



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Department of Computer Science and Engineering

Master Degree in Artificial Intelligence

A Comparison between LLMs and SLMs for Document Processing in the Insurance Sector

Supervisor:
Prof. Paolo Torroni

Defended by:
Alice Turrini

Co-supervisors:
Lorenzo Caimi
Ludovico Granata

Graduation Session: March 2025
Academic Year 2023/2024

Abstract

This thesis provides a comparison over feasibility and performance between state-of-the-art large language models (LLMs) and smaller language models (SLMs), for the task of document classification and data extraction of a real-world case scenario. The research focuses on the development of a robust document processing pipeline. Starting from the raw PDF and encompassing all the necessary steps to obtain a structured format suitable for classification and subsequent metadata extraction.

Modern techniques are integrated throughout the pipeline to ensure efficiency and scalability. The project leverages a dataset of over 8,000 documents, including both labeled and pseudo-labeled data, in the medical and administrative domains. Specifically, the study compares the use of advanced LLMs, particularly GPT-4o, against smaller language models, BERT and LLaMA 3.2, for document classification and key metadata extraction. Key challenges addressed include the efficient extraction of meaningful information from complex domain documents, optimization of model performance for both classification and extraction tasks, and scalability of the proposed methods.

A central focus of this research is identifying the optimal balance between model size and performance. This is explored through fine-tuning smaller models, applying techniques such as knowledge distillation and model quantization, and comparing their results to those of larger models.

Results suggest that finetuning small language models for specific tasks can achieve performance comparable to, or in some cases surpass, LLMs, especially when considering model size and computational efficiency. These findings provide valuable insights for the modern topic of choosing between solutions based on LLMs or SLMs, taking into consideration various aspects such as performances, deployment, privacy, personalization, and cost.

*To myself (yes, we did it!)
and to all those who stood by me,
near or far.*

Contents

Abstract	3
1 Introduction	1
1.1 Motivation	1
1.2 GenAI	2
1.3 Real-world IDP	4
2 Background	7
2.1 IDP pipeline and technology used	7
2.1.1 Optical Character Recognition (OCR)	8
2.2 Language Models	9
2.2.1 LLMs	9
2.2.2 SLMs	12
2.2.3 Differences	18
2.3 Thesis's objective	24
2.4 Models used in the project	25
2.4.1 BERT	25
2.4.2 GPT	27
2.4.3 LLaMA	30
3 Data	35
3.1 Challenges	40
3.2 Dataset exploration	42
3.3 Data preprocessing pipeline	43
4 Classification	45
4.1 Classification model with BERT	46
4.1.1 BERT For Long Texts	48
4.1.2 Performance metrics	51
4.2 Prompting with GPT	54
4.2.1 Different prompt strategies	55
4.2.2 Performance metrics	60
4.3 Comparison and Discussion	61
5 Extraction	65
5.1 Evaluation criteria	65
5.2 LLM	69
5.2.1 Performances	71

5.3	SLMs	72
5.3.1	Knowledge Distillation	73
5.3.2	QLoRA	74
5.3.3	Data augmentation	76
5.3.4	Performances	78
5.4	Comparison and Discussion	79
6	Conclusion	83
6.1	Addressing the initial questions	83
6.1.1	Implications for Real-World Pipelines	84
6.2	Future Research	85
A	Document examples	87
		93
	Bibliography	93

Chapter 1

Introduction

1.1 Motivation

The transition to digital documents has changed how we store and handle information, leading to the adoption of various software tools for managing digital versions of physical paperwork. As the volume of digitalized documents has grown exponentially in recent years, the demand for efficient and accurate document management systems has become increasingly evident. Automated document processing addresses this need by utilizing intelligent systems to interpret and handle documents autonomously.

Intelligent Document Processing (IDP) integrates optical character recognition (OCR)[1], artificial intelligence (AI), and robotic process automation (RPA), allowing machines to understand and interpret information similarly to human cognitive processes. By leveraging these technologies, IDP automates data entry, extraction, and validation workflows, decreasing the need for manual input and enhancing overall efficiency.

The increasing volume and variety of digital documents present significant challenges for businesses and organizations worldwide. Traditional manual document processing is time-consuming, error-prone, and inefficient, often leading to wasted resources and reduced productivity. Studies [2] have shown that 83% of workers spend 1-3 hours daily reading and fixing errors in files, and 73% spend a similar amount of time simply searching for information in specific documents.

The inefficiencies associated with manual data handling are compounded by

the risk of human error and the repetitive nature of these tasks, which are neither engaging nor the best use of human cognitive abilities. These issues underscore the need for automated solutions that can handle large volumes of data with greater speed and accuracy.

Automated document processing, indeed, offers several advantages, including higher reading speed, increased productivity and scalability. By eliminating labour-intensive data entry tasks, organizations can reallocate human resources to higher-value activities. As a result, businesses in industries such as finance, healthcare, insurance, and real estate, which all have to deal with a huge amount of documents, can enhance their operational efficiency.

Despite these benefits, automated document processing has various challenges. Current models, while powerful, can still introduce errors and are often highly specialized for specific tasks and datasets. Generalizing automation remains a complex issue, requiring ongoing research and development to create more flexible and adaptable systems.

1.2 GenAI

Generative Artificial Intelligence (GenAI) includes all AI systems that can generate original content such as texts, images, videos, audio, or even code. These systems are trained on huge amounts of various data, and thanks to deep learning and innovative technologies, such as the transformer architecture and the attention mechanism, they try to mirror human creativity.

Historically, businesses have primarily worked with structured data, such as well-organized information like databases, which constitutes only about 10% of the total available data. However, the remaining 90% of the data is unstructured, consisting of formats such as videos, images, pdf documents, chat, emails, reviews, etc. GenAI represents a transformative shift in the way organizations approach data, it opened up the potential to process and leverage this vast pool of unstructured data, enabling deeper insights and more intelligent automation.

With the emergence of GenAI, the focus on data has intensified, compelling companies to transition toward fully data-driven organizations. This shift is not merely technological but strategic, pushing businesses to reevaluate their processes

and decision-making frameworks. By leveraging GenAI techniques, companies can automate workflows and increase productivity across different levels. Despite its potential, this technology also brings several challenges, including data quality management, model interpretability, and the need for robust governance to mitigate biases and ethical concerns.

At this point, it is clear that organizations that successfully integrate GenAI into their operations gain a significant competitive advantage. In recent years, a historical, technological transition has started, in which the use of GenAI inside companies' businesses will become the new standard. The main characters will be intelligent agents, which can handle repetitive, rule-based tasks while also addressing more complex activities that require contextual understanding and specialized learning. This capability allows businesses to scale their operations quickly, reduce costs, and empower employees to focus on higher-value work.

Certain features become essential when choosing automation tools to ensure reliability, efficiency, and speed. Solutions should include built-in monitoring to manage large-scale automated tasks without disruption and human-in-the-loop validation to improve accuracy. Flexibility and personalization of intelligent systems is also important, to adapt them for different needs. These capabilities align closely with the **objectives of this research**, which aims to explore the role of advanced AI techniques in document processing and metadata extraction. This research aims to explore GenAI's capabilities to establish a basis for building effective, scalable, and intelligent document processing workflows.

Within the broader domain of GenAI, one of the most influential technologies nowadays is **large language models (LLMs)**. These models, typically trained on massive text datasets from diverse sources, leverage deep learning architectures to understand, generate, and manipulate human language. They demonstrate exceptional fluency and contextual awareness and achieve performances in many tasks almost comparable to humans. LLMs are made of hundreds of billions of parameters, which enable them to capture complex patterns and deep relationships in language. Their huge scale empowers them to perform a wide, diverse range of natural language tasks. The popularity of LLMs arises from their versatility and state-of-the-art performance. They are trained on vast corpora, creating a large

background knowledge, and then they can be fine-tuned on specific datasets. Their ability to comprehend nuanced language and generate human-like responses has made them already very useful in many applications. However, the enormous size and computational demands of LLMs introduce **several practical challenges**. Their deployment requires significant hardware resources, which increases their operational costs and energy consumption. The significant infrastructure requirements imply that not every organisation can afford to deploy and maintain them. Their studies and deployment are limited to the private entities, which then generally allow others to utilise their models through various payment methods. In addition, the size and complexity of these models can complicate real-time applications.

Despite their high performances and almost as humans, all these limitations motivated many researchers to explore **Small Language Models (SLMs)**, a class of language models with fewer parameters. SLMs aim to balance performance and efficiency, offering many of the capabilities of their larger counterparts while being more accessible, easier to deploy, and cost-effective.

This thesis will **compare LLMs and SLMs** in the context of automated document classification and metadata extraction. By evaluating their performance, efficiency, and applicability in real-world scenarios, it seeks to determine the feasibility of adopting smaller models without compromising on accuracy.

1.3 Real-world IDP

This study originated during an **internship** at a consulting company collaborating with a client in the **insurance sector**. The project is grounded in a real-world use case, where the client provided the input documents, specified the requirements, and outlined the expected outputs. The formulation of the core methodological approach, however, was independently designed by our team.

The **methodology** adopted for this project follows a structured pipeline. The first step involved collecting all the data supplied by the client, ensuring the presence of labeled data necessary for supervised learning and enabling the validation of model performance through appropriate metrics. Then, the data underwent preprocessing to create a coherent and uniform dataset. This phase included split-

ting multi-page documents into individual pages, saving each page as an image, and extracting the textual content from these images to build natural language processing models. Following the preprocessing phase, a multi-class classification model was developed to classify each document. The final stage involved an extraction model capable of retrieving metadata, specific for each class, from a given document's page in the format of text. Each model implemented underwent a rigorous validation process, with specific metrics and results that were critically assessed.

The overall objective of the project was to compare the performance of large language models and small language models in the two principal stages of the document processing pipeline: the classification and the metadata extraction. Figure 1.1 provides a high-level overview of the document processing pipeline. The objective is to extract specific metadata from documents. The pipeline consists of two main phases: a classification step followed by a metadata extraction step, in each stage a comparative analysis between LLMs and SLMs techniques is performed.

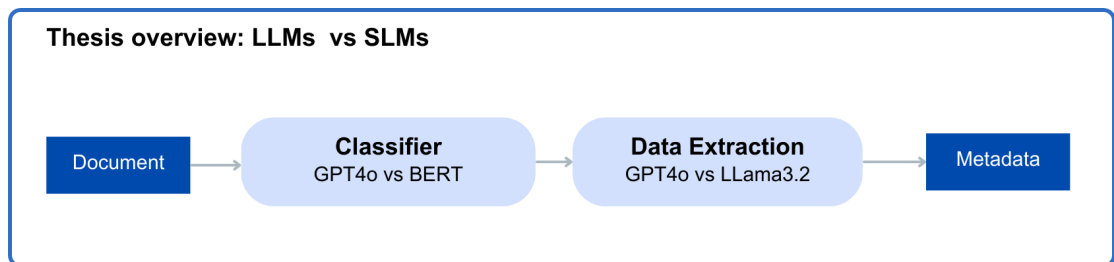


Figure 1.1: A high-level schema of the thesis's document processing pipeline, illustrating the document classification followed by the extraction of class-specific metadata. The focus of the project is the comparison between LLMs and SLMs techniques in these key steps.

In both phases, the LLM utilized was GPT-4o, an OpenAI state-of-the-art model, known for its exceptional performance across numerous NLP tasks. For the classification task, it was compared with BERT [3], a way smaller language model developed by Google. And for the extraction task, GPT-4o was evaluated against LLaMA 3.2 [4], a model from the LLaMA family developed by Meta [5]. Later in the background chapter 2.4 more specific details about the specific models used will be given.

Chapter 2

Background

2.1 IDP pipeline and technology used

Intelligent Document Processing (IDP) combines multiple AI technologies to automate the extraction, validation, and integration of data from various types of documents. By leveraging optical character recognition (OCR), machine learning, and robotic process automation (RPA), an IDP pipeline typically follows these key steps:

Uniformation: The strong aspect of IDP is the flexibility of the input format that it can process. And so, the first step is to uniform all the possible formats into one, usually an image that will be later transformed into text.

Preprocessing: The quality and accuracy of the images are enhanced through techniques such as noise reduction and image cropping. These methods improve the readability and interpretability of documents, ensuring better downstream processing.

Text Reconstruction: Leveraging OCR, the text of the document is extracted and transformed into a readable format.

Classification: Documents are categorized based on their format, structure, and content. This step enables the later data extraction and helps the processing pipeline for specific document types.

Data Extraction: Thanks to advanced AI models, metadata, such as names, dates, numbers, or even more complex concepts, are identified and extracted from the documents.

Validation: It is important to cross-check extracted data to ensure accuracy and consistency. Any flagged inconsistencies can be routed for human review.

Human-in-the-Loop Validation: Despite advanced automation, human oversight remains crucial for quality evaluation. Human feedback is used to refine AI models over time, enhancing their accuracy and adaptability. As powerful as automation technologies have become, there are still many scenarios in which human expertise is indispensable. Human-in-the-loop validation ensures that automated systems maintain high accuracy by incorporating human feedback into the pipeline. In IDP, this validation plays a critical role when dealing with highly variable documents, complex data structures, or low-quality inputs.

Integration: Once validated, the extracted data is seamlessly transferred into the business systems or databases through APIs, enabling automated workflows.

2.1.1 Optical Character Recognition (OCR)

Optical character recognition (OCR) [1] is a technology that digitalises printed texts into electronic format. It is a foundational technology in IDP and serves as the bridge between raw data and textual data. OCR systems analyse a document's visual structure and interpret individual characters, transforming them into a digital format that can be processed by downstream applications. Modern OCRs are now capable of producing accurate results from various types of inputs, such as scanned paper documents, handwritten PDFs, or images.

The underlying technology first cleans the images, classifying at pixel level darker areas as text and lighter areas as backgrounds. With this information, earlier versions analysed the classified image and identified each character, called a glyph, performing pattern matching or extracting symbolic characteristics. Modern OCR technologies, instead, leverage deep learning techniques and ICR (Intelligent Character Recognition), a neural network that mimics human reading of images to improve recognition accuracy, particularly when dealing with complex handwritten text. In this way, modern OCR models achieve high performance in text extraction, even from unstructured or noisy sources.

OCR technologies vary widely based on software and providers, with **AWS Textract** [6] standing out in the real-world document processing landscape. Un-

like traditional OCR, Textract excels at extracting not only text but also relationships and structures within documents, forms, and tables. Its pricing model is usage-based, meaning it depends upon the complexity of the text being analyzed—ranging from simple plain text or signatures to more intricate tables and forms. For users, this translates to a competitive cost of approximately \$15 for processing a million pages within a month, making it a practical solution for various document processing needs.

2.2 Language Models

2.2.1 LLMs

Language models are neural network architectures designed to process and generate natural language text. The underlying architecture is the well-known **transformer** with the **attention** mechanisms, concepts that made the revolution in AI in 2017 with "Attention Is All You Need" [7]. Unlike earlier recurrent neural networks (RNN) that sequentially process inputs, transformers process entire sequences. This allows the use of GPUs during the training process, which significantly reduces the training time, and allows these models to be trained on massive datasets containing billions of words from different contexts. Through this extensive training and the huge number of trainable parameters that these models have, they understand the complexity of language.

Due to their high performance in many NLP tasks and their almost human-like understanding of text during the last five years, they have been one of the most trending topics for research and development of AI. In Figure 2.1 it is shown a timeline with the most popular models released in the recent years, giving the idea of the saturation of this field of studies.

Historically, traditional language models, such as n-grams and statistical models, relied on probabilistic approaches with limited contextual understanding. In contrast, modern language models interpret long contexts of words and capture semantic meanings in a way that mimics human-like understanding. Their ability to generalize from limited labeled data makes them foundational for many contemporary AI-driven document processing systems.

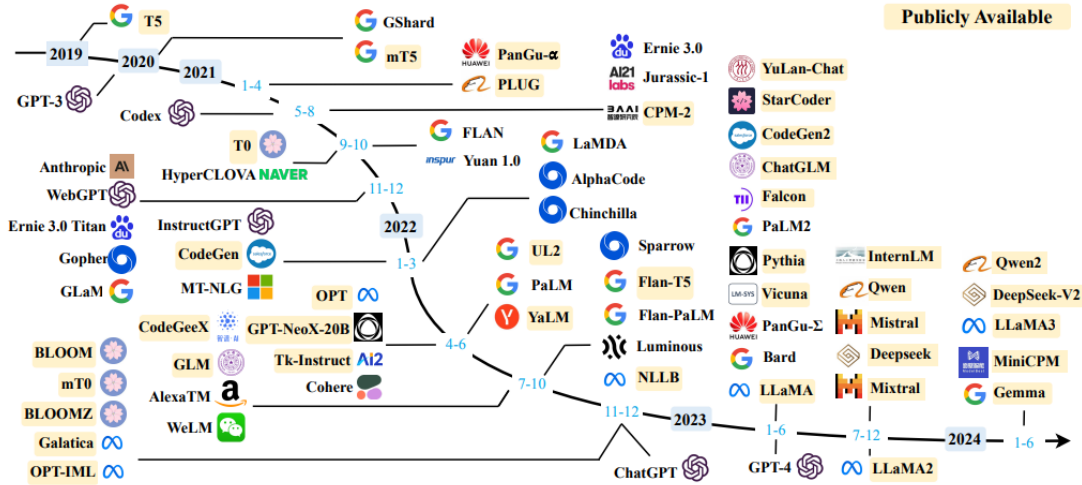


Figure 2.1: The schema comes from the paper “A Survey of Large Language models”[8], and it illustrates the timeline of existing LLMs in recent years. It was established mainly according to the release date (submission date to arXiv) of the technical paper for a model, or in case if there was no corresponding paper, it is reported the date of a model as the earliest time of its public announcement.

A key factor behind the effectiveness of these models lies in how they represent words. Early machine learning approaches like Bag-of-Words, Word2Vec, and GloVe represented words through static numerical tables, which failed to capture contextual relationships between words. This limitation was overcome by the advent of **word embeddings**, multi-dimensional vectors that position semantically similar words closer to each other in the vector space. By leveraging these embeddings, transformers pre-process text as numerical representations through the encoder and interpret the contextual meaning of words and phrases, including grammatical roles and semantic relationships. The decoder then applies this learned knowledge to generate coherent and contextually appropriate output.

Large pre-trained transformer models have demonstrated impressive capabilities in performing tasks for which they were not explicitly trained. However, task-specific performance can often be enhanced through techniques like in-context learning, fine-tuning and Prompt engineering.

In-context learning (ICL) refers to the capacity of pre-trained large language models to perform new tasks by leveraging information provided within the context window, without explicit parameter updates. Instead of adjusting weights via gradient descent, the model adapts its behavior based on examples or instruc-

tions provided in the prompt. This technique allows LLMs to generalize across a wide range of tasks using natural language interactions. Few-shot prompting, for instance, involves providing examples in the context window to help the model infer task structure and apply it to new inputs.

Fine-tuning, on the other hand, involves updating the weights of a pre-trained LLM by training it on task-specific datasets. Techniques like supervised fine-tuning (SFT), reinforcement learning with human feedback (RLHF) [9], and parameter-efficient fine-tuning (PEFT) [10] are among the most common approaches to adapt LLMs for specialized tasks.

Lastly, **Prompt engineering** represents another powerful method to optimize LLM performance without altering the model’s internal structure. By carefully crafting prompts, consisting of the model’s input, inference, and completion—users can maximize output quality for diverse tasks, from simple queries to complex problem-solving. In-context learning complements prompt engineering by providing illustrative examples within the context window:

- **Zero-shot inference:** no examples provided.
- **One-shot inference:** one example provided.
- **Few-shot inference:** multiple examples provided.

The ability of a model to learn and generalise from new data, is proportional to its size, which is determined by the number of parameters it possesses. The bigger the model is, the more data it can process and learn from, and indeed, the higher the performance on many linguistic tasks. Year after year, the dimension of the language models deployed has always increased, and the trend is growing exponentially. However, the increasing size and complexity of LLMs have made them resource-intensive, restricting their accessibility to organizations with substantial computational infrastructure. This underscores the intent of this thesis of exploring efficient alternatives, such as smaller models, which balance performance and resource efficiency. SLMs offer promising solutions for democratizing AI adoption across industries by reducing the costs and energy demands associated with LLM deployment.

2.2.2 SLMs

The term SLMs lacks a precise definition, it is used to denote language models that have a smaller number of parameters, typically fewer than 10 billion, highlighting their distinction from larger models containing hundreds of billions or even trillions of parameters.

