Αγωγιμότητα | 512,00 | μS/cm | Νερό Κορεσμού | | Ολικό CaCO3 | 11,66 % | κ.β. | Ογκομετρικά | | Ενεργό CaCO3 | 1,88 % | κ.β. | C2O4(NH4)2 | | Οργανική ουσία | 1,87 % | | Υγρή οξείδωση | ## Διαθέσιμες μορφές θρεπτικών | ΠΑΡΑΜΕΤΡΟΣ | ΑΠΟΤΕΛΕΣΜΑ | ΜΟΝΑΔΑ | ΜΕΘΟΔΟΣ | |---|---|---|---| | Ολικό Αζωτο (N) | 0,02 % | | Kjeldahl | | Νιτρικό Αζωτο (NO3-N) | 7,44 | mg/Kg | 1N KCl | | Αμμωνιακό Αζωτο (NH4-N) | | mg/100g | 1N KCl | | Φωσφόρος (P) | 13,62 | mg/Kg | Olsen |	# Physicochemical Properties | PARAMETER | RESULT | UNIT | METHOD | |---|---|---|---| | pH | 7.86 | | Paste Extraction | | Electrical Conductivity | 512.00 | μS/cm | Water Extraction | | Total CaCO3 | 11.66 % | vol.% | Volumetrically | | Active CaCO3 | 1.88 % | vol.% | C2O4(NH4)2 | | Organic Matter | 1,87 % | | Wet Oxidation | ## Available Forms of Nutrients | PARAMETER | RESULT | UNIT | METHOD | |---|---|---|---| | Total Nitrogen (N) | 0,02 % | | Kjeldahl | | Nitrate Nitrogen (NO3-N) | 7,44 | mg/Kg | 1N KCl | | Ammonium Nitrogen (NH4-N) | | mg/100g | 1N KCl | | Phosphorus (P) | 13,62 | mg/Kg | Olsen |	13.2

2	### Πληροφορίες Καταχώρησης - **ΕΠΩΝΥΜΟ:** ΖΑΝΝΙΑΣ - **ΟΝΟΜΑ:** ΙΩΑΝΝΗΣ - **ΠΑΤΡΩΝΥΜΟ:** - **Α.Φ.Μ.:** - **ΔΙΕΥΘΥΝΣΗ:** - **ΤΟΠΙΚΗ ΚΟΙΝΟΤΗΤΑ/ ΝΟΜΟΣ:** ΟΡΧΟΜΕΝΟΣ ΒΟΙΩΤΙΑΣ - **ΤΗΛΕΦΩΝΟ / E-MAIL:** ### Πληροφορίες Γης και Δείγματος - **ΤΟΠΟΘΕΣΙΑ ΑΓΡΟΥ:** ORCH 20 - ΚΕΦΑΛΕΣ ΠΕΥΚΟ - **ΧΩΡΙΟ / ΠΟΛΗ:** - **ΕΚΤΑΣΗ (ΣΤΡ):** - **ΚΑΛΛΙΕΡΓΕΙΑ:** ΦΑΚΗ - **ΠΟΙΚΙΛΙΑ:** - **ΚΛΙΣΗ ΕΔΑΦΟΥΣ:** - **ΤΡΟΠΟΣ ΑΡΔΕΥΣΗΣ:** - **ΕΙΔΟΣ ΔΕΙΓΜΑΤΩΝ:** ΕΔΑΦΟΣ - **ΔΕΙΓΜΑΤΟΛΗΨΙΑ:** Ο ΙΔΙΟΣ - **ΗΜ/ΝΙΑ ΠΑΡΑΛΑΒΗΣ:** 27/2/2025 - **ΚΩΔΙΚΟΙ ΔΕΙΓΜΑΤΩΝ:** 250219A - **ΕΙΔΟΣ ΑΝΑΛΥΣΗΣ:** ΠΛΗΡΗΣ	### Registration Information - **SURNAME:** ZANNIAS - **FIRST NAME:** IOANNIS - **PATRONYMIC:** - **TAX ID NUMBER:** - **ADDRESS:** - **LOCAL COMMUNITY/COUNTY:** ORKHOMENOS BOEOTIA - **PHONE / EMAIL:** ### Soil and Sample Information - **FARM LOCATION:** ORCH 20 - CYPRESS HEADS - **VILLAGE / CITY:** - **AREA (STR):** - **CROP:** LENTIL - **VARIETY:** - **SOIL SLOPE:** - **IRRIGATION METHOD:** - **SAMPLE TYPE:** SOIL - **SAMPLE COLLECTION:** THE OWNER - **DATE OF RECEIPT:** 27/2/2025 - **SAMPLE CODES:** 250219A - **TYPE OF ANALYSIS:** COMPLETE	10.6
3	## Φώσφορος Χαμηλή # Περιεκτικότητα ## Υπολειμματικός Φώσφορος - **Υπολειμματικός Φώσφορος:** 5,8 Kg/Στρέμμα - **Προτεινόμενη δόση Φωσφόρου:** 7,00 μονάδες (P) ως P2O5: 7 μον. - **Βασική Εφαρμογή με ενσωμάτωση Φωσφορίτης:** 35,0 Kg/Στρέμμα - **Λίπανση:** (1-20-0) ## Κάλιο (K) - **Υπολειμματικό Κάλιο:** 103,1 Kg/Στρέμμα - **Ποσοστό της CEC:** 1,68% - **Προτεινόμενη δόση Καλίου:** 0,00 μονάδες ως KO2 ## Μαγνήσιο (Mg) - **Υψηλή Περιεκτικότητα επαρκής για 3-4 χρόνια.** - **Ποσοστό της CEC με Mg:** 11,88% ## Ασβέστιο (Ca) - **Υψηλή Περιεκτικότητα:** 83,49% της CEC - **Ανάλυση:** 31,15 cmol/Kg	## Phosphorus Low # Content ## Residual Phosphorus - **Residual Phosphorus:** 5.8 Kg/Struma - **Recommended Phosphorus Dose:** 7.00 units (P) as P2O5: 7 units. - **Basic Application with Rock Phosphate Incorporation:** 35.0 Kg/Stremma - **Fertilization:** (1-20-0) ## Potassium (K) - **Residual Potassium:** 103.1 Kg/Stremma - **Percentage of CEC:** 1.68% - **Recommended Potassium Dose:** 0.00 units as KO2 ## Magnesium (Mg) - **High Content Sufficient for 3-4 Years.** - **CEC Percentage with Mg:** 11.88% ## Calcium (Ca) - **High Content:** 83.49% of CEC - **Analysis:** 31.15 cmol/Kg	11.6

Table 7: Translation of tables and Markdown formats Using The **Translation Module**.

#	Post-Processed Text	Google API Translated Output	Time (s)
1	Το έδαφος στα 0-30 εκατοστά ζυγίζει 421,69 τόνους	The ground at 0-30 cm weighs 421,69 tonnes	1.0
2	Το ενεργό βάθος ριζοστρώματος της καλλιέργειας όπου αναπτύσσεται η κύρια μάζα των ριζών των φυτών και συντελείται ο κύριος εφοδιασμός των ριζών με νερό είναι: 0 εκ.	The active depth of root of crop where the main mass of plant roots develops and The main supply of roots with water is: 0 cm	1.8

Table 8: Translation of simple text from Greek to English using **Google's API** Translation.

#	Post-Processed Text	Google API Translated Output	Time (s)

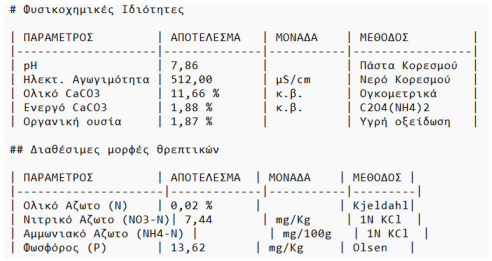
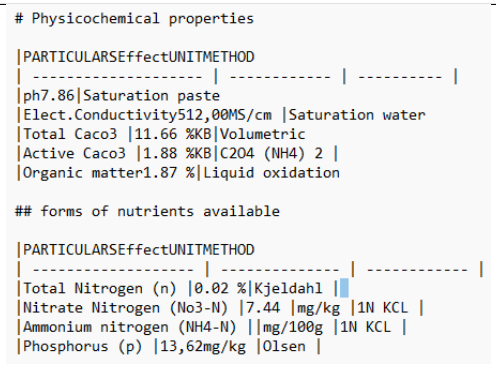
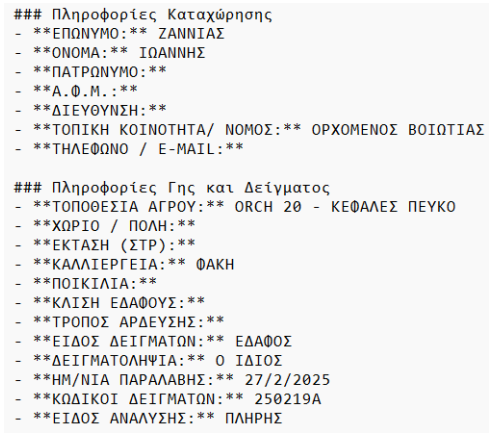
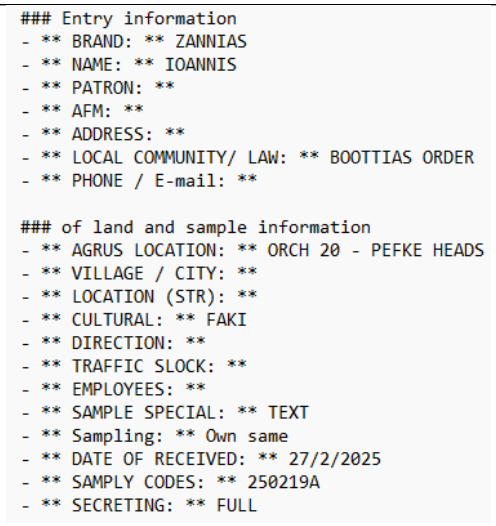
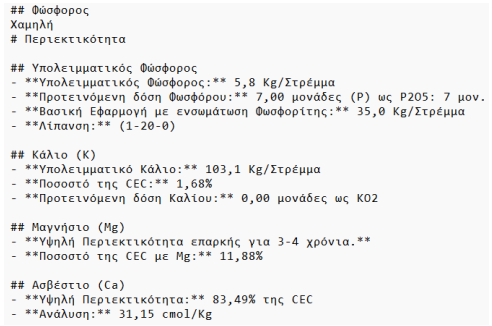
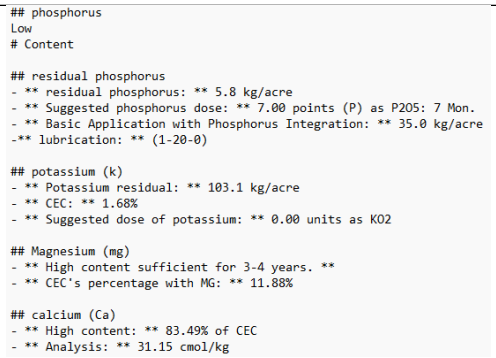
1			1.5
2			1.8
3			1.6

Table 9: Translation of tables and markdown formats using **Google's API** Translation.

#	**Post-Processed Text**	**Google API Translated Output**	**Time (s)**
1	Το έδαφος στα 0-30 εκατοστά ζυγίζει 421,69 τόνους	Soil at 0-30 cm weighs 421.69 tons	0.2

2	Το ενεργό βάθος ριζοστρώματος της καλλιέργειας όπου αναπτύσσεται η κύρια μάζα των ριζών των φυτών και συντελείται ο κύριος εφοδιασμός των ριζών με νερό είναι: 0 εκ.	The active rootbed depth of the crop where the main mass of plant roots grows and the main supply of roots with water occurs is: 0 cm	0.22

Table 10: Translation of simple text from Greek to English using HuggingFace model **`Helsinki-NLP/opus-mt-tc-big-el-en`**

#	Post-Processed Text	Translated Output	Time (s)
1	# Φυσικοχημικές Ιδιότητες | ΠΑΡΑΜΕΤΡΟΣ | ΑΠΟΤΕΛΕΣΜΑ | ΜΟΝΑΔΑ | ΜΕΘΟΔΟΣ | |---|---|---|---| | pH | 7,86 | | Πάστα Κορεσμού | | Ηλεκτ. Αγωγιμότητα | 512,00 | μS/cm | Νερό Κορεσμού | | Ολικό CaCO3 | 11,66 % | κ.β. | Ογκομετρικά | | Ενεργό CaCO3 | 1,88 % | κ.β. | C2O4(NH4)2 | | Οργανική ουσία | 1,87 % | | Υγρή οξείδωση | ## Διαθέσιμες μορφές θρεπτικών | ΠΑΡΑΜΕΤΡΟΣ | ΑΠΟΤΕΛΕΣΜΑ | ΜΟΝΑΔΑ | ΜΕΘΟΔΟΣ | |---|---|---|---| | Ολικό Άζωτο (N) | 0,02 % | | Kjeldahl| | Νιτρικό Άζωτο (NO3-N)| 7,44 | mg/Kg | 1N KCl | | Αμμωνιακό Άζωτο (NH4-N) | | mg/100g | 1N KCl | | Φωσφόρος (P) | 13,62 | mg/Kg | Olsen |	#Phytochemical Properties PARAMETROS EFFECT UNIT ---------- ----------	3.5
2	### Πληροφορίες Καταχώρησης - **ΕΠΩΝΥΜΟ:** ΖΑΝΝΙΑΣ - **ΟΝΟΜΑ:** ΙΩΑΝΝΗΣ - **ΠΑΤΡΩΝΥΜΟ:** - **Α.Φ.Μ.:** - **ΔΙΕΥΘΥΝΣΗ:** - **ΤΟΠΙΚΗ ΚΟΙΝΟΤΗΤΑ/ ΝΟΜΟΣ:** ΟΡΧΟΜΕΝΟΣ ΒΟΙΩΤΙΑΣ - **ΤΗΛΕΦΩΝΟ / E-MAIL:** ### Πληροφορίες Γης και Δείγματος - **ΤΟΠΟΘΕΣΙΑ ΑΓΡΟΥ:** ORCH 20 - ΚΕΦΑΛΕΣ ΠΕΥΚΟ - **ΧΩΡΙΟ / ΠΟΛΗ:** - **ΕΚΤΑΣΗ (ΣΤΡ):** - **ΚΑΛΛΙΕΡΓΕΙΑ:** ΦΑΚΗ - **ΠΟΙΚΙΛΙΑ:** - **ΚΛΙΣΗ ΕΔΑΦΟΥΣ:** - **ΤΡΟΠΟΣ ΑΡΔΕΥΣΗΣ:** - **ΕΙΔΟΣ ΔΕΙΓΜΑΤΩΝ:** ΕΔΑΦΟΣ - **ΔΕΙΓΜΑΤΟΛΗΨΙΑ:** Ο ΙΔΙΟΣ - **ΗΜ/ΝΙΑ ΠΑΡΑΛΑΒΗΣ:** 27/2/2025 - **ΚΩΔΙΚΟΙ ΔΕΙΓΜΑΤΩΝ:** 250219A - **ΕΙΔΟΣ ΑΝΑΛΥΣΗΣ:** ΠΛΗΡΗΣ	Registration Information - **SPECIALITY:** ZANNIA - **DOMINIA **:** IOANNIS - **PATHONY:** **A.F.M.:** - **Directory:** - **Local Community/ LAW:** ORGANIZED BIOTIA -**TELE / E-MAIL:** Land and Sample Information **	0.5
3	## Φώσφορος Χαμηλή # Περιεκτικότητα ## Υπολειμματικός Φώσφορος - **Υπολειμματικός Φώσφορος:** 5,8 Kg/Στρέμμα - **Προτεινόμενη δόση Φωσφόρου:** 7,00 μονάδες (P) ως P2O5: 7 μον. - **Βασική Εφαρμογή με ενσωμάτωση Φωσφορίτης:** 35,0 Kg/Στρέμμα - **Λίπανση:** (1-20-0) ## Κάλιο (K) - **Υπολειμματικό Κάλιο:** 103,1 Kg/Στρέμμα - **Ποσοστό της CEC:** 1,68% - **Προτεινόμενη δόση Καλίου:** 0,00 μονάδες ως KO2 ## Μαγνήσιο (Mg) - **Υψηλή Περιεκτικότητα επαρκής για 3-4 χρόνια.** - **Ποσοστό της CEC με Mg:** 11,88% ## Ασβέστιο (Ca) - **Υψηλή Περιεκτικότητα:** 83,49% της CEC - **Ανάλυση:** 31,15 cmol/Kg	Phosphorus Low # Content Residual Phosphorous** Sublimation C(**C) ** Sublimation Phosphoric:** 5.8 Kg/Stremma - **Recommended Phosphorous Dose:** 7.00 units (P) as P2O5: 7 mon. **Basic Application with Phosphorite Integration:** 35.0 Kg/Strema -**	0.6

Table 11: Translation of tables and markdown formats using HuggingFace model **`Helsinki-NLP/opus-mt-tc-big-el-en`**

#	Post-Processed Text	Google API Translated Output	Time (s)
1	Το έδαφος στα 0-30 εκατοστά ζυγίζει 421,69 τόνους	The soil at 0-30 cm weighs 421.69 tons	0.2
2	Το ενεργό βάθος ριζοστρώματος της καλλιέργειας όπου αναπτύσσεται η κύρια μάζα των ριζών των φυτών και συντελείται ο κύριος εφοδιασμός των ριζών με νερό είναι: 0 εκ.	The active root depth of the cultivation where the main mass of the roots of plants grows and the main water supply of the roots is carried out is: 0 mm	0.6

Table 12: Translation of simple text from Greek to English using HuggingFace model **`facebook/m2m100_418M`**

#	Post-Processed Text	Translated Output	Time (s)
1	# Φυσικοχημικές Ιδιότητες | ΠΑΡΑΜΕΤΡΟΣ | ΑΠΟΤΕΛΕΣΜΑ | ΜΟΝΑΔΑ | ΜΕΘΟΔΟΣ | |---|---|---|---| | pH | 7,86 | | Πάστα Κορεσμού | | Ηλεκτ. Αγωγιμότητα | 512,00 | μS/cm | Νερό Κορεσμού | | Ολικό CaCO3 | 11,66 % | κ.β. | Ογκομετρικά | | Ενεργό CaCO3 | 1,88 % | κ.β. | C2O4(NH4)2 | | Οργανική ουσία | 1,87 % | | Υγρή οξείδωση | ## Διαθέσιμες μορφές θρεπτικών | ΠΑΡΑΜΕΤΡΟΣ | ΑΠΟΤΕΛΕΣΜΑ | ΜΟΝΑΔΑ | ΜΕΘΟΔΟΣ | |---|---|---|---| | Ολικό Άζωτο (N) | 0,02 % | | Kjeldahl | | Νιτρικό Άζωτο (NO3-N) | 7,44 | mg/Kg | 1N KCl | | Αμμωνιακό Άζωτο (NH4-N) | | mg/100g | 1N KCl | | Φωσφόρος (P) | 13,62 | mg/Kg | Olsen |	# Physiochemical properties <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk>	3.5
2	### Πληροφορίες Καταχώρησης - **ΕΠΩΝΥΜΟ:** ΖΑΝΝΙΑΣ - **ΟΝΟΜΑ:** ΙΩΑΝΝΗΣ - **ΠΑΤΡΩΝΥΜΟ:** - **Α.Φ.Μ.:** - **ΔΙΕΥΘΥΝΣΗ:** - **ΤΟΠΙΚΗ ΚΟΙΝΟΤΗΤΑ/ ΝΟΜΟΣ:** ΟΡΧΟΜΕΝΟΣ ΒΟΙΩΤΙΑΣ - **ΤΗΛΕΦΩΝΟ / E-MAIL:** ### Πληροφορίες Γης και Δείγματος - **ΤΟΠΟΘΕΣΙΑ ΑΓΡΟΥ:** ORCH 20 - ΚΕΦΑΛΕΣ ΠΕΥΚΟ - **ΧΩΡΙΟ / ΠΟΛΗ:** - **ΕΚΤΑΣΗ (ΣΤΡ):** - **ΚΑΛΛΙΕΡΓΕΙΑ:** ΦΑΚΗ - **ΠΟΙΚΙΛΙΑ:** - **ΚΛΙΣΗ ΕΔΑΦΟΥΣ:** - **ΤΡΟΠΟΣ ΑΡΔΕΥΣΗΣ:** - **ΕΙΔΟΣ ΔΕΙΓΜΑΤΩΝ:** ΕΔΑΦΟΣ - **ΔΕΙΓΜΑΤΟΛΗΨΙΑ:** Ο ΙΔΙΟΣ - **ΗΜ/ΝΙΑ ΠΑΡΑΛΑΒΗΣ:** 27/2/2025 - **ΚΩΔΙΚΟΙ ΔΕΙΓΜΑΤΩΝ:** 250219A - **ΕΙΔΟΣ ΑΝΑΛΥΣΗΣ:** ΠΛΗΡΗΣ	## Registration Information - **NUMBER:** DEVELOPMENT - **NUMBER:** JOHANNES - **NUMBER:** - **A.F.M.:** - ** RESPONSIBILITY:** - **TOPICAL COMMUNITY/Law:** RELATED NUMBER:** NUMBER:	0.5

3	## Φώσφορος Χαμηλή # Περιεκτικότητα ## Υπολειμματικός Φώσφορος - **Υπολειμματικός Φώσφορος:** 5,8 Kg/Στρέμμα - **Προτεινόμενη δόση Φωσφόρου:** 7,00 μονάδες (P) ως P2O5: 7 μον. - **Βασική Εφαρμογή με ενσωμάτωση Φωσφορίτης:** 35,0 Kg/Στρέμμα - **Λίπανση:** (1-20-0) ## Κάλιο (K) - **Υπολειμματικό Κάλιο:** 103,1 Kg/Στρέμμα - **Ποσοστό της CEC:** 1,68% - **Προτεινόμενη δόση Καλίου:** 0,00 μονάδες ως KO2 ## Μαγνήσιο (Mg) - **Υψηλή Περιεκτικότητα επαρκής για 3-4 χρόνια.** - **Ποσοστό της CEC με Mg:** 11,88% ## Ασβέστιο (Ca) - **Υψηλή Περιεκτικότητα:** 83,49% της CEC - **Ανάλυση:** 31,15 cmol/Kg	## Phosphorus Low # Content ## Residual Phosphorus - ** Residual Phosphorus:** 5.8 Kg/Strack - ** Recommended Phosphorus Dosage:** 7.00 units (P) to P2O5: 7 mon. - ** Basic Application with Integrated Phosphorus:** 35,0 Kg/Strack - ** Sweating:** (1-20-0) ## Potassium (K) - ** Residual Phosphorus:** 103.1 Kg/Strack - ** CEC Percentage:** 1.68% - ** Recommended Phosphorus Dosage:** 0.00 units (P) to P2O5: 7 mon. - ** Basic Application with Integrated Phosphorus:** 35,0 Kg/Strack - **	0.6

Table 13: Translation of tables and markdown formats using HuggingFace model `facebook/m2m100_418M`

Results. Table 14 presents the evaluation of the translation using BLEU, BERTScore, and an embedding-based similarity score for four systems: **Google Translate API**, **Helsinki-NLP/opus-mt-tc-big-el-en**, **facebook/m2m100_418M**, and **Translation Module**.