Figure 2.2 presents some of the most used SLMs released over the past five years. As shown, the number of smaller models has increased in recent years, reflecting a growing interest in more efficient and accessible alternatives to large-scale models.

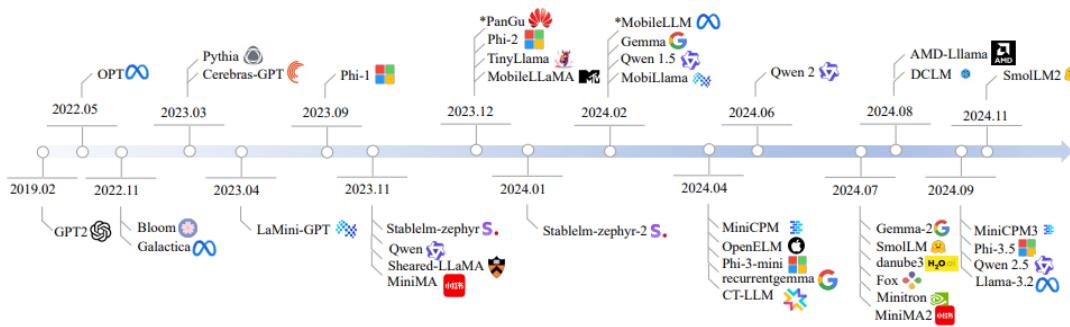


Figure 2.2: An overview timeline of SLMs. The information is taken from [11], in which they set 5B as the upper limit of the size of SLMs, but of course it changes.

Despite their reduced size, SLMs retain much of the capability of their larger counterparts thanks to sophisticated optimization techniques, including:

- **Pruning:** Removing redundant model parameters to reduce size and complexity.
- **Quantization:** Lowering numerical precision (e.g., from 32-bit to 8-bit) to decrease memory footprint and improve inference speed.
- **Low-rank factorization:** Decomposing weight matrices into lower-dimensional components to optimize efficiency.
- **Knowledge distillation:** Transferring knowledge from a large, pre-trained “teacher” model to a smaller “student” model, preserving essential capabilities while reducing complexity.

These techniques ensure that SLMs maintain high efficiency without compromising too heavily on accuracy.

Knowledge Distillation

LLMs present significant deployment challenges due to their enormous computational requirements. Traditional approaches to addressing this challenge have followed two primary paradigms: fine-tuning and standard distillation.

Fine-tuning involves updating pre-trained smaller models using labelled data specific to downstream tasks. While effective, this approach requires extensive human-generated labels, which are expensive and time-consuming to produce.

The idea of knowledge distillation [12] is instead to train smaller models using labels generated by larger models. It still requires large quantities of unlabeled data that can be challenging to collect in sufficient volumes [13]. This approach is based on the idea of transferring the final output distributions from teacher to student models, avoiding for the smaller model to learn all the intermediate reasoning processes that lead to those outputs. In this way, small models struggle to reach the reasoning capabilities that large models possess. This limitation becomes particularly pronounced when training data is scarce, as smaller models lack all the necessary supervision to develop robust reasoning pathways independently.

Despite these efforts, both approaches typically demand substantial amounts of training data to achieve performance comparable to few-shot prompted LLMs, presenting an unfortunate trade-off between model size and data collection costs. Step-by-Step Knowledge Distillation [14] represents a revolutionary approach in the field of knowledge distillation for language model optimization. It enabled significantly smaller models to outperform their much larger counterparts while requiring substantially less training data. This innovative methodology, introduced in the groundbreaking paper "Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes," demonstrated that a 770M parameter T5 model can surpass the performance of the 540B parameter PaLM model, achieving a remarkable 700x reduction in model size while using only 80% of the available training data. The core innovation of this method lies in exploiting LLM as intelligent reasoning agents and making them extract informative natural language rationales [15], which represents the intermediate reasoning steps. These rationales are used as additional, richer supervision in the training phase of the smaller models. These rationales contain valuable task

knowledge that would typically require extensive training data for small models to acquire independently. These reasons essentially teach them how to think through complex problems rather than just what answer to produce.

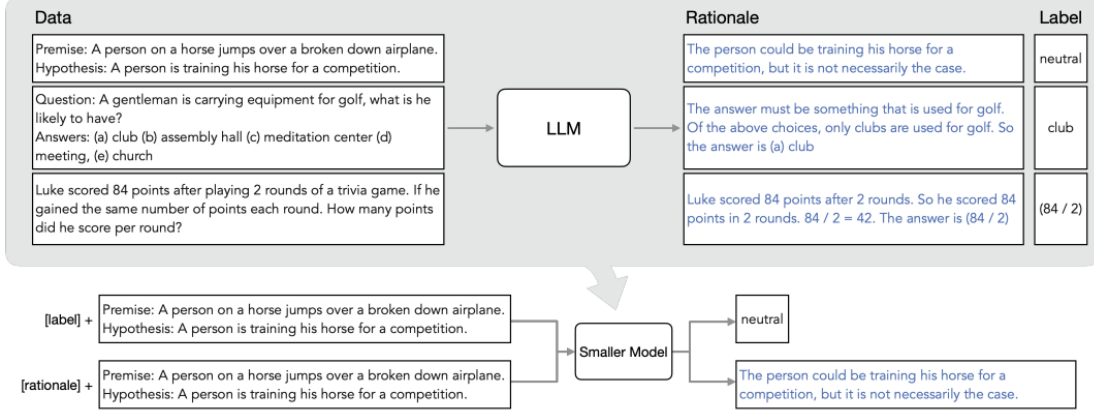


Figure 2.3: The Figure shows an overview of the Distilling step-by-step and it is taken from the original paper [14]. They first utilized CoT (Chain Of Thoughts) prompting to extract rationales from an LLM, on the top left. Then they used the generated rationales to train small task-specific models. They prepend task prefixes to the input examples, as shown in the bottom, and trained the model to output differently based on the given task prefix.

The conceptual framework builds upon **Chain-of-Thought (CoT) prompting techniques**, which have demonstrated LLMs' capacity for explicit step-by-step reasoning. This process involves preparing exemplars for the LLM input prompt, where each example contains a triplet of input, rationale, and output. The LLM then generalizes from these examples to generate rationales for new inputs through in-context learning.

There are multiple ways to incorporate the rationale into the training process, one of which is to input it into the small model. But this implies that at inference time the model needs the rationale, and this is a big disadvantage because at inference time a LLM is needed. To overcome this problem, Step-by-Step Knowledge Distillation, instead imposes the small model to output the reasons $f(x_i) \rightarrow (\hat{y}_i, \hat{r}_i)$, and it implements a multi-task learning approach that trains smaller models simultaneously on two related objectives.

The first objective is the standard label prediction task, while the second is a novel

rationale generation task that teaches models to produce intermediate reasoning steps. This dual-objective framework is formalized mathematically through this composite loss function that balances between accurate prediction and faithful rationale reproduction $L = L_{label} + \lambda L_{rationale}$.

The approach leverages two task prefixes `[label]` and `[rationale]` to differentiate between the two tasks during training, allowing the model to learn both capabilities in parallel, and easily delete `[rationale]` during the inference phase [16].

In conclusion, distilling step-by-step reduces the training dataset required to curate task-specific smaller models, and it also reduces the model size required to achieve, and even surpass, bigger models' performance.

QLoRa

An alternative optimization method is QLoRA [17], or Quantized LoRA, which minimizes GPU memory consumption during the training process. This approach is efficient for fine-tuning as it sufficiently decreases memory requirements, enabling the fine-tuning of a large language model on a single GPU, while maintaining the full performance of 16-bit fine-tuning tasks. QLoRA builds upon Low-Rank Adaptation (LoRA) [18] by introducing quantization to further optimize the fine-tuning process. The standard LoRA technique focuses on low-rank adaptation, which, during the fine-tuning, freezes the original model's parameters while training an external low-rank matrix that adapts those parameters. This technique significantly reduces the number of trainable parameters and memory footprint, making fine-tuning more efficient.

The innovative techniques used by QLoRA [17] to achieve these performance gains include:

- **4-bit NormalFloat (NF4):** a new data type that is information theoretically optimal for normally distributed weights. Unlike traditional quantization formats, NF4 better preserves the distribution characteristics of the original weights, leading to minimal performance degradation despite the lower precision.
- **Double Quantization:** a method that quantizes the quantization constants, reducing the average memory footprint. A small block size is required

for precise 4-bit quantization [19], but this also has a considerable memory overhead. Double quantization mitigates this by treating the quantization constants c^{FP32} from the first quantization as inputs to a second quantization step, yielding the final quantized quantization constants c^{FP8} . This two-step quantization process significantly reduces memory usage while maintaining high precision.

- **Paged Optimizers:** to manage memory spikes leveraging the CPU. It uses the NVIDIA unified memory [20] feature which does automatic page-to-page transfers between the CPU and GPU. This approach ensures error-free GPU processing even in scenarios where the GPU occasionally runs out of memory, improving the stability and efficiency of the fine-tuning process.

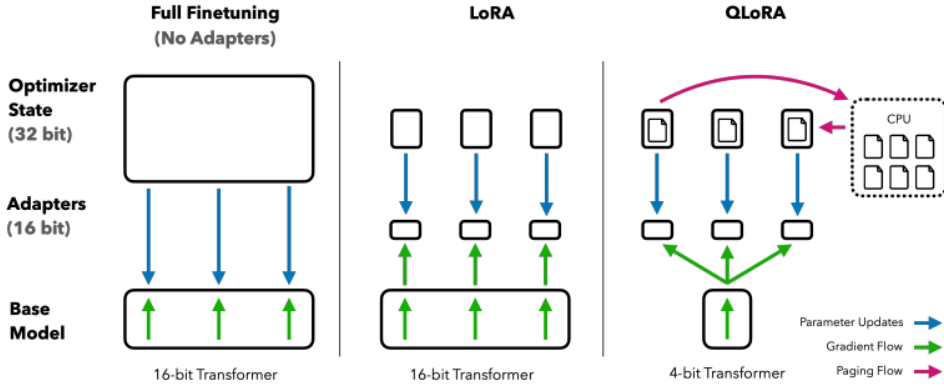


Figure 2.4: The figure shows different fine-tuning methods and their memory requirements, taken from the original [17]. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Regarding hyperparameters, the most critical factor is the number of LoRA adapters used. It is also essential to apply LoRA to all linear transformer block layers to match the performance of full fine-tuning.

Interestingly, accordingly to the official paper [17], other LoRA hyperparameters such as the projection dimension r , have minimal impact on performance.

Mathematically, LoRA operates on the principle that for a pre-trained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$, instead of fine-tuning all parameters, we can approximate the

weight update with a low-rank decomposition:

$$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{B}\mathbf{A} \quad (2.1)$$

where $\mathbf{B} \in \mathbb{R}^{d \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times k}$ are low-rank matrices with rank $r \ll \min(d, k)$, and the update is scaled by $\frac{\alpha}{r}$ during training.

This technique significantly reduces the number of trainable parameters from $d \times k$ to $r \times (d + k)$ and memory footprint, making fine-tuning more efficient.

QLoRA extends the LoRA approach with weight quantization, which reduces numerical precision to 4-bit values, creating a dual optimization strategy that balances memory efficiency and model performance.

The forward pass in QLoRA can be expressed as:

$$\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}} \text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{NF4}}) + \mathbf{X}^{\text{BF16}} \mathbf{L}_1^{\text{BF16}} \mathbf{L}_2^{\text{BF16}}, \quad (5)$$

where $\text{doubleDequant}(\cdot)$ is defined as:

$$\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{k-bit}}) = \text{dequant}(\text{dequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}), \mathbf{W}^{\text{4bit}}) = \mathbf{W}^{\text{BF16}} \quad (6)$$

where \mathbf{X}^{BF16} is the input, $\mathbf{W}^{\text{k-bit}}$ is the quantized pre-trained weight matrix, c_1^{FP32} and $c_2^{\text{k-bit}}$ are the quantization constants, and \mathbf{B}^{BF16} and \mathbf{A}^{BF16} are the trainable LoRA adapters in BFloat16 precision. By combining low-rank adaptation and 4-bit quantization, QLoRA achieves substantial memory savings without sacrificing the quality of the fine-tuning outcomes.

It is fascinating that QLoRA’s 4-bit quantization with the NF4 data type matches the performance of 16-bit full fine-tuning and 16-bit LoRA fine-tuning. Indeed, QLoRA is the first fine-tuning method that enables the fine-tuning of 33B-parameter models on a single consumer GPU and 65B-parameter models on a single professional GPU, without any degradation in performance relative to a full fine-tuning baseline. This remarkable efficiency and scalability make QLoRA a groundbreaking advancement in the field of efficient model fine-tuning, and worth implementing in the experiment of this project.

2.2.3 Differences

The comparison between Large Language Models and Small Language Models can be made across several dimensions, reflecting their different capabilities, resource requirements, and suitability for specific use cases. This section provides an in-depth analysis of these aspects, highlighting both the strengths and limitations of each approach.

Model Size The most fundamental distinction between LLMs and SLMs is the number of parameters, by definition. LLMs possess billions to trillions of parameters, enabling them to capture intricate language structures and extensive general knowledge. LLMs can even exceed the trillion-parameter threshold and demonstrate remarkable linguistic capabilities across diverse domains. This massive parameter count allows LLMs to maintain contextual understanding across lengthy inputs and generate nuanced, contextually appropriate responses.

In contrast, SLMs operate with significantly fewer parameters, generally falling below the 10 billion parameter threshold. This substantial reduction in model size creates a cascade of differences that affect nearly every aspect of model performance, deployment, and practical application. The parameter count distinction represents not just a technical classification but fundamentally shapes the models' capabilities and limitations. Indeed, the smaller architecture of SLMs necessarily constrains their ability to maintain complex contextual relationships but simultaneously enables advantages in computational efficiency and specialized applications.

Resource Requirements and Energy Consumption for Training Training an LLM is an exceptionally resource-intensive process, with substantial implications for accessibility and environmental sustainability. LLM training demands massive datasets and distributed high-performance computing infrastructure, typically requiring hundreds of GPUs or TPUs operating continuously for months. This extraordinary computational demand translates directly into significant financial costs, as high-performance computing hardware has high prices in both purchase and operational dimensions.

The environmental impact of LLM training and usage has become a topic of

increasing concern in the AI community. Major AI research organizations like Google and OpenAI have acknowledged that running their largest models can consume energy equivalent to several hundred households. This substantial carbon footprint raises important questions about the sustainability of increasingly large language models, particularly as deployment scales.

SLMs, on the other hand, require a fraction of these resources, making them more accessible for researchers and organizations with limited computational budgets, and of course more environment friendly. This accessibility democratizes AI development, enabling smaller research teams and organizations to participate meaningfully in language model advancement.

Inference Speed and Latency Reduction Due to their smaller size, SLMs offer significantly reduced inference times. They often achieve response faster than LLMs, which is crucial for real-time applications. This speed advantage is particularly beneficial in latency-sensitive environments. Applications such as interactive conversational agents, real-time document processing, or embedded systems benefit substantially from the reduced computational overhead of smaller models.

Request Efficiency Beyond single-request performance, SLMs excel in managing concurrent requests due to their lower per-instance resource requirements. This efficiency translates directly into improved scalability in production environments. Organizations can serve more simultaneous users with equivalent hardware resources, reducing infrastructure costs while maintaining responsive performance. The scalability advantage becomes particularly significant in high-volume applications, where resource constraints might otherwise force service degradation. The scalability advantage is particularly relevant in high-volume applications, where the ability to efficiently process large request loads can determine the feasibility of real-time document classification and metadata extraction systems.

On the other hand, LLMs often rely on API-based access, which imposes inherent constraints on parallelization. Each request is processed sequentially through the external service, limiting the possibility to batch requests or execute them concurrently. This dependency not only introduces potential bottlenecks but

also increases operational costs. Additionally, API-driven architectures introduce latency variations due to network dependency, making them less predictable in performance-sensitive environments.

Fine-Tuning Ease and Cost For LLMs, fine-tuning presents a significant challenge due to their immense computational and memory demands. Training or adapting these models typically requires high-end GPUs or specialized cloud infrastructure, making the process both very expensive and time-consuming. The cost of fine-tuning an LLM is further amplified by the need for extensive datasets and long training durations, often making it impractical for organizations with limited resources.

In contrast, one of the most significant practical advantages of SLMs lies in their superior adaptability through fine-tuning processes. While both model categories can theoretically be adapted to specific domains, the resource requirements for fine-tuning LLMs often prove prohibitive for many organizations. SLMs can be fine-tuned quickly and cost-effectively, enabling easy adaptation to specialized domains without extensive computational infrastructure. This accessibility democratizes the development of domain-specific language models, allowing a broader range of organizations to benefit from customized AI capabilities.

Advanced techniques have further enhanced the fine-tuning efficiency for SLMs. Methods like Low-Rank Adaptation (LoRA), adapters and quantization significantly streamline the fine-tuning process, reducing computational requirements while maintaining performance.

The domain adaptation capabilities of SLMs have proven surprisingly effective, with studies indicating that properly fine-tuned smaller models can retain approximately 90% of LLM performance in domain-specific applications. This finding challenges the assumption that larger models are always superior, suggesting that for many specialized use cases, the performance gap may be minimal while the resource difference remains substantial. Organizations can achieve comparable functional results within their specific domain while avoiding the significant costs associated with larger models.

Task Adaptability LLMs excel in handling complex, general-purpose tasks which require extensive prior knowledge and contextual understanding. They maintain coherence over long passages and generate detailed responses.

SLMs, instead, have less understanding of the language, and indeed, they are less adept at managing intricate language structures. However, they perform well in specialized domains with targeted optimizations. Studies show that SLMs retain approximately 90% of LLM performance in domain-specific applications, thanks to the easy fine-tuning process explained before.

Open-Source Availability Many leading LLMs remain closed-source proprietary technologies, as organizations that invest millions in their development understandably protect their competitive advantages. This restricted access concentrates advanced AI capabilities among well-resourced technology companies, raising concerns about AI democratization.

In contrast, the SLM ecosystem is more open-source, enabling widespread experimentation and adaptation. This openness fosters a more diverse development community and enables innovations from researchers and organizations that would otherwise lack access to cutting-edge language model technology. The collaborative nature of open-source development further accelerates improvement and specialization of these models across diverse applications.

Deployment Deployment architectures represent another critical dimension of difference between these model categories. LLMs typically require cloud deployment and centralizing processing in data centers with abundant resources, due to their substantial computational demands. This cloud dependency introduces several challenging considerations, including network latency, continuous connectivity requirements, and potential data privacy concerns as information travels to and from remote servers.

SLMs offer more flexible deployment options, including local installation on edge devices or organizational servers. This approach eliminates network transmission delays, resulting in faster response times. It also enhances data security by keeping sensitive information within internal systems and proves particularly valuable in environments with limited or unreliable connectivity.

Data Privacy and Security The data security and privacy implications of deployment architecture have become increasingly important as language models process more sensitive information. A widespread concern about LLM is the risk of exposing sensitive personal data when information is transmitted to external servers, or that the providers of the services could memorize the data to use it for the training process. Ensuring compliance with data protection regulations like GDPR is crucial in the European regions. As this topic has become more important due to the wide usage of the LLM from individuals or businesses, adopting techniques such as data minimization and anonymization to safeguard privacy, and specific subscription in specific regions are now available for most of the models. SLMs enhance privacy protection through on-device processing capabilities, reducing or eliminating the need to transmit potentially sensitive data to external servers. This capability proves particularly valuable in regulated industries like healthcare and finance, where data protection requirements impose strict limitations on information handling.

Cost-Effectiveness The economic considerations encompass the entire lifecycle, including training, deployment, operation, and adaptation. SLMs present substantially lower hardware and operational costs compared to LLMs, making AI language capabilities more accessible to organizations with limited technology budgets. This accessibility enables smaller organizations to leverage advanced language processing without prohibitive infrastructure investments.

The pricing models for cloud-based large language model services typically operate on a per-token basis for both input and output, creating direct relationships between usage volume and cost. The token-based pricing structure varies significantly between models, Table 2.1 shows the pricing of two large language models and a small language model to have a concrete idea of the costs. Inside the same

	GPT-4o	GPT-4o-mini	Claude Sonnet 3.5
Cost 1M input token	2.75 \$	0.165 \$	3 \$
Cost 1M output token	11 \$	0.66 \$	15 \$

Table 2.1: The Table shows the price in \$ for one million input and output tokens. For the GPT-4o model, the referred version is the 2024-08-06.

family and payment method, the small version of the GPT-4o model has approximately a 94% cost reduction.

Considering in more detail the use case of this project of document processing, these pricing differentials become particularly significant when processing substantial document volumes. The input given to the language models is the instruction prompt and the input page of the document, either as text or as image. A text-heavy page typically corresponds to approximately 3200 tokens for text-only processing or 3500 tokens for the image data. With output responses averaging around 100 tokens, depending on the quantity of metadata to extract. At scale, these differences in token consumption lead to significant cost disparities between model categories. Organizations processing thousands of documents per month can achieve substantial operational savings by adopting SLMs. It is crucial to strike the right balance between performance and requirements to maximize efficiency while maintaining accuracy. Beyond direct service costs, the reduced computational requirements of SLMs translate into lower infrastructure investments for organizations deploying models locally. The hardware specifications needed to run SLMs remain substantially more modest than those required for LLMs, including energy consumption, cooling requirements, and maintenance expenses.

Table 2.5 provides an overview of the pricing for the most famous and recent LLMs, offering insight into the scale of the costs involved. The two models on the right, which have the highest pricing, are OpenAI’s most recent releases. This reflects the recent trend in AI industry: the development of increasingly larger and more advanced models, inevitably accompanied by rising operational costs.

Model Performance LLMs have the ability of maintaining context over long passages and generating coherent, contextually rich responses. SLMs, with their limited amount of parameters, struggle with long-term context retention and deep language understanding. While LLMs generally outperform SLMs in terms of linguistic comprehension and contextual awareness, the performance gap narrows through domain-specific fine-tuning, with properly optimized SLMs achieving approximately 90% of LLM performance in specialized applications while requiring significantly fewer resources. This finding challenges simplistic "bigger is better" [22] approaches to model selection and highlights the importance of contextual

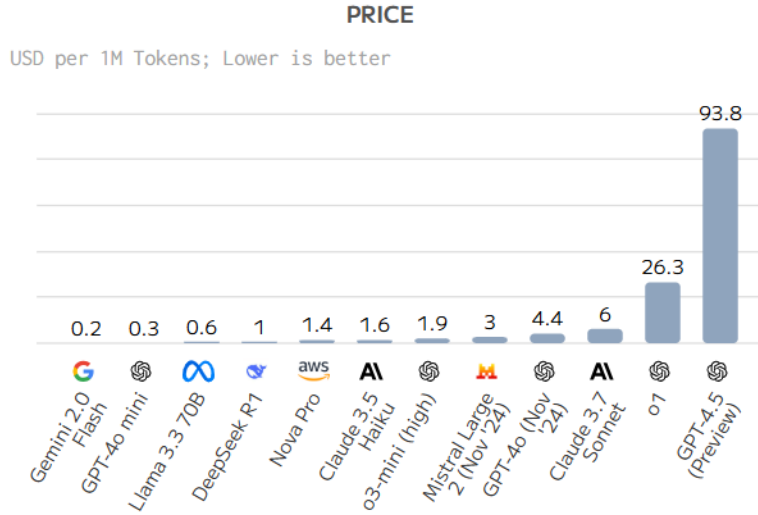


Figure 2.5: The Figure shows different models’ pricing, from the left to the most expensive on the right. The prices are USD per 1M tokens. The plot has been taken from [21] in which it is performed a wide comparison over different topics.

evaluation. Organizations should consider not only raw capability metrics but also operational requirements, budget constraints, and specific performance needs within their domain.

2.3 Thesis’s objective

The comparative analysis of LLMs and SLMs reveals a landscape where optimal model selection depends on specific use cases, resource constraints, deployment requirements and reliability needed. While LLMs demonstrate superior capabilities in general knowledge tasks and complex reasoning scenarios, SLMs offer compelling advantages in specialized domains, particularly when considering resource efficiency, deployment flexibility, and economic factors.

The evolving landscape of language model development suggests several important future directions. Research into efficient training methods, parameter optimization, and specialized architectures may further reduce the performance gap between these model categories while maintaining the efficiency advantages of smaller models.

Hybrid approaches leveraging fine-tuned SLMs for domain-specific tasks, supported by LLMs only in cases requiring extensive reasoning or broad generalization, may define a practical middle ground for many real-world applications.

As language models continue integrating into critical business processes and consumer applications, responsible selection must consider not only technical and economic factors but also broader implications, including environmental impact, accessibility, and data privacy. The comparative analysis of SLMs in these dimensions suggests they may play an increasingly important role in future AI applications.

This thesis investigates these differences in language models, and its objective is to understand the feasibility of SLMs as an efficient and adaptable alternative for specific tasks. The tasks in this case are document classification and metadata extraction. The objective is to analyze the capability of both LLMs and SLMs in delivering reliable solutions, quantify the performance gap between them, and assess the associated computational costs.

2.4 Models used in the project

This section provides a detailed examination of the models selected for this study, focusing on the comparison between LLMs and SLMs. Given the vast landscape of available models, the selection process aimed to include a highly capable, state-of-the-art LLM, potentially one of the largest currently available, and contrast it with a smaller model specifically optimized for the given task. This approach allows for a comprehensive evaluation of the trade-offs between scale, performance, and efficiency in this real-world application.

2.4.1 BERT

BERT, the Bidirectional Encoder Representations from Transformers (BERT), introduced by Google AI in 2018, marked a paradigm shift in natural language understanding and established a new standard for pre-trained language models [3].

It is based on the idea that the major limitation of standard language models was their unidirectional nature, which restricts the choice of architectures that can be employed during pre-training. BERT's architecture relies solely on the transformer encoder, in contrast to the decoder-only architecture (as GPT). A key innovation

of BERT lies in its bidirectional training approach, which enables the model to consider both left and right contexts simultaneously when processing text. This feature contrasts with the autoregressive framework of GPT.

BERT is available in two primary variants: *BERT-Base*, with 110 million parameters, and *BERT-Large*, with 340 million parameters. They are indeed Small Language Models, as their parameter count is significantly lower compared to the vast scale of models. Despite this, BERT's efficiency and effectiveness have made it a cornerstone in natural language processing, especially due to its unsupervised feature-based approaches. The model is trained using two primary objectives: the **Masked Language Model (MLM)** and **Next Sentence Prediction (NSP)**. In the MLM task, BERT randomly masks tokens in the input and learns to predict them based on surrounding context, enabling bidirectional language modelling. In the NSP task, the model learns to determine if two sentences are logically connected, improving its performance on sentence-pair classification tasks.

A key characteristic of BERT is that it can be easily fine-tuned for various tasks, resulting in a powerful, task-specific model. As Figure 2.6 shows, for fine-tuning the BERT model is first initialized with the pre-trained parameters, and all parameters are fine-tuned using labeled data from the downstream task.

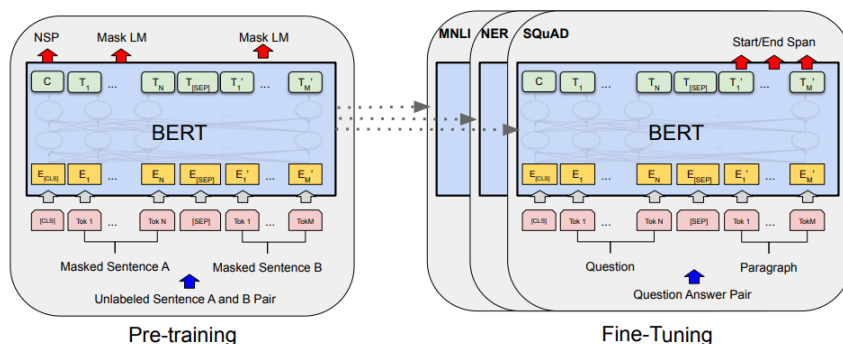


Figure 2.6: The schema shows the two phases of training a model with BERT, the first is a pre-training phase and the second a fine-tuning on a specific dataset for a specific task. This Figure is taken from the official *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* [3].

The impact of BERT on NLP research and applications has been significant. At the time of its release, it achieved state-of-the-art results on numerous benchmark datasets, including the General Language Understanding Evaluation (GLUE) and

the Stanford Question Answering Dataset (SQuAD). It reached a GLUE score of 80.5% and F1 of 93.2 on the SQuAD test, demonstrating exceptional performance on tasks such as text classification, sentiment analysis, named entity recognition, and question answering [23, 24]. The model size of BERT is exactly the same as that of the OpenAI GPT model from that year, GPT-1. The paper indicated that BERT achieved better performance overall. Its success has inspired the development of numerous variants and fine-tuned models tailored for specific languages, domains, and tasks.

One of BERT’s most notable advantages is its open-source availability, which has fostered widespread adoption and extensive research. This transparency enables developers and researchers to fine-tune the pre-trained model on domain-specific data, extending its applicability while maintaining high performance with relatively modest computational resources compared to larger language models. This contrasts with the OpenAI approach, which has increasingly leaned toward proprietary models with restricted access, limiting opportunities for customization and independent evaluation.

Thanks to its moderate size, the model can be deployed directly in private environments. This allows organisations to fine-tune it, implement other tools or pipelines over it, and ensure compliance with data privacy standards. BERT’s efficiency, accessibility, and robust performance in certain tasks make it an optimal choice for real-world applications where data is available to fine-tune the model for enhanced performance, and computational constraints are critical. BERT has demonstrated that rich, unsupervised pre-training enables even low-resource tasks to benefit from deep unidirectional architectures, especially thanks to its deep bidirectional architecture.

2.4.2 GPT

Generative Pre-trained Transformers (GPT) form a family of state-of-the-art language models developed by OpenAI [25]. These models are based on a decoder-only transformer architecture and operate under an autoregressive framework, meaning they generate predictions one token at a time, conditioned on the previous context. Specifically, the model generates a probabilistic distribution over

possible next tokens, and a decoding algorithm selects the actual output token, which can be mathematically expressed as $P(w_t|w_0, \dots, w_{t-1})$. A fundamental feature of this architecture is the use of masked multi-head self-attention, which ensures that each token attends only to its preceding tokens, preventing information leakage from future tokens.

The first GPT model, released in 2018, marked a significant advancement in natural language processing and established OpenAI as a pioneer in this field. Since then, the GPT series has evolved through successive iterations, GPT-2, GPT-3, GPT-3.5, GPT-4, GPT-4o, GPT o1 and the most recent GPT 4.5, introducing increasingly sophisticated learning techniques and architectural improvements. While OpenAI has not disclosed the exact number of parameters in these models, they are estimated to operate on scales reaching hundreds of billions of parameters, with each iteration enhancing the model’s capacity for contextual understanding, reasoning, and coherent language generation.

Figure 2.7 illustrates the evolution of the GPT-series models over the years.

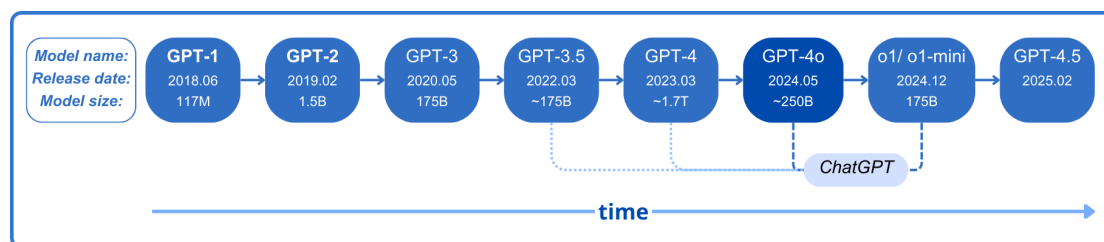


Figure 2.7: The evolution of the GPT-series models over time. This schema draws inspiration from the one in the paper *“A Survey of Large Language Models”* [26]. The information is based primarily on research papers, blog articles, and official OpenAI documentation. Dashed lines represent the models currently integrated into ChatGPT, while dotted lines refer to earlier versions as they were at the initial deployment in 2022. In the model size, B stands for Billions and T for Trillions.

All GPT models are deployed through OpenAI’s centralized cloud infrastructure, which enforces strict data access and usage policies. However, this architecture raises inherent concerns regarding data sovereignty and the potential exposure of sensitive and private information.

GPT models employ a self-supervised learning approach, where the training objective involves predicting the next token in a sequence, enabling the discovery

of intricate patterns inherent in language data. Training occurs in two stages: first, the model is pre-trained on vast Internet datasets, learning general linguistic structures, and then, from GPT-3 onward, models are fine-tuned using reinforcement learning from human feedback (RLHF). This aligns the models with their outputs to human-preferred responses and improving practical usability [?].

Released in 2023, GPT-4 marked a significant advancement in language model performance, demonstrating notable improvements in reasoning, knowledge retention, and coding capabilities compared to its predecessors. Although it is still far from achieving human-level capabilities in many real-world scenarios, GPT-4 attains near-human performance on several professional and academic benchmarks.

Despite these advances, GPT-4 models are not entirely reliable and remain prone to hallucinations, reasoning errors, and the possibility of generating harmful or inaccurate information. To mitigate these issues, GPT-4 incorporates an additional safety reward signal during RLHF training, reducing the likelihood of unsafe outputs while maintaining generative fluency and versatility. OpenAI invested six months in iteratively aligning the model through adversarial testing, which enhanced safety, factual accuracy, and steerability. These efforts show an 82% reduction in the model's tendency to respond to disallowed content compared to GPT-3.5 and a 29% improvement in compliance with OpenAI's safety policies when addressing sensitive queries.

GPT-4o, also known as the "Omni Model", is an advancement over the GPT-4 model. It is designed to unify various AI capabilities into a single, end-to-end framework capable of seamlessly processing and generating text, interpreting images, analyzing audio, and even handling video inputs. Compared to GPT-4's context window of 8192 tokens, GPT-4o significantly extends this capacity to 128k tokens, making it far more adept at handling lengthy and complex text.

A distinctive feature of GPT-4o is its multimodal architecture, where a single neural network processes inputs and outputs across different modalities. This innovation fosters a more natural and fluid human-computer interaction, marking an initial step toward more comprehensive AI systems.

Following the path set by its predecessors, GPT-4o incorporates safety-by-design principles across all modalities. Techniques such as filtering training data,

post-training behavioral refinement, and the implementation of robust guardrails ensure responsible deployment and mitigate risks associated with generative AI.

All these characteristics make GPT-4o particularly suitable for industrial applications where diverse input types and high performance are paramount. Its ability to integrate and process multimodal data efficiently underscores its potential as a foundational tool for advanced document classification and metadata extraction tasks.

2.4.3 LLaMA

LLaMA, Large Language Model Meta AI, represents a family of advanced language models [27] developed by Meta AI, designed to balance state-of-the-art performance with computational efficiency and accessibility [28]. Unlike proprietary models such as GPT-4, LLaMA models are released as open weights, enabling broad experimentation and customization. This approach aligns with Meta’s vision of democratizing access to powerful language models for both research and industry. However, LLaMA’s license enforces an acceptable use policy that prohibits its application for certain purposes, Meta’s description of LLaMA as “open source” has been disputed by the Open Source Initiative, which maintains the official Open Source Definition.

Since the beginning, Meta has consistently released powerful models of varying sizes, but in general, often much smaller than competing models. For example, GPT-4, released the same year as LLaMA’s first model, has approximately 1.7 trillion parameters while LLaMA has a maximum of 65 billion. These differing approaches reflect a broader and modern discourse on the trade-off between model size and efficiency, which will be further explored in the following section on LLMs versus SLMs.

Also LLaMA models are based on the transformer architecture, specifically following a decoder-only configuration similar to GPT, and they employ an autoregressive framework. However, there are some key differences that distinguish LLaMA from other models:

- **SwiGLU** [29] activation function instead of GeLU.
- **Rotary positional embeddings** (RoPE) [30] instead of absolute positional

embeddings.

- **RMSNorm** [31] instead of traditional layer normalization.

In particular, Figure 2.8 illustrates a comparison between the GPT and LLaMA architecture, both based on the transformer [7] but with some important differences..

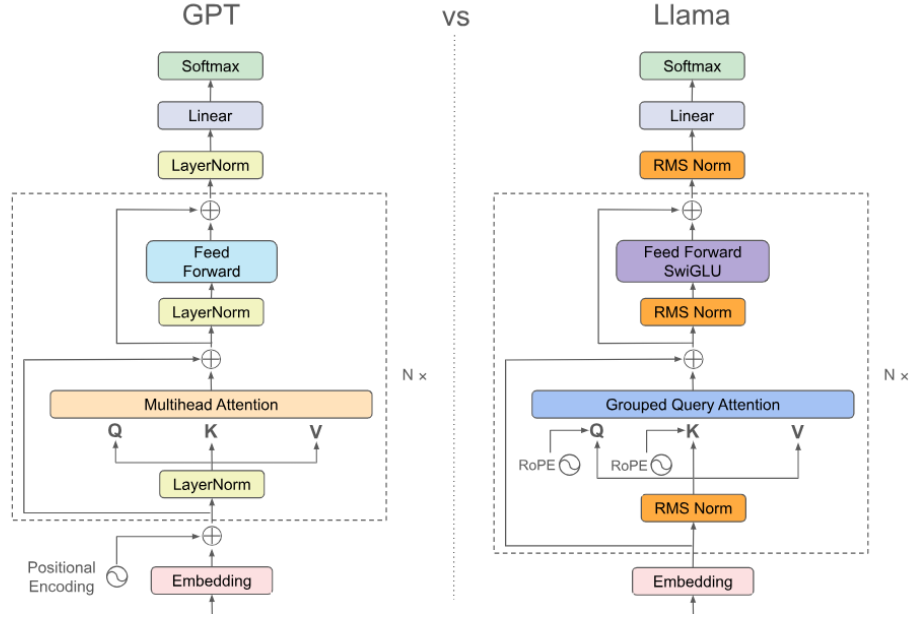


Figure 2.8: A comparison between the architectures of GPT and LLaMA.

A defining characteristic of the LLaMA family is its availability through platforms like Hugging Face [4], underscoring Meta’s commitment to fostering an open AI ecosystem. Hugging Face’s comprehensive integration of LLaMA models has made them widely accessible and easily deployable for various NLP applications. Meta began releasing the LLaMA family of models in 2023, which currently includes LLaMA 1, LLaMA 2, and LLaMA 3. Each iteration introduced models of varying sizes, providing flexibility in terms of computational requirements and performance. The table 2.2 shows the LLaMA models released during the past two years and their available sizes. Since its inception, Meta has consistently released powerful models of varying sizes, often favoring a smaller scale compared to competing models. For instance, GPT-4, released the same year as LLaMA 1, contains approximately 1.7 trillion parameters, whereas LLaMA 1’s largest model at the

Model Name	Release date	Parameters
LLaMA 1	02/2023	6.7B, 13B, 32.5B, 65.2B
LLaMA 2	07/2023	6.7B, 13B, 69B
LLaMA 3	04/2024	8B, 70.6B
LLaMA 3.1	07/2024	8B, 70.6B, 405B
LLaMA 3.2	09/2024	1B, 3B, 11B, 90B
LLaMA 3.2 Vision	09/2024 (no UE)	11B, 90B

Table 2.2: The tables show the models of the LLaMA series, with each corresponding release date and the different released model parameters sizes.

time had 65 billion. These differing approaches reflect two distinct philosophies in AI development: one prioritizing vast scale to achieve maximum performance, and the other focusing on efficiency and accessibility while maintaining competitive capabilities.

LLaMA models consistently demonstrated competitive results compared to larger proprietary model, starting with the 13B parameter LLaMA 1 model which outperformed the much larger GPT-3 model (175B parameters) on several NLP benchmarks, while the 65B model achieve the performances of state-of-the-art models like PaLM [32] and Chinchilla [33].

Later, in July 2023, Meta, in partnership with Microsoft, announced LLaMA 2 [5]. Although its architecture remained largely unchanged from LLaMA 1, it benefited from a 40% increase in training data volume, leading to improved performance and generalization capabilities.

Following, a year later, LLaMA 3 has been released with two model sizes. These models were pre-trained on approximately 15 trillion tokens collected from publicly available sources and enriched with over 10 million human-annotated examples. Compared to its predecessor, LLaMA 3 was trained on eight times more data and featured a new tokenizer with an expanded vocabulary of 128,256 tokens (four times more tokens than the previous version). With these internal evaluations, Meta demonstrated that LLaMA 3 70B outperformed competing models like Gemini Pro 1.5 and Claude 3 Sonnet on most benchmarks.