System	BLEU Score	BERTScore	Embedding Score
Translation Module	0.76	0.70	0.94
Google Translate API	0.48	0.56	0.88
Helsinki-NLP/opus-mt-tc-big-el-en	0.34	0.34	0.79
facebook/m2m100_418M	0.39	0.26	0.79

Table 14: BLEU, BERTScore, and embedding-based similarity results for the translation.

Observations and Analysis.

- **Accuracy:** The proposed Translation Module consistently produces more accurate and domain-aware translations, particularly in technical contexts. Sometimes, the other models cannot produce a correct output due to the input format. Greek agronomic terms, like *"ριζοστρώματος"* and *"Ολικό Άζωτο"*, are properly translated as *"root zone"* and *"Total Nitrogen"*, while the Google API produces incorrect equivalents (e.g., "text", "slock", or "secret"). Moreover, only the Translation Module seems to be able to keep the input alignment correctly to the output.

- **Execution Time:** The Translation Module demonstrates significantly higher latency compared to all the other methods. On average, it takes about 3 to 5 times longer to translate the same Greek text, due to local computationally heavy model inference.

- **Privacy:** Cloud API services do not have the advantage of local execution, when it comes to data protection, since there is no need for transmitting data to external services.

4.2.5 Retrieval from Small PDFs

Methods Evaluated. In the following experiments, the retrieval of values, units, and methods from small soil analysis PDFs is evaluated. For that purpose, the configurations A, B, C, and D, previously described in Section 3.2.1, are tested. Moreover, in order to evaluate the correctness limitations of the text extraction step, the contents of the PDFs were also transcribed manually, providing a reference baseline against which the automated extraction results could be compared. Each experiment introduces one or more enhancements to the basic pipeline, allowing step-by-step evaluation of their contribution.

Experimental Setup.

- **Inputs:** A total of 7 small Greek soil analysis reports were used, including 4 native (digitally generated) PDFs and 3 scanned PDFs consisting of one page each.
- **Prompts:** Fixed field-specific prompts were used across all pipelines to ensure fair comparison.
- **Requested Output:** The prompts contain direct instructions to the LLM to produce a valid JSON output. For each field, the schema that must be produced is the following:

```
{
  "field_name": {
    "value": "",
    "unit": "",
    "method": ""
  }
}
```

- **Evaluation:** Manual prototype JSONs were used for ground truth comparison as described in Section 4.1.2. Each JSON contains the fields together with their values, units and methods.
- **Procedure:** Each pipeline is tested on all PDFs and each experiment is executed 10 times in order to get the average accuracy, ensuring consistent behavior and accounting for potential randomness in the language model's responses.

Metrics And Results

- **Value Accuracy:** Correctness of values extracted for the fields.
- **Unit Accuracy:** Correctness of units extracted for the fields.
- **Method Accuracy:** Correctness of methods extracted for the fields.
- **Total Accuracy:** The average from the value, unit, method accuracy.
- **Time:** Total and per-component execution time measured per PDF.

Pipeline	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Total Time (s)
Baseline (Raw Text)	78.7	71.5	58.4	69.5	21.2
+ Postprocessing	79.9	73.4	58.6	70.6	90.8
+ Translation	88.3	81.6	75.3	81.7	131.2
+ Prompt Expansion	**88.1**	**86.4**	**76.0**	**83.5**	**141.6**

Table 15: Performance of different pipeline variants on small **Native** soil analysis PDFs.

Pipeline	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Total Time (s)
Baseline (Raw Text)	54.5	17.1	25.5	32.4	25.5
+ Postprocessing	62.9	30.6	37.1	43.5	98.2
+ Translation	69.4	53.3	58.1	60.7	140.5
+ Prompt Expansion	**68.5**	**58.5**	**62.3**	**63.1**	**151.2**

Table 16: Performance of different pipeline variants on small **Scanned** soil analysis PDFs.

Pipeline	Text Extraction (s)	Postprocessing (s)	Translation (s)	Inference (s)	Total Time (s)
Baseline (Raw Text)	0.3	–	–	20.9	21.2
+ Postprocessing	0.3	68.8	–	21.7	90.8
+ Translation	0.2	69.5	40.9	20.6	131.2
+ Prompt Expansion	**0.2**	**69.9**	**41.0**	**30.5**	**141.6**

Table 17: Time contribution of each component in pipeline on small soil analysis PDFs.

In the next table, the results of the best pipeline consisting of Post-processing, translation and prompt expansion is compared with **manually transcribed texts from the input PDFs** that are used together with prompt expansion:

Method	Total Accuracy (%)
Auto Extracted PDFs	83.5
Manually Transcribed Texts	**90.3**

Table 18: Comparison between automatically extracted PDFs and manually transcribed texts on retrieval accuracy.

Observations and Analysis.

- **Impact of Text Extraction Quality:** The comparison between automatically extracted and manually transcribed texts demonstrates the importance of accurate text extraction. The manually transcribed input achieved a higher accuracy (+**6.8%**) in native PDFs, confirming that noise, extraction errors, or formatting inconsistencies in auto-extracted text negatively impact the overall retrieval performance.
- **Low Baseline Accuracy in Scanned Inputs:** The initial baseline performance on scanned PDFs is significantly lower than on native PDFs. This highlights the difficulty of extracting structured content from noisy OCR outputs. However, applying post-processing, translation and prompt expansion progressively improves performance, demonstrating their robustness, even on low-quality inputs.

- **Accuracy Improvements:** Each successive addition to the pipeline contributes to increased accuracy. Notably, **translation** and **prompt expansion** yield the most significant gains in both value and unit extraction accuracy.
- **Role of Postprocessing:** While the postprocessing step alone does not produce a huge accuracy improvement compared to the baseline, it plays a **critical role** in enabling correct translation, since it clears the text. Poorly formatted or merged Greek text leads to translation errors or skipped content, which negatively affects retrieval accuracy.
- **Execution Time:** With each addition of a module, the overall time increases significantly, especially with translation and post-processing. Prompt expansion also increases the inference time, since more invocations of the model are being executed.
- **Resource Considerations:** All stages beyond the baseline introduce local computation, since models are executed locally.
- **Final Observation:** The pipeline configuration that yields the highest overall accuracy combines *Postprocessing*, *Translation*, and *Prompt Expansion*, demonstrating the cumulative benefit of each module. While this pipeline achieves the best results, it is also the most demanding in terms of resources and execution time.

4.2.6 Retrieval from Small and Large PDFs using RAG

The Full-Context Prompting strategy used in previous experiments is not suitable for large soil analysis PDFs. As document length increases, the combined size of the input text and prompt instructions often exceeds the language model's context window. To overcome this issue, the retrieval mechanism is replaced with a RAG pipeline.

Methods Evaluated. In the following experiments, the retrieval of values, units, and methods from both small and large soil analysis PDFs is evaluated. The Full-Context Prompting technique used in the previous experiments is no longer suitable due to the increased input size. For that reason, the architecture E, described in Section 3.2.1 is used, that replaces Full Context Prompting with RAG. To fully evaluate this technique, it is firstly tested in the same small soil analysis PDFs and then to the large ones.

RAG Parameters.

1. **Top-k retrieval ($k = 3$):** For each prompt, the 3 most relevant chunks are retrieved based on semantic similarity.
2. **Chunking:** Text is split into chunks of 500 tokens with 50-token overlap using `SentenceSplitter`.
3. **Context window:** The LLM context length is set to 4096 tokens, allowing sufficient room for both prompt and retrieved content.
4. **Embedding model:** `intfloat/multilingual-e5-large` [41, 42] from HuggingFace is used for generating dense vector representations of text.

5. **Language model:** `Qwen2.5:32B` [32] is used for generating the final answer from the input chunks. The temperature is been set to 0 for consistent responses and easier evaluation.

Experimental Setup.

- **Inputs:** A total of 11 Greek soil analysis PDFs, including 4 small documents (1 page each) and 7 large documents ranging from 3 to 5 pages.
- **Prompts:** Fixed field-specific prompts were used across all pipelines to ensure fair comparison.
- **Requested Output:** The prompts contain direct instructions to the LLM to produce a valid JSON output. For each field, the schema that must be produced is the following:

```
{
  "field_name": {
    "value": "",
    "unit": "",
    "method": ""
  }
}
```

- **Evaluation:** Manual prototype JSONs were used for ground truth comparison 4.1.2. Each JSON contains the fields together with their values, units and methods.
- **Procedure:** The pipeline of post-processing, translation and prompt expansion is evaluated using RAG as the retrieval method. It is first applied to the small PDFs for direct comparison with the previous full-context prompting results and then to the large PDFs to assess the pipeline's scalability and robustness on longer inputs. Each experiment is conducted 10 times in order to get the average accuracy, ensuring consistent behavior and accounting for potential randomness in the language model's responses.

Metrics And Results

- **Value Accuracy:** Correctness of values extracted for the fields.
- **Unit Accuracy:** Correctness of units extracted for the fields.
- **Method Accuracy:** Correctness of methods extracted for the fields.
- **Total Accuracy:** The average from the value, unit, method accuracy.
- **Time:** Total execution time measured per PDF.