Among the LLaMA 3 models, LLaMA 3.2 stands out for its efficient inference and reduced deployment costs, making it particularly attractive for real-world applications where computational efficiency is crucial. This optimization is espe-

cially beneficial for on-premises deployments, enabling organizations to maintain data sovereignty and mitigate privacy concerns often associated with cloud-based models.

Overall, LLaMA 3.2 exemplifies the evolving landscape of open-weight language models, offering a powerful yet efficient alternative to more resource-intensive models like GPT-4. Its combination of scalability, performance, and transparency makes it a valuable tool for a wide range of natural language processing tasks.

In this project, LLaMA 3.2 will be fine-tuned for metadata extraction within the document processing pipeline, and its performance will be compared with that of GPT-4o.

Chapter 3

Data

This thesis focuses on the development of a **document processing pipeline** designed for a real-world application, specifically aimed at the **classification** of documents and the subsequent **extraction** of metadata. The ultimate goal of this pipeline is to autonomously extract specific information from a given document, where the type of data to be extracted depends on the document's class. Each class is associated with a distinct set of metadata, reflecting the particular characteristics and informational needs of that class.

The documents considered in this study belong to the **insurance sector**, a domain characterized by a wide variety of documentation from different operational contexts. For this project, the documents are primarily categorized into two subsectors: the medical and the administrative. In this particular case, the dataset includes six primary classes of documents: an employment certificate, leave certificate, death certificate, hospitalization certificate, generic medical certificate, and medical record. Additionally, there exists a residual class, labeled "**others**", which includes all documents that do not belong to any of the six predefined classes. Each of these classes is described in more detail below:

- **Employment:** Belonging to the administrative subsector, this class includes documents related to the hiring processes. They are usually emails, official contracts or formal records of employment agreements and procedural communications.
- **Leave certificate:** Also this one is part of the administrative subsector, this category covers documentation associated with the revocation of vaca-

tion periods. Documents in this class may include formal letters, internal company paperwork, and email correspondence.

- **Hospitalization certificate:** This class instead falls into the medical subsector and consists of admission certificates issued when a patient is admitted to the hospital, clinic, or emergency department. These documents formally record the patient’s entry into the healthcare system.
- **Generic medical certificate:** Also situated in the medical subsector, this class includes various forms of medical documentation, such as prescriptions and sickness certificates. A notable feature of these documents is that they are often handwritten by medical professionals and typically consist of a single page.
- **Medical record:** Representing the most comprehensive category within the medical subsector, this class includes the complete clinical history of a patient. Medical records often comprise multiple pages, detailing the patient’s condition from admission through discharge, including diagnoses, medical examinations, treatment plans, and results. These documents indeed are very long.
- **Death certificate:** As indicated by its name, this class consists of official documents certifying a person’s death. Typically limited to a single page, these documents may describe the cause of death, including medical diagnoses, the circumstances of the event, or legal information related to the case.

The use of real-world documents in this study introduces several **challenges**, primarily due to the organization of the classification schema. The division of document classes was established based on business requirements and was not subject to negotiation, despite some notable ambiguities. For example, certain medical classes, such as the hospitalization certificate and the medical record, exhibit substantial overlap. The differences between them are sometimes very difficult to discern. This overlap complicates the classification task and necessitates robust methods of validation and error analysis.

Regarding the actual dataset, a total of **2079 documents** were initially provided, with each document being either a PDF or an image in JPEG or PNG

format. Specifically, the dataset comprises 482 images and 1597 PDF files, and PDFs often contain multiple pages. Some documents are exceptionally large, the average number of pages per PDF is approximately 9, though it is important to highlight that there are a dozen documents exceeding 100 pages. This variability in document length underscores the complexity of the dataset and the need for efficient document handling and processing techniques. The dataset includes a diverse range of document types, such as scanned images, digitized documents, and photographs of paper documents. The content within the documents is highly heterogeneous, often featuring textual components alongside frontispieces, symbols, tables, forms, signatures, and handwritten sections. And in some cases, entire pages consist solely of handwritten content, further complicating the task of information extraction.

A significant characteristic of the data is the potential for multiple classes to coexist within a single document, a phenomenon prevalent in medical documentation. For instance, medical records, which are typically very long, may contain a hospitalization certificate at the beginning and, in the event of a patient’s death, a death certificate at the end. Another common scenario could involve scanned documents that, due to human error, include pages originating from different original documents. To address this complexity, the **classification process is conducted at the individual page level**. Each page is assigned to one of the seven defined classes, which comprise the six primary categories and the ”others” class. Then, pages belonging to the same class within a given document are grouped together to form what is called a **”logical document.”** This concept of logical documents allows the organization of single physical documents which encompass multiple distinct classes inside. Figure 3.1 illustrates this non-trivial concept, providing a

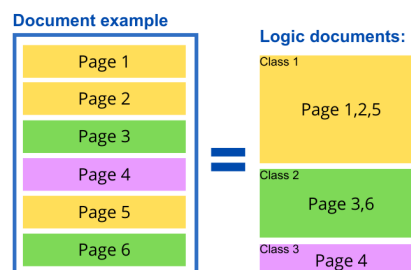


Figure 3.1: High-level schema of the logic documents inside one original document, which are grouped into pages of the same document and classified as the same class.

high-level representation of how logical documents are constructed from the pages of an original document.

Following the classification of individual pages and the subsequent creation of logical documents, **the extraction phase is performed on these logical units**, it is important to note this different granularity. Figure 3.2 provides a broad view of the entire pipeline, illustrating the sequence of steps required to transform an input document into its corresponding extracted metadata.

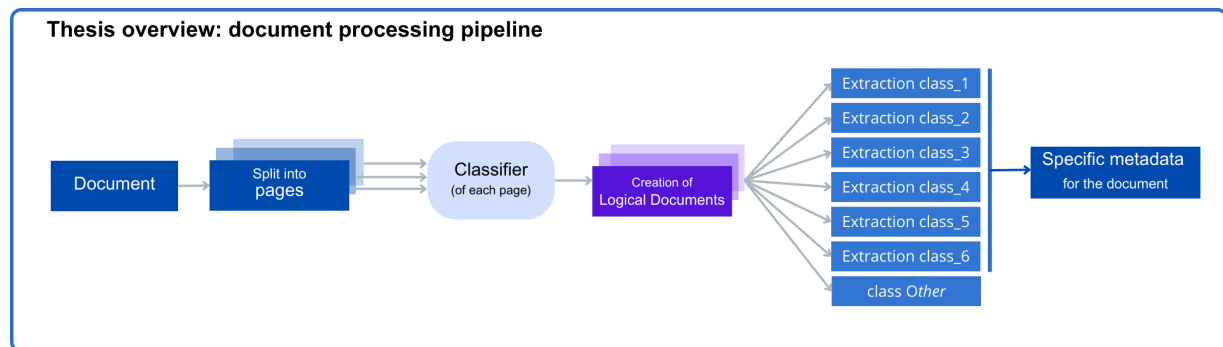



Figure 3.2: The Figure shows a schema of this thesis overview, showing from the input document the main steps to reach the output data. First, splitting each document into its pages and classifying each page into one of the six classes. Then, for each class a specific extraction is performed and gives the output wanted, the metadata of that specific document.

Each document class is associated with a distinct set of metadata to extract. Table 3.1 defines the specific metadata corresponding to each class, highlighting the varying informational requirements across document types.

Class	Metadata
employment	emission_date
leave_cert	emission_date, revoked_date_start/end
death_cert	name, event_date
hospitalization_cert	emission_date, admission_date, diagnosis
generic_medical_cert	emission_date, symptom_onset_date, diagnosis
medical_record	admission_date/time, discharge_date/time, diagnosis


Table 3.1: The table shows the corresponding metadata for each class. In the class name *cert* stands for *certificate*.

At the conclusion of this document processing pipeline, a list of metadata corresponding to the information extracted from a given document is produced. The length of this list reflects the number of logical documents identified within the original document. To provide a clearer understanding of the data, Figure 3.3 presents a concrete example of a medical record from the dataset. Additional examples for each class are available in Appendix A. It is important to note that the documents in this dataset vary significantly in format, and while the most representative types have been selected, they serve only as illustrative examples.



SISTEMA SANITARIO REGIONALE
**ASL
ROMA 2**

OSPEDALE SANDRO PERTINI
Via dei Monti Tiburtini, 385, 00157 Roma RM
Centralino 0641431
PNEUMOLOGIA SP



Direttore:
D.ssa Maria Cristina Zappa

Dirigenti Medici:
D.ssa Carla Condoluci
Dr. Kristopher R. Flores
D.ssa Giuseppina Giotfrè
D.ssa Samantha Lamarra
D.ssa Marianna Lilli
D.ssa Barbara Maggi
Dr. Rosario Rivini
Dr. Mattia Serso
D.ssa Angela M.P. Succu
D.ssa Tiziana Trequattrini
D.ssa Rossana Vignarolo
D.ssa Annalisa Villani

Reparto:
Degenza di Pneumologia Clinica e
Terapia Subintensiva Respiratoria

06 41433383
06 41433363- 06 41433510

Day Hospital di Pneumologia
Tel.: 0641433401
Ambulatorio specialistico di
Pneumologia
Tel.: 06 41433392
Endoscopia Bronchiale Medica
Tel.: 06 41433381

Roma, 20/05/2022

Egregio collega, Dimettiamo in data odierna, il sig. _____ nato/a il _____ a ROMA
residente in via/piazza _____ I ricoverato/a presso il reparto di PNEUMOLOGIA SP in
data 16/05/2022

Motivo del Ricovero
In data _____ giungeva in PS per tosse e febbre insorta tre giorni prima, alla tc torace senza mdc del
15.05.22 evidenziava addensamento parenchimale con broncogramma aereo a livello del lobo inferiore
destro

Diagnosi alla dimissione
polmonite basale LID da Legionella pneumophila

Dati Anamnestici
Progressa cardiopatia ischemica cronica, portatore di PMK, storia di FAC in NAO.

Decorso Clinico e condizioni del paziente alla Dimissione
Il paziente giungeva presso il nostro reparto in data 16/5/2022, proveniente dal PS ove eseguiva tc del
torace con evidenza di un addensamento parenchimale del LID in assenza di insufficienza respiratoria. In
ps veniva valutato dal cardiologo per il riscontro di un minimo movimento enzimatico delle troponine
aspecifico in assenza di sintomatologia di tipo anginoso. Dopo 48 h dall'ingresso in reparto è stata
accertata la positività per l'antigene urinario per Legionella Pneumophila ed è stata iniziata la terapia
antibiotica con Levofloxacina. Durante la degenza il paziente non ha mai necessitato di ossigeno terapia,
si è mantenuto stabile, vigile, eupnoico, collaborante è stata inoltre potenziata la terapia diuretica con
riduzione degli edemi declivi. Gli indici di flogosi sono in netta riduzione. Pertanto dimettiamo in data
odierna il paziente in buone condizioni cliniche e con l'indicazione ad effettuare i seguenti controlli e la
terapia domiciliare prescritta. Ha eseguito oggi un trf rapido ad esito negativo che si allega

Terapia Farmacologica consigliata
Pr: Laszoprazolo 30 mg 1 cp la mattina alle 8
Pr: Triatec 2,5 mg 1 cp ore ore 8
pr: Elquis 5 mg 1 cp ore 8-20
Pr: Zyloric 300 mg 1/2 cp ore 8
pr: Torvast 40 mg 1 cp ore 21.
Pr: Atenololo 50 mg 1 cp ore 8
Pr: Levofloxacina 500 mg 1 cp ore 8 per altri 8 giorni poi sospendere.

Controlli e consigli
Si consiglia di ripetere un prelievo ematico con emocromo, glicemia, AST/ALT, elettroliti sierici da far
visionare al curante, per controllo degli enzimi epatici che hanno mostrato un aumento dei valori soglia
durante la degenza
Il paziente dovrà recarsi in data 8.7.2022 presso l'Ambulatorio di Pneumologia alle ore 10:30 recando in
visione un rx del torace in 2p per confronto del noto addensamento del LID.

Cordiali Saluti
Dott. VILLANI ANNALISA

ASL Roma 2 P.IVA 13665151000
Sede Legale via M. Brighenti, 23 00159 Roma
Sede Operativa via M. Brighenti, 23 00159 Roma

Pagina 1 di 1

Figure 3.3: The image is an example of a document of the dataset, in particular, the one shown is a medical record. The image has been anonymized as it contains sensitive personal data.

3.1 Challenges

Working with real-world data introduces the challenge of **incomplete labeling**, and this is exactly the case for the classification task addressed in this study. The only available labels were the outputs of a pre-existing classification model, accompanied by their respective confidence scores. These labels could not be treated as ground truth. As illustrated in Figure 3.4, an initial filtering step was applied to select only those pages for which the previous model’s confidence score was at least 0.99, and these were considered pseudo-labels.

Pseudo labelling relies on the model’s predictions for unlabeled data. If the model is uncertain or makes incorrect predictions, it can introduce noise into the training set, impacting overall performance. Pseudo labelling assumes that the distribution of pseudo-labeled samples is similar to the original data, but shifts in distribution may occur, leading to suboptimal model performance. This filtering reduced the dataset from approximately 15,000 pages to around **8,000 pseudo-labelled pages**.

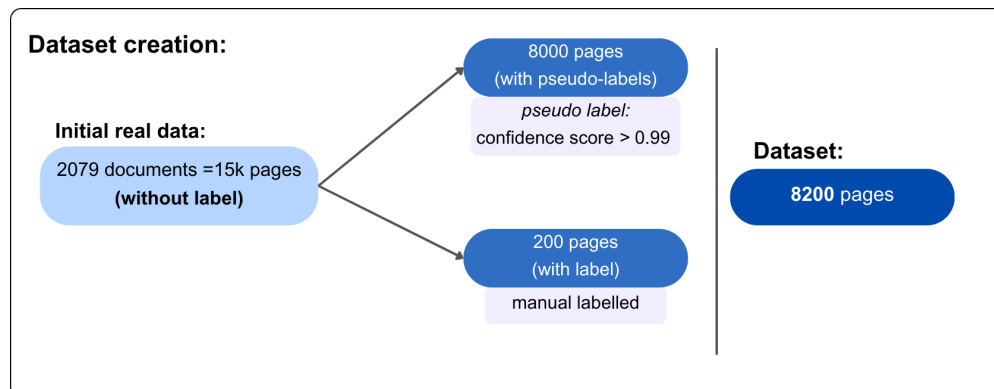


Figure 3.4: The figure shows the schema of the dataset creation, where the biggest challenge was the missing labels. Starting from the original dataset, only the pages with a 0.99 confidence score were selected with their pseudo-labels. In addition, a strategic manual labelling of 200 documents was performed.

This constrained filtering process, while ensuring high-confidence data selection, resulted in the document being fragmented. Indeed, many documents lost pages due to the lower confidence scores of some sections. To mitigate this issue, and improve the quality of the dataset with true labels, a **manual labelling** was performed on 200 additional pages.

An **ad-hoc script** was developed to efficiently assist in the manual labeling process, minimizing the tedious effort and potential for human error.

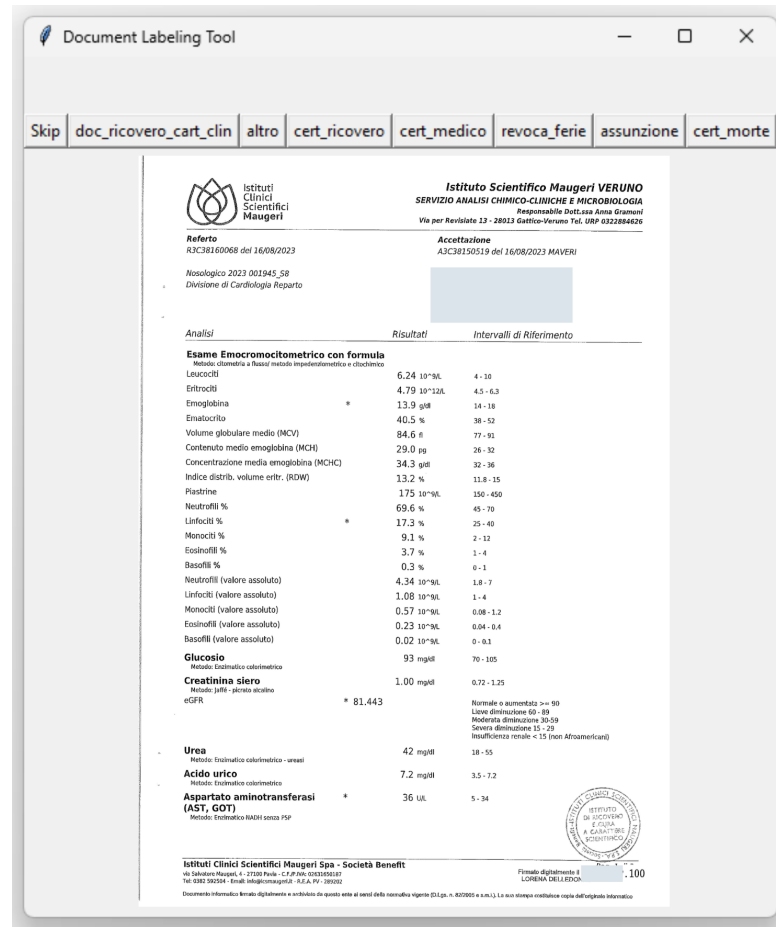


Figure 3.5: The image is a screenshot of the output of running the labelling script. It shows a concrete example of how simple the interface is, with the document and the buttons. The image in this case has been anonymized as it contains sensitive personal data.

This script automated several steps: it displayed each page image, allowed the user to assign a label, recorded the label in an organized and persistent format, and then automatically proceeded to the next file. The script facilitated label assignment through simple buttons corresponding to the document classes. It moved each labeled file to its respective class folder and simultaneously recorded the label data in an Excel file for easy integration into the dataset. An example of how the process looks is in Figure 3.5. To preserve the document-level concept, the manual labeling focused on pages with low-confidence scores from documents that already had a pseudo-label. During the labeling process, the pseudo-label

was visible in case of indecision. Thanks to this approach and the efficiency of the supporting script, it was possible to reconstruct **1,100 documents**, corresponding to approximately **8,200 pages** with a combination of pseudo and true labels.

3.2 Dataset exploration

A dataset exploration of the 8200 page dataset was conducted, and an evident **imbalance in class distribution** emerged. The majority of the pages belonged to the *medical record* class. Figure 3.6 illustrates the class distribution across the dataset. This imbalance is expected given the nature of the domain. Unlike

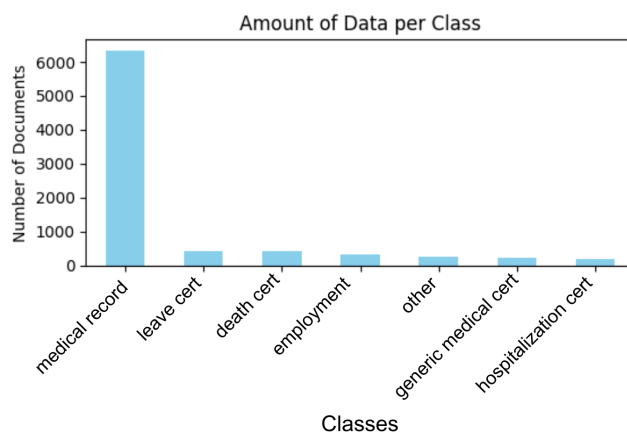


Figure 3.6: The Figure shows the class distribution over the final dataset of 8200 pages. There is a clear unbalanced situation towards the *medical record* class.

the other classes, which represent specific certificates, the medical record covers a patient’s complete clinical history, often consisting of numerous pages. As a result, this class overwhelmingly dominates the dataset.

This class imbalance poses a critical challenge in the subsequent phases of model implementation, influencing the performance of both LLMs and SLMs in distinct ways. For LLMs, which analyze one document at a time and generate responses only based on the provided prompt and their prior knowledge, the impact of class imbalance may be alleviated. In contrast, SLMs undergo a training phase on the entire dataset, making them more prone to the effects of imbalance. This could result in biased predictions, where the model often favors the overrepresented class, ultimately affecting performance and generalization capability.