Input Type	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Total Time (s)
Small PDFs	84.3	87.0	77.0	82.7	149.6
Large PDFs	89.1	85.4	76.4	83.6	515.9

Table 19: RAG Pipeline Performance on Small and Large **Native** Soil Analysis PDFs

Input Type	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Total Time (s)
Small PDFs	68.0	57.6	61.2	62.3	156.4
Large PDFs	65.6	55.2	59.8	60.2	526.3

Table 20: RAG Pipeline Performance on Small and Large **Scanned** Soil Analysis PDFs

Input Type	Text Extraction (s)	Postprocessing (s)	Translation (s)	RAG Inference (s)	Total Time (s)
Small PDFs	0.3	68.4	41.1	39.8	149.6
Large PDFs	0.5	335.4	132.4	47.6	515.9

Table 21: Time Breakdown for RAG Pipeline Components on Small and Large PDFs

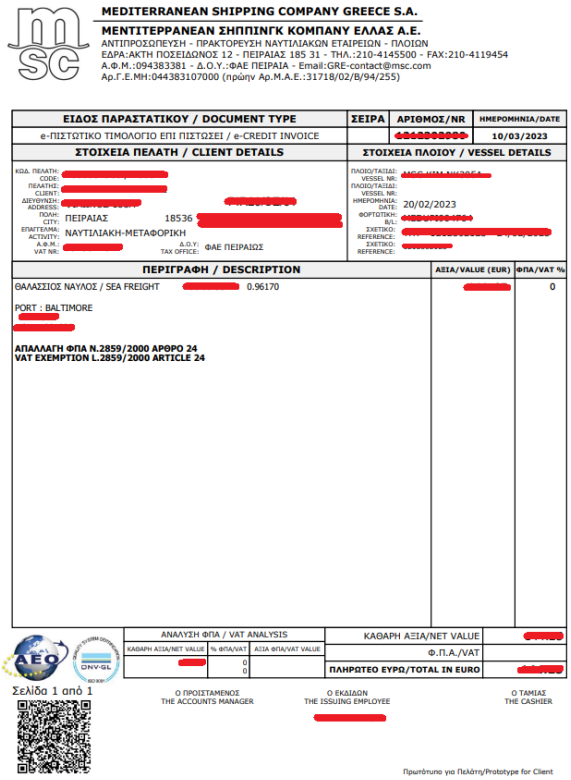
Observations and Analysis.

- **Comparison with Full-Context Prompting:** When applied to the same set of small soil analysis PDFs, the RAG pipeline achieved a total accuracy of **82.7%**, which is nearly identical to the **83.5%** achieved using full-context prompting (Table 15). This indicates that RAG can match the performance of full-context retrieval, even on small documents.
- **Execution Time:** The RAG pipeline requires slightly more time (149.6s vs. 141.6s) on small PDFs, mainly due to the additional steps involved in chunking, embedding, and retrieval. However, this increase is modest and does not significantly affect practical usability.
- **Scalability to Large Documents:** The pipeline was tested on larger soil analysis PDFs (3–5 pages), where full-context prompting becomes infeasible due to context window limitations. The RAG approach maintained high total accuracy (**83.6%**), while scaling effectively to longer documents.
- **Inference Time Stability:** Despite the increase in input size, the RAG inference time remains relatively stable (39.8s for small vs. 47.6s for large PDFs), due to the use of top-k chunk retrieval (with $k = 3$). This ensures that the same amount of context is passed to the language model, regardless of the document's total length.
- **Final Observation:** The RAG pipeline, combining post-processing, translation, and prompt expansion, proves to be the most robust and scalable solution. It achieves consistent accuracy across document sizes, while keeping the execution time increase within acceptable bounds. This makes RAG the preferred method for generalizing the pipeline to diverse and lengthy inputs.

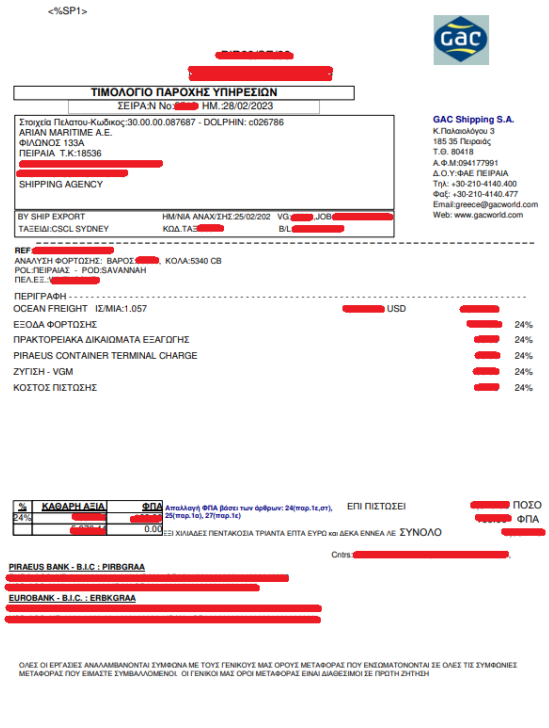
4.3 Part II: Generalized Field Extraction Experiments with Multi-Agent Systems

4.3.1 Dataset Description.

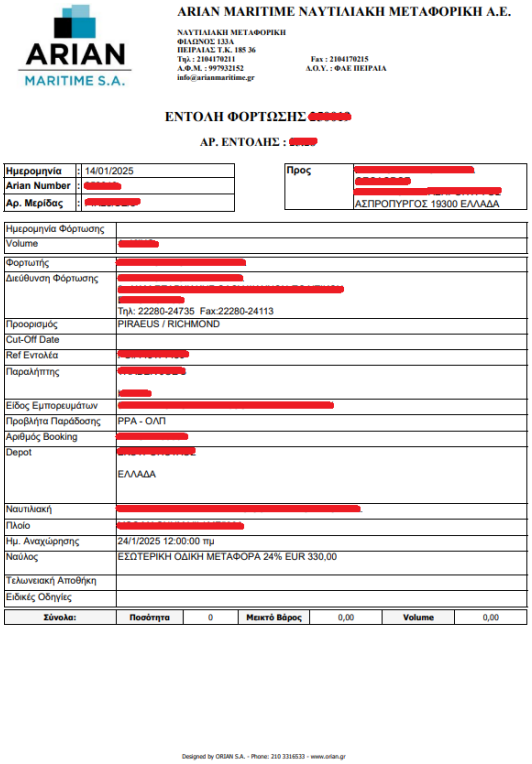
For the evaluation of the multi-agent system, all native documents used in the previous experiments — including both small and large soil analysis reports — were used, but this time, the requested fields increased in order to capture the majority of the PDFs entities. In addition, two new document categories were introduced to test the system's generalization ability: **Greek Invoices** and **Blood Test Results**. The length of these native PDFs range from 1 to 5 pages. Examples are shown in Figures 5, **??**, **??**. This expanded dataset allows for testing across multiple domains with varying structure and terminology.



(a) First PDF.

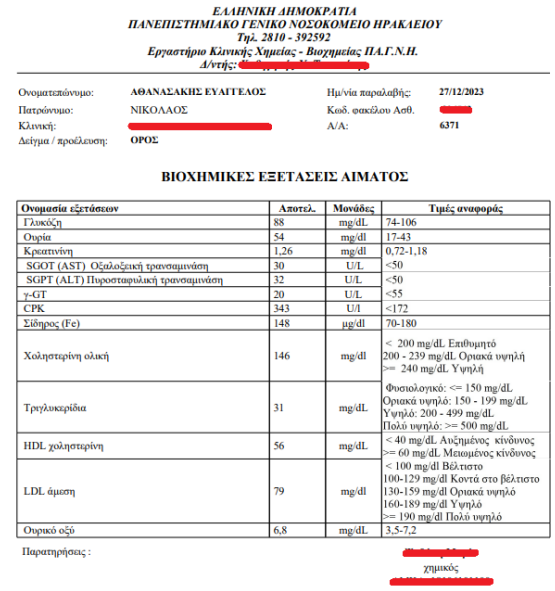


(b) Second PDF.



(c) Third PDF.

Figure 5: Examples Of Invoices



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΓΕΝΙΚΟ ΝΟΣΟΚΟΜΕΙΟ ΗΡΑΚΛΕΙΟΥ
Τηλ. 2810 - 392592
Εργαστήριο Κλινικής Χημείας - Βιοχημείας ΠΑ.Γ.Ν.Η.
Δ/ντής: [redacted]

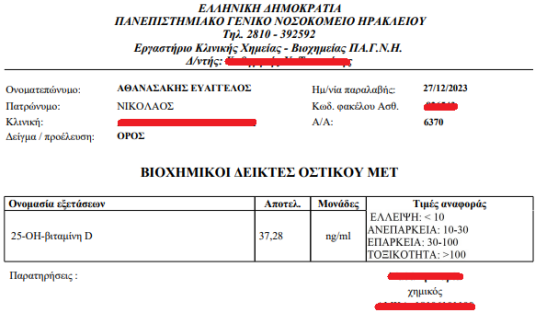
Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ
Πατρώνυμο: ΝΙΚΟΛΑΟΣ
Κλινική: [redacted]
Δείγμα / προέλευση: ΟΡΟΣ
Ημ/νία παραλαβής: 27/12/2023
Κωδ. φακέλου Ασθ. [redacted]
Α/Α: 6371

ΒΙΟΧΗΜΙΚΕΣ ΕΞΕΤΑΣΕΙΣ ΑΙΜΑΤΟΣ

Ονομασία εξετάσεων	Αποτελ.	Μονάδες	Τιμές αναφοράς
Γλυκόζη	88	mg/dL	74-106
Ουρία	54	mg/dl	17-43
Κρεατινίνη	1,26	mg/dl	0,72-1,18
SGOT (AST) Οξαλοξεική τρανσαμινάση	30	U/L	<50
SGPT (ALT) Πυροσταφυλική τρανσαμινάση	32	U/L	<50
γ-GT	20	U/L	<55
CPK	343	U/l	<172
Σίδηρος (Fe)	148	μg/dl	70-180
Χοληστερίνη ολική	146	mg/dl	< 200 mg/dL Επιθυμητό 200 - 239 mg/dL Οριακά υψηλή >= 240 mg/dL Υψηλή
Τριγλυκερίδια	31	mg/dL	Φυσιολογικό: <= 150 mg/dL Οριακά υψηλό: 150 - 199 mg/dL Υψηλό: 200 - 499 mg/dL Πολύ υψηλό: >= 500 mg/dL
HDL χοληστερίνη	56	mg/dL	< 40 mg/dL Αυξημένος κίνδυνος >= 60 mg/dL Μειωμένος κίνδυνος
LDL άμεση	79	mg/dl	< 100 mg/dl Βέλτιστο 100-129 mg/dl Κοντά στο βέλτιστο 130-159 mg/dl Οριακά υψηλό 160-189 mg/dl Υψηλό >= 190 mg/dl Πολύ υψηλό
Ουρικό οξύ	6,8	mg/dL	3,5-7,2

Παρατηρήσεις : [redacted]
χημικός

(a) First Page.