3.3 Data preprocessing pipeline

With the final dataset established, the document processing pipeline could start. The primary objective of this initial stage was to **extract the textual content** from the documents. Figure 3.7 provides an overview of the steps and tools employed for text extraction.



Figure 3.7: The Figure shows the pre-processing schema of a document, from an input PDF to the contained text. illustrates all the steps and techniques used: the *pdf2image* library, the Textract AWS OCR, and a text reconstructor tool.

To ensure consistency in data format, the first step was converting all PDF files into images, so that every document was represented as individual page images. This transformation was carried out using the Python library *pdf2image* [34], which, given the path of a PDF, converts all its pages into a list of images. This library is built on *Poppler* [35], an external tool essential for the conversion process.

Once the documents were converted into images, they were passed through *Textract OCR* [6], an Optical Character Recognition tool, previously detailed in Chapter 2.1.1. Textract produces extensive output, including the recognized text, positional data, and other image metadata. To extract the textual content, a dedicated internal suite of services was used, specifically the *Block Reconstructor* library. The Block Reconstructor plays a crucial role in converting the raw OCR output into coherent, human-readable text by reorganizing detected text blocks according to their spatial relationships on the page. This step ensures the preservation of the original document’s structure and readability, facilitating subsequent processing.

These steps marked the actual beginning of the document processing pipeline. With the textual content successfully extracted, the data were prepared for the first crucial phase: the classification of pages into the seven distinct classes.

Chapter 4

Classification

In this chapter, the implementation of the classification task is presented, with a particular focus on the comparative analysis between the large language model **GPT-4o** and the small language model **BERT**. The primary objective of this task is to classify each page of the dataset into one of seven predefined categories: employment certificate, leave certificate, death certificate, hospitalization certificate, generic medical certificate, medical record, and others. A detailed description of these classes has been provided in Section 3.

After the preprocessing steps, the final dataset is of 8200 rows, each containing the image file name and the corresponding text extracted from the document. This textual data serves as the input for both classification approaches.

One of the principal challenges associated with this dataset is the pronounced imbalance among the classes, as previously discussed in Section 3.2. Specifically, the majority of the data belongs to the *medical record* category, leading to a skewed class distribution. This imbalance impacts particularly the SLM because it has a training phase during which it learns the structure of the documents, the relationships between classes, and their respective frequencies. As a result, the imbalance may lead to biased performance in the SLM, particularly if the model overfits the dominant class.

The documents in this dataset are considerably complex, both in terms of lexical content and document structure. Some pages contain handwritten text, and so the text extract is more confused, while others include tables, images, and various formatting elements. The textual content is often characterized by highly

specialized terminology from the medical and legal domains. This heterogeneity poses significant challenges for automated classification, particularly for models not explicitly fine-tuned on such domain-specific data.

Another key challenge comes from the **subtle distinctions** between certain classes, which lead to ambiguities even for human annotators. For instance, within the medical document category, hospitalization certificates and medical records often exhibit substantial overlap, because the hospitalization is frequently at the beginning of a patient’s clinical history. This inherent ambiguity will be taken into account when interpreting the final performance metrics of the models.

In addition, the documents of the dataset contain a high amount of information per page, leading to very long textual content. As illustrated in Figure 4.1, the distribution of words per page reveals an average length of approximately 500 words. This characteristic imposes an additional challenge, as models must process and interpret long sequences of text while maintaining contextual coherence and classification accuracy.

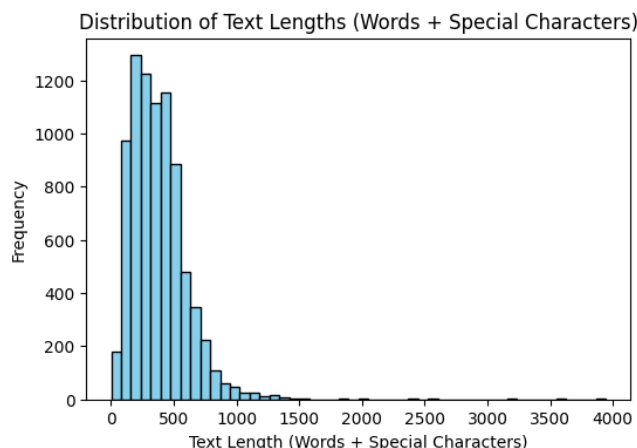


Figure 4.1: The Figure shows the distribution of textual length, counted as words and special characters, of the dataset for each page.

4.1 Classification model with BERT

The implementation of the BERT model for multi-class text classification involved a structured and systematic approach. BERT was used as a feature extractor of the textual information in the pages, on top of that, a classification model was

implemented and trained. As a preliminary step, categorical class labels were numerically encoded to facilitate processing. The dataset was then stratified and split into training (70%), validation (15%), and test (15%) subsets to preserve the class distribution across splits [36].

Class Imbalance

To address class imbalance, class weights were calculated using inverse frequency weighting [37]:

$$\text{weights}_c = \frac{N}{C \cdot n_c} \quad (4.1)$$

where N is the total number of samples, C the number of classes, and n_c the number of samples in class c . The resulting weights were:

Class	Revocation Cert.	Death Cert.	Medical Record	Other	Hospital Cert.	Medical Cert.	Employment
Weights	0.1084	0.1164	0.0075	0.1857	0.2328	0.2079	0.1413

Table 4.1: Class weights for the seven classes, highlighting the class imbalance, particularly towards the `medical_record` class.

These weights reflect class frequency, with the third class being the one over-represented and indeed it has the lowest value (0.0075). Then a weighted cross-entropy loss was used to train the model with the class imbalance:

$$\mathcal{L} = W_c BCE = - \sum_{c=1}^C W_c y_c \log(p_c) \quad (4.2)$$

Tokenization

BERT employs **WordPiece** [3], a subword tokenizer with a fixed vocabulary of 30,522 unique tokens. It includes specific special tokens: `[CLS]` (token ID 101) indicating the start of a sentence, `[SEP]` (token ID 102) marking the end of a sentence, and `[UNK]` (token ID 100) representing out-of-vocabulary words.

The tokenizer preprocesses textual data by managing punctuation, whitespace, and non-ASCII characters. Initially, all spacing characters like tabs and newlines are converted into a single whitespace. Basic punctuation symbols known to the vocabulary are tokenized individually. The subword mechanism then breaks down

words that are not present in the vocabulary: when a word is missing from the vocabulary, it is split into the largest possible subword found in the vocabulary, and is assigned a prefix with **##** to indicate the subword status. If no valid subwords are found, the entire word is assigned the [UNK] token.

For example: the tokenization of word `outperform` is `out` and `##perform`.

4.1.1 BERT For Long Texts

After tokenizing the entire dataset, the next step was adapting BERT for long text sequences. All BERT models [3] impose a maximum input length, but, as shown in Figure 4.1, many pages in the dataset have a lot of content. Considering that a single word can be represented with multiple tokens, the total number of tokens per document is even larger, the average number of tokens per page in the dataset is 708. Figure 4.2 illustrates the distribution of token lengths in the dataset, clearly showing that many pages surpass the 512-token constraint.

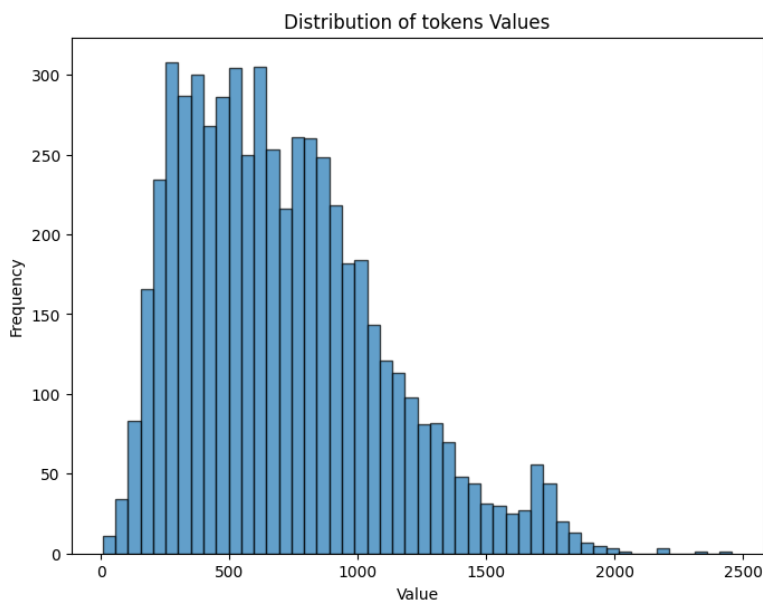


Figure 4.2: Token length distribution of the training dataset.

This token limit is a well-known constraint of BERT, and over the years, alternative models with sparse attention mechanisms like BigBird [38] and Longformer [39] have been developed to address it. However, these models are not modified BERTs, but they have different architectures and they require either pretraining

from scratch or downloading large pretrained weights, making them impractical in many contexts, as in this case. This is why for this project the **BELT** (BERT for Long Text) library [40] was employed. It is a Python implementation inspired by a proposal from Jacob Devlin, co-author of the original BERT paper, in a comment of a user asking about this topic (see the discussion here). The idea is to modify the original tokenization procedure by splitting the long sequences into chunks, and creating a mini-batch of chunks for each long text for faster computation. Each chunk produces a tensor of logits, and a mean pooling strategy aggregates them creating the final prediction:

$$\text{Final Logits}_j = \frac{1}{N} \sum_{i=1}^N \text{Logits}_{ij} \quad \forall \text{class } j \quad (4.3)$$

where N is the number of chunks. A dedicated `DataLoader` was implemented to manage this chunk-based tokenization and ensure the model received appropriately formatted input.

The implementation of BELT required several key hyperparameters of the class `BertClassifierWithPooling` [40]:

- **maximal_text_length**: Specifies the truncation limit for token sequences, either `None` (no truncation) or an integer value. Standard BERT typically uses 510 tokens, reserving 2 for the special tokens `[CLS]` and `[SEP]`.
- **chunk_size**: Defines the number of tokens per chunk, with a maximum of 510 to fit within BERT’s input constraints.
- **stride**: Controls the overlap between chunks, akin to the stride parameter in 1D convolutional networks. As noted in a related discussion, it determines the sliding window step size, ensuring either contiguous or overlapping chunks.
- **minimal_chunk_length**: Sets the minimum token count for a chunk; shorter chunks are discarded.
- **pooling_strategy**: Determines the method for aggregating chunk predictions, typically using mean or max pooling.

An hyperparameter search was conducted to optimize model performance, and these are values tried:

Parameter	All parameters tested
Model Architecture	<code>distilbert-base-uncased</code> , <code>bert-base-uncased</code>
Epochs	3, 5, 7
Batch Size	4, 8, 16
Learning Rate	5×10^{-5} , 1×10^{-5}
Stride	510, 410, 310
Min chunk len	10, 200, 510
Max text len	510, 3000
Pooling strategy	mean, max

Table 4.2: Hyperparameter search space of BERT’s parameters.

The final configuration employed `distilbert-base-uncased` [41] for its computational efficiency, achieving performance comparable to `bert-base-uncased` in a quarter of the time.

Experimental results over the hyperparameter search space 4.2 indicated that introducing the **stride**, of any length, resulted in a lowering of the performances, so it was not considered for the final setup. Another interesting insight is about the document structure, it was found out that relevant information is generally concentrated at the beginning of the text, because truncation of the later sections slightly improves the performance, and speeds up the computation. The final configuration represents a practical and efficient approach for classification with BERT managing long sequences of text.

Parameter	Best parameters
Model Architecture	<code>distilbert-base-uncased</code>
Epochs	5
Batch Size	8
Learning Rate	5×10^{-5}
Stride	510
Min chunk len	10
Max text len	3000
Pooling strategy	Mean

Table 4.3: Final best BERT’s hyperparameters.

The model was trained using the hyperparameters reported in Table 4.3, and the evolution of training and validation losses throughout the training process is illustrated in Figure 4.3. The values reported in the plot were taken every 200 steps

of the training during the 5 epochs. As shown in the figure, the training loss decreased more rapidly compared to the validation loss, which exhibited a consistent downward trend in any case, indicating that the model continued to improve its generalization capabilities over time. During the hyperparameters search, the best number of epochs chosen was 5 because more than that, the model didn't improve its capabilities, but it started to overfit, especially due to the imbalanced dataset.

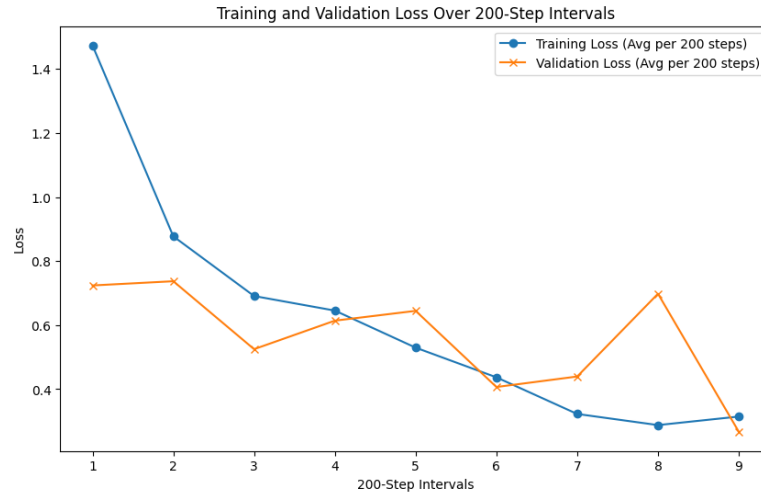


Figure 4.3: Training and Validation loss of the BERT classification model during training.

4.1.2 Performance metrics

The performance of the classification task was evaluated using a set of comprehensive metrics: Accuracy, Precision, Recall, and F1-score. Given the imbalanced nature of our dataset, these metrics are particularly important for assessing the model's ability to handle class disparities effectively.

- **Accuracy** measures the proportion of correctly classified instances out of all instances in the dataset. However, in imbalanced datasets, it can be misleading if the majority class dominates the metric.
- **Precision** quantifies the proportion of true positives among all predicted positive instances, reflecting the model's ability to avoid false positives.
- **Recall** measures the proportion of true positives among all actual positive

instances, indicating the model’s ability to detect all instances of a particular class.

- **F1-score** is the harmonic mean of Precision and Recall, offering a balanced measure of both. It is especially useful in scenarios with imbalanced datasets, as it provides a more nuanced evaluation than Accuracy alone by penalizing models that perform well on the majority class but poorly on minority classes. The **macro F1-score** calculates the F1-score independently for each class and then takes the average, treating all classes equally regardless of their frequency. Since it gives equal weight to each class, it prevents the model’s performance on majority classes from overshadowing its performance on minority classes.

These metrics collectively allow for a thorough assessment of the model’s strengths and weaknesses, especially in handling class imbalance, which is a common challenge in multi-class classification tasks.

Class	Precision	Recall	F1-Score	Accuracy
Leave cert	0.950	0.966	0.958	0.966
Death cert	1.000	0.955	0.977	0.955
Medical record	0.987	0.994	0.991	0.994
Other	0.762	0.762	0.762	0.762
Hospitalization cert	0.923	0.828	0.873	0.828
Medical cert	0.842	0.842	0.842	0.842
Employment	0.929	0.907	0.918	0.907

Table 4.4: BERT’s classification performance for each class, of the model with the best hyperparameters in the test set.

The overall F1-score of the model with best combination of parameters is **0.903** and an accuracy of **0.971**. The difference between these two metrics is due to the class imbalance, because the accuracy does not make any differentiation between classes, but it considers only the total number of correct predictions over all the predictions, and in this case, many data come from the same class for which the model is almost always right. It is important to check, especially in this case, that the model does not simply learn to always predict the most frequent class and never the other. In that case, the accuracy would stay high, but for sure the F1

would be lower, because it is an average of the precision and recall over all classes proportionally.

The results show strong overall performance for most classes, with particularly high metrics values for “*medical_record*” and “*death_certificate*”, the first one because it is the most frequent one, and “*death_certificate*” because it is the simplest and clearest document to identify. These results align with the confusion matrix, where these classes show very few misclassifications.

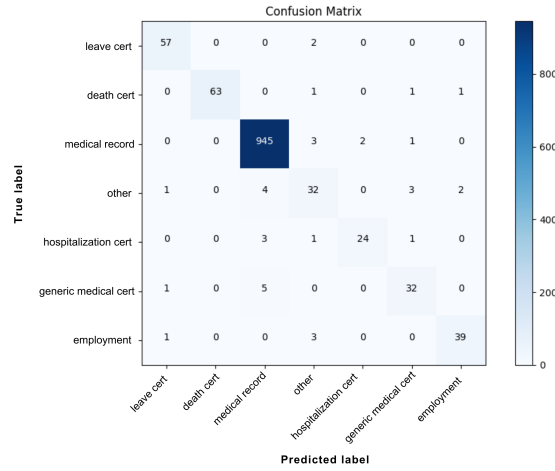


Figure 4.4: Confusion matrices of the seven classes in the test set, of the model with the best configuration of parameters.

On the other hand, the class “*others*” has lower performance across all metrics (precision, recall, and F1-score at 0.762). This is reflected in the confusion matrix, where there are more misclassifications for this class, indicating that it’s harder for the model to distinguish. As expected, similarly, “*hospitalization_certificate*” also shows moderate performance, with some degree of confusion with especially the “*medical_record*” because these two classes sometimes overlap.

The confusion matrix confirms that most errors occur between semantically similar classes, for example between the “*hospitalization_certificate*” and “*medical_record*” or “*medical_certificate*”. These misclassifications suggest that further refinement of features or model tuning might improve differentiation among these closely related classes, or that business experts are needed to validate specific cases of misclassification and in case adjust the labels.

4.2 Prompting with GPT

LLMs are not traditionally employed for classification tasks, given that their primary strength lies in generating coherent text rather than assigning labels. But their deep understanding of language enables them to interpret complex content similarly to human reasoning, thus facilitating accurate document classification.

In this study, GPT-4o, deeply discussed at Chapter 2.4.2, was tested using two distinct prompting methodologies to better analyze the capabilities of the LLM and to understand how much contextual information helps the model generate correct answers. GPT-4o did not undergo any fine-tuning or additional training on labeled examples for this specific classification task, only background knowledge and prompting techniques were used. Both methodologies were tested in a **zero-shot** setting, meaning no examples were given in the prompt as reference to follow.

A one-shot prompting approach was also implemented by including an example document and its corresponding class in the prompt. However, this approach did not improve the task’s overall performance, instead it slightly biased the model toward the provided example class in cases of uncertainty. Few-shot prompting with one example for each class was also considered, but discarded due to the prohibitive cost associated with lengthy prompts. It is worth remarking that working with large language models like GPT-4o is expensive, especially when prompts are very long. The **longer the prompt, the higher the cost**, as LLM providers typically charge a cost based on the number of input and output tokens. In this case, the average document length was approximately 900 tokens per page, and the prompt itself ranged around 1000 tokens, depending on the approach, so additional example pages would have increased the cost unnecessarily.

Another possible input format was to use document images instead of text. However, this approach did not bring better results, likely because, for the classification task, the crucial information lies in the text rather than the document’s visual structure.

The documents originated from an Italian company, so their content is in **Italian**. GPT-4o is a multilingual, meaning it is a model trained on diverse language datasets, reflecting the remarkable versatility of large language models for different tasks across different languages. As the primary language of GPT-4o’s training

data is English, it is well-documented that English prompts often yield more accurate results compared to those in other languages, such as Italian. This observation was confirmed through experimentation in this study, by **prompting in English** and Italian, the first one led to better results. To mitigate this language mismatch, the most effective strategy was to structure the prompt in English while retaining class names and **keywords in Italian**. This hybrid approach allowed the model to leverage its English-language training for understanding task instructions while drawing on its knowledge of Italian for domain-specific terminology.

To help the model deeply understand the classes and their differences, the prompt included detailed descriptions of each class, along with relevant metadata that can serve as distinguishing features. Some metadata are shared between classes, for example, the *"emission_date"* or the *"diagnosis"* are present in all the medical documents. But there are other metadata which are unique to specific classes, such as *"event_date"* for the *"death_certificate"*, or *"revoked_date_start/end"* for the *"leave_certificate"*. If one of these metadata is found in the text, the model is sure to identify that specific class. This approach aligns with the subsequent phase of data extraction from documents, providing the model with crucial contextual clues for classification. Although not every document page contains all the expected metadata, their presence often strongly indicates a particular class, making them a useful heuristic for classification.

This method leverages the multi-capability of LLMs, allowing them to transition between different types of tasks, in this case, classification and metadata identification.

4.2.1 Different prompt strategies

Single-Prompt Classification

In the first approach, the model was given a single, comprehensive prompt containing a detailed description of each class and its defining characteristics. The prompt concluded with the raw text of the document page to be classified. GPT-4o was then instructed to output a single class label based on its interpretation of the provided text. This method exploits GPT-4o's capacity for language comprehension, as the entire context—including class definitions and the textual content—is

fed to the model in one prompt. Although this approach does not extensively utilize the model's generative abilities, its deep semantic understanding can yield high accuracy in zero-shot classification, particularly when class descriptions and metadata are well-structured and clearly differentiated.

Listing 4.1: GPT-4o Multiclass Classification Prompt

```
1 "system_prompt":
2     You are an AI assistant particularly skilled and meticulous in the classification of specific
3     documents given as a text, akin to how an experienced human operator would perform the task.
4     You are aware of the need to provide an answer exactly in the requested format, without
5     adding any additional information regarding the reasoning process.
6
7 "user_prompts":
8     "Task"
9     Your task is to classify the document and return the output in the requested format.
10    The classes to assign are: Documento di ricovero con cartella clinica, Certificato medico,
11    Certificato di ricovero, Certificato di morte, Revoca ferie, Assunzione, and Altro
12
13    "Context":
14    The document to classify contains details related to the medical and work fields.
15    Document characteristics:
16    - Document structure may vary between the same type of document.
17    - Document of the same class contains very similar information.
18    - Document contains some specific information and a pattern that helps you to classify it.
19
20    "Explanation of the classes"
21    - "assunzione":
22        It is a document that certifies the hiring of a person.
23        Consider that inside it should be present the metadata: data emissione.
24    - "cert_morte":
25        It is a document that certifies and explains the death of a person.
26        Consider that inside it should be present the metadata: data decesso.
27    - "revoca_ferie":
28        It is a document that certifies the revocation of holidays with some explanation. It could
29        be written as an email or letter.
30        Consider that inside it should be present the metadata: data emissione, periodo revocato
31        start/end.
32    - "cert_medico":
33        It is a document that certifies the health condition of a person. It is usually written by
34        hand by the doctor, which could state the drugs to take.
35        Consider that inside it should be present the metadata: data emissione, data insorgenza
36        sintomi, diagnosi.
37    - "cert_ricovero":
38        It is a document that certifies the hospitalization of a person. It is quite a short
39        document with the main information about the hospitalization.
40        Consider that inside it should be present the metadata: data emissione, data ingresso,
41        diagnosi. Here there is no reference to the end of the hospitalization period, it is usually
42        written at the beginning.
```



```

32 - "cart_clin":
33     It is a document that contains the clinical folder of a person together with the
information about the hospitalization and the exams conducted in the clinic.
34     It is a long document with all the information about the patient. It contains the clinical
history of the patient, and there is a reference to the end of the hospitalization period.
35     Consider that inside it should be present the metadata: data emissione, data ingresso/
uscita, ingresso/uscita orario, diagnosi.
36 - "altro":
37     These are all the documents that do not belong to the other classes and do not contain any
of the previous information.
38     They are blank pages, or images or documents that do not contain any specific information.
Indeed, the document does not have specific metadata to extract.
39     It is very important that you use this class as a specific category, and **not** when you
are not sure about the classification. Try to avoid using it too much.
40
41 "Output":
42     The output must be just the class of the document. Select the class that best fits the
document considering all the previous information.
43     Don't report or extract any metadata; use those details only to help classify the document.
44     You will be provided with a document in text format, the text has been extracted in blocks
by an OCR.
45     Your task is to classify the document and return the expected output in the requested format
46     "placeholder__input_text"

```

Binary Classification in Cascading Prompts

The second approach involved decomposing the classification task into a series of binary decisions. Specifically, six separate prompts were written, each corresponding to a single class (e.g., *leave_certificate* vs. not *leave_certificate*, *death_certificate* vs. not *death_certificate*, and so forth). A script sequentially used these prompts in a cascading structure: the LLM was first asked whether the text belonged to a certain class. If the response was positive, the cascade ended, if negative instead, the next class-specific prompt was issued, continuing until one class was confirmed. If none of the binary prompts returned a positive classification, the text was assigned to the *other* class. This fallback mechanism was chosen to reduce misclassifications, as having a dedicated prompt for the *other* class led the model to overpredict this category when uncertain.

This methodology effectively breaks down the multi-class task into a chain of simpler yes-no queries, leveraging GPT-4o's interpretative ability in each binary decision. The **order of the cascade** was chosen carefully, because as soon as

the model answered positively to a class, the loop was ended so the first classes needed to be the most clear and simple one to be sure that the model got them right at the beginning. The final order chosen was first the employment and leave certificate, as they are not from the medical field, and then the death and generic medical certificate, the hospitalization certificate, and finally the medical record. Two variations of this approach were tested, differing in the amount of context provided in each prompt. In one case (4.2), the prompt included only the definition and context related to the specific class being evaluated. In the other case (4.3), the prompt described all possible classes in detail, even when the binary classification focused on just one class. As expected, the latter method improved the model's awareness of alternative class possibilities, reducing the likelihood of misclassification by clarifying distinctions between classes.

Here there are the two prompt variations implemented:

Listing 4.2: GPT-4o Single-class classification prompt with context of only that specific class, as an example the *revoca_ferie* class is shown

```

1 "system_prompt":
2     You are an AI assistant particularly skilled and meticulous in the "binary classification" of
3     specific documents given as a text, akin to how an experienced human operator would perform
4     the task.
5     You are aware of the need to provide an answer exactly in the requested format, without adding
6     any additional information regarding the reasoning process.
7 "user_prompts":
8     "Task"
9     Your task is to classify the document and return the output in the requested format.
10    You need to say if the document is of the class "revoca_ferie" or not.
11 "Context":
12    The document to classify contains details related to the medical and work fields.
13 "Document characteristics":
14    - Document structure may vary between the same type of document.
15    - Document of the same type contains very similar information.
16    - Document contains some specific information and a pattern that helps you to classify it.
17 "Explanation of the class"
18    Consider this information to have a better knowledge of the domain. At the end of the
19    reasoning, you need to decide if it is a Documento di ricovero con cartella clinica or not.
20 "revoca_ferie":
21    It is a document that certifies the revocation of holidays with some explanation. It could
22    be written as an email or letter. Consider that inside it should be present the metadata:
23    data emissione, periodo revocato start/end.
24 "Important":
25    - The document may contain some OCR errors.
26    - The document may contain some missing information.

```

```

21     - Pay attention to not mistake the class cart_clin and cert_ricovero. The first one contains
      the clinical folder of the patient, the second one is a short document with the main
      information about the hospitalization.
22     - Pay attention to not mistake the class cert_medico and cart_clin. The first one is a
      document that certifies the health condition of a person, and it is usually written by hand,
      the second one contains the clinical folder of the patient.
23 "Output":
24     The output must be just a string: "TRUE" or "FALSE".
25     - TRUE: if the document is a "revoca_ferie".
26     - FALSE: if the document is "not a revoca_ferie".
27     Don't report or extract any metadata, use that information just for reference to help classify
      the document. You will be provided with a document in text format, the text has been
      extracted in blocks by an OCR. Your task is to classify the document and return the expected
      output in the requested format
28     "placeholder__input_text"

```

Listing 4.3: GPT-4o Single-class classification prompt with context of all classes, as an example the *cert_ricovero* class is shown

```

1 "system_prompt":
2     You are an AI assistant particularly skilled and meticulous in the "binary classification" of
      specific documents given as a text, akin to how an experienced human operator would perform
      the task.
3     You are aware of the need to provide an answer exactly in the requested format, without adding
      any additional information regarding the reasoning process.
4 "user_prompts":
5     "Task"
6     Your task is to classify the document and return the output in the requested format.
7     You need to say if the document is of the class "cert_ricovero" or not.
8     "Context": The document to classify contains details related to the medical and work fields.
9     "Document characteristics":
10     - Document structure may vary between the same type of document.
11     - Document of the same type contains very similar information.
12     - Document contains some specific information and a pattern that helps you to classify it.
13     "Explanation of all the class"
14     Consider this information to have a better knowledge of the domain. At the end of the
      reasoning, you need to decide if it is a Documento di ricovero con cartella clinica or not.
15     "assunzione":
16     It is a document that certifies the hiring of a person. Consider that inside it should be
      present the metadata: data emissioni.
17     "cert_morte":
18     It is a document that certifies and explains the death of a person. Consider that inside it
      should be present the metadata: data decesso.
19     "revoca_ferie":
20     It is a document that certifies the revocation of holidays with some explanation. It could be
      written as an email or letter. Consider that inside it should be present the metadata: data
      emissione, periodo revocato start/end.
21     "cert_medico":
22     It is a document that certifies the health condition of a person. It is usually written by

```

```

hand by the doctor, which could state the drugs to take. Consider that inside it should be
present the metadata: data emissione, data insorgenza sintomi, diagnosi.
23 "cert_ricovero":
24 It is a document that certifies the hospitalization of a person. It is quite a short document
with the main information about the hospitalization. Consider that inside it should be
present the metadata: data emissione, data ingresso, diagnosi. Here there is no reference to
the end of the hospitalization period, it is usually written at the beginning.
25 "cart_clin":
26 It is a document that contains the clinical folder of a person together with the information
about the hospitalization and the exams conducted in the clinic. It is a long document with
all the information about the patient. It contains the clinical history of the patient, and
there is a reference to the end of the hospitalization period.
27 Consider that inside it should be present the metadata: data emissione, data ingresso/uscita,
ingresso/uscita orario, diagnosi.
28 "altro":
29 These are all the documents that do not belong to the other classes and do not contain any of
the previous information. Indeed, the document does not have specific metadata to extract.
Use this as a specific category, not when you are not sure about the classification.
30 "Important":
31 - The document may contain some OCR errors.
32 - The document may contain some missing information.
33 - Pay attention to not mistake the class cart_clin and cert_ricovero. The first one contains
the clinical folder of the patient, the second one is a short document with the main
information about the hospitalization.
34 - Pay attention to not mistake the class cert_medico and cart_clin. The first one is a
document that certifies the health condition of a person, and it is usually written by hand,
the second one contains the clinical folder of the patient.
35 "Output":
36 The output must be just a string: TRUE or FALSE.
37 - TRUE: if the document is a "cert_ricovero".
38 - FALSE: if the document is "not a cert_ricovero".
39 Don't report or extract any metadata, use that information just for reference to help classify
the document. You will be provided with a document in text format, the text has been
extracted in blocks by an OCR. Your task is to classify the document and return the expected
output in the requested format
40 "placeholder__input_text"

```

4.2.2 Performance metrics

The performance results presented in Tables 4.6 and 4.5 provide insights about the two approaches, the multiclass classification versus the cascading binary classification method. The overall performances are presented in Table 4.5.

The multiclass prompt method achieves slightly better overall performance, with an overall F1-score of **71%**, and in comparison, the binary classification approach achieves just 67%. These results suggest that the additional context

	Accuracy	Precision	Recall	F1 Score
Multiclass	0.749	0.766	0.770	0.709
Binary	0.628	0.790	0.725	0.668

Table 4.5: GPT-4o overall performances of the multiclass and binary classification approaches, in the test set. In bold are highlighted the best values of the comparison.

provided in the multiclass prompt likely helps the decision-making process by enabling the model to consider multiple class-specific cues simultaneously. With this experiment, we understood that for GPT-4o it is better to have a comprehensive understanding of all the possible classes, so it can infer from its background knowledge, to choose more carefully which class to assign the input document.

Interesting is that the binary model, which decomposes the problem into a series of simpler yes-no decisions, has a slightly higher precision than the multiclass, indicating a better ability to identify correctly positive instances, even if some classes remain difficult to distinguish due to overlapping definitions and ambiguous metadata. The method simplifies the task by decomposing it into a series of simpler yes-no decisions, its lower recall suggests that the model tends to answer negatively to a higher number of relevant instances. Possibly because the cascade process imposes the model to be either very sure and answer positively to the correct class or make a mistake because it was not so sure, and at the end of the cascade process, another class will be assigned wrongly. This reasoning is confirmed with a higher precision, when the model is sure it answers correctly, and when the document is complex, it makes more mistakes than the multiclass because it can't choose the correct class directly, but there is the cascade mechanism.

Overall, the results indicate that while the classification task appears a straightforward task, the model struggled with distinguishing between certain classes, likely due to overlapping definitions and the complexity of the textual content.

4.3 Comparison and Discussion

Prompting is essential when working with LLMs, and the results of this study underscore the importance of context-rich, well-structured prompts. The combi-

	Precision		Recall		F1 score	
	multiclass	binary	multiclass	binary	multiclass	binary
Medical record	0,986	0,982	0,728	0,578	0,837	0,727
Leave cert	1	1	0,920	0,893	0,958	0,944
Employment	1	1	0,615	0,654	0,762	0,791
Hospitalization cert	0,684	1	0,520	0,320	0,591	0,485
Other	0,027	0,018	0,727	0,727	0,051	0,034
Death cert	0,947	0,877	0,986	0,986	0,966	0,928
Medical cert	0,717	0,654	0,892	0,919	0,795	0,764

Table 4.6: GPT-4o performance over all classes in the test. The table shows the comparison between *multiclass* and *binary* classifier approaches. In bold are highlighted the best values of the comparison.

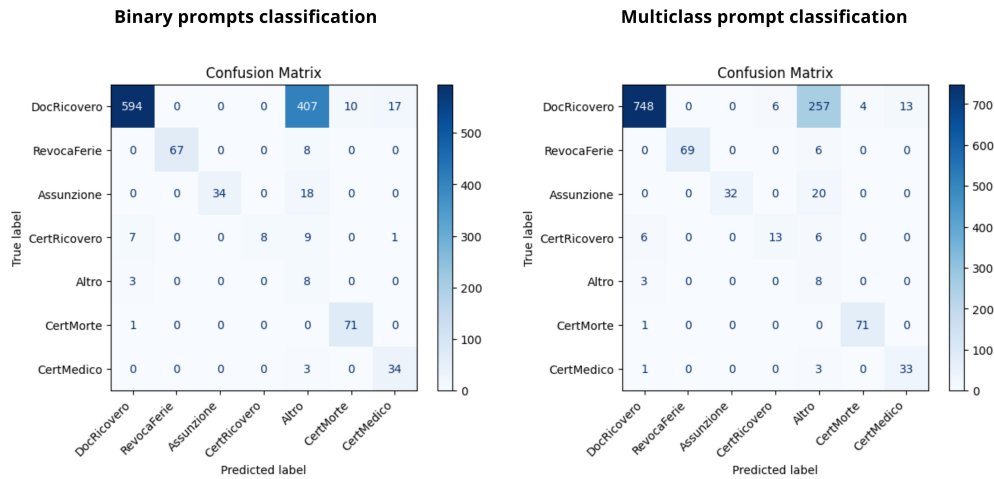


Figure 4.5: Comparison of the two confusion matrices, on the left the *binary* classifier and on the right the *multiclass* classifier. It shows the True Label and Predicted values of each of the seven classes in the test set.

nation of zero-shot learning, careful prompt engineering, and the strategic use of metadata enabled GPT-4o to achieve high precision in certain classes, without task-specific training, thereby demonstrating its potential for document classification in multilingual and domain-specific settings. However, as illustrated in Table 4.7, the final comparison reveals notable differences between the GPT-4o and BERT approaches, with BERT achieving the best performances overall.

Overall, the BERT-based classifier consistently had higher recall and F1-scores across nearly all document categories. This suggests that the SLM, benefiting from robust word representations and a fine-tuned classification head, is more ef-

	Precision		Recall		F1 score	
	GPT-4o	BERT	GPT-4o	BERT	GPT-4o	BERT
Medical Record	0,986	0,987	0,728	0,994	0,837	0,991
Leave Cert	1	0,950	0,920	0,967	0,958	0,959
Employment	1	0,857	0,615	0,837	0,762	0,847
Hospitalization Cert	0,684	0,871	0,520	0,931	0,591	0,901
Other	0,027	0,762	0,727	0,762	0,051	0,762
Death Cert	0,947	1	0,986	0,955	0,966	0,977
Medical Cert	0,717	0,842	0,892	0,842	0,795	0,842

Table 4.7: Performance comparison between GPT-4o and BERT classifiers over all classes in the test.

fective at capturing relevant instances in these complex documents. For instance, GPT-4o demonstrates high precision for well-defined classes such as *Employment* and *Leave Certificate*, likely because the prompt 4.1 provides clear definitions and context that allow the model to leverage its extensive background knowledge.

However, GPT-4o exhibits significantly lower recall in most classes and struggles particularly with the *Other* class (4.7 shows a precision of 0.027 and an F1-score of 0.051). This low performance in the *Other* class is attributed to its role as a default assignment when the model is not sufficiently confident about any other category, which leads to a high rate of misclassification in ambiguous cases.

In contrast, the BERT-based approach, with its targeted classification training on a large number of documents, achieves a more balanced and effective classification outcome. Its success highlights the fact that, for a relatively straightforward task like this one, where the output is a fixed label and the generative capabilities of LLMs are not fully exploited, a well-tuned SLM can outperform even the most powerful LLM.

Additionally, the **cost implications** are significant: the GPT-4o method incurs a per-document expense based on token usage, with the multiclass prompt alone consuming approximately 1400 tokens, in addition to an average of 900 tokens for the input document. This results in a cost of roughly **0.7 cents per document**. In contrast, the BERT-based approach operates at a fixed, much lower computational cost, enabling local deployment on affordable hardware, a 16GB T4 GPU

was used for this project.

This raises important questions regarding the practical deployment of large-scale LLMs versus solution based on smaller models, maybe it is not always true that they are perfect for all tasks and with their knowledge, they can answer to all the queries and prompts. In this case, a lower-cost, easier-to-deploy, with higher performance solution was found implementing a classification model based on BERT, a much smaller model.

This thesis further investigates this topic, analysing the more complex task of data extraction, aiming to determine whether SLMs can match or even exceed the performance of LLMs while significantly reducing resource demands and operational costs.

Chapter 5

Extraction

The next step of the document processing pipeline, as shown in 1.1 is the extraction of the metadata based on the classification outputs, each class has its own specific data to extract, which are shown in Table 3.1.

The classification and extraction tasks operate at different levels of granularity: classification is performed at the page level, whereas **extraction is conducted over logical documents**, which group pages of the same class within a single document (as detailed in Chapter 3). Following the structure illustrated in Figure 3.2, the first step of this stage is to create the actual logical documents, and processing only those associated with one of the six meaningful classes and discarding all pages labeled as *Other*.

The goal of this chapter is double. First, to evaluate the performance of state-of-the-art models on the metadata extraction task, assessing the reliability of LLM in a real-world pipeline and how trustworthy their responses are. Second, to adapt and fine-tune a smaller language model for this task and analyze its results, evaluating the feasibility of using SLMs in real-world, task-specific scenarios. For this comparison, GPT-4o was used as the LLM and LLaMA 3.2 as the SLM.

5.1 Evaluation criteria

For this task, the availability of labelled metadata across all metadata of the six classes enabled an extensive evaluation, supported by some performance metrics and various iteration runs.

The performance of the extraction task was evaluated using a comprehensive set of metrics: Accuracy, Precision, Recall, and F1-score. Given the increased complexity of extraction compared to classification, it is crucial to clearly redefine each metric and their specific interpretation in this context. Some data are relatively simple to identify and retrieve, while others require more complex reasoning and advanced consideration. For each metadata field of each class, a separate metric was computed, revealing significant performance variation both between classes and, within individual classes, across different metadata fields.

Firstly, it is essential to precisely define the concepts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). A true positive (TP) occurs when the model extracts the correct value. A false positive (FP) arises when the model extracts a value, but it is incorrect. A true negative (TN) is when there is no data to extract, and the model correctly returns no output. Finally, a false negative (FN) occurs when the model should not extract anything but instead it returns a value.

Table 5.1 provides concrete examples to illustrate these definitions:

True Label	Extracted Value	Interpretation
1	1	TP
1	2	FP
1	-	FN
-	-	TN
-	1	FP

Table 5.1: Interpretation of true positives, true negatives, false positives, and false negatives based on model predictions and true labels. The values reported are just an example of possible numbers extracted, and the "-" means no data.