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΓΕΝΙΚΟ ΝΟΣΟΚΟΜΕΙΟ ΗΡΑΚΛΕΙΟΥ
Τηλ. 2810 - 392592
Εργαστήριο Κλινικής Χημείας - Βιοχημείας ΠΑ.Γ.Ν.Η.
Δ/ντής: [redacted]

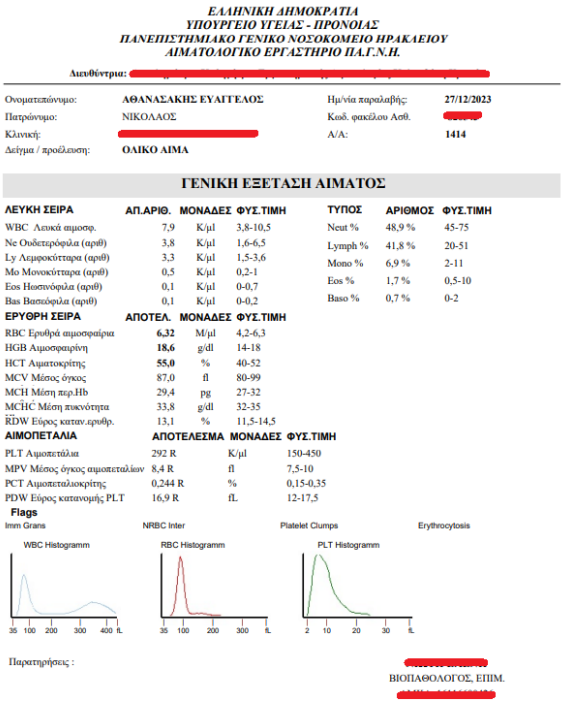
Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ
Πατρώνυμο: ΝΙΚΟΛΑΟΣ
Κλινική: [redacted]
Δείγμα / προέλευση: ΟΡΟΣ
Ημ/νία παραλαβής: 27/12/2023
Κωδ. φακέλου Ασθ. [redacted]
Α/Α: 6370

ΒΙΟΧΗΜΙΚΟΙ ΔΕΙΚΤΕΣ ΟΣΤΙΚΟΥ ΜΕΤ

Ονομασία εξετάσεων	Αποτελ.	Μονάδες	Τιμές αναφοράς
25-OH-βιταμίνη D	37,28	ng/ml	ΕΛΛΕΙΨΗ: < 10 ΑΝΕΠΑΡΚΕΙΑ: 10-30 ΕΠΑΡΚΕΙΑ: 30-100 ΤΟΞΙΚΟΤΗΤΑ: >100

Παρατηρήσεις : [redacted]
χημικός

(b) Second Page.



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΥΠΟΥΡΓΕΙΟ ΥΓΕΙΑΣ - ΠΡΟΝΟΙΑΣ
ΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΓΕΝΙΚΟ ΝΟΣΟΚΟΜΕΙΟ ΗΡΑΚΛΕΙΟΥ
ΑΙΜΑΤΟΛΟΓΙΚΟ ΕΡΓΑΣΤΗΡΙΟ ΠΑ.Γ.Ν.Η.
Διευθύντρια: [redacted]

Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ
Πατρώνυμο: ΝΙΚΟΛΑΟΣ
Κλινική: [redacted]
Δείγμα / προέλευση: ΟΛΙΚΟ ΑΙΜΑ
Ημ/νία παραλαβής: 27/12/2023
Κωδ. φακέλου Ασθ. [redacted]
Α/Α: 1414

ΓΕΝΙΚΗ ΕΞΕΤΑΣΗ ΑΙΜΑΤΟΣ

ΛΕΥΚΗ ΣΕΙΡΑ	ΑΠ.ΑΡΙΘ.	ΜΟΝΑΔΕΣ	ΦΥΣ.ΤΙΜΗ
WBC Λευκά αιμοσφ.	7,9	K/μl	3,8-10,5
Ne Ουδετερόφιλα (αριθ)	3,8	K/μl	1,6-6,5
Ly Λεμφοκύτταρα (αριθ)	3,3	K/μl	1,5-3,6
Mo Μονοκύτταρα (αριθ)	0,5	K/μl	0,2-1
Eos Ηωσινόφιλα (αριθ)	0,1	K/μl	0-0,7
Bas Βασεόφιλα (αριθ)	0,1	K/μl	0-0,2

ΤΥΠΟΣ	ΑΡΙΘΜΟΣ	ΦΥΣ.ΤΙΜΗ
Neut %	48,9 %	45-75
Lymph %	41,8 %	20-51
Mono %	6,9 %	2-11
Eos %	1,7 %	0,5-10
Baso %	0,7 %	0-2

ΕΡΥΘΡΗ ΣΕΙΡΑ	ΑΠΟΤΕΛ.	ΜΟΝΑΔΕΣ	ΦΥΣ.ΤΙΜΗ
RBC Ερυθρά αιμοσφαίρια	**6,32**	M/μl	4,2-6,3
HGB Αιμοσφαιρίνη	**18,6**	g/dl	14-18
HCT Αιματοκρίτης	**55,0**	%	40-52
MCV Μέσος όγκος	87,0	fl	80-99
MCH Μέση περ.Hb	29,4	pg	27-32
MCHC Μέση πυκνότητα	33,8	g/dl	32-35
RDW Εύρος καταν.ερυθρ.	13,1	%	11,5-14,5

ΑΙΜΟΠΕΤΑΛΙΑ	ΑΠΟΤΕΛΕΣΜΑ	ΜΟΝΑΔΕΣ	ΦΥΣ.ΤΙΜΗ
PLT Αιμοπετάλια	292 R	K/μl	150-450
MPV Μέσος όγκος αιμοπεταλίων	8,4 R	fl	7,5-10
PCT Αιμοπεταλιοκρίτης	0,244 R	%	0,15-0,35
PDW Εύρος κατανομής PLT	16,9 R	fL	12-17,5

Flags
Imm Grans NRBC Inter Platelet Clumps Erythrocytosis

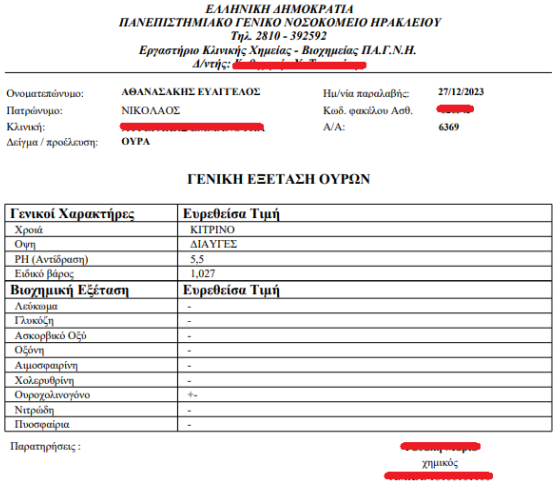
WBC Histogramm

RBC Histogramm

PLT Histogramm

Παρατηρήσεις : [redacted]
ΒΙΟΠΑΘΟΛΟΓΟΣ, ΕΠΙΜ.

(c) Third Page.



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΓΕΝΙΚΟ ΝΟΣΟΚΟΜΕΙΟ ΗΡΑΚΛΕΙΟΥ
Τηλ. 2810 - 392592
Εργαστήριο Κλινικής Χημείας - Βιοχημείας ΠΑ.Γ.Ν.Η.
Δ/ντής: [redacted]

Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ
Πατρώνυμο: ΝΙΚΟΛΑΟΣ
Κλινική: [redacted]
Δείγμα / προέλευση: ΟΥΡΑ
Ημ/νία παραλαβής: 27/12/2023
Κωδ. φακέλου Ασθ. [redacted]
Α/Α: 6369

ΓΕΝΙΚΗ ΕΞΕΤΑΣΗ ΟΥΡΩΝ

Γενικοί Χαρακτήρες	Ευρεθείσα Τιμή
Χροιά	ΚΙΤΡΙΝΟ
Όψη	ΔΙΑΥΓΕΣ
PH (Αντίδραση)	5,5
Ειδικό βάρος	1,027
Βιοχημική Εξέταση	**Ευρεθείσα Τιμή**
Λεύκωμα	-
Γλυκόζη	-
Ασκορβικό Οξύ	-
Οξόνη	-
Αιμοσφαιρίνη	-
Χολερυθρίνη	-
Ουροχολινογόνο	+-
Νιτρώδη	-
Πυοσφαίρια	-

Παρατηρήσεις : [redacted]
χημικός

(d) Fourth Page.

Figure 6: First example of blood test results.

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΓΕΝΙΚΟ ΝΟΣΟΚΟΜΕΙΟ ΗΡΑΚΛΕΙΟΥ
Τηλ. 2810 - 392229
Εργαστήριο Κλινικής Ανοσολογίας / Εργαστηριακής Ενδοκρινολογίας ΠΑ.Γ.Ν.Η.
Διευθύντρια: [redacted]

SID :	**2077**	Κλινική:	[redacted]
Ονοματεπώνυμο:	**ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ**	Ημ/νία παραλαβής:	**27/12/2023**
Πατρώνυμο:	ΝΙΚΟΛΑΟΣ	Ημ/νία εκτύπωσης:	**27/12/2023 11:26**
Κωδ. φακέλου Ασθ.	[redacted]	Ηλικία:	

ΑΠΑΝΤΗΣΕΙΣ ΕΞΕΤΑΣΕΩΝ

ΕΛΕΓΧΟΣ ΟΡΜΟΝΩΝ

Ονομασία εξετάσεων	Αποτελέσματα	Μονάδες	Φυσιολογικές τιμές
TSH	1,3091	μUI/mL	0,25-3,43
Φερριτινη	23,44	ng/ml	21,81-274,66
Vitamin B12	747	pg/ml	189-883

Υπογραφή Υπευθύνου

[redacted] Βιοπαθολογος Διευθυντρια ΕΣΥ [redacted]

Παρατηρησεις

(e) Fifth Page.

Figure 6: (continued) First example of blood test results.

Κωδικός Ασθενή: 146643 **Ημερομηνία Εξέτασης:** 16/04/2021 09:58
Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ ΝΙΚΟΛΑΟ **Ηλικία:** 19
ΕΞΩΤΕΡΙΚΟΣ ΑΣΘΕΝΗΣ

ΒΙΟΧΗΜΙΚΟ

Εξέταση	Τιμή	Μ.Μ.	Φυσιολογικές Τιμές
ΣΑΚΧΑΡΟ(GLUCOSE)	103	mg/dl	74 - 99
ΟΥΡΙΑ(UREA)	48	mg/dl	8 - 71
ΚΡΕΑΤΙΝΙΝΗ	0,9	mg/dl	0,70 - 1,30
ΟΥΡΙΚΟ ΟΞΥ	6,6	mg/dl	2,5 - 7,0
ΧΟΛΗΣΤΕΡΙΝΗ(CHOLESTEROL)	116	mg/dl	< 170 (σε συνεκτίμηση με τα επιθυμητά επίπεδα LDL χοληστερόλης)
ΤΡΙΓΛΥΚΕΡΙΔΙΑ(TRIGLYCERIDES)	27	mg/dl	< 150
ΥΨΗΛΗΣ ΠΥΚΝΟΤΗΤΑΣ ΛΙΠΟΠΡΩΤΕΙΝΗ HDL	59	mg/dl	> 40
SGOT ΟΞΑΛΟΞΕΙΚΗ ΤΡΑΝΣΑΜΙΝΑΣΗ	19	U/L	< 38
SGPT(ALT) ΠΥΡ/ΚΗ ΤΡΑΝΣΑΜΙΝΑΣΗ	15	U/L	< 40
ΑΛΚΑΛΙΚΗ ΦΩΣΦΑΤΑΣΗ(ALP)	79	U/L	40 - 129
γ-GT	13	U/L	< 60
CPK	115	U/L	< 170
ΝΑΤΡΙΟ	141,6	mmol/l	136 - 145
ΚΑΛΙΟ	4,1	mmol/L	3,5 - 5,1
ΑΣΒΕΣΤΙΟ	9,9	mg/dl	8,0 - 10,3
ΣΙΔΗΡΟΣ	118	μg/dl	59 - 158

(a) First Page.

Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ ΝΙΚΟΛΑΟ **Ηλικία:** 19
Κωδικός Ασθενή: 146643 **Ημερομηνία Εξέτασης:** 16/04/2021 10:09 **ΕΞΩΤΕΡΙΚΟΣ ΑΣΘΕΝΗΣ**

ΓΕΝΙΚΗ ΕΞΕΤΑΣΗ ΑΙΜΑΤΟΣ

ΛΕΥΚΑ ΑΙΜΟΣΦΑΙΡΙΑ		Τιμή	Μ.Μ.	Φ.Τ.
WBC	ΛΕΥΚΑ (WBC)	5,47	K/μL	4,5 - 10,5
NEUT%	ΠΟΛΥΜΟΡΦΟΠΥΡΗΝΑ	48,5	%	40 - 75
LYMPH%	ΛΕΜΦΟΚΥΤΤΑΡΑ	40,8	%	20 - 45
MONO%	ΜΟΝΟΠΥΡΗΝΑ	8,2	%	2 -10
EOS%	ΗΩΣΙΝΟΦΙΛΑ	2,4	%	0,0 - 7,0
BASO%	ΒΑΣΕΟΦΙΛΑ	0,1	%	0,0 - 2,0
ATL%		-		
IMM%		-		
NEUT#	ΠΟΛΥΜΟΡΦΟΠΥΡΗΝΑ#	2,65	K/μL	2,0 - 7,7
LYMPH#	ΛΕΜΦΟΚΥΤΤΑΡΑ#	2,23	K/μL	1,5 - 4,0
MONO#	ΜΟΝΟΠΥΡΗΝΑ#	0,45	K/μL	0,2 - 1,0
EOS#	ΗΩΣΙΝΟΦΙΛΑ#	0,13	K/μL	0,0 - 0,7
BASO#	ΒΑΣΕΟΦΙΛΑ#	0,01	K/μL	0,0 - 0,2
ATL#		-		
IMM#		-		

ΕΡΥΘΡΑ ΑΙΜΟΣΦΑΙΡΙΑ		Τιμή	Μ.Μ.	Φ.Τ.
RBC	ΕΡΥΘΡΑ ΑΙΜΟΣΦΑΙΡΙΑ	6,13	M/μL	4,20 - 6,00
HGB	ΑΙΜΟΣΦΑΙΡΙΝΗ (HGB)	17,4	g/dL	13,5 - 18,0
HCT	ΑΙΜΑΤΟΚΡΙΤΗΣ (HCT)	50,6	%	40,0 - 52,0
MCV	MCV	82,5	fL	79,0 - 98,0
MCH	MCH	28,3	p g	26 - 32
MCHC	MCHC	34,3	g/dl	32-36
RDW	RDW	13,7	%	11,5 - 14,5

ΑΙΜΟΠΕΤΑΛΙΑ		Τιμή	Μ.Μ.	Φ.Τ.
PLT	ΑΙΜΟΠΕΤΑΛΙΑ (PLT)	213	K/μL	140 - 440
MPV	Μέσος όγκος PLT	7,6	fL	7,0 - 10,0
PDW	Εύρος κατανομής PLT	12,0	fL	12,0 - 18,0
PCT	Αιμοπεταλιοκρίτης	0,162	%	0,140 - 0,420

ΜΙΚΡΟΣΚΟΠΙΚΗ ΕΞΕΤΑΣΗ

Οξεόφιλα -	ΜΕΤΑΜΥΕΛΟΚΥΤΤΑΡΑ -	ΥΠΟΧΡΩΜΙΑ -	Στοχοκυτταρωση -
Αμφοκύτταρα -	ΜΥΕΛΟΚΥΤΤΑΡΑ -	ΑΝΙΣΟΚΥΤΤΑΡΩΣΗ -	Σφαιροκυτταρωση -
Μονοκύτταρα -	ΠΡΟΜΥΕΛΟΚΥΤΤΑΡΑ -	ΜΙΚΡΟΚΥΤΤΑΡΩΣΗ -	ΠΟΛΥΧΡΩΜΑΤΟΦΙΛΙΑ -
Ηωσινόφιλα -	ΕΡΥΘΡΟΒΛΑΣΤΕΣ -	ΜΑΚΡΟΚΥΤΤΑΡΩΣΗ -	Σχιστοκυτταρωση -
Βασεόφιλα -	Άτυπα -	ΠΟΙΚΙΛΟΚΥΤΤΑΡΩΣΗ -	ΒΑΣΕΟΦΙΛΟΣ ΣΤΙΞΗ -
ΡΑΒΔΟΠΥΡΗΝΑ -	Εμπ. Ερυθρ. -		

Παρατηρήσεις:

(b) Second Page.

Κωδικός Ασθενή: 146643 **Ημερομηνία Εξέτασης:** 16/04/2021 09:57
Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ ΝΙΚΟΛΑΟ **Ηλικία:** 19
ΕΞΩΤΕΡΙΚΟΣ ΑΣΘΕΝΗΣ

ΟΡΜΟΝΟΛΟΓΙΚΟ

Εξέταση	Τιμή	Μ.Μ.	Φυσιολογικές Τιμές
ΒΙΤΑΜΙΝΗ Β12	546	pg/ml	225 - 1000
ΦΕΡΡΙΤΙΝΗ	24	ng/ml	30 - 400
25 (OH) ΒΙΤΑΜΙΝΗ D3	18,1	ng/ml	ΜΕΓΑΛΗ ΕΛΛΕΨΗ : < 10 ΕΛΛΕΨΗ : < 20 ΑΝΕΠΑΡΚΕΙΑ : 20 - 29 ΕΠΑΡΚΕΙΑ : 30 - 100 ΤΟΞΙΚΟΤΗΤΑ : > 100

(c) Third Page.

Κωδικός Ασθενή: 146643 **Ημερομηνία Εξέτασης:** 16/04/2021 09:57
Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ ΝΙΚΟΛΑΟ **Ηλικία:** 19
ΕΞΩΤΕΡΙΚΟΣ ΑΣΘΕΝΗΣ

ΑΝΟΣΟΛΟΓΙΚΟ

Εξέταση	Τιμή	Μ.Μ.	Φυσιολογικές Τιμές
CRP (ΠΟΣΟΤΙΚΟΣ ΠΡΟΣΔΙΟΡΙΣΜΟΣ)	1,6	mg/L	< 5,0
ΑΝΟΣΟΣΦΑΙΡΙΝΗ IgE	90,27	IU/ml	< 160
C4 ΣΥΜΠΛΗΡΩΜΑ	23,0	mg/dl	10 - 40
C3 ΣΥΜΠΛΗΡΩΜΑ	122	mg/dl	90 - 180
ΑΝΤΙΠΥΡΗΝΙΚΑ ΑΝΤΙΣΩΜΑΤΑ ΑΝΑ	ΑΡΝΗΤΙΚΟ ΣΕ ΑΡΑΙΩΣΗ 1/80		ΑΡΝΗΤΙΚΑ : < 1/80 ΑΣΘ.ΘΕΤΙΚΑ : 1/80 - 1/160 ΘΕΤΙΚΑ : > 1/160
ΤΚΕ(ΤΑΧΥΤΗΤΑ ΚΑΘΙΖΗΣΗΣ ΕΡΥΘΡΩΝ)	5	mm/ 1h	έως 40 ετών : < 20 > 40 ετών : < (ηλικία/2)

(d) Fourth Page.

Figure 7: Second example of blood test results.

Κωδικός Ασθενή: 146643 Ημερομηνία Εξέτασης: 16/04/2021 09:58
Ονοματεπώνυμο: ΑΘΑΝΑΣΑΚΗΣ ΕΥΑΓΓΕΛΟΣ ΝΙΚΟΛΑΟ Ηλικία: 19
ΕΞΩΤΕΡΙΚΟΣ ΑΣΘΕΝΗΣ

ΓΕΝΙΚΗ ΟΥΡΩΝ

ΒΙΟΧΗΜΙΚΗ ΕΞΕΤΑΣΗ

ΑΝΤΙΔΡΑΣΗ (PH).	6.0	
ΕΙΔΙΚΟ ΒΑΡΟΣ	>= 1030	
ΛΕΥΚΩΜΑ ΟΥΡΩΝ	ΑΡΝΗΤΙΚΟ	g/l
ΑΙΜΟΣΦ/ΝΗ ΟΥΡΩΝ	ΑΡΝΗΤΙΚΗ	ery/µl
ΣΑΚΧΑΡΟ ΟΥΡΩΝ	ΑΡΝΗΤΙΚΟ	mmol/L
ΚΕΤΟΝΗ	ΑΡΝΗΤΙΚΗ	mmol/l
ΧΟΛΕΡΥΘΡΙΝΕΣ	ΑΡΝΗΤΙΚΕΣ	
ΧΟΛΟΧΡΩΣΤΙΚΕΣ	-	
ΧΟΛΙΚΑ ΑΛΑΤΑ	-	
ΟΥΡΟΧΟΛΙΝΟΓΟΝΟ	ΑΡΝΗΤΙΚΟ	
ΝΙΤΡΩΔΗ	ΑΡΝΗΤΙΚΑ	

ΦΥΣΙΚΗ ΕΞΕΤΑΣΗ

ΟΨΗ	ΔΙΑΥΓΗΣ
ΧΡΟΙΑ	ΚΙΤΡΙΝΗ

ΜΙΚΡΟΣΚΟΠΙΚΗ ΕΞΕΤΑΣΗ

ΠΥΟΣΦΑΙΡΙΑ(κοπ)	0 - 2
ΕΡΥΘΡΑ ΑΙΜΟΣΦΑΙΡΙΑ(κοπ)	0 - 2
ΕΠΙΘΗΛΙΑ(κοπ)	-
ΒΛΕΝΝΗ	ΑΡΚΕΤΗ
ΜΙΚΡΟΟΡΓΑΝΙΣΜΟΙ	-
ΚΥΛΙΝΔΡΟΕΙΔΗ ΒΛΕΝΝΗΣ	-
ΚΥΛΙΝΔΡΟΙ ΥΑΛΩΔΕΙΣ(κοπ)	-
ΚΥΛΙΝΔΡΟΙ ΥΑΛΟΚΟΚΚΩΔΕΙΣ(κοπ)	-
ΚΥΛΙΝΔΡΟΙ ΚΟΚΚΩΔΕΙΣ(κοπ)	-
ΚΥΛΙΝΔΡΟΙ ΑΙΜΟΡΡΑΓΙΚΟΙ(κοπ)	-
ΚΥΛΙΝΔΡΟΙ ΕΠΙΘΗΛΙΑΚΟΙ(κοπ)	-
ΚΥΛΙΝΔΡΟΙ ΚΗΡΩΔΕΙΣ(κοπ)	-
ΑΜΟΡΦΑ ΑΛΑΤΑ	-
ΚΡΥΣΤΑΛΟΙ	ΛΙΓΟΙ ΒΕΛΟΝΟΕΙΔΕΙΣ

(e) Fifth Page.