When dealing with metadata, it is necessary to account for the variability in data types, which are not always in simple numerical formats. In this specific scenario, the metadata fields consist of dates, times, and textual strings. The definition of equality for these data types must be rigorously established to ensure consistency and coherence in the later evaluation.

For date fields, all **dates** are formatted uniformly as **dd/mm/yyyy**, and two dates are considered equal only if they match exactly.

For **textual fields**, the evaluation criteria differ based on their semantic nature.

Two primary types are distinguished, and the evaluation is customized based on their characteristics:

- **diagnosis descriptions:** these are typically long, complex strings containing domain-specific terminology. These strings allow minor variations due to potential rephrasing or summarization without altering the meaning. Therefore, a more flexible similarity threshold is applied.
- **personal names:** these simply consist of a first and last name. These strings indeed require stricter matching, given their shorter length and the higher importance of exact identity.

Both types of text fields undergo preprocessing, including lowercasing, removal of punctuation and special characters, and trimming of leading or trailing spaces. After preprocessing, string similarity is assessed using multiple techniques:

- **Levenshtein ratio** measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another, normalized by the string length.
- **Jaro-Winkler similarity** emphasizes common prefixes and adjusts the similarity score based on the number of matching characters and their positions.
- **Jaccard similarity** measures the intersection over the union of character bigrams or word sets, reflecting the proportion of shared elements.

For the diagnosis strings, equality is confirmed if the Levenshtein ratio is **>0.75**. Instead, two names are considered equals if the maximum similarity score among the three (Levenshtein, Jaro-Winkler, and Jaccard similarities) exceeds **0.85**. The string evaluation process can be summarized in the following pseudocode:

```
function is_equal(str1, str2, metadata_type):
    str1, str2 = preprocess(str1), preprocess(str2)
    if str1 == str2:
        return True
    if metadata_type == 'diagnosis':
        return levenshtein_ratio(str1, str2) > 0.75
    if metadata_type == 'name':
        max_score = max(
            levenshtein_ratio(str1, str2),
```

```

        jaro_winkler_similarity(str1, str2),
        jaccard_similarity(str1, str2)
    )
    return max_score > 0.85
return False

```

Given these considerations, the performance metrics are defined as follows:

- **Accuracy** measures the proportion of correctly classified instances out of all instances in the dataset, providing an overall measure of correctness.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision** quantifies the proportion of true positives among all predicted positive instances, reflecting the model's certainty and the trustworthiness of its extractions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

High Precision indicates a low rate of false positive predictions, which is crucial in real-world scenarios for trustworthy extractions.

- **Recall** measures the proportion of true positives among all actual positive instances, indicating the model's ability to detect all relevant instances of a particular class.

$$\text{Recall} = \frac{TP}{TP + FN}$$

High Recall ensures that the model captures most of the relevant data, it is a form of coverage of the metadata extraction, even at the risk of increasing false positives.

- **F1-score**, being the harmonic mean of Precision and Recall, offers a balanced measure of both metrics. It is especially useful for providing an overall assessment of the model's performance on metadata extraction.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1-score indicates a good balance between Precision and Recall, making it an essential metric for evaluating overall performance.

Achieving high performance across these metrics is of course the ultimate goal. Evaluating them together is crucial, as they each highlight distinct aspects of the model’s behavior. For instance, given the same F1-score, the trade-off between higher Recall and higher Precision can lead to very different real-world implications. The prioritization of one metric over the other often depends on business requirements. For example, in scenarios with a modest number of documents, and a possibly human-in-the-loop validation, high Precision is preferable, as missing values due to lower Recall can be manually addressed. Conversely, when dealing with large-scale data for statistical analyses, higher Recall becomes more valuable to ensure comprehensive data collection.

5.2 LLM

The large language model GPT-4o was employed for metadata extraction, leveraging its extensive general-purpose capabilities refined through careful prompt engineering. As established in the previous Classification chapter 4, the quality and specificity of prompts are crucial for guiding the model’s performance. LLMs possess remarkable adaptability across various tasks but require clear and precise instructions to produce accurate results.

For each document class, a **distinct prompt** was crafted to instruct the model on extracting its specific metadata. These prompts provided detailed contextual information based on each document type, enhancing the model’s comprehension of the specific domain. An extensive fine-tuning of these prompts was conducted to identify configurations yielding optimal performance.

Due to their length, it is infeasible to report all prompts in this paper, however, key insights from the tuning process are discussed here. First significant observation was that including examples of actual documents within the prompts substantially improved the model’s accuracy and contextual understanding of what to extract. The number of examples varied by document class, optimized as a hyperparameter to achieve effective **few-shot prompting**.

A delicate balance was required: using only one example proved insufficient, while too many examples were confusing and increased the token count too much. Prompts which are too long and with numerous examples often caused the model

to mistakenly return metadata from the examples rather than extract them from the input text, leading to noise and reduced precision. This effect was even worse in this case, by the dense and lengthy nature of the documents, for which each example added approximately 1000 tokens.

Table 5.2 presents a statistical analysis of the final versions of the six prompts used for metadata extraction. It highlights the number of examples, token count, and associated costs, reflecting the varying complexity and verbosity of each document class.

	Empl.	LeaveCert	Hospit.Cert	Med.Cert	Med.Record	DeathCert
Num. examples	3	3	4	3	4	2
Words	2620	1490	2160	1750	6520	1020
Input Tokens	3670	2080	3020	2450	9120	1430
Cost input (cent)	1.01	0.57	0.83	0.67	2.51	0.39
Output Tokens	35	70	80	35	80	25
Cost output (cent)	0.036	0.036	0.029	0.087	0.080	0.087
Total Cost (cent)	1.05	0.61	0.86	0.76	2.59	0.48

Table 5.2: Statistical analysis of the final version of the six prompts used by GPT-4o to extract the metadata.

Another critical insight was the **complexity of certain metadata** fields, such as diagnoses, which required the model to identify and select the most relevant information from multiple possible candidates within the text. In these cases, domain experts clarified some extraction rules, which were remarked in the prompt to help the model distinguish between various possibilities in case of uncertainty. Prompt reinforcement techniques were employed to mitigate the risk of losing important instructions in longer prompts. Given the known tendency of LLMs to prioritize information at the beginning and end of prompts while overlooking the middle sections [42], key extraction guidelines were emphasized both at the start and the conclusion of the prompts.

5.2.1 Performances

At the fine-tuning process, conducted in collaboration with **domain experts** to validate the model’s outputs, the final evaluation metrics were computed. Table 5.3 presents the performance results for each document class and their respective metadata fields.

Classes	Metadata	Accuracy	Precision	Recall	F1
hospital. cert	emission_date	0.97	0.99	0.97	0.98
	admission_date	0.95	0.98	0.97	0.98
	diagnosis	0.94	0.94	0.71	0.81
medical record	admission_date	0.86	0.89	0.91	0.90
	discharge_date	0.83	0.87	0.88	0.87
	admission_time	0.90	0.82	0.86	0.84
	discharge_time	0.88	0.76	0.80	0.78
	diagnosis	0.67	0.65	0.69	0.67
medical cert	emission_date	0.83	0.86	0.83	0.85
	symptom_onset_date	0.89	0.59	0.67	0.62
	diagnosis	0.60	0.75	0.59	0.66
death cert	event_date	0.97	0.97	0.97	0.97
	name	0.92	0.95	0.92	0.96
leave cert	emission_date	0.98	0.99	0.97	0.98
	revoked_date_start	0.94	0.97	0.95	0.96
	revoked_date_end	0.92	0.96	0.93	0.94
employment	emission_date	0.90	0.91	0.94	0.92

Table 5.3: Performance metrics of GPT-4o extraction of each metadata for each document class.

The results reflect the model’s **overall effectiveness**, with high performance across most metadata fields. They also reflect the varying levels of complexity of the different metadata types. Highly structured data, such as emission dates, achieved near-perfect performance across document classes. In contrast, fields like diagnosis posed greater challenges, often requiring more context and careful prompt design. These fields exhibited lower recall and F1 scores, underscoring the complexity of selecting the correct information from multiple textual candidates.

The **cost analysis** reported in Table 5.2 reveals the substantial expense associated with input token processing, particularly for longer documents like medical records. This highlights a trade-off between prompt completeness and operational costs, suggesting the need for other strategies.

These results **confirm the ability of the LLM** in tasks like this one, where a complex text needs to be analysed and specific information needs to be extracted following the specific instruction of the prompt. They also emphasize the importance of custom and specific prompt engineering techniques to maximize the model’s accuracy and reliability.

5.3 SLMs

The objective of this section is to assess whether Small Language Models can achieve performance comparable to that of Large Language Models on a specific metadata extraction task through effective fine-tuning. To address this challenge, advanced techniques have been employed to enhance the fine-tuning process, such as **knowledge distillation**, **QLoRA**, and **data augmentation**. These strategies were fundamental in overcoming the limitations of SLMs, particularly in terms of their reduced model size and capacity, by leveraging knowledge transfer, efficient parameter optimization, and enriched training data.

To systematically explore and analyze the potential of SLMs, this section focused on a single document class out of the six available. The selected class needed to offer a combination of structural simplicity and straightforward nature metadata to be extracted, leading to the choice of **death certificates** (see Figure A.4 for an example). This class was ideal for experimentation due to its standard format and its use of specific, well-defined legal or medical terminology. At the same time, the metadata extraction task presents a non-trivial challenge: the need to identify the deceased person’s name and the date of the event, as shown in Table 3.1. The documents often contain multiple personal names—such as those of family members, medical professionals, and legal representatives, and various dates, including the date of the emission or printing of the document, which can lead to ambiguity. The models must therefore demonstrate not only the ability to parse structured information but also to accurately interpret context and resolve potential conflicts

when extracting the correct information.

The SLM used for these experiments was **LLaMA 3.2** [4], in its version with few parameters.

5.3.1 Knowledge Distillation

To enhance the performance of SLMs, knowledge distillation was adopted, a well-established technique for transferring the knowledge and reasoning capabilities of a larger model (the teacher) to a smaller model (the student). Described in more detail in the section 2.2.2. Following the "*Step-by-step distillation*" methodology [14], the LLM served as the teacher model, demonstrating high-level reasoning and task-specific expertise. The SLM learns from the outputs of the LLM, acquiring the ability to replicate its performance with significantly reduced computational requirements and without the huge training done by the LLM, acquiring its reasoning capabilities.

A new prompt was written to instruct the LLM to extract both the metadata of the *death certificates* and the reasoning behind each identified metadata. The rationale for each extraction includes details on where, why, and how the model found the specific information.

Through the prompt development process, it became clear that the **order** in which the metadata and reasoning were asked to be returned was critical to the model's effectiveness. The less efficient approach was discovered to be the one where the metadata was extracted first, followed by the reasoning. Instead, the more effective structure was found by asking the model to provide the reasoning immediately after extracting each individual piece of metadata. This change in the output structure significantly improved the clarity and accuracy of the extractions. The output section of the final prompt indeed looked like this:

Listing 5.1: Output section of the LLM prompt to extract also the reasonings

```
1      **Output:**  
2      {"event_date": "dd/mm/yyyy", "reasoning_event_date": "I found the date in the phrase: ...",  
3      "name": "name surname", "reasoning_name": "I found the name in the phrase ..."}  

```

For example, the reasoning for a name and date appear as shown in the Table 5.4 below:

Name	Reasoning Name	Date	Reasoning Date
Ferrari Giuseppe	I found the name in the phrase: 'FERRARI GIUSEPPE è morto' referring to the deceased person.	09/06/2024	I found the date in the phrase: 'è morto il nove giugno duemilaventiquattro' which translates to 'died on the ninth of June two thousand twenty-four'.
Pasquina Montabelli	I found the name in the phrase: 'MONTEBELLI PASQUINA Nata il' referring to the deceased person. Other names are doctors or officials.	06/08/2021	I found the date in the phrase: 'È mortal il 06/08/2021'.

Table 5.4: Example of reasoning output by the LLM. The examples provided are from the data augmentation process and are not real, ensuring the privacy of the individuals involved.

These reasonings made through knowledge distillation contribute significantly to the goal of enabling SLMs to handle complex metadata extraction tasks while maintaining efficiency and accuracy, as Table 5.5 will better show.

5.3.2 QLoRA

To facilitate the efficient fine-tuning and deployment of larger SLMs on affordable hardware, the Quantized Low-Rank Adaptation (QLoRA) [17] technique was employed. QLoRA is designed to optimize both memory usage and computational efficiency by integrating low-rank adaptations (LoRA) [18] with quantization [43]. This hybrid approach allows the training of models that would typically require multiple powerful GPUs, significantly reducing hardware requirements while maintaining performance.

As an illustration of QLoRA’s efficiency, fine-tuning a model with 8 billion parameters would typically require 16 GB of memory for just the model parameters. However, by leveraging QLoRA, only a quarter of this memory was necessary, thanks to the introduction of low-rank matrices. The gradients and optimizers, which would normally require an additional 16 GB and 32 GB of memory, respectively, required only 120 MB and 240 MB with QLoRA due to the reduced number of parameters from the adapter. In total, a full fine-tuning session would typically demand 64 GB of GPU memory, whereas with QLoRA, the required memory was

reduced to approximately 5 GB, demonstrating its remarkable efficiency.

For this experiment, the **ml.g4dn.xlarge** instance from AWS was utilized. This machine is equipped with a single NVIDIA T4 Tensor Core GPU featuring 16 GB of GPU memory, 16 GB of RAM, and 125 GB of SSD storage, which is well-suited for fine-tuning pre-trained models, inference tasks, and prototyping deep learning pipelines. Its balance of performance and cost-effectiveness made it an excellent choice for lightweight training tasks and GPU-based experimentation.

Despite the hardware’s memory constraints, with a maximum of 16 GB GPU memory, the use of QLoRA enabled the fine-tuning of the LLaMA 1B and LLaMA 8B models efficiently. Specifically, the LoRA technique was combined with 4-bit quantization from the BitsAndBytes library. The quantization utilized the NF4 (Normalized Float 4) data type, which strikes an optimal balance between efficiency and precision. Furthermore, double quantization was applied to reduce memory usage further by quantizing both the model weights and the quantization constants. Computational operations were conducted in bfloat16 format, which provides a good trade-off between memory efficiency and numerical stability.

In this setup, the LoRA configuration required careful tuning of two key parameters: α and r . The r is the parameter that defines the dimensionality of the low-rank update matrices. While the α is a scaling parameter which controls the impact of the low-rank updates on the model’s original weights, ensuring that the adapted model retains meaningful task-specific information. As per best practices, these parameters were set such that one was double the value of the other. The final optimal configuration found was with a rank of $r = 128$, this choice ensured efficient adaptation without imposing excessive memory overhead, and the scaling parameter $\alpha = 256$. To prevent overfitting, a dropout rate of 0.1 was applied to the LoRA layers, promoting better generalization and robustness.

Based on the QLoRA paper, LoRA should be applied to all linear layers within transformer blocks to match the full fine-tuning performance. This was achieved by targeting the following linear layers: q_proj, k_proj, v_proj, o_proj, gate_proj, down_proj, up_proj, and lm_head.

By applying this calibrated combination of quantization and low-rank adaptation, it was possible to train the LLaMA models without exceeding the limited 16 GB

GPU memory, while preserving performance, as the results will demonstrate. This demonstrates the effectiveness of QLoRA in real-world applications with resource-constrained environments,

5.3.3 Data augmentation

A significant challenge encountered during the fine-tuning was the limited size of the available *death certificate* dataset, which comprised **only 400 documents**, partitioned into 300 for training, 50 for validation, and 50 for testing. This scarcity of training data was an obstacle for the model’s ability to generalize effectively, as the small training set restricts its capacity to learn diverse representations. To address this issue, data augmentation techniques were employed, utilizing the GPT-4o LLM to generate synthetic training samples. The objective was to create additional documents that closely mirrored the style, structure, and content of the original data while maintaining the necessary diversity for robust learning. This augmented dataset, now containing 700 documents, was generated solely from the training set, with the validation and test sets left intact for unbiased evaluation.

Given the sensitive nature of the documents, which include legally and medically relevant information, privacy remained a primary concern throughout the project. The synthetic data generation process not only addressed the scarcity of training data but also adhered to privacy principles by generating entirely fictitious information. Thanks to its wide background general knowledge, the LLM was capable of generating realistic Italian cities, hospital names, and even inventing people’s names, thus performing a form of anonymization while preserving the realism of the data.

These synthetic documents, created for fine-tuning the SLM model, required labels. To achieve this, the model was configured to output both the generated text and the metadata of that new document. This labeling process was crucial for ensuring the quality and relevance of the synthetic data used in training.

A well-known challenge inherent in working with LLMs is their non-deterministic nature, which affects precision and reproducibility. However, for this specific task of dataset augmentation, the creativity of the LLMs was an advantage to generate a diverse set of documents that follow the structural guidelines while avoiding

repetition. Initially indeed the model generated outputs with similar names and locations, and the dates were often unrealistic, either too far in the past or future. To control the degree of randomness in the model’s output, the **temperature** parameter was adjusted. It influences how deterministic or creative the generated text is, depending on the sampling from the model’s probability distribution over potential next tokens.

- Low Temperature (0 - 0.5): Generates more deterministic outputs, selecting the token with the highest probability. While this results in more stable and reliable outputs, it can lead to rigidity and repetition.
- High temperature (1.0 or higher): Increases randomness and diversity by reducing the preference for the most likely tokens, fostering more creative outputs. However, this may come at the cost of generating less coherent and off-topic responses.

During the iterative refinement of the prompt for this augmentation task, various temperature values were tested, ultimately setting the value to **0.8** to strike a balance between creativity and adherence to the prompt.

For this specific task, where the model needed to replicate the structure of certificates, the **few-shot prompting** was crucial. The more documents the model saw as examples, the better its understanding of the context, all while considering the token cost. To explore all possibilities, both the number of examples and which specific documents to input, were varied to optimize the diversity. Despite the high temperature setting, the model tended to closely follow the structure of the examples provided. To overcome this, the final prompt includes three texts as examples, selected through a **sliding window approach** over the entire training dataset. Additionally, the model was instructed, based on the three input examples, to generate six distinct outputs per prompt, with the remarked requirement that the generated documents share a similar structure but contain different information. This approach led to great results of coherence in the structure, but at the same time, both following the prompt instructions and seeing all the time different documents, guaranteed that the outputs were always different.

The final prompt configuration is illustrated in Listing 5.2.

Listing 5.2: Data Augmentation with GPT-4o of the class *death_certificate*

```

1 "system_prompt":
2   You are a highly trained AI, exceptionally skilled in reproducing text and generating text from
   given examples, as a human operator would do.
3 "user_prompts":
4   "Task":
5     Your task is to augment a dataset of documents. You have to read the given input texts and
   generate new text that is similar to the input text. Understand the meaning of the input
   texts, familiarize yourself with the context, and generate text that is coherent with the
   input text.
6   "Input": You will receive the text of a scanned document, containing a series of information
   . In particular, the document is a death certificate, which includes the name and the date of
   death of a person.
7   "placeholder__example_text"
8   "Output": Be creative and imaginative when inventing names, dates, locations and situations.
   Strive for variety and avoid repetition to ensure the generated texts are diverse.
9   "Important": Do not repeat yourself. Generate something new, especially changing dates,
   names, and locations. Dates in the generated text can be written in textual form (e.g., "
   August 6, 2020") or in the format DD/MM/YYYY.
10  Ensure dates are realistic, before February 2025. Occasionally write dates in textual form.
11  "Step-by-Step Instructions":
12    - Analyze the given input texts carefully.
13    - Identify commonalities and differences.
14    - Generate new text, changing especially dates, names, and locations.
15    - Ensure the output text is coherent with the input text.
16    - Do not repeat yourself; generate something new each time.
17  "Output format": Return a list of dictionaries containing the generated text, the name of
   the deceased person, and the date of death. Each dictionary should follow this structure:
18    [{ "output_text": "output text 1",
19       "name": "name 1",
20       "event_date": "DD/MM/YYYY"
21     },
22     ...,
23     { "output_text": "output text 6",
24       "name": "name 6",
25       "event_date": "DD/MM/YYYY"
26     }]

```

5.3.4 Performances

The effectiveness of the techniques described above was rigorously evaluated by measuring the performance of the fine-tuned SLMs on the metadata extraction task. Table 5.5 presents a detailed comparison of precision, recall, and F1 scores achieved by the LLaMA 1B model when employing data augmentation, knowledge distillation, and their combination. QLoRA was used as a default to be able to

run the model on the machine.

Model	Techniques	Name			Event Date		
		Precision	Recall	F1	Precision	Recall	F1
LLaMA 1B	data augmented	0.853	1	0.921	0.971	1	0.985
	KD	0.838	1	0.912	0.892	1	0.943
	data augmented+KD	0.941	1	0.970	1	1	1

Table 5.5: Performance metrics for the LLaMA 1B model using data augmentation and knowledge distillation (KD) techniques.

The combination of both techniques yields the highest F1 scores across both metadata fields, with a perfect score of **100%** for the event date values, which is particularly remarkable. This outcome illustrates that even with the application of QLoRA, the performance of the model is not compromised. These results highlight the potential of these techniques to enhance the performance of small models, enabling them to compete with larger models on specific tasks.

5.4 Comparison and Discussion

The primary objective of this project was to compare the performance of LLMs and SLMs on a specific metadata extraction task. Table 5.6 presents the final comparison of precision, recall, and F1 scores for the LLM GPT-4o and two LLaMA models (1B and 7B parameters), both of which employed QLoRA and Knowledge Distillation techniques. As previously discussed, the combination of these techniques significantly enhanced the performance of the LLaMA models, which, thanks to QLoRA, were fine-tuned.

The results are impressive and surpass initial expectations. Even the smallest model, LLaMA 1B, achieved the same performance as GPT-4o, and in some cases, exceeded it. Specifically, the LLaMA 1B model achieved the highest F1 scores across both metadata fields, including a perfect score for all the metrics for the "Event Date". This demonstrates that, even with a significantly smaller parameter size, it can compete effectively with larger models like GPT-4o.

It is worth noting that the test set used for evaluation was relatively small, compris-

Model	Name			Event Date		
	Precision	Recall	F1	Precision	Recall	F1
GPT-4o	0.941	0.972	0.959	0.974	1	0.987
LLaMA 1B	0.941	1	0.970	1	1	1
LLaMA 8B	1	1	1	0.971	1	0.985

Table 5.6: Comparison metrics between LLM GPT-4o and two SLMs, the LLaMA 1b and LLaMA 8b, for which both data augmentation and Knowledge Distillation techniques were applied

ing only 50 documents. While this may limit the generalizability of the results, the observed performance suggests that the combination of data augmentation, Knowledge Distillation, and QLoRA can significantly enhance the abilities of small models in specific tasks, potentially enabling them to rival larger models in terms of performance.

Another interesting aspect, apart from the performances, is the comparison of the **costs** of the solution with the LLM and with the SLM. For the LLM, the cost as already explained depends on the number of input and output tokens in the prompt. So it depends on the length of the input documents. An average statistic of how much does the GPT-4o cost per document, based always on the cost shown at 2.1, is **0.5 cents**.

The solution with the SLM instead depends on many factors:

- Type of instance: In this case, the ml.g4dn.xlarge instance was used, with a pricing of 0.68€/h.
- Batch size: The larger the batch size, the less time is required for the machine to process the documents. In this case, the maximum batch size that could be set was 16.
- Model: The choice of model directly impacts the cost, as the best instance type on AWS is selected based on the model’s requirements. Smaller models are preferred to improve processing speed on the machine.
- Framework: This thesis utilized PyTorch, though other efficient frameworks could potentially reduce costs.
- Document length: Longer documents may require more time to process, affecting the overall cost.

So, the cost of the SLM doesn’t depend on the quantity of calls made to the model,

but on the usage timing of the machine, the more data that can be fit into less time, the less is the cost per document.

At parity of performances, for a few documents, the LLM solution is easier to implement because the API is ready-to-use, and it is just writing the best prompt. But in real-world cases with lots of documents to process, the SLM solution is worth it. The Figure 5.1 shows exactly the trend of the cost for the SLM solution varying the number of documents to process in one hour. The dotted line is the division in which on the left the LLMs are preferred and on the right instead the SLMs are cheaper.

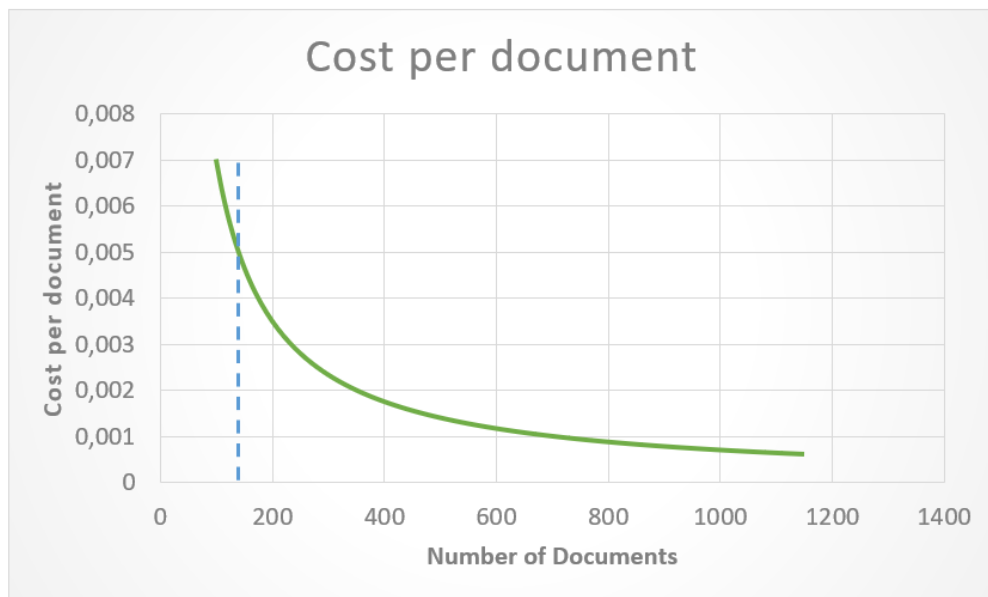


Figure 5.1: Cost comparison for the SLM solution as the number of documents processed per hour increases. The dotted line indicates the point where LLMs are more cost-effective on the left, and SLMs are cheaper on the right.

To be considered also that these *death certificates* were short of just one page, in cases of longer documents for the LLMs the cost increases. For the SLMs, it increases the speed and difficulty in the training, but, hoping for still promising results despite the length of the text, they are still parallelizable in batch, so the cost would almost remain the same as now.

Chapter 6

Conclusion

6.1 Addressing the initial questions

This thesis was based on an **internship project** regarding the techniques for a document processing pipeline, which includes the main steps of the **classification of documents** and based on that, the **metadata extraction** of specific classes. The primary objective of this thesis was to explore the potential of Small Language Models in performing specific tasks, such as document classification and metadata extraction tasks, traditionally dominated by Large Language Models. This investigation was driven by the need to evaluate whether SLMs could match the performance of LLMs on specific tasks while offering benefits in terms of cost, computational efficiency, and deployment feasibility. Through the research, advanced techniques such as data augmentation, knowledge distillation, and QLoRA were employed to enhance the performance of SLMs and mitigate their inherent limitations.

The results unequivocally demonstrate that, with effective training or fine-tuning, SLMs can indeed achieve and, in some cases, improve the performance of LLMs on well-defined, specific tasks. In particular, the usage of a solution for a classification model based on BERT achieved higher performances than the usage of the LLM GPT-4o. The training over a labelled dataset helped the model to learn specialized concepts and the relationship between classes and documents. Then, by focusing on a single document class, the ability of SLMs to extract key metadata such as the person's names and dates, emerged with high precision and

recall. The combination of data augmentation and knowledge distillation played a pivotal role in this success, allowing the smaller models to generalize well despite their limited capacity.

SLMs approach not only **reduced memory usage** but also maintained high performance, demonstrating that they can be adapted and optimized for deployment in resource-constrained environments. The successful implementation of these techniques suggests that the perceived necessity of high-end computational infrastructure for advanced NLP tasks can be reconsidered.

Moreover, the **cost analysis** highlights a significant advantage of SLMs over LLMs when considering real-world deployment. The cost of using LLMs via API is driven by the number of input and output tokens. In contrast, SLMs depend on the cost of the GPU instance machine, or they can even run on local hardware. This cost efficiency becomes even more pronounced when processing large document batches, where the per-document savings accumulate rapidly.

In real-world applications, the choice between SLMs and LLMs ultimately depends on a balance of performance, cost, and operational constraints. LLMs remain the preferable option when handling highly complex, multi-faceted tasks requiring extensive generalization and broad linguistic capabilities. However, SLMs offer a compelling alternative for domain-specific tasks, especially where privacy, cost-efficiency, and deployment flexibility are paramount. This study provides a robust foundation for adopting SLMs in scenarios where high performance must be achieved with limited computational and financial resources.

6.1.1 Implications for Real-World Pipelines

The findings of this research hold significant implications for document processing pipelines. Traditional approaches to document classification and metadata extraction often rely on rule-based systems or require substantial human oversight. While LLMs provide strong generalization capabilities, in the case of many documents to process, they remain computationally expensive and difficult to adapt for domain-specific applications.

SLMs offer a more sustainable solution, particularly when integrated into structured document processing workflows. By training task-specific SLMs to handle

classification, metadata extraction, and other key processing steps, it is possible to build efficient pipelines that scale without excessive computational costs. Moreover, the ability to fine-tune SLMs locally allows for greater adaptability, enabling organizations to continuously refine models without relying on proprietary API services..

SLMs offer also greater **deployment flexibility**. Unlike LLMs, which often require access to cloud-based APIs with associated latency and dependency issues, SLMs can be integrated directly into on-premises systems. This independence from external services ensures consistent performance and reduces the risk of service interruptions, changes in API pricing and availability, and eliminates the need to transmit sensitive data to external servers, thus mitigating data exposure risks. Additionally, the reduced computational footprint of SLMs allows for more scalable and efficient processing pipelines. In scenarios in which the volume of documents to process is very high, the ability to deploy multiple SLM instances in **parallel** without incurring prohibitive costs or infrastructure demands can lead to significant performance gains and operational efficiency. This is particularly relevant in industries such as finance, healthcare, and legal services, where document processing must be precise, secure and cost-effective.

6.2 Future Research

Building on the findings of this thesis, several future research directions can be explored.

First, expanding the analysis of the metadata extraction to **other document classes** would provide a broader validation of SLM capabilities across different data types and task complexities. Investigating the performance of SLMs on documents with more varied structures, specific terminology, handwritten elements, or embedded tables, could further assess their robustness and adaptability, or help define their limitations.

Another interesting aspect to explore would be the **multimodal approach**, which can often help the resolution of complex tasks for which there are documents in which the visual component can help a lot. Many LLMs are multimodal, meaning they can manage different input sources such as text, images, audio and even

videos. Some SLMs are vision-language models like Qwen2.5 VL, LLaMA 3.2 11B-Vision, or DeepSeek-VL2, meaning they can process text and images. For the document pipeline, this could address the limitations of OCR-based text extraction by directly interpreting document images. This approach has the potential to enhance information extraction accuracy from complex document formats where textual content alone may be insufficient.

Thinking even out of just the context of the document pipeline, another possibility is the development of **agent-based systems**. This represents an innovative and efficient architecture where the subjects are specialized SLMs, one for each distinct task. By training individual SLMs on specific tasks and employing techniques like QLoRA to create adaptable model components, it becomes possible to construct a collaborative agent system. It is a shift towards the last trend of creating big LLM trained on huge datasets and with a wide broader knowledge, towards a decomposition of intelligence in SLM very specialized in specific subtasks that create a powerful and robust intelligent system. This system could achieve LLM-level performance across a diverse range of functions while maintaining the benefits and modularity of SLMs.

In conclusion, this thesis underlines the **potentiality of SLMs** as a powerful alternative to LLMs for specific NLP tasks. Through strategic application of advanced training techniques and careful consideration of deployment requirements, SLMs deliver high performance with a greater operational flexibility. Future research in this area is crucial to contrast the growing trend of developing increasingly large models, which are often in the hands of a few organizations. The results of this project demonstrate the valuable opportunities for further advancing SLMs, promoting their broader use and impact across diverse real-world applications.

Appendix A

Document examples

This appendix section reports one example per class of the dataset's documents. The images have been anonymized as they contain sensitive personal data.

P. IVA / C.F. 06371851004
Insc. R.E.A. 964818
CAP. SOC. EURO 100.000
C.A.P. 00187 ROMA

CALTAHOTEL S.R.L.
SEDE IN ROMA - VIA BARBERINI, 22

Sesto San Giovanni, lì

RACCOMANDATA A MANO

Egr. Sig.

OGGETTO : LETTERA DI IMPEGNO.

A seguito delle intese intercorse, Le confermiamo la nostra intenzione di procedere alla Sua assunzione a tempo parziale - 20 ore settimanali - e determinato a decorrere dal 11/04/2022 per mesi 6 (sei), con la qualifica di Segretario di Ricevimento cassa turnante 4^o Livello.

Dalla stessa data il Suo rapporto di lavoro alla nostra dipendenza presso la scrivente sarà regolato dalle norme di Legge e dalle norme vigenti del C.C.N.L. Alberghi Confcommercio.

Il trattamento economico sarà di €. 1.550,69.= lordi mensili riproporzionati al 50%.

Durante il periodo di prova contrattuale di 10 (dieci) giorni, ognuna delle parti può recedere dal contratto senza l'obbligo di preavviso o diritto ad indennità sostitutiva.

Per quanto non previsto si fa riferimento al C.C.N.L. Alberghi Confcommercio.

La preghiamo di volerci restituire copia della presente da Lei sottoscritta per accettazione e benessere in ogni sua singola parte e nella sua interezza.

Distinti saluti

CALTAHOTEL S.R.L.

Figure A.1: The image is an example of an **employment** document. The image has been anonymized as it contains sensitive personal data.



Servizio Risorse Umane
Istituto Cardiocentro Ticino

Signora [redacted]
Cure intensive
In sede

Lugano, 2 dicembre 2022

Dichiarazione

Attestiamo che la signora

[redacted],

è alle dipendenze dell'Istituto Cardiocentro Ticino dall'1 ottobre 2018 in qualità di Inf. dipl. in cure generali, con un contratto a tempo indeterminato ed un'attività del 100%.

Confermiamo che per questioni professionali abbiamo dovuto revocarle le vacanze già programmate dal 2 al 9 gennaio 2023.

Cordiali saluti

Theo Mastelli
Imp. serv. risorse umane


Istituto Cardiocentro Ticino
Via Tessarete 48, CH-6900 Lugano – T +41 (0)91 811 51 11 – F +41 (0)91 811 52 13

Figure A.2: The image is an example of a **leave certificate**. The image has been anonymized as it contains sensitive personal data.

Dott. PAOLA MELZI
Specialista in Pediatria
Amb. Via Mazzini, 5 - MUGGIO'
Tel. 329.8223238

19. 6. 23

Certifico che il piccolo
Tommaso G., ricoverato
dal 12. 6 al 17. 6 per Sindrome
di Kawasaki, viene al
rifiuto assoluto per 30 gg.

Per fede



Figure A.3: The image is an example of a **generic medical certificate**. The image has been anonymized as it contains sensitive personal data.



CITTÀ DI CONEGLIANO

PROVINCIA DI TREVISO

CERTIFICATO DI MORTE

L'UFFICIALE DELLO STATO CIVILE

CERTIFICA

dal registro degli atti di morte di questo Comune,

Anno 2022

Parte II

Serie B

Numero 245

che [REDACTED]

nato/a in [REDACTED]

il [REDACTED]

era vedova di [REDACTED]

è morto/a in [REDACTED]

il giorno [REDACTED]

del mese di [REDACTED]

dell'anno [REDACTED]

Si rilascia in carta libera ai sensi dell'art.7, comma 5°, della legge 29 dicembre 1990, n. 405
Il presente certificato non può essere prodotto agli organi della pubblica amministrazione o ai privati
gestori di pubblici servizi.

Data 05/05/2022



L'UFFICIALE DELLO STATO CIVILE

CAMOZZI Roberta

Istruttore Amministrativo

Figure A.4: The image is an example of an **death certificate**. The image has been anonymized as it contains sensitive personal data.

STABILIMENTO OSPEDALIERO DI VOGHERA
VIA VOLTURNO, 14 27058 VOGHERA
TEL. 0383.6951

NOTIFICA DI RICOVERO

Si notifica, ai sensi di Legge, che il/la Sig./Sig.ra [REDACTED]
nato/a a [REDACTED] il [REDACTED],
Codice Fiscale [REDACTED]
residente a [REDACTED]
in [REDACTED]
è stato/a ricoverato/a il **03/07/2023** con numero di pratica **2023005336**
presso questo Stabilimento Ospedaliero.

VOGHERA, li 28/07/2023

IL MEDICO DI REPARTO



Azienda Socio Sanitaria Territoriale (ASST) di Pavia
P.O. di Voghera
Medicina Interna
Dott.ssa FRANCESCA MARCHESE
C.F. 02613080189

Figure A.5: The image is an example of an **hospitalization certificate**. The image has been anonymized as it contains sensitive personal data.

Bibliography

- [1] AWS. What is ocr (optical character recognition)? https://aws.amazon.com/what-is/ocr/?nc1=h_ls.
- [2] Zapier Editorial Team. Meetings aren't killing productivity; data entry is, 2021. <https://zapier.com/blog/report-how-office-workers-spend-time/>.
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [4] Huggingface. Llama on huggingface. <https://huggingface.co/meta-llama>.
- [5] Huggingface. Llama on huggingface. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>.
- [6] AWS. Amazon textract. <https://aws.amazon.com/it/textract/>.
- [7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. <https://arxiv.org/abs/1706.03762>.
- [8] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2024. <https://arxiv.org/abs/2303.18223>.

- [9] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback, 2024. <https://arxiv.org/abs/2312.14925>.
- [10] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. <https://arxiv.org/abs/2312.12148>.
- [11] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights, 2025. <https://arxiv.org/abs/2409.15790>.
- [12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.
- [13] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help, 2021.
- [14] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023. <https://arxiv.org/abs/2305.02301>.
- [15] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- [16] Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. Mixkd: Towards efficient distillation of large-scale language models, 2021.
- [17] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.
- [18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

- [19] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws, 2023.
- [20] NVIDIA. Cuda c++ programming guide. <https://docs.nvidia.com/cuda/cuda-c-programming-guide/>.
- [21] Artificial Analysis. Intelligence, performance & price analysis. <https://artificialanalysis.ai/models/grok-3>.
- [22] Aili McConnon IBM Tech Reporter. Are bigger language models always better?, 2024. <https://www.ibm.com/think/insights/are-bigger-language-models-better>.
- [23] GLUE Benchmark. General language understanding evaluation. <https://gluebenchmark.com/>.
- [24] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. <https://arxiv.org/abs/1606.05250>.
- [25] Huggingface. Openai official site. <https://openai.com/>.
- [26] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2024. <https://arxiv.org/abs/2402.06196>.
- [27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. <https://arxiv.org/abs/2302.13971>.
- [28] Meta. Llama. <https://www.llama.com/>.
- [29] Noam Shazeer. Glu variants improve transformer, 2020. <http://arxiv.org/abs/2002.05202v1>.

- [30] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. <https://arxiv.org/abs/2104.09864>.
- [31] Rico Sennrich Biao Zhang. Root mean square layer normalization. *arXiv:1910.07467*, 16 October 2019. <https://arxiv.org/abs/1910.07467>.
- [32] Sharan Narang and Software Engineers Google Research Aakanksha Chowdhery. Pathways language model (palm). <https://research.google/blog/pathways-language-model-palm-scaling-to-540-billion-parameters-for-breakthrough-performance/>.
- [33] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. <https://arxiv.org/abs/2203.15556>.
- [34] Edouard Belva. pdf2image 1.17.0, 2024. <https://pypi.org/project/pdf2image/>.
- [35] Poppler. <https://poppler.freedesktop.org/>.
- [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.
- [37] Scikit-learn. compute_class_weight. https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html.
- [38] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang,

- Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021. <https://arxiv.org/abs/2007.14062>.
- [39] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. <https://arxiv.org/abs/2004.05150>.
- [40] Marek Wachnicki Michał Brzozowski. Belt (bert for longer text), 2023. https://github.com/mim-solutions/bert_for_longer_texts.
- [41] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. <https://arxiv.org/abs/1910.01108>.
- [42] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. <https://arxiv.org/abs/2307.031721>.
- [43] Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. Exploiting llm quantization, 2024.