Figure 7: (continued) Second example of blood test results.

4.3.2 Field Extraction Tests

The purpose of this subsection is to evaluate the detection of fields in a given PDF document by the **Field Detection Agent** proposed in Section 3.1.6.

Parameters.

1. **Chunking and Overlap:** Text is split into chunks of 480 tokens with 20-token overlap using `SentenceSplitter`. Overlap is used in order to eliminate the risk of losing important contextual information that may occur at chunk boundaries, ensuring smoother transitions between chunks.

2. **Language model:** Qwen2.5:32B [32] is used for generating the final answer from the input chunks. The temperature is been set to 0 for consistent responses and easier evaluation.

Experimental Setup.

- **Input:** Chunks of translated, postprocessed text extracted from Soil Analysis PDFs of both sizes, Invoices and Blood Test Results.

- **Evaluation:** For each input PDF, a manually constructed JSON contains all the important fields that need to be extracted, as described in Section 4.1.2. Each field accounts for possible variations in naming (e.g., `name`, `Surname`) to ensure accurate matching, regardless of formatting or capitalization. The evaluation score is defined as the percentage of important fields correctly extracted from the total number of expected fields. Any additional fields that are not part of the important set but are extracted, are ignored during scoring.

- **Procedure:** Each PDF from the dataset is firstly chunked and then passed to the input of the Field Detection Agent, which produces a JSON object containing a list of extracted fields. The process is repeated, until all chunks have been processed. Each PDF is tested five times to compute an average score, ensuring consistent behavior and accounting for potential randomness in the language model's responses.

Results The results of the test are presented in Table 22. The overall accuracy, reported as **87%**, is calculated as a weighted average of the individual scores to account for the different number of documents in each category.

Document Type	**Field Detection Accuracy (%)**
Soil Analysis Reports	93.9
Invoices	72.7
Blood Test Results	90.1

Table 22: Field Detection Accuracy across different document types.

Observations And Analysis As seen from the results, there is a significant drop in the performance of the agent, when dealing with invoices. The cause of this drop lies in the structure of these documents, which is more inconsistent and complex than the other two types, with tables often lacking clear separators, such as lines. Even after post-processing, the format remains challenging for the agent leading to weak performances and lack of understanding of the format.

4.3.3 Prompt Builder Tests

The purpose of this subsection is to demonstrate the capabilities of the Prompt Builder Agent, introduced in Section 3.1.8, in constructing prompts based on the provided instructions.

Parameters.

1. **Chunking and Overlap:** Text is split into chunks of 480 tokens with 20-token overlap using `SentenceSplitter`. Overlap is used in order to eliminate the risk of losing important contextual information that may occur at chunk boundaries, ensuring smoother transitions between chunks.
2. **Language model:** Qwen2.5:32B [32] is used for generating the final answer from the input chunks. The temperature is been set to 0 for consistent responses and easier evaluation.

Experimental Setup.

- **Inputs:** Receives a list of extracted field names (e.g ph, name, address, etc.) from a document.
- **Evaluation:** The Prompt Builder Agent is evaluated on its ability to correctly follow the provided instructions and construct a valid, unambiguous prompt for the extraction method. The evaluation considers whether the generated prompt enforces the required constraints and contains all necessary elements. The key evaluation criteria are:
  - **Rule 1: JSON requirement.** The produced prompt must explicitly instruct the extraction method to return output strictly in JSON format, without any additional text or commentary.
  - **Rule 2: Schema conformity.** The prompt must explicitly contain the requested output format that consists of a full JSON schema with `value`, `unit`, and `method` keys for every field.

```
{
  "<field_1>": {"value": value,
                "unit": unit,
                "method": method},

  "<field_2>": {"value": value,
                "unit": unit,
                "method": method},
      ...
```

```
}
```

- **Rule 3: Field completeness.** The prompt must include all fields specified in the input list.
- **Rule 4: Missing values handling.** The prompt must state that missing attributes must be filled with empty strings ("").
- **Rule 5: Separation of value and unit.** The prompt must enforce the rule that `value` and `unit` are strictly separated (the unit must not be included in the value).

The production of each prompt gets a score from 0/5 to 5/5, since there are 5 rules that must be followed.

- **Procedure:** Fields extracted from the Field Extractor Agent are passed through the input of the agent. For batches of 5 fields, the agent creates the corresponding prompts. Each prompt is evaluated with the criteria described above. The procedure is repeated for all the dataset.

Results: The results of the experiments are given in the following table:

Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Total Score (Avg.)
100 %	98 %	100 %	100 %	100 %	99.6 %

Table 23: Evaluation of the Prompt Builder Agent across individual criteria and total average score.

Example Output: For the input field names: *Surname, Name, LocalCommunityRegion, FieldLocation, Crop* the agent produces the following response:

```
You are an Extractor Agent tasked with extracting the following fields
from the source document:
Surname, Name, LocalCommunityRegion, FieldLocation, Crop.

Provide your output in this exact JSON format:

{"Surname": {"value": "", "unit": "", "method": ""},
 "Name": {"value": "", "unit": "", "method": ""},
 "LocalCommunityRegion": {"value": "", "unit": "", "method": ""},
 "FieldLocation": {"value": "", "unit": "", "method": ""},
 "Crop": {"value": "", "unit": "", "method": ""}
 }.
If a value is not found, set Value="", unit="", method="".
SEPARATE the value, from the unit. Do not include the unit in the value.
Return ONLY valid JSON.
Do NOT add any introductory or explanatory text - return the dictionary ONLY.
```

Observations And Analysis The evaluation demonstrates that the Prompt Builder Agent achieves near-perfect accuracy across all defined rules. The only small inconsistency appears in the 2nd rule, which is the most complex one. This outcome can be attributed to the relative simplicity of the task and the limited input size, which consists only of the instruction template and the list of fields leading the agent to achieve great performance.

4.3.4 Multi-Agent Configuration Tests

This section presents the evaluation of the different architectures of the multi-agent system described in Section 3.2.2. The configurations (A, B, C, and D) explore variations in system design, including the integration of an intermediate Postprocessor Agent between the Field Detection and the Prompt Builder Agent, as well as the choice of retrieval method employed in the final stage, namely Chunk-Level Retrieval (Section 3.1.5) and classic RAG Retrieval (Section 3.1.4).

Parameters

1. **Chunking And Overlap:** All of the Field Detector, Post Processor Agents and the Chunk-Level Retrieval method recieves the text splitted in chunks. Text is split into chunks of 480 tokens with 20-token overlap using `SentenceSplitter`. Overlap is used in order to eliminate the risk of losing important contextual information that may occur at chunk boundaries, ensuring smoother transitions between chunks.

2. **Language model:** Qwen2.5:32B [32] is used from every agent for generating the responses. The temperature is been set to 0 for consistent responses and easier evaluation. The same model and temperature is also used by the Chunk-Level Retrieval method.

3. **RAG Parameters**

   (a) **Top-k retrieval ($k = 3$):** For each prompt, the 3 most relevant chunks are retrieved based on semantic similarity.

   (b) **Chunking:** Text is split into chunks of 500 tokens with 50-token overlap using `SentenceSplitter`.

   (c) **Context window:** The LLM context length is set to 4096 tokens, allowing sufficient room for both prompt and retrieved content.

   (d) **Embedding model:** `intfloat/multilingual-e5-large` [42] from HuggingFace is used for generating dense vector representations of text.

   (e) **Language model:** `Qwen2.5:32B` [32] is used for generating the final answer from the input chunks. The temperature is been set to 0 for consistent responses and easier evaluation.

Experimental Setup

- **Inputs** The extracted, post-processed, translated text from PDFs in the expanded dataset consisting of both small and large soil analysis reports, invoices and blood test results.

- **Requested Output:** For each field, the schema that must be produced is the following:

```
{
  "field_name": {
    "value": "",
    "unit": "",
    "method": ""
  }
}
```

- **Evaluation:** Manual prototype JSONs were used for ground truth comparison, as described in Section 4.1.2. Each JSON contains the fields together with their values, units and methods.

- **Procedure:** For every input PDF, its content is extracted, post-processed, translated and then passed through the multi-agent system. At first, the Field Detector Agent extracts the possible field names. Depending on the configuration, the Post-processor Agent evaluates and refines the extracted fields. This step is present only in the Configurations C and D. The fields pass through the Prompt Builder for prompt construction. Finally, the prompts pass through one of the two different retrieval methods. This procedure is repeated 5 times for every PDF in the dataset to get the average accuracy, ensuring consistent behavior and accounting for potential randomness in the language model's responses.

Metrics And Results

- **Value Accuracy:** Correctness of values extracted for the fields.
- **Unit Accuracy:** Correctness of units extracted for the fields.
- **Method Accuracy:** Correctness of methods extracted for the fields.
- **Total Accuracy:** The average from the value, unit, method accuracy.
- **Correct Fields:** The ratio of correctly extracted field names to the total number of fields that were expected to be extracted.

Results: The results are reported separately, according to the type of input PDF, with a dedicated table for each document category. The tables contain the dedicated accuracy of each one of the value, unit and method parameters, a total accuracy and the percentage of the correctly extracted field names. An additional summary table is provided at the end to present the overall performance across all types. For Soil Analysis Reports specifically, two tables are included. The first one (Table 24) reports results for the subset of approximately 30 fields that were originally targeted in Part 1 (Section: 4.2.5) tests of the thesis. The second table (Table 25) provides the overall performance for Soil Analysis Reports, when using the Field Detection Agent from Part 2, which considers a much larger set of fields. Importantly, the original 30 fields from Part 1 are fully contained within the broader field set of Part 2, enabling both focused evaluation on the original subset and overall assessment on the extended set.

Configuration	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)
A	**88.8**	**87.7**	**81.2**	**85.9**
B	81.7	80.9	73.4	78.7
C	82.2	85.0	79.2	82.1
D	77.5	79.5	75.2	77.4

Table 24: Performance of each Multi-Agent Architecture for Part 1 Set Soil Analysis Reports

Configuration	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Correct Fields (%)
A	77.0	77.0	73.0	75.7	93.9
B	**78.3**	**77.5**	**73.3**	**76.4**	**93.9**
C	70.0	74.2	71.0	71.7	90.1
D	73.4	76.0	72.2	73.9	90.1

Table 25: Performance of each Multi-Agent Architecture for Part 2 Soil Analysis Reports

Configuration	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Correct Fields (%)
A	56.4	67.5	71.8	65.2	72.7
B	**57.8**	**67.0**	**71.8**	**65.5**	**72.7**
C	49	63.3	61.8	58.0	62.6
D	51.5	59.8	62.8	58.0	62.6

Table 26: Performance of each Multi-Agent Architecture for Invoices

Configuration	Value (%)	Unit (%)	Method (%)	Total Accuracy (%)	Correct Fields (%)
A	80.6	83.0	82.0	81.9	90.1
B	**80.3**	**83.3**	**83.3**	**82.3**	**90.1**
C	75	76.3	75.7	75.7	83.2
D	75.3	76.7	76	76.0	83.2

Table 27: Performance of each Multi-Agent Architecture for Blood Test Results

Depending on the input size, the execution time of the agents varied significantly. Specifically, the Field Detection Agent required between **10–40 seconds**, the Field Postprocessing Agent between **15–60 seconds**, and the Prompt Builder Agent between **40–160 seconds**.

Observations And Analysis

- **Effect of the Postprocessor Agent:** The introduction of the Field Postprocessor Agent, although designed to filter noise and clarify ambiguous fields, does not lead to overall improvements in performance. This is because, in several cases, it mistakenly removes fields of actual interest, resulting in a decrease in the number of correctly extracted values. The cost of losing valid fields outweighs the benefits of cleaning noisy ones.

- **Relation between Correct Fields and Overall Accuracy:** The results show a strong correlation between the percentage of correctly detected fields and the overall extraction accuracy. When a field is not detected, it is excluded from the prompt built for the retrieval phase, meaning that no extraction instruction is passed forward. Consequently, the completeness of field detection is a critical factor in determining the quality of the overall pipeline.

- **Comparison with Part 1 Field Subset:** As seen in Table 24, the performance on the subset of approximately 30 fields is the same or slightly better in Part 2 compared to Part 1. This may be attributed to the improved prompts generated by the Prompt Builder in Part 2, which are smaller, more structured and precise than those used in the first phase.

- **Configuration Comparison:** In general, Configurations A and B demonstrate superior performance compared to C and D. Between the two, Configuration A has a slight advantage, achieving marginally higher overall accuracy and correct field percentages across most document types. This suggests that the addition of the Postprocessor Agent in Configurations C and D does not provide consistent benefits.

- **Performance on Invoices:** The most pronounced performance drop is observed in invoices, where total accuracy falls **below 66%**. This decline is primarily due to the inconsistent formatting of invoices, which often lack clear tabular structures or separators, such as lines. Even with additional processing, the irregularity and variability of invoice layouts present a significant challenge for field detection and extraction.

5 Conclusion

5.1 Summary

This thesis project aimed to develop and evaluate methods for extracting and retrieving information from semi-structured PDF documents, focusing on Greek soil analysis reports at first, but later generalizing across different domains, such as invoices and blood test results. The objective was to design an effective pipeline that combines text extraction, postprocessing, translation, and retrieval augmented generation (RAG).

Both native PDF extraction and OCR-based approaches were tested to address digital and scanned documents. Postprocessing module corrected errors and improved formatting, while translation standardized the text into English. Several retrieval workflows were compared, and a Field Extraction Multi-Agent System was introduced, consisting of detection, postprocessing, and prompt-building agents to structure field information for RAG.

Experiments showed that OCR on Greek scanned documents is highly dependent on scan quality, whereas native extraction was more reliable. Postprocessing and translation significantly improved text quality, and the multi-agent framework proved effective in automating the extraction process, by identifying fields of interest, constructing prompts for retrieval methods and finally extracting the values, units and methods of each field.

Furthermore, the experiments demonstrated that RAG achieved comparable accuracy to full-context prompting in small documents, while also scaling effectively to larger reports, overcoming the context length limitations of LLMs.

Overall, the thesis contributes a comparative study of extraction and retrieval tasks on technical PDFs, the design of a multi-agent framework for field extraction, and insights into the application of large language models and RAG to heterogeneous document processing.

5.2 Limitations

Despite the positive outcome of data retrieval automation, several limitations of the present research must be acknowledged. A primary challenge lies in the accuracy of text extraction, which directly affects the quality of subsequent retrieval. In particular, errors or inconsistencies introduced during extraction propagate through the pipeline, reducing the reliability of the final results.

A further limitation concerns the use of OCR for scanned PDF documents. Due to the Greek content and the variability in scan quality, OCR-based extraction often produced incomplete or noisy outputs, highlighting the sensitivity of this approach to document quality.

Resource constraints also imposed restrictions. LLMs require a huge amount of computational resources, especially on GPU memory, in order to be invoked. The experiments were conducted using locally executed large language models, which limited the scale of testing and the range of models evaluated. Inference time was another bottleneck, as multi-stage pipelines—particularly those involving postprocessing, translation, and RAG retrieval—introduced significant delays, making deployment in real-time applications challenging.

Finally, even when the models were tuned to produce deterministic responses by setting the temperature to zero, instances of unexpected outputs still occurred. This

inconsistency posed challenges during the evaluation phase, as it required multiple experimental runs in order to obtain reliable average results.

These limitations underline the dependence of the proposed methods on both document quality and computational resources, and they point to areas where future work can improve robustness and efficiency.

5.3 Future Work

The field of document understanding and retrieval is highly dynamic, with new models and techniques emerging at a rapid pace. Consequently, there are several directions for extending the present research.

Firstly, future work can explore alternative models at all stages of the pipeline. More recent embedding models, as well as advanced LLMs, may be integrated both within the agents of the multi-agent system and in the supporting modules, potentially improving performance and efficiency.

Secondly, additional methods can be investigated in the extraction phase. Beyond the current text-based libraries, machine learning approaches specifically designed for document layout analysis could be applied. Vision-language models may also be employed either for direct extraction or for reconstructing complex document layouts, offering more robust handling of heterogeneous inputs. Moreover, machine learning models can be fine-tuned on unstructured PDFs in order to automatically recognize and classify fields within the content.

Finally, the retrieval process itself can be extended by adopting more sophisticated paradigms, such as agentic RAG, where autonomous agents manage retrieval strategies adaptively. Such approaches may enhance scalability, robustness, and adaptability in real-world applications.

References

[1] Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. Prior disambiguation of word tensors for constructing sentence vectors. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[2] J. Ignacio Toledo, Sounak Dey, Alicia Fornés, and Josep Lladós. Handwriting recognition by attribute embedding and recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1038–1043, 2017.

[3] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20, page 1192–1200. ACM, August 2020.

[4] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding, 2022.

[5] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking, 2022.

[6] Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer, 2022.

[7] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. Docformer: End-to-end transformer for document understanding, 2021.

[8] Microsoft Azure Architecture Blog. Complex data extraction using document intelligence and rag. https://techcommunity.microsoft.com/blog/azurearchitectureblog/complex-data-extraction-using-document-intelligence-and-rag/4267718, 2024. Accessed: 2025-09-11.

[9] J. Wright. pdfplumber: Extract text, tables, and metadata from pdf files. https://www.pdfplumber.com/, 2025. Version 0.11.0, Accessed: 2025-09-11.

[10] Unstructured Technologies, Inc. Unstructured: Document partitioning into semantic elements. https://github.com/Unstructured-IO/unstructured, 2025. Version 0.15.0, Accessed: 2025-09-11.

[11] A. Pearson and the PyMuPDF developers. Pymupdf (fitz): Python bindings for mupdf. https://pymupdf.readthedocs.io/, 2025. Version 1.24.9, Accessed: 2025-09-11.

[12] J. Parnell and contributors. Docling: Structured text extraction for pdf documents. https://github.com/DS4SD/docling, 2025. Version 0.3.4, Accessed: 2025-09-11.

[13] K. Finney. Ocrmypdf: Adds an ocr text layer to scanned pdf files. https://ocrmypdf.readthedocs.io/, 2025. Version 15.4.0, Accessed: 2025-09-11.

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

[15] Jeffrey L. Elman. Finding structure in time. https://doi.org/10.1207/s15516709cog1402_1, 1990. Published in Cognitive Science, 14(2), 179–211.

[16] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

[17] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022.

[18] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024.

[19] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023.

[20] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023.

[21] Jörg Tiedemann. The tatoeba translation challenge – realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online, November 2020. Association for Computational Linguistics.

[22] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal, November 2020. European Association for Machine Translation.

[23] Helsinki-NLP team and contributors. Helsinki-nlp/opus-mt-tc-big-el-en on hugging face. https://huggingface.co/Helsinki-NLP/opus-mt-tc-big-el-en, 2020. Accessed: 2025-09-11.

[24] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. Beyond english-centric multilingual machine translation, 2020.

[25] Facebook AI Research (FAIR) team and contributors. facebook/m2m100_418m on hugging face. https://huggingface.co/facebook/m2m100_418M, 2020. Accessed: 2025-09-11.

[26] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[27] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey, 2024.

[28] Harrison Chase and contributors. Langchain: Building applications with llms through composable components. https://www.langchain.com/, 2023. Version 0.1+, Accessed: 2025-09-11.

[29] Jerry Liu and contributors. Llamaindex: A data framework for llm applications. https://www.llamaindex.ai/, 2023. Version 0.10+, Accessed: 2025-09-11.

[30] Ollama contributors. Ollama: Run llms locally. https://ollama.com/, 2023. Version 0.1+, Accessed: 2025-09-11.

[31] Hugging Face team and contributors. Hugging face: Transformers, datasets, and tools for machine learning. https://huggingface.co, 2016. Accessed: 2025-09-11.

[32] Alibaba Group and contributors. Qwen2.5:32b on ollama model library. https://ollama.com/library/qwen2.5:32b, 2024. Accessed: 2025-09-18.

[33] Ollama. Llama 3.1. https://ollama.com/library/llama3.1. Accessed: 2025-09-19.

[34] Institute for Language, Speech Processing (ILSP), and contributors. ilsp/llama-krikri-8b-instruct on hugging face. https://huggingface.co/ilsp/Llama-Krikri-8B-Instruct, 2024. Accessed: 2025-09-11.

[35] Hugging Face. all-mpnet-base-v2. https://huggingface.co/sentence-transformers/all-mpnet-base-v2. Accessed: 2025-09-19.

[36] Hugging Face. all-minilm-l6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed: 2025-09-19.

[37] Hugging Face. st-greek-media-bert-base-uncased. https://huggingface.co/dimitriz/st-greek-media-bert-base-uncased. Accessed: 2025-09-19.

[38] Nils Reimers and contributors. sentence-transformers/paraphrase-multilingual-minilm-l12-v2 on hugging face. https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2, 2020. Accessed: 2025-09-18.

[39] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[40] Dimitris Roussis, Leon Voukoutis, Georgios Paraskevopoulos, Sokratis Sofianopoulos, Prokopis Prokopidis, Vassilis Papavasileiou, Athanasios Katsamanis, Stelios Piperidis, and Vassilis Katsouros. Krikri: Advancing open large language models for greek, 2025.

[41] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[42] Luyu Wang and contributors. intfloat/multilingual-e5-large on hugging face. https://huggingface.co/intfloat/multilingual-e5-large, 2023. Accessed: 2025-09-18.